RFID数据仓库上的多时空粒度近似聚集查询

屈 啸 王永利
(南京理工大学计算机科学与技术学院 南京 210094)

摘 要 随着物联网的发展,以RFID为代表的物联网传感器数据的存储、查询、处理等课题正成为研究的热点。结合数据仓库时空维度和列存储的思想,建立了一种列式RFID数据仓库,并根据RFID的时空特性,设计了一种支持连续聚集查询的多时空粒度数据结构和快速更新算法。它去除了传统聚集查询的部分冗余操作,适合处理大规模RFID数据仓库上的连续实时聚集查询。通过实验证明,该模型与算法在一些典型的物联网应用中取得了较高的效率,可广泛地适用于海量RFID数据仓库上的OLAP分析。

关键词 RFID,数据仓库,聚集查询,多时空粒度
中图法分类号 TP311 **文献标识码** A

Multi-granularity Temporal and Spatial Approximate Aggregate Query on RFID Data Warehouse

QU Xiao WANG Yong-li
(Department of Computer Science, Nanjing University of Science and Technology, Nanjing 210094, China)

Abstract With the development of Internet of Things (IOT), the research concerning storage, query, processing, etc. of sensor data, such as RFID, is becoming more and more popular. In this paper, with the thoughts of temporal and spatial dimensions in data warehouse and column-style storage, we built a column-style warehouse for RFID data. Moreover, according to the temporal and spatial characteristics of RFID data, we designed a data structure for continuous aggregate query of multiple temporal and spatial granularity and a fast updating algorithm, which can remove a part of redundant operation in traditional aggregate query. The structure and algorithm can be applied for processing large-scale continuous aggregate query in RFID data warehouse. The experiment results show that the model and algorithm can achieve high efficiency in some typical application of IOT (Internet of Things), and can be widely applied to massive data OLAP analysis in RFID data warehouse.

Keywords RFID, Data warehouse, Aggregate query, Multi-granularity temporal and spatial

1 引言

随着计算机技术、通信技术、人机交互技术的快速发展,人类迎来物联网(the internet of things)时代。物联网的定义是:通过射频识别、红外感应器、全球定位系统、激光扫描器等信息传感设备,按约定的协议,把任何物品与互联网连接起来进行信息交换和通讯,以实现智能化识别、定位、跟踪、监控和管理的一种网络。目前,物联网中使用最多的也是最成熟的技术是RFID(Radio Frequency Identification,即射频识别)技术[1]。

射频识别是一种非接触式的自动识别技术(Automatic Identification Technology, AIT)。这项技术利用射频电磁波在阅读器和贴有标签的移动物品之间传输数据,达到识别和跟踪物品等目的。有观点认为,RFID是继PC、互联网和无线通信之后的第四次信息技术革命。RFID具有重大科学意义和应用价值,已引起国际学术界与工业界的极大关注。

RFID数据与传统数据相比,有着原始数据元组结构简单、时态性和空间性、数据不准确、连续流产生巨量数据等特点。在传统数据仓库技术中,并没有充分考虑到RFID数据的这些特点[2]。

另一方面,随着企业信息化程度达到一定水平,对极大量数据的存储、处理及分析后对决策的支持等一些不同于传统数据库应用的新需求就变得十分迫切。在海量数据中获取有用信息,速度是一个关键因素和指标。基于这样的需求和背景,近年来对于列存储数据库的研究与开发越来越多。目前使用的绝大部分数据库都是基于行的数据库系统,它将数据库表的每一条记录中的不同属性进行连续存储。基于列存储的数据库与传统的数据库不同,在表中按属性而不是按行进行连续存储。列存数据库的优势在于读取表中属性的子集的速度更快,因为同类型数据的连续存储更利于压缩和优化[3]。

到稿日期:2011-08-02 返修日期:2011-10-19 本文受国家自然科学基金(60803001, 61170035),江苏省自然科学基金重大专项(BK2011022),中国博士后科学基金特别资助项目(200902517),江苏省自然科学基金(BK2011702),江苏省博士后基金(0801043B),江苏省高校2010年"青蓝工程"优秀青年骨干教师项目,南京市科技计划重点项目(020142010)以及南京理工大学2009年度紫金之星项目资助。

屈 啸(1986—),男,硕士生,主要研究领域为物联网数据管理、数据仓库等;王永利(1974—),男,博士,副教授,主要研究领域为数据库技术、情境感知、物联网数据处理、模式识别等。

在RFID数据仓库领域，已有一些人做了相关研究。Gonzalez等人[1]提出了RFID数据立方体，充分考虑了物件的成批移动、数据泛化、部分路径的融合和展开，但未对RFID数据的时态性做充分考虑。F. Wang等人[5]在RFID数据模型方面做了相应研究，提出了动态关系实体关系模型，通过简单增加一个称为“动态关系”的关系，对ER模型进行时态扩展，没有涉及RFID事件/数据泛化等问题。S. Chawathe等人[7]提到一个RFID数据基础设施区别于一般数据仓库的显著特点，即RFID数据仓库更强调“站-区域”活动；指出RFID数据仓库更加强调在中央数据仓库聚合数据，而在数据来源处的查询却并不常见。刘青宝提出了一种多层次时间窗口模型，其能支持在不同时段对数据流进行不同时间粒度的建模，为实时聚集数据查询提供了一种有效的方案，但该模型只是涉及到时间粒度，并未对空间维度做深入研究。

本文贡献如下。

设计了一种基于阈值和时间粒度的聚集更新算法，其大致思想是：为不同维度的属性设计不同的阈值和时间粒度，只有当数据变化超过阈值范围和时间粒度范围时，聚集结果才会被更新并传至更高维度。

设计了一种时间滑窗的冗余消除操作，其大致思想是：在一个时间滑窗中，根据聚集查询的粒度，有些元组可相互抵消，以消除那些对聚集结果不会产生影响的更新操作。

结合以上内容，设计了一种实时近似聚集查询算法，其可共享一个时间滑窗和多时间粒度空间维度树。

2 相关模型

2.1 多粒度时间滑窗模型

2.1.1 模型描述

把时间窗口分成 m 个粒度，令顶层窗口的长度为最长的时间段 T_m，时间区间为$[t_{now}-T_m, t_{now}]$。最底层窗口长度为 T_1，时间区间为$[t_{now}-T_1, t_{now}]$，其他窗口分别为$[t_{now}-T_i, t_{now}]$，其中 $i=2,3,\cdots,m-1$，每个时间窗口$[t_a, t_b]$表示 t_a 到 t_b 时间内检测到的RFID事件集合。这样划分的每个粒度的窗口依次为 $W_1, W_2, \cdots, W_m$，每个粒度的窗口所表示的时间粒度为 $G_1, G_2, \cdots, G_m$，则称三元组序列$\{(W_i, T_i, G_i)\}(i=1, 2, \cdots, m)$为多粒度时间滑窗。

多粒度时间滑窗模型如图1所示。

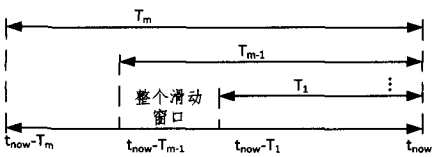

图1 多粒度时间滑窗模型

2.1.2 结构描述

设多粒度时间滑窗模型为三元组序列$\{(W_i, T_i, G_i)\}(i=1,2,\cdots,m)$，其中时间滑窗分为 m 个层次，顶层滑窗长度为 T_m，时间区间为$[t_{now}-T_m, t_{now}]$；最底层的窗口为最短的时间长度，时间区间为$[t_{now}-T_1, t_{now}]$，则多粒度时间窗口结果描述如下：

(1)非底层滑窗节点

设非底层滑窗节点为第 $i(1<i\leqslant m)$ 层滑窗的节点，其信息结构为$\{EventSet, [t_s, t_e), [ptr_1, ptr_2, \cdots, ptr_{k_i}], lptr\}$，其中 $EventSet$ 表示RFID语义事件集合；$[t_s, t_e)$为节点的时间区间，t_s 为起始时间，t_e 为结束时间，$I=t_e-t_s$ 由第 i 层时间滑窗的粒度 G_i 决定；$[ptr_1, ptr_2, \cdots, ptr_{k_i}]$为指向子节点的指针数组，$k_i$ 为第 i 层滑窗的节点扇出度；$lptr$ 为指向左兄弟节点的指针，当左兄弟不存在或者流出时间滑窗时，$lptr$ 指向空指针。

(2)底层滑窗节点

底层滑窗节点为第1层时间滑窗的节点，其信息结构为$\{EventSet, [t_s, t_e), lptr\}$，其中 $EventSet$ 表示RFID语义事件集合；$[t_s, t_e)$为节点的时间区间，t_s 为起始时间，t_e 为结束时间，$I=t_e-t_s$ 由第 i 层时间滑窗的粒度 G_i 决定；$lptr$ 为指向左兄弟节点的指针，当左兄弟不存在或者流出时间滑窗时，$lptr$ 指向空指针。

(3)尾指针组

尾指针组$\{tail_i\}$为指向每层中具有最新结束时间的节点。

多粒度时间滑窗具有以下性质：

(1)若非底层节点 $Node_i$ 的子节点指针$[ptr_1, ptr_2, \cdots, ptr_{k_i}]$均非空，则它们所指向的子节点序列具有时间可加性，并且 $Node_i$ 的时间区间为$[t_{s_1}, t_{e_k})$。

(2)非底层节点 $Node_i$ 的子节点指针在以下两种情况下可以为空：一是当其所指向的子节点流出下层时间滑窗时，子节点指针被设为Null；二是子节点还没有生成。

(3)新的底层节点总是从最右端的尾指针处插入。

下面的例子中(见图2)，将时间划分为3个粒度，由低层到高层的时间粒度分别表示秒、分钟、小时。每个上层例子都由下层例子聚集而成，每个粒度滑窗中，节点的展开数分别为1,60和60，设当前时间为1000,1001时刻的节点即将插入多粒度时间滑窗。

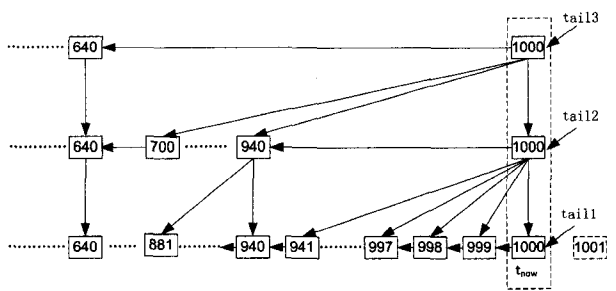

图2 多时间粒度滑窗例子

2.1.3 更新算法描述

多粒度时间滑窗的更新作为多时间粒度空间维度树更新的一部分，其具体过程描述如下：

(1)滑窗 W_1 的更新

W_1 包含的是最近 T_1 时间段内的RFID事件集合，对应的是基粒度 G_1。设 W_1 滑窗序列为 $S_1[0], S_1[1], \cdots, S_1[k_1-1], S_1[k_1]$为即将到达的节点，$S_1[0]$为即将过时的节点。当新的数据到达时，$S_1[k_1]$插入滑窗，若 W_1 滑窗已满，则将最老的 $S_1[0]$移出滑窗。

(2)滑窗 $W_{i+1}(1<i<m)$的更新

在滑窗 W_i 中，按照上层滑窗 W_{i+1} 的粒度对 W_i 中的数据粒子进行分组。当每组的第一个例子到达时，创建新节点 $newNode_{i+1}$，并把 $newNode_{i+1}$ 插入上层滑窗 W_{i+1} 中。以此类推，直至滑窗 W_m。

(3)RFID 语义事件集合的聚集

在多粒度时间滑窗中，每个节点中都还有该节点所表示的时间区间内发生的 RFID 语义事件的集合。当细粒度节点向上层对应的较粗粒度节点聚合信息时，就涉及到多个 RFID 语义事件集合的聚集，因为聚集的操作会因语义的不同而有差异，这里以地点 A 的人数检测的场景为例进行简要介绍。

该场景中，RFID 语义事件的 op 为 $\{in, out\}$，其分别表示进入 A 地区和离开 A 地区。假设 $[t_a, t_b)$ 和 $[t_b, t_c)$ 时间段内，RFID 语义事件集合分别为 $\{(p_1, A, in, t_1), (p_2, A, in, t_2)\}$ 和 $\{(p_2, A, out, t_3)\}$，则在 $[t_a, t_c)$ 时间段内，两个集合的聚集结果为 $\{(p_1, A, in, t_1)\}$。易见，p_2 先后在 A 点进行了 in 和 out 两次操作，这两个操作在语义上可以相互抵消，所以只用记录 p_1 的事件即可。

下面给出多粒度时间滑窗更新算法的伪代码描述。

Algorithm1 TimeSlideWinUpdate (Node $newNode_i$, Pointer $tail_i$)

//$newNode_i$ 待插入的第 i 层的新节点

//$tail_i$ 多粒度时间滑窗第 i 层的尾指针

1. BEGIN
2. IF (第 i 层滑窗已满) THEN
3. 移出第 i 层最老的节点
4. END IF
5. $newNode_i$. lptr=tail. lptr;
6. $tail_i = newNode_i$;
7. IF ($newNode_i$ 是第 i+1 层节点的首子节点) THEN
8. $newNode_{i+1}.[t_s, t_e) = newNode_i.[t_s, t_e)$;
9. $newNode_{i+1}.ptr_1 = tail_i$;
10. $newNode_{i+1}$. RFIDEventSet=$newNode_i$. RFIDEventSet;
11. TimeSlideWinUpdate($newNode_{i+1}$, $tail_{i+1}$);
12. ELSE IF ($newNode_i$ 是第 i+1 层节点的第 j 个子节点) THEN
13. $Node_{i+1} = *tail_{i+1}$;
14. $Node_{i+1}.ptr_j = newNode_i$;
15. //RFID 语义事件集合的聚集
16. $Node_{i+1}$. RFIDEventSet=$Node_{i+1}$. RFIDEventSet+$newNode_i$. RFIDEvent;
17. $Node_{i+1}.t_e = newNode_i.t_e$;
18. END IF
19. END

2.1.4 算法复杂度分析

设当前多粒度时间滑窗的粒度层次数为 m，第 i 层滑窗尾指针所指向节点设为 $tailNode_i$，其中 $|tailNode_i.RFIDEventSet| = n_i$，即将插入的滑窗节点为 $newNode$，$|newNode.RFIDEventSet| = n$。$newNode$ 达到最底层滑窗时，会由低层次到高层次与相应层次尾节点进行更新操作。最好的情况下，$newNode$ 只会与 $tailNode_1$ 进行一次更新操作；最坏的情况下，$newNode$ 会与 $tailNode_1$ 进行更新操作，更新过后的节点 $tailNode_1$ 向上传递并与 $tailNode_2$ 进行更新操作，以此类推，直至最高层次的 $tailNode_m$ 得到更新。

滑窗节点进行更新时，主要操作为 RFID 语义事件集合的聚集运算。设两个 RFID 语义事件集合 $RFIDEventSet_1$ 和 $RFIDEventSet_2$，$|RFIDEventSet_1| = n_1$，$|RFIDEventSet_2| = n_2$，则聚集运算的复杂度为 $O(n_1 * n_2)$。不失一般性，设 $max_n = max\{n_1, n_2\}$，则复杂度可表示为 $O(max_n^2)$。设进行滑窗更新算法时，RFID 语义事件集合的平均大小为 n，则最好情况下，更新算法的复杂度为 $O(n^2)$；最坏情况下，复杂度为 $O(m * n^2)$。

2.2 多时间粒度空间维度树模型

2.2.1 模型描述

与传统数据仓库中的数据不同，在典型的 RFID 应用中，数据更新是频繁的，而且更加需要最近的数据以及分析。基于这种背景，本篇论文设计了多时间粒度空间维度树模型。该模型的结构是一颗多叉树，从根节点到叶子节点，空间维度逐渐降低，叶子节点为包含最细粒度 RFID 信息的节点。每个节点中，包含根据不同维度而设置的时间粒度和聚集阈值。

多时间粒度空间维度树模型的结构如下：

(1)叶子节点

叶子节点表示最细物理维度的多种时间粒度 RFID 聚集信息，包含以下属性：多时间粒度维度树、维度编码、多粒度聚集结果、多粒度聚集结果变化值、聚集结果阈值。

(2)非叶节点

非叶节点表示通过底层的叶子节点和非叶节点得到的更高物理维度的 RFID 聚集信息，包含以下属性：维度编码、多粒度聚集结果、多粒度聚集结果变化值、聚集结果阈值。

2.2.2 建立多时间粒度空间维度树

在实际应用中，空间属性通常含有不同维度，并且即使对于同一个物理空间，不同角色的用户对空间维度也会有不同的需求。以图 3 为例，在物流应用中，假设货物要通过仓库、运输车辆、商店库房、货架和付款处 5 个环节。对于商店，其并不关心货物具体是在仓库还是运输车上，只把它们共同看作运输环节；而对于运输商，其只要将货物送到商店即可，并不关心之后的货架以及付款处环节。

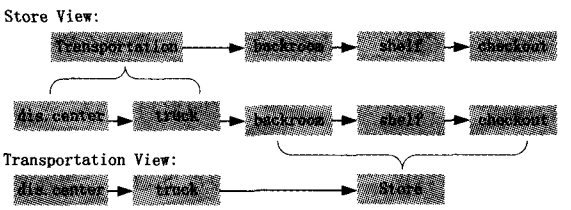

图 3 空间维度例子

根据上述应用以及多时间粒度空间维度树模型，可以建立针对该应用场景的多时间粒度空间维度树。结构如图 4 所示。

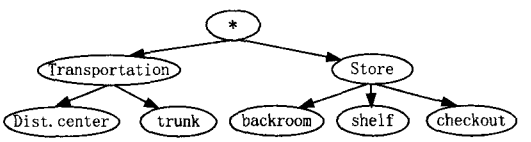

图 4 多时间粒度空间维度树模型的例子

2.2.3 更新多时间粒度空间维度树

在对空间维度树上的聚集结果进行更新时，只有节点的聚集结果与时间粒度以及阈值满足一定条件时，该节点的聚集结果才会传至上一层节点用于更新。

多时间粒度空间维度树的更新过程为：

获取最近最细粒度 RFID 语义事件集合，并将集合按物理地点分组；

对于每个分组过后的 RFID 语义事件集合，通过维度编

码判断该地点是否包含在查询区域，如果不在，则退出；

如果该节点为叶子节点，则执行多粒度时间滑窗更新算法 TimeSlideWinUpdate；

更新多粒度聚集结果变化值；

如果变化值大于该节点聚集结果阈值，则更新聚集结果，将聚集变化值提交到父节点，并以此类推，最后聚集变化值置0。

下面给出多时间粒度空间维度树更新算法的伪代码描述。

Algorithm 2 MTGSDTreeUpdate(curNode, resultDelta)

```
1. BEGIN
2.   IF (curNode 在查询区域内)
3.   THEN
4.       IF (该节点是叶子节点)
5.       THEN
6.           //创建多粒度时间滑窗节点
7.           Node. RFIDEventSet=RFIDEventLocationSet
8.           //获取最低一层时间滑窗的尾指针
9.           Tail=curNode. slideWindow[1];
10.           TimeSlideWinUpdate(Node, Tail);
11.       END IF
12.       //更新聚集变化值
13.       UpdateResultDelta(curNode);
14.       IF (resultDelta > threshold )
15.       THEN
16.           UpdateRsult(curNode);
17.           //递归调用更新算法
18.           IF (curNode ! = rootNode )
19.              MTGSDTreeUpdate(curNode. parent, curNode. re-
                 sultDelta);
20.           curNode. resultDelta=0;
21.       END IF
22.   END IF
23. END
```

2.2.4 算法复杂度分析

设多时间粒度空间维度树的节点数为 T，节点的平均展开数为 k，节点中 RFID 语义事件集合的平均大小为 n，叶子节点中多粒度时间滑窗的粒度种类为 m。在对不同物理维度的聚集变化值和聚集结果进行更新的过程中，最好的情况下，只对最低维度节点进行更新，即对其中的多粒度时间滑窗、聚集变化值和聚集结果进行更新；最坏情况下，除了对最底层节点进行更新，还需对其父亲节点中的聚集变化值和聚集结果进行更新，以此类推，直至更新根节点。

节点更新分为两种方式：叶子节点更新和非叶子节点更新。

更新过程中，多粒度时间滑窗更新复杂度为 $O(m*n^2)$，聚集变化值和聚集结果的更新为 $O(C)$。因此，最好情况下，算法复杂度为 $O(m*n^2+C)$；最坏情况下，算法复杂度为 $O(m*n^2+C*\log_k T)$。

3 实验分析

本节首先对本文提出的多时间粒度空间维度树的更新算法的查询响应时间和查询结果误差进行实验分析，并与目前比较流行的近似查询算法进行对比，以验证多时间粒度空间维度树模型以及算法的优越性。另外，将本文提出的算法分别在传统数据仓库和列式数据仓库上进行试验，以验证列式存储在数据仓库领域的性能优势。

目前，在聚集查询近似处理研究中，已有抽样技术、直方图技术以及小波变换技术等多种经典技术。本次实验中，将采用 S. Acharya[8] 提出的 congressional sample 算法与本文提出的算法进行对比。

第一组实验对比了在不同数据集大小以及不同查询频率的情况下，两种算法响应查询所需的时间。第二组实验对比了在不同数据集大小以及不同查询频率的情况下，两种算法的查询结果与准确结果之间的误差。第三组实验构建了传统数据仓库和列式数据仓库两种环境，并对比了本文提出的算法在这两种环境下的性能。

本组实验基于 Linux 平台，内存为 1G，CPU 为 Intel Core i3 2.9GHz，实验环节采用 J2EE 作为开发工具，传统数据仓库利用 MySQL 构建，列式数据仓库利用 HBase 构建。实验采用的数据为某 RFID 仓库管理应用中的数据。应用场景是某 RFID 仓库管理应用，需要实时统计某个地理维度上的货物存储量。在实验时，针对海量数据以及实时性的要求，分别采取不同的数据集大小以及数据更新频率，测量对于同一个查询或多种不同查询，使用多时间粒度空间维度树模型时查询的响应时间。

3.1 多时间粒度空间维度树查询及更新响应时间实验及分析

3.1.1 数据集大小对于查询响应时间的影响

在不同的数据集大小的情况下，对于使用多时间粒度空间维度树和 congressional sample 的查询响应时间的测量结果如图 5 所示。图中横坐标表示数据集的大小，单位为 Mb；纵坐标表示查询相应时间，单位为 ms。

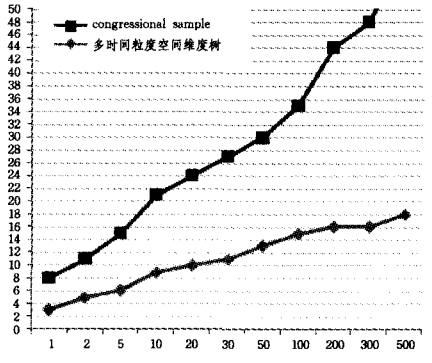

图 5 在不同数据集大小，查询响应时间测量结果

由图 5 可知，随着数据集大小的增长，两种算法的查询响应时间都随之增加，但使用多时间粒度空间维度树的响应时间的增长幅度较为平缓。在数据集从 1Mb 提高到 500Mb 的过程中，查询响应时间最高为 18ms，查询的效率较高。

3.1.2 数据插入频率对于查询响应时间的影响

在不同的数据插入频率下，对于使用多时间粒度空间维度树和 congressional sample 算法的查询时间的影响如图 6 所示。图中横坐标表示数据插入的频率，单位为插入次数/s；纵坐标表示查询相应时间，单位为 ms。

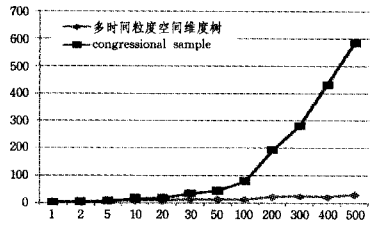

图 6　不同数据插入频率下，查询时间和更新时间测量结果

由图可知，随着数据插入频率的增长，congressional sample算法的增长幅度较大，而多时间粒度空间维度树算法由于采用近似聚集查询思路，因此查询相应时间的变化则不大。

3.2　多时间粒度空间维度树查询误差实验及分析

图 7 和表 1 表示，随机查询下的准确聚集结果以及使用多时间粒度空间维度树得到的近似聚集结果及误差。

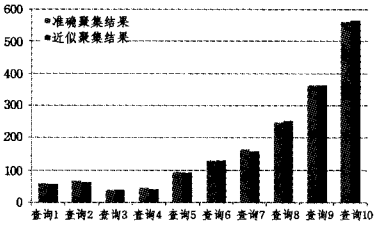

图 7　随机查询下的准确聚集结果以及近似聚集结果

表 1　随机查询下的准确聚集结果、近似聚集结果及其误差

	准确聚集结果(ms)	近似聚集结果(ms)	误差(%)
查询 1	59	57	3.39
查询 2	68	65	4.41
查询 3	39	39	0.00
查询 4	46	42	8.70
查询 5	95	92	3.26
查询 6	130	131	7.69
查询 7	165	159	3.64
查询 8	250	253	0.12
查询 9	365	365	0.00
查询 10	563	566	5.33

由图 7 和表 1 可以看出，使用多时间粒度空间维度树的近似聚集结果的误差最低为 0%，最高为 8.7%。

3.3　传统数据仓库与列式数据仓库性能对比

图 8 表示多时间粒度空间维度树在传统行式数据仓库和列式数据仓库环境中，对于不同数据集进行同一种查询的相应时间对比。图中，横坐标表示已有数据集的大小，纵坐标表示响应时间，单位为 ms。

由图 8 可知，数据集在 1M～1G 的情况下，两种数据仓库的查询响应时间还没有明显差距。但当数据集大小为 10G 或者更大的情况下，响应时间的差距就十分明显，行式数据仓库环境下的查询时间增长较快，而列式数据仓库则比较平缓。

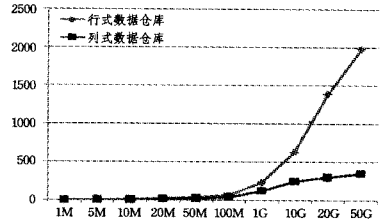

图 8　不同数据集下的行式/列式数据仓库查询响应时间

结束语　在数据仓库环境中，由于大数据量以及查询的复杂性使得一个查询的执行通常需要很长时间，并且目前对实时查询的需求越来越迫切，因此，近似查询技术成为处理该类查询的一个有效手段。本文结合 RFID 数据的特点以及聚集查询技术设计了支持连续不同时空粒度的聚集查询的数据结构——多时间粒度空间维度树，用以对 RFID 数据的聚集信息进行快速索引，以去除部分在连续实时聚集查询中的冗余操作。通过实验证明，该模型与算法在不同数据集大小以及数据插入频率的情况下，查询响应时间较小，符合对于 RFID 数据在列式数据仓库中进行实时查询的基本要求。但是，本文的模型和算法还存在许多不足，比如模型的构造是静态的，无法动态改变其结构；模型的查询效率较高，但更新时间仍会随着数据集大小以及数据插入频率的增加而显著提高。这些不足都是我们下一阶段改进的重点。

参 考 文 献

[1] Want R. An Introduction to RFID Technology[J]. Pervasive Computing, IEEE, 2006(6): 25-33

[2] 李战怀，聂艳明，陈群，等. RFID 数据管理的研究进展[J]. 中国计算机学会通讯，2007(8)

[3] Abadi D J, Boncz P A, Harizopoulos S. Column-oriented Database Systems[C]// VLDB '09. 2009: 24-28

[4] Han Jia-wei, Gonzalez H, Li Xiao-lei, et al. Warehousing and Mining Massive RFID Data Sets[C]// ADMA. 2006: 1-18

[5] Wang Fu-sheng, Liu Pei-ya. Temporal Management of RFID Data[C]// Proceedings of the 31st VLDB Conference. 2005: 1128-1139

[6] Chawathe S S, Krishnamurthy V, Ramachandrany S, et al. Managing RFID Data[C]// VLDB. 2004: 1189-1195

[7] Chawathe S S, Krishnamurthy V, Ramachandrany S, et al. Managing RFID Data[C]// VLDB. 2004: 1189-1195

[8] Acharya S, Gibbons P B, Poosala V. Congressional Samples for Approximate Answering of Group-By Queries[C]// SIGMOD. 2000

(上接第 154 页)

[6] Jin Che-qing, Yi Ke, Chen Lei, et al. Sliding-window top-k queries on uncertain Streams [J]. Proceedings of the 34th International Conference on Very Large Data Bases, 2008, 1(1): 301-312

[7] Hua Ming, Pei Jian, Zhang Wen-jie, et al. Ranking queries on uncertain data: A probabilistic threshold approach [C]// SIGMOD. 2008: 673-687

[8] Yi Ke, Li Fei-fei, Kollios G. Efficient Processing of Top-k Queries in Uncertain Databases with X-Relations [J]. Knowledge and Data Engineering, 2009, 21(1): 1-14

[9] 刘德喜，万常选，刘喜平. 不确定数据库中基于 X-tuple 的高效 Top-k 查询处理算法[J]. 计算机研究与发展，2010，47(8)：1415-1423

[10] 王爽，王国仁. 基于不确定数据的分布式 Top-k 查询算法[J]. 东北大学学报：自然科学版，2010，31(2)：177-180

[11] 王晓伟，贾焰，杨树强，等. 存在级不确定数据上的概率 Skyline [J]. 计算机研究与发展，2011，48(1)：68-76