(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 105740312 B
(45)授权公告日 2020.10.23

(21)申请号 201510993945.2
(22)申请日 2015.12.25
(65)同一申请的已公布的文献号
申请公布号 CN 105740312 A
(43)申请公布日 2016.07.06
(30)优先权数据
14307192.6 2014.12.27 EP
(73)专利权人 达索系统公司
地址 法国韦利济－维拉库布莱
(72)发明人 I·贝勒吉提
(74)专利代理机构 永新专利商标代理有限公司 72002
代理人 邬少俊 王英
(51)Int.Cl.
G06F 16/2453(2019.01)
G06F 16/28(2019.01)
审查员 孙巍巍

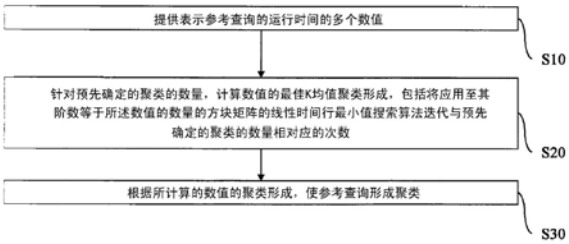
权利要求书2页 说明书13页 附图2页

(54)发明名称

使数据库查询形成聚类以用于运行时间预测

(57)摘要

本发明特别涉及一种计算机实现的用于使数据库中的参考查询形成聚类，从而基于目标查询与参考查询的相似度来预测数据库中的目标查询的运行时间的方法。所述方法包括以下的步骤：提供(S10)表示参考查询的运行时间的多个数值；针对预先确定的聚类的数量，计算数值的最佳K均值聚类形成，其中，计算步骤(S20)包括将应用至其阶数等于所述数值的数量的方块矩阵的线性时间行最小值搜索算法迭代与预先确定的聚类的数量相对应的次数；并且根据计算的所述数值的聚类形成而使参考查询形成聚类。这种方法改进了数据库查询时间预测的领域。



CN 105740312 B

1.一种计算机实现的使数据库中的参考查询形成聚类以用于基于所述数据库中的目标查询与所述参考查询的相似度来对所述目标查询的运行时间进行预测的方法,其中,所述方法包括以下的步骤:

提供(S10)表示所述参考查询的所述运行时间的多个n个数值$x_1,...,x_n$;针对预先确定的聚类的数量K个,计算(S20)所述数值的最佳K均值聚类形成,其中,所述计算步骤(S20)包括将应用至具有等于所述数值的数量的阶数n的方块矩阵H的线性时间行最小值搜索算法迭代与所述预先确定的聚类的数量相对应的次数;并且

根据所述数值的所计算的聚类形成,使所述参考查询形成聚类(S30),其中,所述数值$x_1,...,x_n$被进行排序并相应地被编制了索引,并且所述计算步骤(S20)内的所述迭代包括,在每个相应的迭代等级k并且对于低于所述数值的数量n的每个相应的索引j,最小总失真$TD_{min}(j,k)$的计算对于所编制的索引小于该相应的索引的数值x_i,$i\leqslant j$,的子集是能够实现的,其中根据应用至所述方块矩阵H的所述线性时间行最小值搜索算法,聚类的数量对应于相应的迭代等级k,并且其中,在每个相应的迭代等级k并且对于低于所述数值的数量n的每个相应的索引j,对于每个行索引i和每个列索引j,矩阵条目H(i,j)对应于以下项的和:

在针对该个行索引之前的索引i-1的先前的迭代中计算的最小总失真$TD_{min}(i-1,k-1)$,以及

在该个行索引和该个列索引之间的所述数值的连续的子集$x_i,...,x_j$的失真disto(i,j)。

2.根据权利要求1所述的方法,其中,所述方法还包括,在每个相应的迭代等级k,存储由所述行最小值搜索算法返回的索引$Cut_{min}(j,k)$。

3.根据权利要求2所述的方法,其中,所述方法还包括,在所述计算步骤(S20),从所存储的索引确定最佳聚类形成。

4.根据权利要求3所述的方法,其中,从所存储的索引确定所述最佳聚类形成包括:从所存储的索引Cut_{min}中的最后被索引的数值$Cut_{min}(n,K)$开始迭代地将所述数值进行划分,其中,在每个相应的迭代等级q,针对等于当前形成的聚类的所述最后被索引的数值的索引的所述行索引,所述当前形成的聚类的起始数值的索引等于在所述计算步骤(S20)内的迭代期间在等级K-q的迭代处所存储的索引,K-q等于所述预先确定的聚类的数量减去该相应的迭代等级q。

5.一种用于预测数据库中的目标查询的运行时间的方法,其中,所述方法包括:

提供能够通过权利要求1-4中的任何一项所述的方法获得的所述数据库中的参考查询的聚类形成,并且提供所述参考查询的运行时间;

基于所述目标查询与所述参考查询的相似度而将所述目标查询与所述聚类形成的聚类相关联;并且

根据与所述目标查询相关联的所述聚类的所述参考查询的运行时间,预测所述目标查询的运行时间。

6.一种数据存储介质,其上存储有计算机程序,其包括用于执行权利要求1-5中的任何一项所述的方法的指令。

7.一种系统,其包括耦合至存储器的处理器,所述存储器上记录有权利要求6所述的计算机程序。

8.根据权利要求7所述的系统，其中，所述存储器还存储数据库，所述系统被配置用于对所述数据库中的参考查询和/或对所述数据库中的目标查询执行所述程序。

使数据库查询形成聚类以用于运行时间预测

技术领域

[0001] 本发明特别地涉及数据库工程的领域，并且尤其涉及计算机实现的用于使数据库中的参考查询形成聚类的方法、计算机程序、和系统。

背景技术

[0002] 数据库工程常常包含关于对数据库的查询的不同种类的优化，并且尤其针对执行工作量预测。执行查询所花费的时间被称为工作量，或简单地称为执行时间、或查询运行时间。这通常是优化器将最小化的量，即使常常必须考虑其他成本，如所使用的存储器空间和资源。重要的是要注意到，回答查询所需要的时间是用于计算查询计划与用于执行查询计划的时间的和。一些技术倾向于寻找这两个量之间的权衡(例如，在文档US20050071331A1中)。

[0003] 查询运行时间预测的最重要的应用是查询优化，所述查询优化依靠这些预测以从常常很大数量的候选者中选择特定的执行计划。在现实世界的应用中，数据库系统在必须一些时间内回答很多查询，这就是为什么其运行查询调度(详见文章“Distribution-Based Query Scheduling”，作者Chi，Hacigum，Hsiung，Naughton，2013)，该调度基于不同的标准，如期望的查询运行时间、从其发送查询的用户的优先级、有关的任务的优先级。

[0004] 因此，查询运行时间是需要进行估计以便调度查询的中央信息。特别地，本领域技术人员想要避免可能延迟其他重要的查询的瓶颈查询。此外，评估查询的运行时间以便量化对其计算投入(put)多少资源以使得查询将在给定的时间限制之前被执行，可以是有趣的。这在文档US20050192937中进行了说明。如所提及的，预测查询的运行时间是查询调度的核心，并且因此该问题已经被集中地研究过。

[0005] 一种用于预测查询的运行时间的自然的方式是去寻找已经被执行的查询(并且对于那些已经存储了所使用的时间的查询)的“相似”查询的运行时间。为了实现这种方法，必须要找到好的表示以便利用例如在文章“Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning”，作者Ganapathi，Kuno，Dayal，Wiener，Fox，Jordan，和Patterson(2009)中的度量，或文章“Learning to Predict Response Times for Online Query Scheduling”，作者Macdonald，Tonellotto，和Onnis(2011)中的模型选择，来将好的表示之间的查询进行比较，并且学习如何对其运行时间建模。对查询的表示的选择和根据该表示对运行时间进行建模的方法取决于构建查询的语言。例如，研究报告“Predicting SPARQL Query Execution Time and Suggesting SPARQL Queries Based on Query History”，作者Hasan和Gandon处理SPARQL语言的情况。

[0006] 接着可以在机器学习算法已经被应用到训练集之后，评估对查询的运行时间的预测。特别地，论文“Dynamic Cost Models for Query Runtime Prediction”作者Lima分析了如PostgreSQL的关系数据库的情况，并且利用特定的查询的表示来测试不同的经典机器学习算法。有监督的平均、有监督的线性回归、有监督的K近邻回归、有监督的Nadaraya-Watson回归都是常用的概念。

[0007] 多篇文章已经论证了强大的元方法(meta-method)在于将训练集的查询划分成执行时间的区间,并且接着对每个区间应用机器学习方法。特别地,文章“PQR:Predicting Query Execution Times for Autonomous Workload Management”,作者Gupta,Mehta,Dayal(2008)使用了该时间上的分隔。一些方法也递归地应用该范式并且获得PQR树(如在文档US7895192B2和US20080270346中),其中,树的每个节点都使用在训练数据上学习到的预测子(回忆一下,训练数据是其执行时间已知的一组查询)。最鲁棒性的一般方法中的一种在于使训练集查询的查询时间形成聚类,并且接着预测发现距离其最近的聚类的给定的(新)查询的执行时间(利用特定的相似度的概念,常常基于该查询和该聚类的质心(centroid)之间的相似度),并且接着根据该聚类的查询时间来计算查询时间。上文中提及的Hasan和Gandon的研究报告“Predicting SPARQL Query Execution Time and Suggesting SPARQL Queries Based on Query History”,与Wang和Wong的文章“Optimal k-means Clustering in One Dimension by Dynamic Programming”示出了例如,该最后的计算可以通过评估在聚类上学习到的预测的模型(可以是简单的平均——在这种情况下该最后的计算相当于以一种方式来量化(即,根据预先确定的距离,由值的预先确定的集中的最接近的一个值来替换输入值),或如SVM的更复杂的机器学习方法)来执行。该框架涉及更普遍的聚类分析的领域。

[0008] 聚类分析涉及将对象的集划分成组(称为聚类)的任务,使得在每个组中,数据是相似的(见Jain等的文章“Data Clustering:A Review”)。这表现为数据挖掘(见Chen等的文章“Data mining:an overview from a database perspective”)、机器学习(见Murphy的书,“Machine Learning,A Probabilistic Perspective”)、和大规模搜索(见Goodrum的文章,“Image Information Retrieval:An Overview of Current Research”)中的中心问题。聚类分析是量化的重要工具:向每个聚类分配一个中心,该聚类具有包含在将每个点量化至其聚类的中心的简单量化。

[0009] K均值聚类形成(K-means clustering)问题是聚类分析中最著名的问题,并且由Stuart Lloyd在贝尔实验室在1957年作为针对脉冲编码调制的技术而提出。Lloyd算法将p维的点的集合作为输入,并且将这些点的分区(partition)作为输出,该分区的目的在于最小化“总失真”。该算法仅仅是启发式的(其不提供最佳的聚类形成)。但是事实上,我们不能期待准确的算法,这是因为K均值聚类形成问题在非一维情况下是NP难(NP-hard)的。Lloyd算法现在仍然被广泛使用。也已经提出了几个变体(见J.A.Hartigan(1975),“Clustering algorithms”,John Wiley&Sons,Inc.)

[0010] 一维的应用是特别重要的。针对该问题最著名的算法之一被称为Jenks natural breaks optimization,其在1967年被开发(见Jenks的书,“The Data Model Concept in Statistical Mapping”,在International Yearbook of Cartography中),并且为了制图的目的而被提出。像Lloyd算法一样,其仅仅是启发式的。在2011年,由Wang和Song开发了一种叫做CKmeans的准确的算法(见Wang和Wong的文章,“Optimal k-means Clustering in One Dimension by Dynamic Programming”)。该算法是文档US1543036A的基础。其在时间O(K*n^2)内运行,其中K是所要求的聚类的数量并且n是实数的数量。即使在最近(在2013年),Maarten Hilferink已经开发了更高效的算法并且提供了其实现。该实现致力于制图学,更精确而言用于等值区域图,然而该算法的唯一的文件材料是维基百科的页面

(Fisher's Natural Breaks Classification,可以在优先权日在以下的URL访问:http://wiki.objectvision.nl/index.php/Fisher%27s_Natural_Breaks_Classification)。

[0011] 然而,所有这些现有的方法都是有限制的,因为它们要么不产生最佳K均值聚类,要么太慢。在这样的情况下,仍然有对使查询形成聚类的改进的解决方案存在需要。

发明内容

[0012] 因此提供了一种计算机实现的、使数据库中的参考查询形成聚类以用于基于目标查询与参考查询的相似度而对数据库中的目标查询的运行时间进行预测的方法。所述方法包括提供表示参考查询的运行时间的多个数值的步骤。所述方法也包括针对预先确定聚类的数量,计算数值的最佳K均值聚类形成的步骤。所述计算步骤包括将应用至方块矩阵(其阶数等于所述数值的数量)的线性时间行最小值搜索算法迭代与所述预先确定的聚类的数量相对应的次数。并且,所述方法也包括根据计算的数值的聚类形成,使参考查询形成聚类。

[0013] 所述方法可以包括以下中的一项或多项:

[0014] -将数值被进行排序并相应地被编制了索引,并且在计算步骤内的迭代包括,在每个相应的迭代等级并且对于低于数值的数量的每个相应的索引,最小总失真的计算对于所编制的索引的小于该相应的索引数值的子集是能够实现的,其中根据应用至所述方块矩阵的所述线性时间行最小值搜索算法,聚类的数量对应于相应的迭代等级。

[0015] -在每个相应的迭代等级,并且对于低于数值的数量的每个相应的索引,对于每个行索引和每个列索引,矩阵条目对应于以下项的和:在针对该个行索引之前的索引的先前的迭代中计算的最小总失真,以及在该个行索引和该个列索引之间的所述数值的连续的子集的失真;

[0016] -所述方法还包括,在每个相应的迭代等级,存储由行最小值搜索算法返回的索引;

[0017] -所述方法还包括,在计算步骤,从所存储的索引确定最佳聚类形成;并且/或者

[0018] -从存储的索引确定最佳聚类形成包括,从所存储的索引中的最后的被索引的数值开始迭代地将数值进行划分,其中,在每个响应的迭代等级,针对等于当前形成的聚类的所述最后被索引的数值的索引的所述行索引,当前形成的聚类的起始数值的索引等于在所述计算步骤内的迭代期间在等级的迭代处所存储的索引,所述迭代的等级等于所述预先确定的聚类的数量减去个相应的迭代等级。

[0019] 还提供了用于预测数据库中的目标查询的运行时间的方法。所述预测方法包括提供能够通过上述聚类形成方法获得的对数据库中的参考查询的聚类形成,并且提供参考查询的运行时间。所述预测方法也包括基于目标查询与参考查询的相似度,将目标查询与聚类形成的聚类相关联。并且预测方法接着包括根据与目标查询相关联的聚类的参考查询的运行时间,预测目标查询的运行时间。

[0020] 还提供了计算机程序,其包括用于执行聚类形成和/或预测方法的指令。

[0021] 还提供了计算机可读存储介质,其具有记录在其上的计算机程序。

[0022] 还提供了系统,其包括耦合至存储器的处理器,其中所述存储器具有记录在其上的计算机程序。

[0023] 在示例中，所述存储器还存储数据库，所述系统被配置用于对所述数据库中的参考查询和/或对所述数据库中的目标查询执行所述程序。

附图说明

[0024] 现在将通过非限制性示例的方式，并且参考附图而描述本发明的实施例，其中：

[0025] -图1示出了所述方法的示例的流程图；

[0026] -图2示出了所述系统的示例；并且

[0027] -图3示出了所述方法

具体实施方式

[0028] 参考图1的流程图，提出了计算机实现的、使数据库中的查询(即，关于数据库的查询)形成聚类以用于基于目标查询与参考查询的相似度(例如，根据任何预先确定的查询相似度标准)而预测(即，参考查询及它们的聚类相关于/适合于这样之后的预测，如本身已知的)数据库中的目标查询(即，关于例如实质上同一数据库的另一查询)的运行时间。所述方法包括提供表示参考查询的运行时间的多个数值(例如，任意数)(即，每个数例如实质上对应于/表示/等于各个参考查询的运行时间，各个参考查询的运行时间确实是公知的)的步骤S10。所述方法也包括针对预先确定的聚类的数量的来计算该数值的最佳K均值聚类形成的步骤S20。计算步骤S20包括将应用至方块矩阵(其阶数等于数值的数量)的线性时间行最小值搜索算法迭代与预先确定的聚类的数量相对应的次数。并且，所述方法包括根据所计算的数值的聚类形成而使参考查询形成聚类的步骤S30(即，将每个参考查询分配给其各自的运行时间的聚类)。这种方法改善了参考查询的聚类形成，以用于将来预测目标查询的运行时间。

[0029] 特别地，也如在本领域中所知的，所述方法允许基于参考的运行时间(其运行时间已知)而使参考形成聚类S30。如从数据库工程的领域公知的，因为所述方法通过计算最佳K均值聚类形成S20来执行这样的聚类形成S30，所以从将来的运行时间预测的观点而言，所述方法执行相对好的聚类形成。但是，最重要的是，所述方法通过以下操作执行这样的计算S20：将应用至方块矩阵(其阶数等于数值的数量)的线性时间行最小值搜索算法迭代与预先确定的聚类的数量相对应的次数。由于由所述方法实现的该具体的算法的框架，对最佳K均值聚类形成的计算被快速执行，如在下文中详述。

[0030] 所述方法是计算机实现的。这意味着所述方法的步骤(或基本上所有的步骤)是由至少一个计算机，或任何相似的系统执行的。因此，所述方法的步骤是由计算机可能完全自动地(例如，除了提供S10之外的所有步骤)或半自动地执行的。在示例中，所述方法的步骤中的至少一些的触发可以通过用户-计算机交互(例如，提供S10)来执行。所要求的用户-计算机交互的等级可以取决于所预知的自动控制的等级，并且与实现用户的愿望的需要进行平衡。在示例中，该等级可以是用户定义的和/或预先定义的。

[0031] 所述方法的计算机实现的典型示例是要利用适合于该目的的系统来执行所述方法。该系统可以包括耦合至存储器的处理器，所述存储器具有记录在其上的计算机程序，所述计算机程序包括用于执行所述方法的指令。所述存储器也可以存储适合于保存由所述方法处理的数据的数据库。所述存储器是适合于这种存储的任何硬件，其可能包括几个物理

上不同的部分(例如,一个用于程序,并且有可能一个用于数据库)。

[0032] 图2示出了系统的示例,其中所述系统是客户计算机系统,例如,用户的工作站。示例的客户计算机包括连接至内部通信总线1000的中央处理单元(CPU)1010,也连接至总线的随机存取存储器(RAM)1070。客户计算机还提供有与连接至总线的视频随机存取存储器1100相关联的图形处理单元(GPU)1110。视频RAM 1100在本领域中也被称为帧缓冲器。大容量存储设备控制器1020管理对大容量存储器设备(例如,硬盘驱动器1030)的访问。适于有形地实施计算机程序指令和数据的大容量存储器设备包括所有形式的非易失性存储器,其通过实例方式包括半导体存储器设备,例如EPROM、EEPROM、以及闪速存储器设备;磁盘,例如内置硬盘和可移动磁盘;磁光盘;以及CD-ROM盘1040。前述的任何项可以由专门设计的ASIC(专用集成电路)进行补充或包含在专门设计的ASIC中。网络适配器1050管理对网络1060的访问。客户计算机也可以包括触觉设备1090,例如光标控制设备、键盘等。在客户计算机中使用光标控制设备以允许用户选择性地将光标置于显示器1080上的任意期望的位置处。此外,光标控制设备允许用户选择各种命令,并且输入控制信号。光标控制设备包括用于将控制信号输入系统的多个信号生成设备。通常而言,光标控制设备可以是鼠标,鼠标上的按钮用于生成信号。替代地或额外地,客户计算机系统可以包括感应板,和/或感应屏幕。

[0033] 计算机程序可以包括可以由计算机执行的指令,所述指令包括用于使上述系统执行所述方法的单元。所述程序可以可记录在任何数据存储介质上,包括系统的存储器上。所述程序可以例如在数字电子电路、或在计算机硬件、固件、软件或它们的组合中实现。所述程序可以被实现为装置,例如,有形地在机器可读存储设备中实施的、用于由可编程处理器执行的产品。可以由执行指令的程序以通过对输入数据进行操作并生成输出来执行所述方法的功能的可编程处理器来执行方法步骤。因此,处理器可以是可编程的并且是耦合的以从以下项中接收数据和指令,并且将数据和指令发送至以下项:数据存储系统、至少一个输入设备、以及至少一个输出设备。可以以高级面向过程或面向对象的编程语言来实现应用程序,或者如果愿意的话,可以以汇编语言或机器语言来实现应用程序。无论如何,所述语言可以是编译的或翻译的语言。所述程序可以是完全安装的程序或更新程序。所述程序在系统上的应用在任何情况下都得到用于执行所述方法的指令。

[0034] 所述方法提出了一种改进的数据聚类形成算法,具体应用于查询运行时间预测。但在详述所述方法的算法上的解决方案之前,现在对其上下文(即,应用)进行详述。

[0035] 所述方法用于使数据库中的参考查询形成聚类。执行聚类形成以用于基于目标查询与参考查询的相似度(例如,任何相似度标准,例如现有技术中的那些标准)来(之后)预测在数据库中的目标查询的运行时间(即,用于计算查询计划的时间加上/或用于执行该查询计划的时间,或者甚至在输入到所述方法的目标查询是子查询的情况下(例如,从计算常规查询的查询计划所输出的)用于执行子查询的时间)(即,关于在所考虑的具体数据库上运行所用的时间)。换句话说,所述方法允许基于对(可能是之前的)参考查询的先前的聚类形成而预测给定(可能是将来)的查询(即,目标查询)的运行时间,其中,所执行的聚类形成是基于参考查询的运行时间的。因此,如在本领域中所知的并且如之前所讨论的,参考查询形成训练集。例如,参考查询可以仅仅是运行时间是已知的任何一组查询(例如,在所述方法之前,提供了数据库,提供了一组查询,并且在该数据库上执行该组查询,并且保持对运

行时间的追踪允许确定待在S10被提供的数据)。在所述方法不断地被迭代并且聚类形成不断更新的情况下,一旦已经执行了所输入的目标查询并且其运行时间已知,则之后甚至可以将所输入的目标查询加入到参考查询中。

[0036] 使参考查询形成聚类,并且它们关联于它们的(已知并且真实的)运行时间(即,因此它们在S10以表示持续时间的数值的形式被提供)。在示例中,在方法的最后,针对每个聚类仅保留一个代表性的运行时间,例如,聚类的中心,即,聚类的平均值。接着,当提供了(新的)目标查询时,预测算法可以基于目标查询与参考查询的相似度,将该目标查询与聚类中的一个相关联。在这里可以实现任何基于相似度的关联方案,例如,上文那些结合关于现有技术的讨论所讨论的方案。这本身是已知的。例如,一旦聚类形成S30可用,则可以根据所评估的在目标查询与所有查询之间的、或在每个查询的一个或多个代表性查询(例如,运行时间最接近于聚类的中心的(多个)参考查询)之间的预先确定的查询距离标准,将目标查询与聚类中的一个相关联。该评估导致确定“最接近的”聚类(基于查询距离标准)。接着可以基于这样的信息来预测目标查询的运行时间。

[0037] 接着预测算法的确可以根据与目标查询相关联的聚类的参考查询的运行时间来预测目标查询的运行时间。再一次,如本身已知的,这可以例如根据任何现有技术来执行。例如,该步骤的高效且直接的实现是取相关联的聚类的中心(即,针对目标查询所预测的运行时间是相关联的聚类的参考查询的平均运行时间)。在更复杂的实现中,可以使用支持向量机。所述方法因此对于避免查询时间调度(例如,一旦同时进行的查询的总数低于预先确定的阈值,和/或在已知数据库较少被查询时,例如在晚上,则可将具有高于预先确定的阈值的预测的运行时间的目标查询推迟执行)、执行计划优化中的瓶颈特别有用,并且特别适合于联邦数据库系统。

[0038] 现在,进一步讨论在S10处提供的数值$(x_1,...,x_n)$。

[0039] 这些数值通过表示对数据库数据库的查询/数据库中的查询(参考查询)的运行时间而与数据库相关。因此,数值与预先确定的数据库相关联。当然,数据库可以随着时间略微进化,但接着,所述方法可以定期迭代,以便获得预测方案的更新,如在本领域中所知的。在过渡期,在S10处提供的运行时间可能不准确(因为数据库已经进化了),但它们仍然形成好的近似。在任何情况下,这种更新主题对本领域技术人员来说是显而易见的,并且在本讨论的范围之外。应当注意,本讨论指的是在数据库“中”的查询,意味着所述查询及其运行时间被认为相关于预先确定的数据库(查询的运行时间是执行对所述数据库的查询所用的时间)。

[0040] 现在讨论所述方法所构想的数据库和其中的查询的示例。

[0041] 数据库(DB)是有组织的数据的集合,其中所述数据以这样的方式存储:用户可以利用以明确定义的语言表达的查询来检索信息的具体片段。用户和数据库之间的界面称为数据库管理系统(DBMS)。最著名的DBMS有MySQL、PostgreSQL、Microsoft SQL Server、Oracle、SAP和DB2。

[0042] 数据库的概念非常泛化,并且包含许多种类的功能不同的系统。所述方法的数据库可以是以下的种类中的任何一种:

[0043] • 演绎数据库(例如,在“Logic Programming and Database”,Tanca,1990中所描述的)

[0044] • 分布式数据库(例如,在“Principle of Distributed Database”,Ozsu,Valduriez,2011中所描述的)

[0045] • 联邦数据库系统(例如,在“A Federated Architecture for information management”,McLeod,Heimbigner,1985中所描述的)

[0046] • 图形数据库(例如,在“Survey of graph database models”,Angles,Gutierrez,2008中所描述的)

[0047] • 知识库(例如,在“Report on a knowledge-based software assistant”,Green,Cordell,Luckham,Balzer,Cheatham,Rich,1986中所描述的)

[0048] • 操作数据库(例如,在“Management Information Technology Systems”,O'Brien,Marakas,2008中所描述的)

[0049] • 概率数据库(例如在“Efficient query evaluation on probabilistic databases”,Dalvi,Suciu,2007中所描述的)

[0050] • 时态数据库(例如,在“TSQL2 Temporal Query Language”,Snodgrass,2009中所描述的)

[0051] 所有这些种类的数据库使用不同的算法数据结构以存储它们的信息,然而,这些种类的数据都通过查询的概念联系起来。即使用于表达查询的语言取决于数据的本质,但它们的结构常常共享共同的形状。特别地,联邦数据库系统被设计为使得用户可以在统一的框架下与具有不同本质的协作数据库进行交互。由于所述方法的核心涉及查询及其运行时间,因此就其可以用于任何类型的数据库而言,其是泛化的。

[0052] 所述方法构想的查询可以由用户写入,并且因此常常以说明性语言(例如,SQL)表达,并且可以变形以便被高效地回答。该变形由称为查询优化器的单元执行(并且其可以如在文章“Query optimization”,作者Ioannidis和Yannis,1996中那样被执行),该查询优化器目的是重新表达用户查询使得所述用户查询能够分解成更小的子查询,这些子查询将根据被称为查询计划或执行计划的从属方案被处理。常常将查询计划表示为“平面节点”的树,其中每个节点对应于一个原子操作,并且其中从叶子到根来完成运算,所述根的结果对应于给用户的回答(如在上文所引用的文章“An Overview of Query Optimization in Relational Systems”,作者Chaudhuri和Surajit,1998,或者在文档US5694591或US20060031189中)。

[0053] 所述方法属于使用对查询时间的聚类形成的方法的类别,所述方法的核心涉及标量聚类形成。如在上文中提及的,这些方法与聚类分析高度相关。所述方法产生比当下的其他最佳聚类形成设备更快的最佳聚类形成(关于“总失真”,其是标准目标函数)。实际上,如在下文所示,所述方法的计算时间可以比得上(产生非最佳量化的)最好的启发式的方法(并且常常甚至更快)。

[0054] 所述方法可以遵循以下的管线(也在上文中呈现了):

[0055] • 步骤1:使查询时间形成聚类

[0056] • 步骤2:在每个聚类上学习模型

[0057] • 步骤3:预测查询运行时间

[0058] 所述方法的核心在步骤1上并且提出了一种快速的方法以计算最佳聚类形成方案。由于已经在背景技术部分中提及了步骤2和步骤3,并且取决于在查询上具有的特定的

模型(取决于其表达所用的语言、其涉及的数据库的类型、也取决于具体的基本应用),所以本讨论将主要仅集中于步骤1,并仅对步骤2和步骤3的概念做简要的回顾。

[0059] 现在简要地进一步详述已经在之前讨论过的在每个聚类上学习模型的步骤。这些细节全都是从现有技术可知的。

[0060] 所述方法在聚类上建立的最简单的模型在于认为位于该聚类中的任何查询都将具有等于所述聚类的运行时间的平均值的预测的运行时间(这是在例如文档WO2014060050-"A federated database system"中所采取的选择)。

[0061] 更复杂的技术使用基于查询的表示的机器学习算法(所述表示在于可以被视为查询的内部统计的一系列的特征中,参见文章"Predicting Multiple Metrics for Queries:Better Decisions Enabled by Machine Learning",作者Ganapathi等,2009)。例如,上文所引用的Hasan和Gandon的研究报告"Predicting SPARQL Query Execution Time and Suggesting SPARQL Queries Based on Query History"使用支持向量机以学习每个聚类中的预测子。

[0062] 现在简要地进一步详述已经在之前讨论的预测查询运行时间的步骤。这些细节全都是从现有技术可知的。

[0063] 当给定了新的查询,经典的策略在于找到所述查询的"最相似的聚类",并且接着简单地应用在步骤2中所学习到的模型。取决于聚类的数量和每个聚类的大小的数量,目标查询与特定的聚类中的一些之间执行的对比数可以不同。一种方法是如在文档US1543036A中提出的,利用如之前所讨论的特定的相似度的概念,仅将目标查询与每个聚类的质心进行比较。

[0064] 到这里,已经提供了所述方法的上下文及其许多示例,但是还没有提供关于所述方法的核心的细节(即,定义之后的聚类形成S30的计算S20步骤)。这在下文中完成,应当注意,在下文中提供的所有实现示例可以应用在在上文中提供的示例应用的任何一个中。

[0065] 如在前文中指出的,所述方法包括针对聚类的预先确定的数量K来计算S20数值的最佳K均值聚类形成。但是这没有被粗暴地完成。的确,计算步骤S20包括将线性时间行最小值搜索算法迭代对应于K的次数的数(在之后所讨论的示例中K-1次)。由于其对行最小值搜索算法的公知的类别的任何预先确定的线性时间算法的使用,计算步骤S20具有低复杂度。所述方法因此实现了K均值聚类形成问题的新的并且在算法上高效的解决方案。

[0066] 在提供更多关于计算步骤S20的细节之前,现在讨论标量量化。实际上,所述方法的聚类形成S20-S30在示例中可以遵循标量量化的框架,这是因为其可以通过虚拟地"量化"这些运行时间并且定义与虚拟量化值相关联的聚类,来关于运行时间而使参考查询形成聚类。实际上,聚类的参考查询关联于等于所述聚类的中心(即,所述查询的参考查询的运行时间的平均值)的虚拟的运行时间。参考查询的运行时间因此以这种方式被量化。

[0067] 如已知的,标量量化是利用有限集合 $V = \{c_1, \dots, c_K\} \subset \mathbb{R}$ 来近似实数的计算工具,其中被称为数位步进的V的元素是用作近似的值。标量量化器在数学上被定义为映射:

[0068] $$q: \mathbb{R} \longrightarrow V = \{c_1, \dots, c_K\},$$

[0069] 使得x和q(x)之间的距离是小的,所述距离是任何预先确定的距离(其中,距离的概念可以取决于上下文),例如欧几里得距离。

[0070] 在实践中，标量量化器总是通过将 $\mathbb{R}$ 划分成间隔$I_1=]-\infty,a_1[,I_2=[a_1,a_2[,\ldots,I_K=a_{K-1},\infty[$(其中$a_1<\cdots<a_{K-1}$)来定义的，并且接着对于每个 $x\in\mathbb{R}$，表示I_i为唯一的间隔，使得$x\in I_i$，我们关联值$q(x)=c_i$。实数$a_1,\ldots,a_{K-1}$被称作“决策边界”。Gray和Neuhoff的文章“Quantization”提供了量化的完整的概况。

[0071] 如公知的和例如在MacQueen的最广泛使用的文章“Some Methods for classification and Analysis of Multivariate Observation”中所定义的，所述方法集中于K均值设定上。在该设定中，对于按照增加的顺序排序的给定的元组 $(x_1,\ldots,x_n)\in\mathbb{R}^n$ 以及对于给定的整数K(通常$K=2^b$，其中b是可用于对每个x_i进行编码的比特数)所述方法在S20处利用达到最小总失真的K个数位步进而找出量化器q，其中总失真被定义为：

[0072] $$TD(q)=\sum_{1\le i\le n}\left(x_i-q(x_i)\right)^2$$

[0073] 显然，为了最小化该量，所述方法可以仅处理将每个实数值x映射至其最接近的数位步进的量化器。因此，所述问题正好等同于找出将总失真最小化的K个中心$c_1,\ldots,c_K$：

[0074] $$TD(\{c_1,\ldots,c_K\})=\sum_{1\le i\le n}\min_{1\le k\le K}(x_i-c_k)^2$$

[0075] 图3示出了针对10个值和4个数位步进的示例。最佳量化由集合$\{c_1,\ldots,c_K\}$给出，使得：$\forall c_1',\ldots,c_k',TD(\{c_1',\ldots,c_k'\})\geq TD(\{c_1,\ldots c_K\})$。

[0076] 最小化该总失真必须被理解为，在量化步骤(对于给定的K)期间损失尽可能少的信息。应当注意的是，每个点x_i被隐含地分配至其最近的中心，因此所述方法可以将分区构建成多个聚类，其中每个聚类对应于分配至给定的中心的点的集合(找到最好的量化器因此是聚类形成问题，如由Lloyd在“Least square quantization in PCM”中所说明的)。对于每个$k\in\{1,\ldots,K\}$，让我们用C_k表示对应于中心c_k的聚类。显而易见的是，每个中心实际上是其相对应的聚类的点的平均值。此外还应当注意的是，由于假设$x_1<\cdots<x_n$，因此每个聚类存在于点的连续的子集中。例如，如果我们有我们想划分成K=4个聚类的47个实数，则可能的最佳聚类形成可以是：

[0077] $$(\underbrace{[x_1,\ldots,x_{17}]}_{C_1},\underbrace{[x_{18},\ldots,x_{24}]}_{C_2},\underbrace{[x_{25},\ldots,x_{42}]}_{C_3},\underbrace{[x_{43},\ldots,x_{47}]}_{C_4})$$

[0078] 对于所有$1\leq a\leq b\leq n$，我们引入了记法 $mean(a,b)=\frac{1}{b-a+1}\sum_{\{a\le i\le b\}}x_i$ 并且我们也表示$disto(a,b)=\sum_{\{a\le i\le b\}}(x_i-mean(a,b))^2$。可以将之前的示例的相对应的总失真写为：

[0079] TD=disto(1,17)+disto(18,24)+disto(25,42)+disto(43,47)

[0080] 如之前所提及，已经存在针对该问题的解决方案，但是这些解决方案比所述方法慢，并且对于其中的大多数来说只是启发式的(即，不产生最佳量化)。

[0081] 在所述方法的示例实现中，数值$(x_1,\ldots,x_n)$并排序并相应地被编制了索引。在计算步骤S20内的迭代包括，在每个相应的迭代等级k，并且对于低于(inferior to)数值的数量n的每个相应的索引j，对最小总失真(记作$TD_{min}(j,k)$)的计算对于所编制的索引小于j的数值x_i(因此$i\leq j$)的子集是可实现的，其中根据应用至方块矩阵H的线性时间行最小值搜

索算法,聚类的数量k对应于相应的迭代等级(因此,是k)。

[0082] 在该示例中,在每个相应的迭代等级k,并且对于低于数值的数量n的每个相应的索引j,对于每个行索引i和每个列索引j,矩阵条目H(i,j)可以对应于以下项的总和:

[0083] • 最小总失真($TD_{min}(i-1,k-1)$),其是在针对该个行索引之前的索引(i-1)的先前的迭代中计算的,以及

[0084] • 在该个行索引和该个列索引之间的数值的连续的子集($x_i,...,x_j$)的失真(disto(i,j))。

[0085] 该实现提供了一种胜过现有的聚类形成方法的系统,由于其也产生最佳K均值聚类形成但运行地更快,具体而言,在时间O(K*n)内。应当注意的是,对于典型的使用,该示例的方法比"好的"启发式方法的运行速度快10倍以上。

[0086] 现在讨论示例的聚类形成算法的更完整的概述。

[0087] 为了计算最佳分区,所述方法使用动态编程范式(如在Bellman的文章"The theory of dynamic programming"中所描述的)。特别地,示例的所述方法针对每个$j\in\{1,...,n\}$和每个$k\in\{1,...,K\}$来计算值$TD_{min}(j,k)$,其被定义为在如果我们只考虑前j个点($x_1,...,x_j$),则我们可以利用至多k个聚类而达到的最小总失真。

[0088] 通过定义,由于将点的集合分成一个聚类的唯一方式是取其所有,因此对于所有$j\in\{1,...,K\}$,我们有:$TD_{min}(j,1)=disto(1,j)$。此外,我们有以下的方程,对于所有$k\in\{2,...,K\}$和所有$j\in\{1,...,n\}$:

[0089] $$TD_{min}(j,k)=\min_{1\le i\le j}\{TD_{min}(i-1,k-1)+disto(i,j)\}$$

[0090] 该方程表达了这样的事实:对于($x_1,...,x_j$)我们可以利用至多k个聚类而达到的最小总失真在于,对于某个i,利用最多k-1个聚类并且[$x_1,...,x_j$]作为最后的聚类而对前i-1个点的最佳聚类形成。之前的方程是所述方法的核心。应当注意,如果对于给定的$k\in\{2,...,K\}$,已经针对所有j计算了值$TD_{min}(j,k-1)$,那么所述方法可以针对所有j,通过测试之前的方程中的所有可能的$i\in\{1,...,j\}$来计算值$TD_{min}(j,k)$。然而,该假想的技术将会得出一种非常慢的算法。

[0091] 为了克服这点,所述方法针对特定的矩阵中的行最小值搜索使用特定类别的算法。

[0092] 现在讨论示例的所述方法依赖的行最小值搜索和完全单调性的概念。

[0093] 行最小值搜索算法(如在Bradford和Reinert的文章"Lower Bounds for Row Minima Searching",1996中所描述的)是这样的一种算法:将函数$f:[\![1,R]\!]\times[\![1,C]\!]\longrightarrow\mathbb{R}$作为输入,使得对于所有$1\le i\le R$,$1\le j\le C$,值f(i,j)可以在恒定时间内进行计算,并且输出整数向量$p=(p_1,...,p_R)$,使得:

[0094] $$\forall 1\le i\le R, p_i=argmin_{1\le j\le C}f(i,j)$$

[0095] 在下文中,我们用F表示矩阵$F=(f(i,j))_{i,j}$。应当注意,出于完备性的原因,如果矩阵F没有特定的性质,那么可以请求其所有元素以便计算向量p。然而,在特定的关于F的条件下,可以实现极其快速的算法。

[0096] 如果矩阵F满足以下条件,则其可以被称为完全单调的:"如果对于i,j,k,其中i<j,则我们有H(k,i)<H(k,j),接着对于所有k'≤k,我们也有F(k',i)<F(k',j)。"

[0097] 在完全单调矩阵中存在针对行最小值搜索的线性时间算法(如在Alon和Azar的文章“Comparison-Sorting and Selecting in Totally Monotone Matrices”中所说明的)。可以通过所述方法在S20处针对矩阵H实现这些预先确定的算法(即,线性时间行最小值搜索算法)中的任何一个。特别地,发明者已经利用公知的呈现在Alon和Azar的论文中的SMAWK算法测试了所述方法具有极快速的收敛(相对于现有技术)。

[0098] 现在将讨论允许所述方法极快速地运行的基本性质。在这之前,应当注意的是,该性质的识别使得K均值聚类形成问题和针对行最小值搜索提供的公知且强大的算法之间建立了桥梁,并且对K均值聚类的研究的漫长的历史上从未确立过这样的桥梁。

[0099] 定理:

[0100] 对于所有1≤i<j<n,我们有:

[0101] disto(i,j)+disto(i+1,j+1)≤disto(i,j+1)+disto(i+1,j)

[0102] 证明:

[0103] 首先,注意对于1≤a≤b≤n,由定义,disto(a,b)等于$(x_a,...,x_b)$的方差乘以(b-a+1)。

[0104] 因此,我们可以从König-Huygens 方程得到:

[0105] $$disto(a,b)=\sum_{a\le l\le b}x_l^2-\frac{1}{b-a+1}$$

[0106] 让我们考虑i和j使得1≤i<j<n。

[0107] 表示p=(b-a+1),$S=\sum_{i\le l\le j}x_l$,$\alpha=x_{j+1}$并且$\beta=x_i$,我们从之前的等式中可以得到:

[0108] $$disto(i,j)=\sum_{i\le l\le j}x_l^2-\frac{S^2}{p}$$

[0109] $$disto(i,j+1)=\sum_{i\le l\le j+1}x_l^2-\frac{(S+\alpha)^2}{p+1}$$

[0110] 并且因此:$disto(i,j+1)-disto(i,j)=\alpha^2+\frac{S^2}{p}-\frac{(S+\alpha)^2}{p+1}$.(1)

[0111] 我们另外有:

[0112] $$disto(i+1,j)=\sum_{i+1\le l\le j}x_l^2-\frac{(S-\beta)^2}{p-1}$$

[0113] $$disto(i+1,j+1)=\sum_{i+1\le l\le j+1}x_l^2-\frac{(S-\beta+\alpha)^2}{p}$$

[0114] 并且因此:$disto(i+1,j)-disto(i+1,j+1)=-\alpha^2-\frac{(S-\beta)^2}{p-1}+\frac{(S-\beta+\alpha)^2}{p}$.(2)

[0115] 让我们表示Δ=disto(i,j+1)-disto(i,j)+disto(i+1,j)-disto(i+1,j+1)

[0116] 接着我们想证明的定理简单地等价于Δ≥0。

[0117] 此外,将等式(1)和(2)相加,我们可以得到:

[0118] $$\Delta=\frac{S^2}{p}-\frac{(S+\alpha)^2}{p+1}-\frac{(S-\beta)^2}{p-1}+\frac{(S-\beta+\alpha)^2}{p}$$

[0119] 我们的目标现在是使用该表达式来证明Δ≥0。

[0120] 在不损失普遍性的情况下，由于所述问题是平移不变的（其对应于将所有点平移$-\frac{s}{p}$），我们能够假设S＝0，使我们有：

[0121] $$\Delta = -\frac{\beta^2}{p-1} - \frac{\alpha^2}{p} + \frac{(\alpha-\beta)^2}{p} = \frac{1}{p(p-1)(p+1)}\Delta'$$

[0122] 其中

[0123] $\Delta' = -p(p+1)\beta^2 - p(p-1)\alpha^2 + (p-1)(p+1)(\alpha-\beta)^2$

[0124] 式子整理后，我们能够写成：

[0125] $\Delta' = -(p+1)\beta^2 + (p-1)\alpha^2 - 2(p+1)(p-1)\alpha\beta$

[0126] 现在应注意的是，由于对于所有的$l \in \{i+1,\ldots,j\}$，有$\alpha \geq x_l$，所以有$S = x_i + \cdots + x_j = \beta + \cdots + \alpha \leq \beta + (p-1)\alpha$（记住$x_1 < \cdots < x_n$）。由于我们假设S＝0，因此我们得到：

[0127] $(p-1)\alpha \geq -\beta$

[0128] 此外，由于总和S的较小的项等于零，因此我们明确得到$\beta \leq 0$，于是可得

[0129] $-(p-1)\alpha\beta \geq \beta^2$

[0130] 将该不等式重新带入Δ的最后的表达式，我们得到：

[0131] $\Delta' \geq -(p+1)\beta^2 + (p-1)\alpha^2 + 2(p+1)\beta^2$

[0132] $\geq (p-1)\alpha^2 + (p+1)\beta^2$

[0133] 所以，我们有Δ’≥0，并且因此Δ≥0，证明结束。

[0134] 现在，对于固定的$k \in \{2,\ldots,K\}$，假设所述方法已经针对所有j计算了所有的$TD_{min}(j,k-1)$。让我们回忆，可以通过以下关系式来针对所有j检索$(TD_{min}(j,k))_j$：

[0135] $$TD_{min}(j,k) = \min_{1 \leq i \leq j} \{TD_{min}(i-1,k-1) + disto(i,j)\}$$

[0136] 现在，我们将看到上述的性质将如何帮助所述方法在时间O(n)内从$TD_{min}(j,k-1)$计算所有$(TD_{min}(j,k))_j$。

[0137] 首先，让我们表示$H(i,j) = TD_{min}(i-1,k-1) + disto(i,j)$。

[0138] 由于$disto(i,j) + disto(i+1,j+1) \leq disto(i,j+1) + disto(i+1,j)$，因此通过在两边都加上$TD_{min}(i-1,k-1) + TD_{min}(i,k-1)$，我们可以得到：

[0139] $H(i,j) + H(i+1,j+1) \leq H(i,j+1) + H(i+1,j)$

[0140] 该性质被称为矩阵$H = (H(i,j))_{i,j}$的Monge的性质（参见Cechlárová和Szabó的文章“On the Monge property of matrices”）（实际上，当j<i时，所述方法可能抛弃了H(i,j)的定义，但是在实践中这样的缺失值不是真正的问题，并且这将不会进一步被讨论）。在一些文献中，其也被称为Knuth-Yao四边形不等式（参见例如Bein、Golin、Larmore和Zhang的文章“The Knuth-Yao quadrangle-inequality speedup is a consequence of total-monotonicity”）。

[0141] 依据定理，矩阵H是完全单调的，即：如果对于i,j,k，其中i<j，则我们有H(k,i)<H(k,j)，接着对于所有k’≤k，我们也有H(k’,i)<H(k’,j)。这实际上是Monge的矩阵的公知性质并且不需要被证明。

[0142] 现在，注意计算$(TD_{min}(j,k))_j$等价于计算矩阵H的每一行的最小值。这里，示例的方法调用任何预先确定的线性时间行最小值搜索算法(例如，SMAWK算法)，其恰好是针对在时间O(n)内解决的该子问题而精确地设计的。应当注意，矩阵H为n×n的大小，但是所述方法不需要完全对其进行构建。所述方法可以仅向例如SMAWK子例程提供方法以计算恒定时间内的任何H条目。

[0143] 因此在实践中，所述方法的算法可以首先计算第一层$(TD_{min}(j,0))_j$，然后其将利用行最小值搜索(RMS)子程序计算第二层$(TD_{min}(j,1))_j$，然后利用第二次RMS算法计算第三层$(TD_{min}(j,2))_j$，以此类推，直到所述方法得到所有$(TD_{min}(j,k))_{j,k}$为止。由于K层中的每层都消耗时间O(n)来进行计算，因此整个算法在时间O(Kn)内运行。

[0144] 此时，在示例中，所述方法还可以包括，在每个相应的迭代等级k，将由行最小值搜索算法返回的索引存储在例如专用矩阵Cut_{min}中。该示例的所述方法还可以包括，在计算步骤S20处，从所存储的索引确定最佳聚类形成。

[0145] 在简单且直接的实现中，从存储的索引确定最佳聚类形成包括在矩阵Cut_{min}内工作。具体而言，示例的所述方法从最后被索引的数值$(Cut_{min}(n,K))$开始迭代地将数值进行划分。在每个相应的迭代等级q，当前形成的聚类的起始数值的索引等于在等级K-q的迭代中的(在计算步骤S20内进行迭代期间的)所存储的索引。

[0146] 换句话说，如果注意到，每次所述方法计算出最小值，则所述方法显然也可以得到达到该最小值的索引。更加准确地说，将$(TD_{min}(j,k))_{j,k}$的每个值计算为最小值，所述最小值的索引可以被存储在矩阵$(Cut_{min}(n,K))_{j,k}$中。从所述方法能够仅通过查看表格Cut_{min}而容易地得到最佳分区。

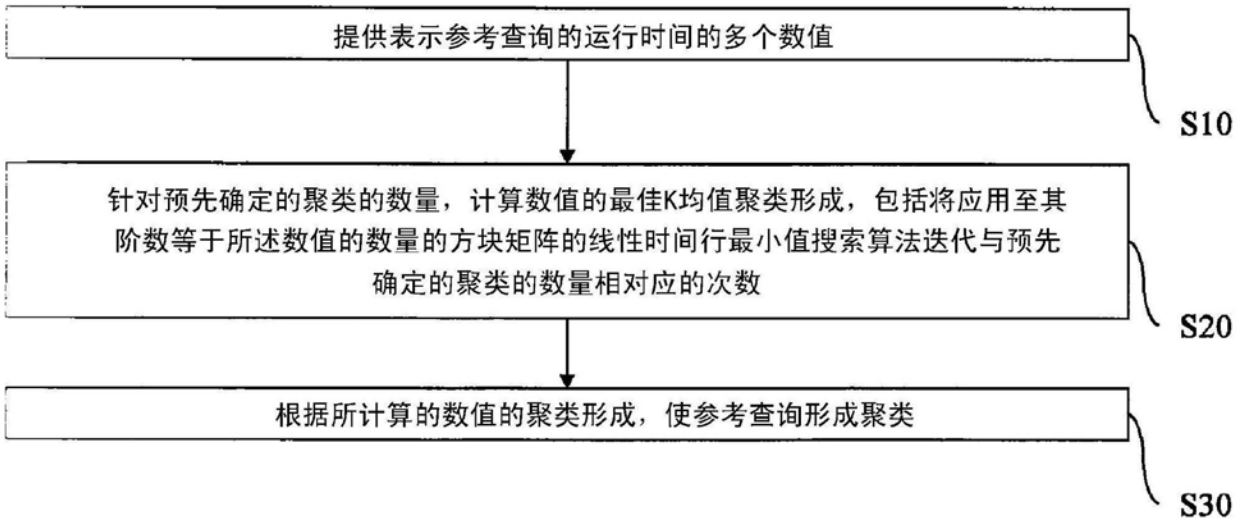

提供表示参考查询的运行时间的多个数值
S10
针对预先确定的聚类的数量，计算数值的最佳K均值聚类形成，包括将应用至其阶数等于所述数值的数量的方块矩阵的线性时间行最小值搜索算法迭代与预先确定的聚类的数量相对应的次数
S20
根据所计算的数值的聚类形成，使参考查询形成聚类
S30

图1

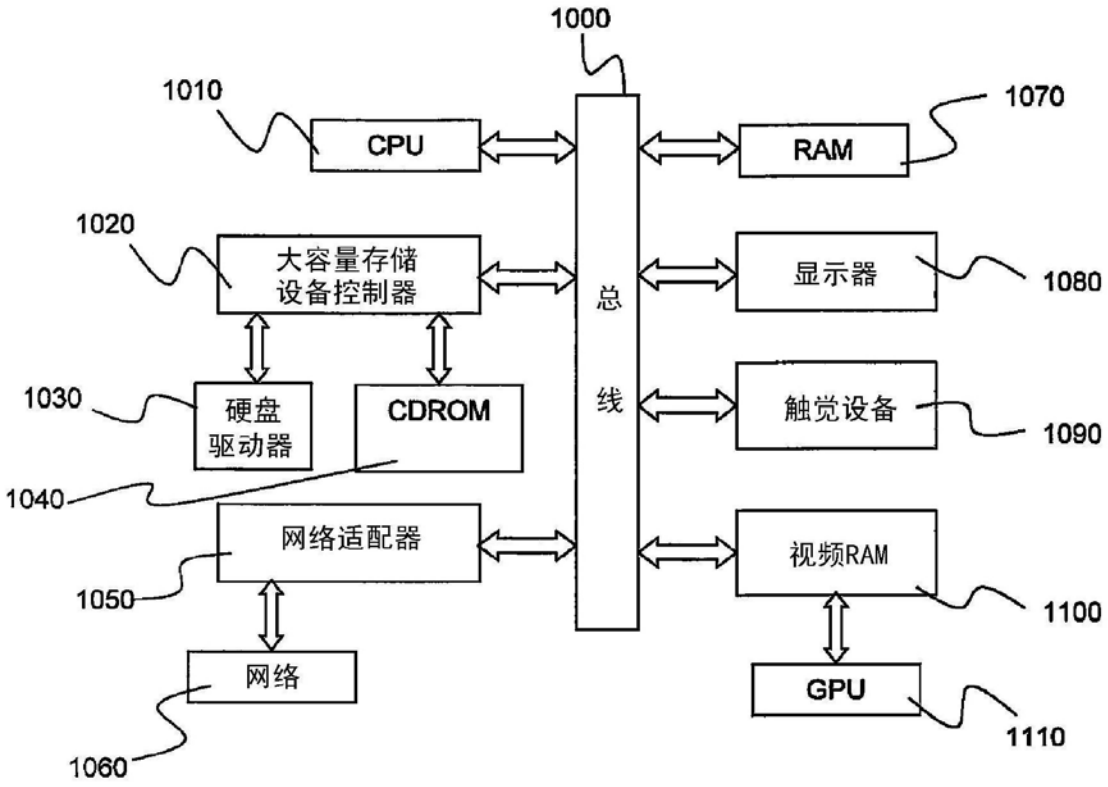

1000
1010
CPU
1070
RAM
1020
大容量存储设备控制器
显示器
1080
总线
1030
硬盘驱动器
CDROM
触觉设备
1090
1040
网络适配器
视频RAM
1050
1100
网络
GPU
1060
1110

图2

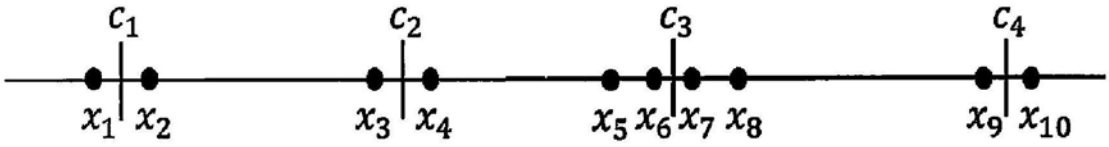


图3