维普资讯 http://www.cqvip.com

一种支持数据流条件过滤的批处理策略*)

杨兴华[1] 宋宝燕[2] 欧征宇[1] 苏 东[1] 于亚新[1] 于 戈[1]

(东北大学信息科学与工程学院 沈阳110004)[1] (辽宁大学信息科学与技术学院 沈阳110036)[2]

摘 要 介绍了一种支持数据流条件过滤的批处理策略。该策略采用红黑树对查询建立索引,通过把查询条件分解为单独的布尔因子的方法,将每个布尔因子加到红黑树的相应节点上,其中相同的布尔因子加到红黑树的同一个节点上。在查询处理过程中,每处理一个索引节点,就将所有的布尔因子同时处理。因此,使用这种批处理策略能够实现高效查询。

关键词 数据流,条件过滤,红黑树,布尔因子,批处理策略

A Batch Strategy Supporting Condition Filtration on Data Streams

YANG Xing-Hua[1] SONG Bao-Yan[2] OU Zheng-Yu[1] SU Dong[1] YU Ya-Xin[1] YU Ge[1]

(School of Information Science and Engineering, Northeastern University, Shenyang 110004)[1]

(School of Information Science and Technology, Liaoning University, Shenyang 110036)[2]

Abstract A strategy supporting condition filtration on data streams is introduced. The strategy uses red-black tree indices to index queries. Query conditions are disassembled into single boolean factors, and the boolean factors are added to the red-black trees indices. The same boolean factors are added to the same node in the trees. So during the query processing, when one node is processed, all the boolean factors are efficiently processed together using the batch strategy. In this paper, the basic ideas will be introduced first, and then some extended ideas will also be introduced.

Keywords Data stream, Condition filtration, Red-black trees, Boolean factors, Batch strategy

1 引言

最近,数据流的应用变得越来越广泛。例如一些具体的例子:在传感器网络中由传感器得到的各类数据,电信通话的记录,网络安全中的人侵检测,股票金融信息,网络日志等等都是数据流的应用。在这些数据流的应用中,一个显著的特点是数据不再被看作是一个固定的数据集,而是被看作是动态到达且其速度无法控制的流的形式,这与传统的数据库应用有着本质的区别。通常情况下,数据流应用中的数据是大量的、连续的,数据到达的速度是随时间变化的,而且数据一旦流过就无法重现。数据流上的查询是连续查询。传统的数据库技术已经不能满足数据流应用中的种种需求,因此需要研究新的数据查询处理技术以适用于数据流的应用。

现在国内外有许多机构、学者都在致力于数据流的研究工作[1],开发了一些数据流管理系统的模型。TelegraphCQ 系统[2]是 UC Berkeley 大学研究开发的系统,基于适应性查询处理的机制,实现了一个适应性查询处理引擎。应用于数据量大的传感器网络,在复杂多变的网络环境中,快速高效地处理查询。Stream 系统[3]是由 Stanford 大学开发的系统,目标是开发一个较通用的数据流管理系统用于实际应用。Aurora 系统[4]是由 Brown、MIT 和 Brandeis 大学联合开发的一个系统,其核心包括大量的网络触发器,主要是针对数据流上的监督应用。香港科学技术大学的 Mouratidis 对当前数据流的研究工作概况作了综述性的介绍[5]。

Avnur 和 Hellerstein 介绍了一种适应性的查询处理机制"eddy"[6],这种机制能够在一个查询方案执行过程中连续地重组算子。Madden 等人介绍了一个支持连续查询的系统[7],这个系统使用了"eddy"的机制。Chandrasekaran 等人介绍了一个查询处理器 Psoup[8]。Psoup 是基于 UC Berkeley 大学的 Telegraph 查询处理框架[9]开发的,采用了文[7]中介绍的处理连续查询的思想。它不仅将数据看作是流,而且将查询也看作流来对待,允许对历史数据的访问。为了支持范围查询,需要使用一种基于树的索引。考虑到红黑树的效率较高且维护代价较小,Psoup 采用红黑树对查询和数据建立索引,并实现了包含合取关系的条件查询,但并未实现析取条件查询。而在实际查询过程中,很多查询条件既包含合取关系也包含析取关系,因此如何有效实现析取关系的条件查询也很重要。

我们目前开发了一个支持嵌入式实时应用的数据流原型系统 RealStream,该系统仍采用红黑树作为索引,但对查询条件进一步扩展,使红黑树不仅支持包含合取关系的条件查询,而且也支持析取条件查询。

本文第2节介绍了相关工作;第3节阐述了批处理的查询思想并对红黑树索引的性能进行了定性分析;最后总结了全文。

2 相关工作

红黑树由 Bayer 在《Symmetric binary B-trees: Data structures and maintenance algorithms》一书中首先介绍,当时名为"对称二叉 B 树"。当前的红黑树术语则由 Guibas 和 Sedgewick 在1978 年提出。另外,Cormen 等人较详细地阐述

*)本课题得到国家"八六三"高技术计划 CIMS 主题(编号:2002AA1Z2308,2002AA118030),辽宁省自然科学基金(编号:20022027),教育部优秀青年教师科研教育奖励计划资助。

维普资讯 http://www.cqvip.com

了红黑树的特点以及相关算法[10]。

由于红黑树支持范围查询，同时红黑树的查询效率较高且维护代价较小，因此本文选择红黑树对查询建立索引。查询条件分解为单独的布尔因子，每个布尔因子形式上可以表示为 *S.attr RELOP c*，其中，*S* 是数据流的名称，*attr* 是这个流上的一个属性，*RELOP* 是关系算符，而 *c* 是一个具体的值。在索引建立的过程中，根据每个布尔因子的条件值 *c* 来生成树上的节点。在红黑树上节点间的值是不重复的，所以条件值 *c* 相同的布尔因子被加入到同一个节点上。然后在处理的过程中可以对它们进行批处理，提高查询效率。

下面，简要介绍与黑红树有关的一些内容。

2.1 红黑树的定义

1）一棵二叉搜索树，树上的每一个节点都有一种颜色。平衡条件通过约束节点的排列方式（基于其颜色）得以维持。

2）满足如下的几个条件：

① 每个节点的颜色或者为红色，或者为黑色；

② 每一个叶子节点（空节点）的颜色为黑色；

③ 如果一个节点颜色为红，则它的两个子节点颜色全为黑；

④ 从一个节点到叶子节点的各条路径上包括的黑节点的数目相同。

2.2 红黑树的性质

定义节点的黑高度（black-height）：是在从一个节点向下到其一个叶子节点之间路径上的黑色节点的数目，不包括这个节点本身。

带有 n 个节点的红黑树的高度在 $\log_2(n+1)$ 和 $2\log_2(n+1)$ 之间。

在一棵红黑树中，从根节点到叶子节点的任意一条路径上至少有一半的节点的颜色是黑色的。

3 批处理查询

在 RealStream 系统中，使用一种称为状态模块（State Modules）的数据结构来存储系统中的查询和数据元组。在系统中只有一个查询状态模块存储所有注册的查询，而系统中每一个流都对应一个数据状态模块。当系统中新注册一个查询后，这个查询首先被加入到查询状态模块中，然后再搜索数据状态模块。同样，系统中新到达一个数据元组，这个元组首先被加入到相应的数据状态模块中，然后再搜索查询状态模块。在搜索的过程中，如果有数据元组满足查询的情况，则需要记录数据元组满足查询的相关信息。

系统中注册的查询格式如下：

SELECT * FROM Stream S

WHERE ($S.attr_1 < C_1$ [$\wedge S.attr_2 > C_2$])

一个查询的查询条件中可以有多个布尔因子，每个查询条件中的各个布尔因子间现在限定为合取的关系。各个查询之间的查询条件中的布尔因子可能有重复的部分，如果单独地测试每一个布尔因子显然是低效率的，因此在 RealStream 系统中采用了一种对查询建立红黑树索引的方法。

在 Realstream 系统中采用同样的方法对数据元组建立红黑树索引。具体的建立方法以及查询处理过程在下面一节中详细介绍。

3.1 合取关系查询

上面介绍了 RealStream 的查询处理主要思想，现在对其查询处理过程详细介绍。

查询状态模块用来存储和索引查询，整个系统中只有一个查询状态模块。在 RealStream 中对于注册的查询中涉及到的每一个属性建立一个红黑树的索引。如前面所叙述的那样，每个查询中的布尔因子的一般形式为 *S.attr* RELOP *C*，其中 *S* 是流的名称，*attr* 是这个流上的一个属性，RELOP 是关系算符，而 *C* 是一个具体的值。索引是根据查询条件中的值 *C* 建立起来的，*C* 作为节点的键值。条件值 *C* 相同的布尔因子被加入到索引树的同一个节点上，因此每一个节点可以对应多个布尔因子。另外还需要扩展红黑树的节点结构来加入查询的信息。所以在每个节点上加入5个阵列，用来存储该节点对应的布尔因子所在所属查询的查询号，每一个阵列对应一个关系算符（<，<=，=，>=，>），加入的阵列称为关系算符阵列。

下面举例说明。例如：系统有许多个查询如下：

q1 ：S.attr>23 AND S.attr<212

q2 ：S.attr<=107

……

……

q15：S.attr>23 AND S.attr<107

q16：S.attr<=212

……

……

则建立索引，在图1中给出了查询索引的图示。

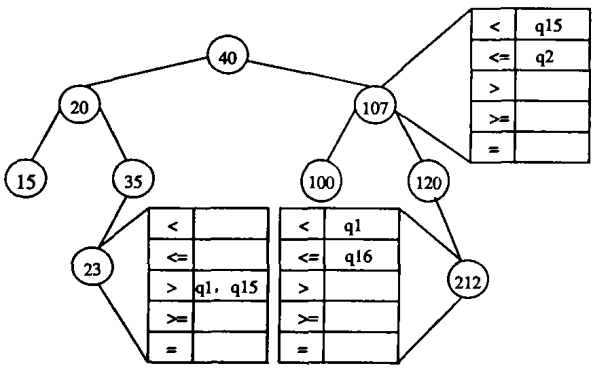

图1 查询索引图示

查询处理的过程为：当有一个新的数据元组到达后，先被加入到相应的数据状态模块中，然后用它的各个属性去搜索相应的查询索引。由于查询是被分解成为若干个独立的查询条件来建立索引树的，因此需要一种技术来综合得到最后的结果，方法如下：

在查询状态模块中，包含一个标志数组，数组中的每一个元素对应于系统中的一个查询。在一个数据元组开始搜索查询索引之前，将数组中的每一个元素的值赋值为其对应查询的查询条件中布尔因子的个数。在元组搜索查询索引的过程中，将元组的属性值与树上每个节点的值进行比较，若数据元组能够满足这个节点所对应的布尔因子，则依据节点的阵列中的查询号将该查询对应的数组元素的值减1。如果某个数组元素的值为零时，则说明元组已经满足了这个数组元素所对应的查询的所有布尔因子，即这个数据元组是满足这个查询的。

这里值得注意的是，当对查询建立索引的时候，不同查询之间的布尔因子如果有相同的 *C* 值，那么它们将被加入到索引树的同一个节点上，在这个节点的某一个关系算符阵列中顺序的加入了不同查询的查询号。而在数据元组搜索的过程

维普资讯 http://www.cqvip.com

中，可以根据关系算符阵列中的查询号一次将相应的查询对应于该节点的布尔因子全部处理。

同样，数据状态模块是用来存储和索引数据元组的。每一个流都对应一个数据状态模块，对流上每个属性都建立一个数据索引。当一个查询搜索数据状态模块时，查询被分解为单个的布尔因子，然后用每个布尔因子到相应的数据索引上去搜索。最后再综合得到结果，综合的方法和上面的查询状态模块的方法是相类似的。

3.2 析取关系查询

在上面的阐述中，一个查询的查询条件中的各个布尔因子间都限定为是合取的关系，现在对其进行扩展以允许布尔因子间有析取的关系。

首先，对红黑树索引的节点有一点扩展，就是在节点中再加入一个关系算符阵列，这个阵列代表关系算符!=，这样使得在注册的查询中允许出现形式为 $S.attr!=C$ 的布尔因子，即每个红黑树节点现在有6个关系算符阵列。

注册查询一般形式如下：

SELECT * FROM Stream S

WHERE ($S.attr_1 < C_1$ [∧/∨] [$S.attr_2 > C_2$])

在这种情况下，将每个查询的都分解为单个布尔因子，并将所有的布尔因子放在一起编号。仍然是采用前面介绍的办法来建立索引，但是在每个节点上的6个阵列中存储的并不是该节点所对应的布尔因子所属查询的查询号，而是该节点对应的布尔因子的编号。搜索的策略也没有改变，但是在查询处理过程中则需要改变处理的策略。现在采用的方法是对每个查询建立查询条件树，将每个单独的布尔因子的编号作为叶子节点，把逻辑算符(∧和∨)作为中间节点建立一棵查询条件树。

例如查询条件 R.a>10∧(R.a<=15∨R.b>3)建立的查询条件树如图2和表1所示。

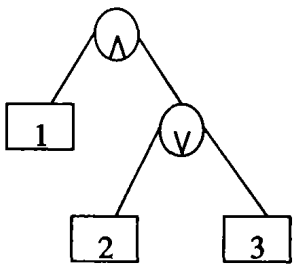

表1 查询条件

布尔条假设编	
R.a>10	
R.a<=15	
R.b>3	

图2 查询条件树

查询处理的过程也有所改变。原来的情况是在查询状态模块中包含一个标志数组，数组中每一个元素对应一个查询。现在查询条件中加入了析取条件后，原来的办法无效了。因此现在采取这样的办法，在查询状态模块中包含一个布尔数组(只用来存储每个布尔因子的真假值)，这个布尔数组中的每一个元素与查询条件中的一个布尔因子相对应。在一个数据元组开始搜索查询索引之前，将数组中的每一个元素赋值为假。在元组搜索查询索引的过程中，将元组的属性值与树上每个节点的值进行比较，若数据元组能够满足这个节点所对应的布尔因子，则依据其阵列中布尔因子的编号将其所对应的布尔数组中的元素赋值为真。在一个数据元组完成搜索后，这个元组对于每个布尔因子是否满足都可以通过布尔因子所对应的布尔数组中的元素来判断。然后，检查每个查询的查询条件树。由查询条件树叶子节点中的编号找到这个编号对应的布尔数组中的元素的值为真还是为假，最后在查询条件树的根节点返回一个值，这个值可以用来判断数据元组是否满足该查询，若这个值为真，说明元组是满足这个查询的；反之，说明元组不满足这个查询。

数据状态模块存储和索引方法与前面部分所述的方法相同，处理的方法与扩展的处理方法类似。

3.3 时间复杂度分析

假设在系统维护的查询中总共有 n 个布尔因子，则依次比较每一个查询的布尔因子的时间复杂度是 $O(n)$。

在建立索引树时，是将每个布尔因子加入到树上的节点中，每个布尔因子对应一个树上的节点。如果有重复的布尔因子，则重复的布尔因子被加入到同一个节点上，所以树上节点的个数 N 与布尔因子的个数 n 的关系一定是 $N<=n$ 的。在索引树上查询的时间复杂度就是树的查找时间复杂度 $O(\log_2 N)$，这个时间复杂度是小于时间复杂度 $O(n)$ 的。

结束语 本文介绍了一种查询处理方法。先给出了有限定条件的查询处理思想，限定查询条件中只允许有合取条件。然后介绍了扩展查询条件的方法，使得查询条件中允许有析取条件。在系统中使用红黑树对查询建立索引，利用红黑树建立索引的方法优点在于将查询中的重复的布尔因子合并在一起，处理过程中每个布尔因子不必单独地进行判断而是进行批处理。但是这种方法存在局限性，要求查询间的布尔因子有重复的部分，重复的布尔因子越多效果越好。如果查询间相互独立，布尔条件重复很少或几乎没有，则效果就不显著。

参 考 文 献

1 Babcock B, Babu S, Datar M, et al. Models and Issues in Data Stream Systems. In SIGMOD POS, 2002

2 Chandrasekaran S, Cooper O, Deshpande A, et al. TelegraphCQ: Continuous DataFlow Processing for an Uncertain World. In CIDR, 2003

3 Motwani R, Widom J, Arasu A, Babcock B. Query Processing, Resource Management, and Approximation in a Data Stream Management System. In CIDR, 2003

4 Carney D, Cetintemel U, Cherniack M, et al. Monitoring streams-A New Class of Data Management Applications. In VLDB, 2002

5 Mouratidis K. Data Stream Processing: An Overview of Recent Research. Submitted to the Qualifying Examination Committee In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science, 2003

6 Avnur R, Hellerstein J. Eddies: Continuously adaptive query processing. In ACM SIGMOD, 2000

7 Madden S, Shah M, Hellerstein J, et al. Continuously Adaptive Continuous Queries over Streams. In SIGMOD, 2002

8 Chandrasekaran S, Franklin M. Streaming Queries over Streaming Data. In VLDB, 2002

9 Hellerstein J, Franklin M, Chandrasekaran S, et al. Adaptive Query Processing: Technology in Evolution. In IEEE, 2000

10 Cormen T, Leiserson C, Rivest R. Introduction to Algorithms. MIT Press, 1990