(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 1601541 B
(45)授权公告日 2011.11.30

(21)申请号 200410055239.5
(22)申请日 2004.07.14
(30)优先权数据
10/670,561 2003.09.26 US
(73)专利权人 微软公司
地址 美国华盛顿州
(72)发明人 G·B·齐克德罗夫 R·Z·加森 E·A·里尔 余春
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 张欣
(51)Int.Cl.
G06F 17/30(2006.01)
(56)对比文件
US 6496909 B1,2002.12.17, 全文.
US 20030149702 A1,2003.08.07, 全文.
审查员 林亮亮

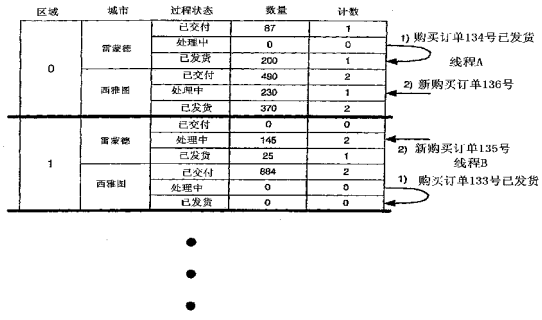
权利要求书 4 页 说明书 10 页 附图 9 页

(54)发明名称

用于自维护的实时数据聚集的方法和数据处理装置

(57)摘要

形成能够作为聚集数据表中的多个分区的多个聚集群。每个群包括多个聚集记录;每条聚集记录包括多个数据库记录的不同子集所包含的各值的聚集。当单个程序线程在聚集群更新事务期间访问聚集群时,不允许其他线程访问该群。各聚集群被组合成单个聚集记录表。多个数据库记录中的每一个可以与组织活动的一实例相对应且包括具有指示处于若干过程状态之一的相对应实例的值的字段。每个聚集群还可以包含按时间排序的聚集记录,每个按时间排序的聚集记录包含在与该按时间排序的聚集记录相关联的时间段期间处于若干过程状态之一的实例的聚集值。删除与在预选时窗之外完成的实例相对应的聚集记录。



CN 1601541 B

1. 一种用于聚集来自多个数据库记录的数据，从而概括关于组织活动的多个实例的信息的方法，其中所述多个数据库记录的每一个对应于所述组织活动的多个实例中的一个，其中所述多个数据库记录的每一个中的数据反映对应于该记录的组织活动实例的属性，并且其中所述方法由至少一个计算机中的多个程序线程来执行，该方法包括：

创建至少一个表示多个分区的聚集表，每个分组包括多个聚集记录，每个聚集记录包括聚集值，该聚集值表示所述多个数据库记录的一个独立子集的各字段所包含的各值的聚集；

当插入或更新所述多个数据库记录中的第一数据库记录时，选择所述多个分区中的第一分区，其中，选择第一分区是响应于多个程序线程中的第一程序线程访问所述多个分区之一的请求而启动的；

在第一分区的聚集记录的至少一个中更新聚集值，其中所述更新是由第一程序线程来执行并作为第一分区更新事务的一部分，并且所述第一分区更新事务是基于所插入或所更新的第一数据库记录中的一个或多个值的；

阻止其它程序线程访问所述第一分区，直到所述第一程序线程不再请求对所述第一分区的访问；

当执行所述第分区更新事务时，选择第二分区，其中，选择第二分区是响应于多个程序线程中的第二程序线程访问所述多个分区之一的请求而启动的；

在第二分区的聚集记录的至少一个中更新聚集值，其中所述更新由第二程序线程来执行并作为第二分区更新事务的一部分，并且所述第二分区更新事务是基于所插入或所更新的第二数据库记录中的一个或多个值的，并且其中所述第二分区更新事务是在第一分区更新事务执行期间执行的；以及

聚集来自所述多个分区的聚集值，并输出经聚集的聚集值以作为所述多个组织活动实例的一部分。

2. 如权利要求 1 所述的方法，其特征在于，每个分区是多分区聚集表的一个单独分区。

3. 如权利要求 1 所述的方法，其特征在于，还包括：

当启动更新所述多个数据库记录中的第一数据库记录的后续事务时，选择第三分区；以及

基于后续更新的第一数据库记录中的一个或多个值，修订所述第三分区的聚集记录的聚集值。

4. 如权利要求 1 所述的方法，其特征在于，所述聚集来自所述多个分区的聚集值的步骤包括：

将所述多个分区组合成单个聚集记录表，该单个表的每个记录聚集了来自所述多个分区的每一个的聚集记录的值。

5. 如权利要求 1 所述的方法，其特征在于，所述创建至少一个聚集表包括创建至少三个分区。

6. 如权利要求 1 所述的方法，其特征在于，所述创建至少一个聚集表包括创建至少十个分区。

7. 如权利要求 1 所述的方法，其特征在于，该方法在带有至少一个处理器的计算机上执行，并且其中所述创建至少一个聚集表包括创建一定数目的聚集群，并且其中所述数目

超过所述计算机中的处理器的数目。

8. 如权利要求 1 所述的方法，其特征在于，还包括：

当收到来自第一程序线程的访问分区的请求时，确定发出请求的程序线程的系统标识符；以及

基于所确定的系统标识符来将分区标识符分配给第一程序线程。

9. 如权利要求 1 所述的方法，其特征在于：

所述多个数据库记录中的每一个包括具有指示对应的实例处于若干过程状态之一的值的字段，以及

每个分区包括按时间排序的聚集记录，每个按时间排序的聚集记录包括在与该按时间排序的聚集记录相关联的时间段期间处于所述若干过程状态之一的实例的聚集值。

10. 如权利要求 9 所述的方法，其特征在于，所述若干过程状态之一包括实例已完成，并且所述方法还包括：

修订所述按时间排序的聚集记录的聚集值，从而从所述经修订聚集值中排除在预选时窗之外完成的实例相对应的记录的影响。

11. 如权利要求 9 所述的方法，其特征在于，包括：

对于一个聚集记录，当确定在与其相关联的时间段期间，所述多个数据库记录中没有一个处于若干过程状态之一，则将该聚集记录从分区删除。

12. 如权利要求 9 所述的方法，其特征在于，所述过程状态之一与实例已完成相对应，并且其中所述多个数据库记录中的每一个都包括完成时间字段；并且所述方法还包括：

为与未完成的实例相对应的数据库记录的完成时间字段分配空值；

为与处于已完成过程状态的实例相对应的数据库记录的完成时间字段分配非空值。

13. 如权利要求 12 所述的方法，其特征在于，还方法包括：

确定所述多个数据库记录中的一个记录是否已经被更新；

当确定所述数据库记录已经被更新时，基于所述更新修订所述多个聚集记录中的所述一个的聚集值；

进一步确定已更新的数据库记录是否包括指示所述相应实例处于已完成过程状态的值；

当确定所述相应实例处于已完成过程状态时，生成所述实例的完成时间值；以及

基于所述实例的完成时间值来更新所述多个聚集记录中的所述一个。

14. 一种用于聚集来自多个数据库记录，从而概括关于组织活动的多个实例的信息的数据处理装置，其中所述多个数据库记录的每一个对应于所述组织活动的多个实例中的一个，其中所述多个数据库记录的每一个中的数据反映对应于该记录的组织活动实例的属性，该数据处理装置，包括：

至少一个数据储存设备；

至少一个用户输入设备；以及

在操作上连接到所述储存设备和所述用户输入设备的处理器，其中所述至少一个数据储存设备上存储有一组指令，所述一组指令在被执行时将所述处理器配置成：

创建至少一个表示多个分区的聚集表，每个分组包括多个聚集记录，每个聚集记录包括聚集值，该聚集值表示所述多个数据库记录的一个独立子集的各字段所包含的各值的聚

集；

当插入或更新所述多个数据库记录中的第一数据库记录时，选择所述多个分区中的第一分区，其中，选择第一分区是响应于多个程序线程中的第一程序线程访问所述多个分区之一的请求而启动的；

在第一分区的聚集记录的至少一个中更新聚集值，其中所述更新是由第一程序线程来执行并作为第一分区更新事务的一部分，并且所述第一分区更新事务是基于所插入或所更新的第一数据库记录中的一个或多个值的；

阻止其它程序线程访问所述第一分区，直到所述第一程序线程不再请求对所述第一分区的访问；

当执行所述第分区更新事务时，选择第二分区，其中，选择第二分区是响应于多个程序线程中的第二程序线程访问所述多个分区之一的请求而启动的；

在第二分区的聚集记录的至少一个中更新聚集值，其中所述更新由第二程序线程来执行并作为第二分区更新事务的一部分，并且所述第二分区更新事务是基于所插入或所更新的第二数据库记录中的一个或多个值的，并且其中所述第二分区更新事务是在第一分区更新事务执行期间执行的；以及

聚集来自所述多个分区的聚集值，并输出经聚集的聚集值以作为所述多个组织活动实例的一部分。

15. 如权利要求 14 所述的数据处理装置，其特征在于，所述一组指令包括附加指令，所述附加指令在被执行时将所述处理器配置成：

当启动更新所述多个数据库记录中的第一数据库记录的后续事务时，选择第三分区，以及

基于后续更新的第一数据库记录中的一个或多个值，修订所述第三分区的聚集记录的聚集值。

16. 如权利要求 14 所述的数据处理装置，其特征在于，所述聚集聚集值的步骤包括：

将所述多个分区组合成单个聚集记录表，该单个表的每个记录聚集了来自所述多个分区中的每一个的聚集记录的值。

17. 如权利要求 14 所述的数据处理装置，其特征在于，所述一组指令包括附加指令，所述附加指令在被执行时将所述处理器配置成：

当收到来自第一程序线程的访问分区的请求时，确定第一程序线程的系统标识符，以及

基于所确定的系统标识符来将分区标识符分配给第一程序线程。

18. 如权利要求 14 所述的数据处理装置，其特征在于，

所述多个数据库记录中的每一个包括具有指示对应的实例处于若干过程状态之一的值的字段，以及

每个分区包括按时间排序的聚集记录，每个按时间排序的聚集记录包括在与该按时间排序的聚集记录相关联的时间段期间处于所述若干过程状态之一的实例的聚集值。

19. 如权利要求 18 所述的数据处理装置，其特征在于，所述一组指令包括附加指令，所述附加指令在被执行时将所述处理器配置成：

修订所述按时间排序的聚集记录的聚集值，从而从所述经修订聚集值中排除在预选时

窗之外完成的实例相对应的记录的影响。

20. 如权利要求 18 所述的数据处理装置，其特征在于，所述过程状态之一与实例已完成相对应，并且其中所述多个数据库记录中的每一个都包括完成时间字段，并且其中所述一组指令包括附加指令，所述附加指令在被执行时将所述处理器配置成：

为与未完成的实例相对应的数据库记录的完成时间字段分配空值；

为与处于已完成过程状态的实例相对应的记录的完成时间字段分配非空值。

21. 如权利要求 20 所述的数据处理装置，其特征在于，所述一组指令包括附加指令，所述附加指令在被执行时将所述处理器配置成：

确定所述多个数据库记录中的一个记录是否已经被更新，

当确定所述数据库记录已经被更新时，基于所述更新修订所述多个聚集记录的所述一个记录的聚集值，

进一步确定已更新的数据库记录是否包括指示所述相应实例处于已完成过程状态的值，

当确定所述相应实例处于已完成过程状态时，生成所述实例的完成时间值，以及

基于所述实例的完成时间值来更新所述多个聚集记录中的所述一个记录。

用于自维护的实时数据聚集的方法和数据处理装置

发明领域

[0002] 本发明涉及用于监视公司或者其他组织的工作流程的方法和计算机系统。本发明更具体地涉及用于聚集与活动的多个实例相关的信息以及用于维护该聚集的方法。

[0003] 发明背景

[0004] 公司和其他组织使用计算机，并且尤其是计算机数据库应用程序，来监视和记录关于组织活动的信息。通常，该组织会有必须执行且频繁再现的各种过程或者活动。的确，在任何给定时间，公司具有活动的处于不同完成阶段的多个实例是很常见的。作为一个示例，一公司可能基于从客户接收到的订单来销售商品。感兴趣的活动可以是履行这些客户订单；每一个购买订单表示该活动的一个单独实例。在任何特定时间，该公司可具有该活动的处于完成的各个阶段的多个实例（即，来自多个客户的多个订单）。作为另一示例，一个金融机构可基于来自客户的申请而贷款给这些客户。感兴趣的活动可以是处理贷款申请以完成（例如，批准或拒绝），每一个贷款申请表示该活动的一个单独实例。在任何特定时间，司存在处于处理的不同阶段的多个贷款申请实例。在又一示例中，负责签发许可的政府实体可具有处于不同的处理阶段的多个许可申请。

[0005] 为了能够监视活动的多个实例，许多组织将关于这些活动实例的信息存储在数据库程序中。具体而言，可为活动的每个实例创建记录或其他数据对象。然后建立一个单独的字段或该记录的其他组成，以持有每个实例所共有的一些信息类型的值。使用前述示例之一作为说明，销售商品的公司可为每个客户订单创建单独的数据库记录。在该记录中，可以存在用于以下各项的分开字段：接收到订单的时间，订单是从何处接收到的，订购的是什么，订单何时发货，等等。数据库程序的这种使用通常被概念化为一个表。活动的每个实例被分配了该表的一个单独的行（或元组）。多个实例所共有的信息的每个类型随后被分配了该表的一个单独的列。

[0006] 尽管有时需要个别记录中的个别字段的值，但许多组织频繁需要关于一群记录的信息。而且，经常实时地需要该信息。诸如，许多销售商品的公司需要了解当前有多少订单正待处理，有多少订单已被完成，以及有多少订单正处于完成的一个或更多个中间阶段。某些数据库程序可以通过聚集该数据库的多个记录中的值来提供这样的报告。然而，仅此而已，但在数据库很庞大的情况下，这通常是不可接受的解决方案。

[0007] 随着越来越多的记录累积，数据库的访问速度会显著降低。对于诸如每天接收到成百上千订单的商品销售商等大型公司而言，记录的数量能达到数十万或数百万。在每次查询数据库时，需要有限的时间来搜索盘驱动器或其他存储设备。类似地，当创建新记录和更新旧记录时，需要有限的时间来创建或更新每个记录。随着记录数目的增长，找到特定记录所需的时间在增加。在具有数百（或数千）用户和数十万（或数百万）数据库记录的公

司或者组织中，数据库系统访问的等待时间变得相当长。此外，在众多用户尝试访问数据库中的同一信息的情况下，在尝试访问同一记录的用户之间会发生死锁。如果众多用户将记录插入或者更新到大型数据库中而其他用户尝试访问该数据库以生成各个字段的摘要，则所有用户都将体验欠满意的数据库性能。

[0008] 另一个可能的解决方案是为数据库中的数据生成联机分析处理（OLAP）立方体。然而，生成 OLAP 立方体所需要的处理也相当耗时。如果存在大量数据库记录，则通常只能在每天的的基础上（或有时是在每小时的基础上）生成 OLAP 立方体。如果组织实时地需要聚集信息，则 OLAP 立方体常常会力不从心。

[0009] 发明概述

[0010] 本发明解决与维护关于活动的多个实例的聚集信息相关联的上述和其他挑战。在至少一个实施例中，本发明包括一种用于维护多个数据库记录的各字段所包含的各值的聚集的方法。该方法包括创建多个聚集群。每一个群包括多个聚集记录，并且每个聚集记录包括该多个数据库记录的不同子集的各字段所包含的各值的聚集的值。该方法还包括，在插入或更新该多个数据库记录的第一数据库记录时，选择第一聚集群。该方法还包括基于所插入或更新的第一数据库记录中的一个或多个的值并作为第一聚集群更新事务的一部分来修订第一聚集群的各聚集记录之一的聚集值。阻止对该第一聚集群的随后选择，直至完成第一聚集群的更新事务为止。在执行第一聚集群的更新事务且在插入或更新该多个数据库记录中的第二数据库记录时，选择第二聚集群。基于所插入或更新的第二数据库记录中的一个或多个值且在第一聚集群更新事务期间，修订第二聚集群的各聚集记录之一的聚集值。在本发明的其他方面，各聚集群被组合成聚集记录的单个表。

[0011] 在至少另一实施例中，本发明包括一种用于维护关于组织活动的多个实例的聚集数据的方法，该活动的每个实例具有多个过程状态中的一个状态。该方法包括在聚集数据表中创建多个记录，每条记录包含在同一时间段期间处于同一过程状态的多个实例的子集的聚集值。该方法还包括更新聚集数据表来反映将与处于各过程状态中的一个状态的实例相对应的数据删除到预选时间窗口之外。

[0012] 结合附图，本发明的这些和其他特征和优点将从优选实施例的以下详细描述中将显而易见并易于完全理解。

[0013] 附图简述

[0014] 图 1 是示出了假设的批发公司对客户订单的处理的框图。

[0015] 图 2 是用于图 1 的公司的实例数据表的一部分。

[0016] 图 3 是聚集了来自图 2 的表的各个字段的数据的表。

[0017] 图 4 示出了对聚集数据表的更新以反映新实例数据记录。

[0018] 图 5 示出了对聚集数据表的更新以反映已有实例数据记录的更新。

[0019] 图 6 示出了在多个程序线程之间的死锁。

[0020] 图 7 示出了根据本发明的至少一个实施例的多分区聚集表。

[0021] 图 8 示出了根据本发明的至少一个实施例的图 7 的聚集表的实现。

[0022] 图 9 示出了根据本发明的至少一个实施例的组合了多分区聚集表的各分区的视图。

[0023] 图 10 示出了用于将各分区分配给各程序线程的存储过程的一个实现。

[0024] 图 11 示出了提供聚集数据且其中使用公司里程碑时间戳来作为聚集准则的另一个表。

[0025] 图 12 示出了根据本发明的至少一个实施例方便对陈旧聚集数据进行删除的表。

[0026] 图 13 是示出了根据本发明的至少一个实施例的用于维护实时数据聚集表的触发器的逻辑的流程图。

[0027] 优选实施例的详细描述

[0028] 本发明可通过结合于 2002 年 5 月 31 日提交的、序号为 10/157,968、题为“Support for Real-Time Queries Concerning Current State.Data and history of a process(对关于过程的当前状态、数据和历史的实时查询的支持)”的美国专利申请中所描述的方法、装置和系统来有利地使用,该专利申请的内容通过引用结合与此。

[0029] 将参考可在可从华盛顿州雷德蒙市的微软公司获得的 SQL SERVER™2000 关系数据库管理系统 (RDBMS) 软件和相关联的 RDBMS 联机分析处理 (OLAP) 软件中找到的结构化查询语言 (SQL) 指令和其他数据分析特征来描述本发明。尽管在这里描述了可用来实现本发明的某些实施例的 SQL 指令的某些方面,然而,只要向本领域技术人员提供了这里所提供的说明书,则用于实现本发明的其他指令、程序设计算法和过程对于这些人员而言将是显而易见的。可从各种源获得 SQL SERVER™ 2000RDBMS 软件和相关联的 OLAP 服务软件的概括描述,包括可从 http://www.microsoft.com/sql/techinfo/productdoc/2000/ ＞获得的 Karen Delaney(2001Microsoft press) 的 Inside Microsoft® SQL SERVER™ 2000 和 Microsoft® SQL SERVER™ 2000Books Online。本发明不限于使用 SQLSERVER™2000 RDBMS 软件和相关联的 OLAP 服务软件的实现,并且可使用其他类型的 RDBMS 软件和 OLAP 软件来实现。

[0030] 还可以参考在服务器上操作并由一个或多个客户机访问的 RDBMS 软件(例如上述的 SQL SERVER™2000 软件)来描述本发明。这种配置在本领域中是已知的,且在例如在先前包括的美国专利申请 10/157,968 中描述。然而,客户机 - 服务器配置仅是可以实现本发明的方式的一个示例。本发明也能够在其他物理系统配置中实现。

[0031] 图 1 是示出了基于客户购买订单来向客户销售商品的假设的批发公司对客户订单的处理的框图。为方便起见,公司在此被称为“公司 X”。公司 X 接收来自多个客户的购买订单。在接收到每个购买订单时,公司 X 将关于该订单的信息输入到在数据库服务器上维护的数据库中。具体而言,公司 X 在数据库中为该购买订单创建新记录,连同用于各单个购买订单所共有的各种信息类型的字段。在该示例中,每条记录包括用于购买订单数目、接收的日期和时间、客户所在城市、订购的商品的数量的各个字段。公司 X 确定每个订单将被接受还是拒绝,并接着更新数据库内的一个或者多个其他字段来反映这一接受或拒绝。如果一个购买订单被接受,则将生成相对应的销售订单并将其发送给公司 X 的处于订购客户所在城市的仓库。当商品被发货给客户时,将发货时间输入(通过仓库之一处的客户计算机)到该记录的另一字段。当商品交付时,更新又一字段。

[0032] 尽管创建了上述示例来描述本发明,但是该示例(包括在此描述的附加方面)表示许多实际公司,尽管是以简化的形式。此外,本领域技术人员将明白,相对于假设的公司 X 所描述的概念广泛适用于公司和其他组织活动。的确,可以一般参考“组织”而非“公司 X”来替换地描述本发明。类似地,代替参考购买订单和订单履行的各个阶段,可以使用一般术

语来描述本发明，例如“组织过程的实例”、组织过程实例的“状态”、组织过程实例的完成，等等。虽然使用了实际公司类型的简化示例来提供更易读的描述，但本发明不限于特定类型的组织或组织活动。

[0033] 图 2 是来自图 1 的数据库的数据表的一部分，而且由公司 X 维护并用于存储关于各单独的购买订单实例的数据。这一实例数据表具有每个购买订单的单独记录（例如，行）和各种数据类型的单独字段（列）。在该示例中，“PO#”是购买订单号码。“接收时间”是接收购买订单的日期和时间，“城市”是最接近于发出购买订单的客户的仓库的所在城市，且“数量”是订购的项目的数目。“发货时间”是购买订单的商品的发货时间，且“交付时间”是商品交付的时间。“过程状态”是用来描述购买订单当前正在经历的那一部分过程的变量。如果已接收了购买订单但还未发货，则购买订单是“处理中”。如果已经发货但尚未交付，则购买订单是“已发货”。如果商品已经交付，则购买订单是“已交付”。尽管没有示出，但还可存在其他过程状态值（例如，被拒绝的购买订单的“已拒绝”）。商品尚未发货的购买订单在发货时间字段会有一个 <NULL>（空）值。类似地，尚未交付的购买订单在“交付时间”字段中会有一个 <NULL> 值。在购买订单发货和 / 或交付时，将用适当的时间值来更新这些字段。

[0034] 如可从图 2 看到的，并且即使在只有有限数目的记录的情况下，当需要聚集信息的时候，原始数据的表将变得难以使用。例如，如果需要来自特定仓库的所交付的所有商品，则必须检查每一行。用于获取聚集数据的更加有用的表格式在图 3 中示出。具体而言，图 3 通过仓库示出当前已交付、处理中或已发货的购买订单的总数（“计数”）。图 3 还在“数量”列中示出了每个仓库已交付、已发货和正在处理以供发货的商品的总数。

[0035] 图 4 示出如何更新图 3 的表以反映接收到来自雷蒙德市的 30 个单位的新购买订单 135 号。具体而言，雷蒙德市的“处理中”总数增加了 30 且“计数”总数增加了 1。图 5 示出了当购买订单 135 号已发货时对图 4 的表的更新。雷蒙德 / 处理中购买订单的“数量”和“计数”总数分别减少了 30 和 1，而雷蒙德 / 已发货购买订单的相同字段分别增加了 30 和 1。

[0036] 尽管图 4 和图 5 的更新相对简单，然而当多个编程线程同时访问同一的聚集数据表时会发生问题。图 6 示出了尝试同时对同一个表执行事务的两个程序线程。为简单起见，假定存在关于“城市”和“过程状态”的群集索引（即，数据行基于群集索引键而按顺序存储），从而允许以图 6 所示的次序从上至下地访问记录。在许多数据库环境中，多个记录作为访问表的单个事务的一部分而被修改。由于每一事务所需的处理开销，因此批处理多个记录通常更高效。要求在“全部或没有”的基础上执行事务也是一般惯例。换言之，没有记录被修改，直至完成了事务的所有部分为止。否则，将会破坏表的完整性。

[0037] 这些数据库处理约束可导致死锁。当程序线程 A 尝试在一个事务中更新三个记录时：更新雷蒙德 / 处理中和雷蒙德 / 已发货以反映购买订单 134 已发货，并更新西雅图 / 处理中以反映新购买订单 136，死锁在图 6 中发生。同时，程序线程 B 也尝试在一个事务中更新三个记录：更新西雅图 / 处理中和西雅图 / 已发货以反映购买订单 133 已发货，以及接下来更新雷蒙德 / 处理中以反映新购买订单 135。为了执行其事务，线程 A 在雷蒙德 / 处理中和雷蒙德 / 已发货上获得排它锁，以便在线程 A 更新这些记录的时候防止其他线程影响这些记录。线程 A 随后尝试在西雅图 / 处理中上获得排它锁。与此同时，线程 B 在记录西雅

图 / 处理中和西雅图 / 已发货上获得排它锁并进行修改，并且随后尝试在雷蒙德 / 处理中上获得排它锁。由于线程 A 已经锁定了雷蒙德 / 处理中，所以线程 B 不能完成其事务的第一部分。然而，线程 B 能够在线程 A 尝试锁定西雅图 / 处理中之前锁定西雅图 / 处理器和西雅图 / 已发货。线程 A 从而被阻止完成其事务的第二部分。事实上，每个线程都在等待另一线程结束，而且两个都不能完成其事务。一些系统将选择线程之一作为死锁牺牲品，退回该牺牲品的事务的任何部分完成的部分，并向该牺牲品报告错误。尽管这允许另一线程继续进行，且尽管牺牲品可重试其事务，但是系统性能仍然降级。具体而言，取消部分完成的事务浪费了处理资源和存储器，并且随后由于重复事务先前所完成的部分而导致了进一步的浪费。

[0038] 图 7 示出了根据本发明的至少一个实施例来如何防止死锁。将聚集值的表分割成各个单独的表。图 7 示出了两个分区（由粗线分开），并且附加部分的存在由垂直省略号来指示。每个单独的分区表 0、1 等包含相同的记录，但是这些记录中的值在分区之间通常将各不相同。在图 7 的示例中，分区 0 具有雷蒙德 / 已交付、雷蒙德 / 处理中、雷蒙德 / 已发货、西雅图 / 已交付、西雅图 / 处理中以及西雅图 / 已发货的记录。分区 1 也具有雷蒙德 / 已交付、雷蒙德 / 处理中、雷蒙德 / 已发货、西雅图 / 已交付、西雅图 / 处理中以及西雅图 / 已发货的记录。然而，分区 0 中的雷蒙德 / 已交付的“数量”和“计数”的值与分区 1 中雷蒙德 / 已交付“数量”和“计数”的值不同，等等。一次只准许单个程序线程访问特定分区。为了说明，图 6 中的线程 A 和线程 B 现在在图 7 中示出。在图 7 中，线程 A 在分区 0 上执行其事务，而线程 B 同时在分区 1 上执行其事务。因为这些线程不再竞争对同一记录的访问，因而避免了死锁。

[0039] 在本发明的使用 SQL SERVER™2000 RDBMS 软件来实现的一个实施例中，图 7 的分区表被实现成单独的 SQL 表，如图 8 所示。分区号作为附加字段（“分区 ID”）而被包括。该多分区聚集表由在图 2 的实例数据表上的触发器来维护。如本领域所公知的，“触发器”是由 INSERT（插入）、UPDATE（更新）或 DELETE（删除）语句自动地调用的特殊类型的 SQL 存储过程。无论在何时将新记录插入到图 2 的实例数据表中，触发器都会使得对图 8 的多分区聚集表的分区之一作出贡献。类似地，当更新实例数据表（图 2）中的记录时，多分区聚集表（图 8）的对应记录也被更新。

[0040] 在任何给定时间，图 8 的表中的单独分区将不具有关于公司 X 的完整信息。对实例数据表（图 2）的一个更新可造成分区 0 内的更新，而对该实例数据表的另一个更新可造成分区 1 内的更新，等等。此外，在实例数据表中创建记录可造成图 8 的一个分区的表中的记录的更新，且该同一实例数据记录的稍后更新可造成不同分区中的记录的更新。例如，将新购买订单 140 号（雷蒙德，数量＝ 200）添加到图 2 可造成图 8 中的 < 分区 ID ＝ 1>/< 城市＝雷蒙德 >/< 过程状态＝处理中 > 记录的“数量”和“计数”字段增加 200 和 1。然而，当购买订单 140 的商品已发货时，图 8 中的 < 分区 ID ＝ 3>/< 城市＝雷蒙德 >/< 过程状态＝处理中 > 记录的“数量”和“计数”字段可减少 200 和 1，且 < 分区 ID ＝ 3>/< 城市＝雷蒙德 >/< 过程状态＝已发货 > 记录的相同字段增加 200 和 1。因此，组合各分区以便提供公司 X 的完整的数据聚集表。具体而言，下面的 SQL 代码在至少一个实例使用：

[0041] CREATE VIEW<view_name>

[0042] AS

[0043] SELECT City,ProcessState,SUM(Quantity),SUM(Count)

[0044] FROM<multi_partition_table>

[0045] GROUP BY City,ProcessState

[0046] 第一个斜体名（“view_name（视图名）”）是在其中组合各单独的分区的视图的名称。第二个斜体名（“multi_partition_table（多分区表）”）是图 8 的多分区聚集表的名称。所得表（“view_name”）将是图 9 所示的形式。由于多分区聚集表的每个分区包含相对少量的记录，因此各分区可以相对快速地进行组合。事实上，概括了概要。在其他实施例中，每个分区可以实现成单独的表，并随后使用 SQL JOIN（加入）语句来加入到 CREATE VIEW（创建视图）命令。

[0047] 因而每个分区一次只由一个程序线程访问，一个线程必须拥有虚拟权标才能访问一个分区。在本发明的一个实施例中，通过对名为 Get_Mutex（获得互斥）的特殊存储过程的调用来获得该权标。如上所述，多分区聚集表由 SQL 触发器来维护。在可以更新多分区聚集表中的一行之前，触发器中的一个或多个指令必须指示要更新哪一行。作为这些指令的一部分，调用 Get_Mutex 存储过程。该 Get_Mutex 过程随后返回该表的分区 ID 列的值。当线程在具有该所返回的分区 ID 值的记录上执行事务时，阻止其他线程获得同一值，并且从而在分区正由另一线程更新时阻止对该分区的访问。

[0048] 参考先前示例，当最初在实例数据表（图 2）中创建购买订单 140 的记录时，激发触发器。该触发器随后：

[0049] 调用 Get_Mutex 并接收分区 ID 的值；

[0050] 确定该分区中的要被更新的行；

[0051] 标识多分区聚集表的、其中“分区 ID”等于 Get_Mutex 过程所返回的分区 ID 值、其中“城市”等于购买订单 140 的实例数据记录中的城市的值、并且其中“过程状态”等于“处理中”的行；

[0052] 将该行中的“数量”的值增加购买订单 140 的实例数据记录中的“数量”的值；以及

[0053] 将该行中的“计数”值加 1。

[0054] 作为进一步的说明，稍后更新购买订单 140 的已有实例数据表记录，以将“过程状态”从“处理中”改为“已发货”。触发器中的附加逻辑随后：

[0055] 调用 Get_Mutex 并接收分区 ID 的值；

[0056] 确定该分区中的要被更新的行；

[0057] 标识多分区聚集表的、其中“分区 ID”等于 Get_Mutex 过程所返回的值、

[0058] 其中“城市”等于购买订单 140 的实例数据记录中的城市值、并且其中“过程状态”等于处理中的行；

[0059] 将该行中的“数量”值减去购买订单 140 的实例数据记录中的“数量”值；

[0060] 将该行中的“计数”值减 1；

[0061] 标识多分区聚集表的、其中“分区 ID”等于 Get_Mutex 过程所返回的值、其中“城市”等于订单 140 的实例数据记录中的城市值、并且其中“过程状态”等于已发货的行；

[0062] 将该行中的“数量”值增加购买订单 140 的实例数据记录中的“数量”值；以及

[0063] 将该行中的“计数”值加 1。

[0064] 当购买订单 140 过程状态从已发货改为已交付时，类似的逻辑将标识并修改多分区聚集表中的适当的行。

[0065] Get_Mutex 的一个实现在图 10 中示出。首先，创建名为 RTA_Mutex 的单列表（“create table RTA_Mutex(创建 RTA_Mutex)”）。RTA_Mutex 表具有与图 8 的多分区聚集表中的分区数相同的行数；接着为每个字段分配分区 ID 值之一。在图 10 的示例中，假定图 8 的表包括 10 个分区（即，分区 ID 具有从 0 到 9 的整数值）。接下来，创建 Get_Mutex 存储过程。在声明局部变量 @par 之后，向局部 @par 变量分配来自 RTA_Mutex 表中的各行之一的值。具体而言，“select(选择)”语句将来自 RTA_Mutex 表的其中分区 ID 等于“@@spid% 10”的分区 ID 的值分配给 @par。系统函数 @@spid 返回当前用户进程的服务进程标识符。换言之，该 @@spid 函数返回标识调用 Get_Mutex 存储过程的程序线程的数字。模算术运算符（“%”）随后返回程序线程标识符除以 10 的余数。该过程随后找出 RTA_Mutex 的分区 ID 的值等于该余数的行，并使用排它锁锁定提示（“(xlock)”）来锁定该行。该锁定的 RTA_Mutex 行中的分区 ID 的值随后作为 Get_Mutex 存储过程的结果而被返回。因为该行上的排它锁，所以没有其他程序线程被允许访问 RTA_Mutex 表的该行，因此直到先前获得该分区 ID 值的事务完成为止都不能获得该分区 ID 值。

[0066] 作为说明，具有标识符 231 的程序线程调用 Get_Mutex。231 除以 10 的余数是 1；从而 Get_Mutex 返回给该程序线程的值是 RTA_Mutex 中的分区 ID 等于 1 的字段的值。在从 Get_Mutex 返回了值之后，线程 231 收能够更新图 8 中分区 1 的各行。当线程 231 更新分区 1 时，程序线程 161 调用 Get_Mutex。然而，由于线程 231 的事务没有完成，因此线程 161 不能够访问 RTA_Mutex 的持有分区 ID 的值为 1 的行。线程 161 随后排队，直到必需的 RTA_Mutex 行上的排它锁被释放（即，线程 231 的事务完成）时为止。如果附加线程尝试访问该 RTA_Mutex 行，则它们也在先进先出（或其他）的基础上排队。值得注意的是，线程 161 不会继续进行，直至从它对 Get_Mutex 的调用返回了值为止，并且线程 161 因而不在线程 231 访问分区 1 时尝试访问分区 1。当线程 161 在队列中等待释放 xlock 时，线程 154 调用 Get_Mutex。由于当前在 RTA_Mutex 的其中分区 ID ＝ 4 的行上没有锁，因此 Get_Mutex 向线程 154 返回值 (4)。

[0067] 图 10 只是可实现 Get_Mutex 过程的方式的一个示例。作为另一示例，尝试访问一个分区的所有线程可进入先进先出的队列。该队列中的每个线程随后可被分配第一可用分区。

[0068] 在一个实施例中，分区的数量至少等于且优选地大于数据库服务器上的处理器的数量。

[0069] 如上所示，实例数据表（图 2）的大小将会随着时间的推移接收到越来越多的购买订单而不断增加。然而，图 9 的聚集表（或者图 8 的多分区聚集表）的大小不会增加，除非增加附加聚集字段（例如，雷蒙德 / 已拒绝和西雅图 / 已拒绝）。尽管表的大小不会增加，然而“数量”和“计数”列的值将会增加。通常，组织仅仅需要某个时“窗”内的事件的聚集数据。例如，公司 X 可能需要当前正在处理的购买订单的聚集数据、当前正在发货的订单的聚集数据、以及已在最近 24 小时内交付的订单的聚集数据。此外，公司 X 可能想要基于例如事件在一天中发生的时间来对数据进行排序。参考图 11，公司 X 管理人员们希望了解在最近的上午 8 点钟期间且尚未发货（在该示例中是 0）、在最近的上午 9 点钟期间（在该示

例中也是 0)、在最近的上午 10 点钟期间 (2 和 1) 等等在雷蒙德和西雅图有多少购买订单变得正在处理中。类似地,这些管理人员们还想了解有多少购买订单已在上午 8:00 发货但是尚未交付、有多少购买订单已在上午 9:00 发货但尚未交付,等等。这些管理人员们还想知道在这些时间段期间有多少购买订单已从它们的位置处交付。然而,这意味着聚集表每小时都增长。与“处理中”或者“已发货”的且最终将变成“已交付”的购买订单不同,“已交付”购买订单不转到另一状态。换言之,一旦购买订单在特定一天中的特定几点钟中变成“已交付”,则记录将持久存储在聚集表中的该时刻 / 天,除非采取某些动作。随着时间的推移,该表将因而变得巨大且查询缓慢。另一方面,这些管理人员们对了解在较长时间段 (例如,上周,等等) 期间已从一个地点交付的购买订单的总数不感兴趣。尽管该信息对于某些目的而言也许有用,但其相对较少需要。

[0070] 图 12 示出了在本发明的至少一个实施例中在不活动的购买订单 (例如,已交付的购买订单) 的数据到了一定年龄时可如何将该数据从实时聚集表中清除。为简单起见,只示出了单个分区。然而,并且如本领域技术人员将基于以下描述所明白的,可结合上述多分区实施例来实现图 12 的实施例。

[0071] 如在图 12 中看到的,“时间片”和“时刻”列已经被添加到图 9 的实时聚集表中。“时间片”是更新该实例数据表 (图 2) 以反映一个实例已完成的服务器日期和时间 (精确到小时)。在该示例中,当订购的商品已交付给客户时,购买订单就完成了。“时刻”是感兴趣的事件在当天发生的钟点。在该示例中,“时间片”的时间组成与完成记录的“时刻”相同,但不必是这种情况。

[0072] 如先前示例一样,图 12 的实时聚集表的数据由触发器修改,当该实例数据表的数据记录被插入或更新时该触发器就激发。“时间片”列的值由触发器根据两个规则自动生成。如果一个记录预期有更多更新 (例如,商品尚未交付),则“时间片”为 NULL(空)。否则,生成表示当前时间 (精确到小时) 和日期的数字。值得注意的是,已完成的购买订单的非空“时间片”值并不阻止已完成的购买订单的数据的聚集。因为“时间片”的时间组成只精确到小时,所以在同一个小时时间段期间从同一个仓库交付的购买订单可聚集在单个记录中。例如,购买订单 126 和 128(图 2) 在分别在下午 3:10 和 3:05 交付。如在图 12 中看到的,1 月 20 日下午 3 点 /3/ 西雅图 / 已交付示出了“计数”2 和“数量”790,这将图 2 中的购买订单 126 和购买订单 128 的数据记录联系起来。

[0073] 每当触发器修改图 12 的实时聚集表时,该触发器还确定该表中的已完成实例聚集数据中的任一个是否是陈旧的,即不再需要的。在该示例中,假定超过 24 小时以前完成的实例的聚集数据是陈旧的且应该从聚集数据表中移除。因此,每次触发器修改图 12 的表时,该触发器删除具有时间片期间中的最近时间超过当前时间的 24 小时之前的“时间片”值的记录。如果触发器在 1 月 21 日下午 1:05 更新了图 12 的表,则 1 月 20 日下午 12 点 /12/ 西雅图 / 已交付的记录就会被删除,因为该记录中的所有数据与超过 24 小时之前所完成的购买订单相对应。在至少一个实施例中,图 12 的表被显示在向用户隐藏“时间片”数据的视图中。

[0074] 图 13 是示出了在多分区中维护实时数据聚集表 (例如图 12) 的触发器的逻辑的流程图。具体而言,图 13 的流程图示出了在更新分区中的一个或多个聚集数据记录的事务期间,每个单独的程序线程中的触发器所遵循的逻辑。每个线程更新一个单独分区,每个分

区都具有与图 12 的表相似的格式。在执行期间，触发器访问由 SQL 数据库服务器在存储器（例如 RAM）中维护的两个名为“已插入”和“已删除”的系统表。“已插入”和“已删除”表由数据库服务器自动生成，并且临时储存来自在先前尝试将记录插入到实例数据表或更新实例数据表的现有记录（比如图 2）时受影响的行的数据的副本。具体而言，“已插入”表包括插入到实例数据表的一行中的值，“已删除”表包括在实例数据表的更新的行中被替换的值。

[0075] 开始后，在框 102，触发器首先获得权标。如上所述，触发器通过对诸如 Get_Mutex 之类的过程的调用来获得权标。在框 104，触发器随后确定是否正在更新实例数据记录。具体而言，触发器确定“已删除”表是否为空。如果“已删除”表不为空，则在框 106，使用“已删除”表中的“数量”值来减少分区的适当记录的“数量”值。具体而言，触发器标识分区中、其中“时刻”等于实例数据记录被更新到其以前的“过程状态”值期间的钟点、“城市”等于已更新实例数据记录的城市、并且其中“过程状态”等于该已更新记录的先前过程状态的记录。该记录的“计数”值也减 1。在框 108，触发器随后确定该分区中的任何记录是否具有“时间片”值 NULL 和“计数”值 0。如果是，则删除这些记录。

[0076] 在框 110，触发器确定正被更新的（或插入的）实例数据记录是否完成。换言之，该触发器决定是否预期有进一步的更新。这可以用各种方式来执行。例如，可以将“完成”列添加到每个实例记录中并且当更新完成时将其设定成“1”。在其他实施例中，各种过程状态值（例如，“已交付”或“已拒绝”）可以通过信号来通知实例的过程完成。如果实例数据记录没有完成（即，关于正被更新的购买订单实例的过程仍然活动且预期有进一步的更新），则在框 112，使用来自“已插入”表的“数量”值来增加聚集表分区中的、其中“时间片”等于 NULL、“时刻”等于实例数据记录被更新为其当前”过程状态”值的钟点、“城市”等于已更新的实例数据记录的城市、且“过程状态”等于已更新的实例数据记录的当前过程状态值的记录的“数量”值。该分区记录中的“计数”值也增加 1。触发器随后终止。

[0077] 如果在框 110，触发器确定正被更新的实例数据记录已完成，则在框 114 生成“时间片”值。在框 116，触发器删除具有处于所选择的时窗之外（例如 24 小时之前的）的“时间片”值的所有记录。触发器随后标识（如果如有必要则创建）其中“时间片”的值等于刚生成的“时间片”值、其中“时刻”等于完成的钟点、其中“城市”等于已更新的实例数据记录的城市、并且其中“过程状态”等于来自已完成的实例数据记录（在该示例中是“已交付”）的值的分区记录。随后使用来自“已插入”表的“数量”值来增加所标识（或创建）的分区记录的“数量”值。该分区记录的“计数”值也增加 1。触发器随后终止。

[0078] 在其他实施例中，图 13 的触发器逻辑可以被修改以适应实例数据表中的多个记录的更新。代替在框 112 和 118 之后结束，触发器将确定是否要更新附加实例记录来作为实例数据表的批量更新的一部分。如果是，则触发器逻辑将返回框 104。如果否，则触发器随后终止。在其他实施例中，时窗可以由用户来调整。

[0079] 虽然使用假设的公司类型作为示例描述了本发明，但是应该记住，本发明不限于特定类型的公司、组织或活动。此外，本发明也不限于其中通过对数据库记录中的各字段的值进行求和来聚集各个值的实现。仅作为一个示例，可以创建其中聚集值表示数据库记录中各字段的值的平均值的聚集表。因此，虽然描述了执行本发明的各具体示例，但本领域技术人员将明白，存在着落入所附权利要求书所阐明的本发明的精神和范围内的、上述系统

和技术的众多变型和置换。这些和其他修改都处于由所附权利要求书所限定的本发明的范围之中。

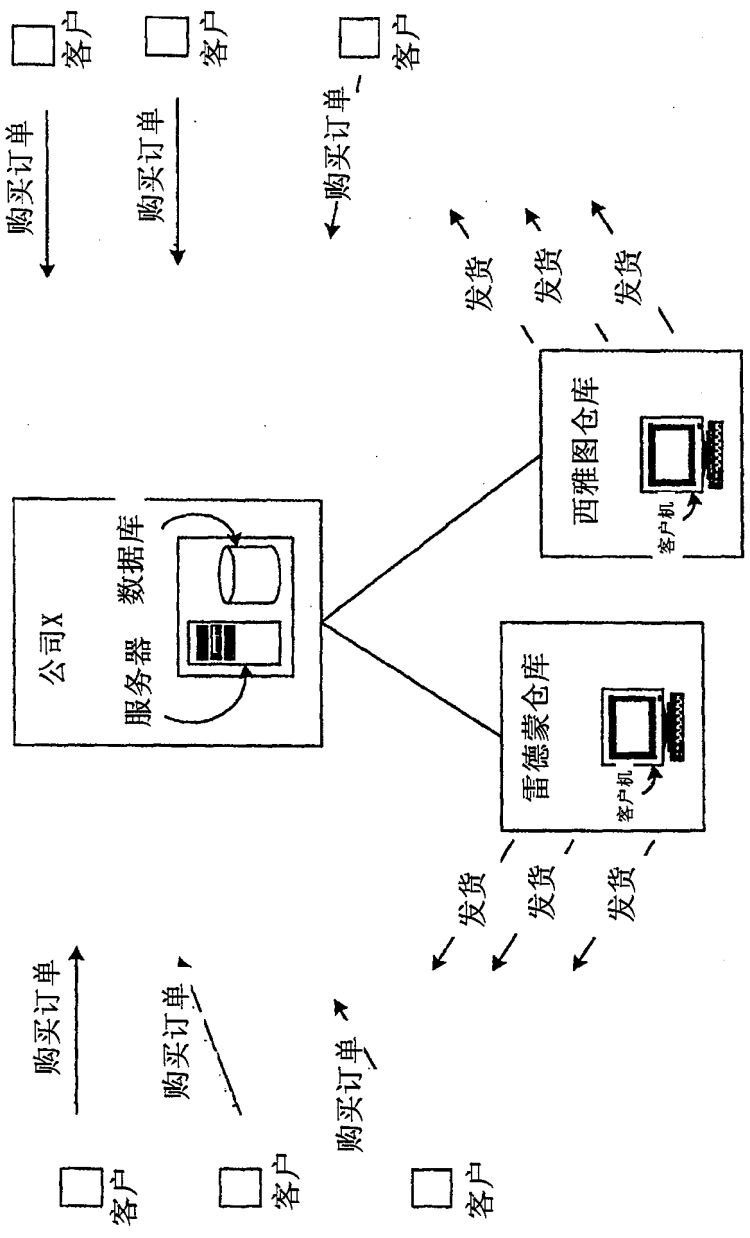

客户
客户
客户
购买订单
购买订单
购买订单
公司X
服务器
数据库
发货
发货
发货
西雅图仓库
客户机
雷德蒙仓库
客户机
发货
发货
发货
购买订单
购买订单
购买订单
客户
客户
客户

图 1

PO#	接收时间	城市	数量	发货时间	交付时间	过程状态
...	...	...	...	...	...	...
123	上午8:00	西雅图	150	上午8:24	下午12:45	已交付
124	上午8:30	西雅图	234	上午8:45	下午1:20	已交付
125	上午8:35	雷蒙德	87	上午9:05	下午2:30	已交付
126	上午8:45	西雅图	450	上午9:20	下午3:10	已交付
127	上午8:55	雷蒙德	200	上午9:30	<NULL>	已发货
128	上午8:57	西雅图	340	上午9:20	下午3:05	已交付
129	上午9:12	西雅图	120	上午9:45	<NULL>	已发货
130	上午9:30	雷蒙德	25	上午10:15	<NULL>	已发货
131	上午9:45	西雅图	250	上午10:35	<NULL>	已发货
132	上午10:00	雷蒙德	100	<NULL>	<NULL>	处理中
133	上午10:15	西雅图	230	<NULL>	<NULL>	处理中
134	上午10:25	雷蒙德	45	<NULL>	<NULL>	处理中
...	...	...	...	...	...	...

图 2

城市	过程状态	数量	计数
雷蒙德	已交付	87	1
雷蒙德	处理中	145	2
雷蒙德	已发货	225	2
西雅图	已交付	1174	4
西雅图	处理中	230	1
西雅图	已发货	370	2

图 3

接收到来自雷德蒙市的30个集装箱的新购买订单135号

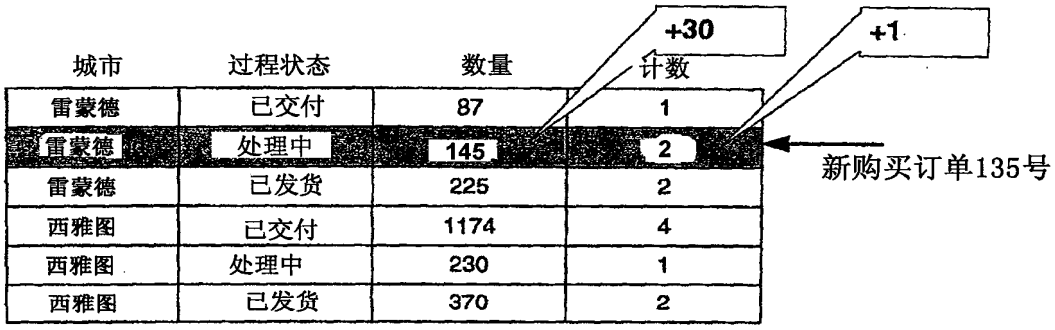


图 4

购买订单135号已发货

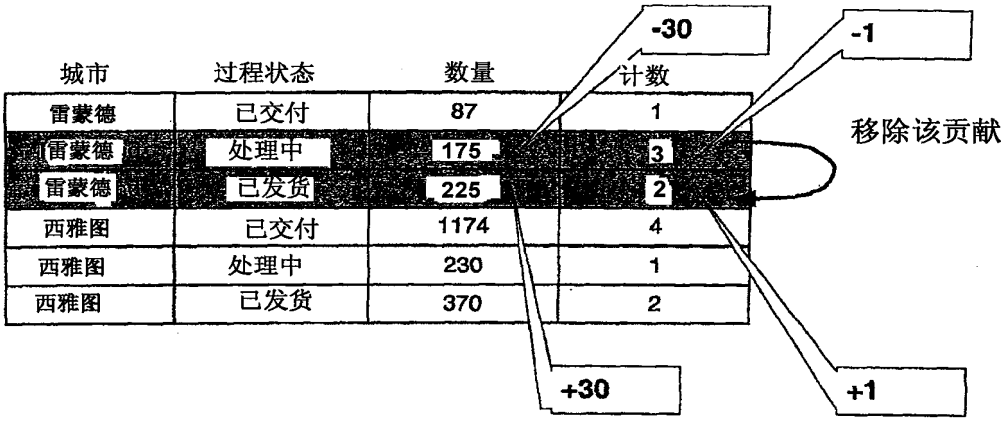


图 5

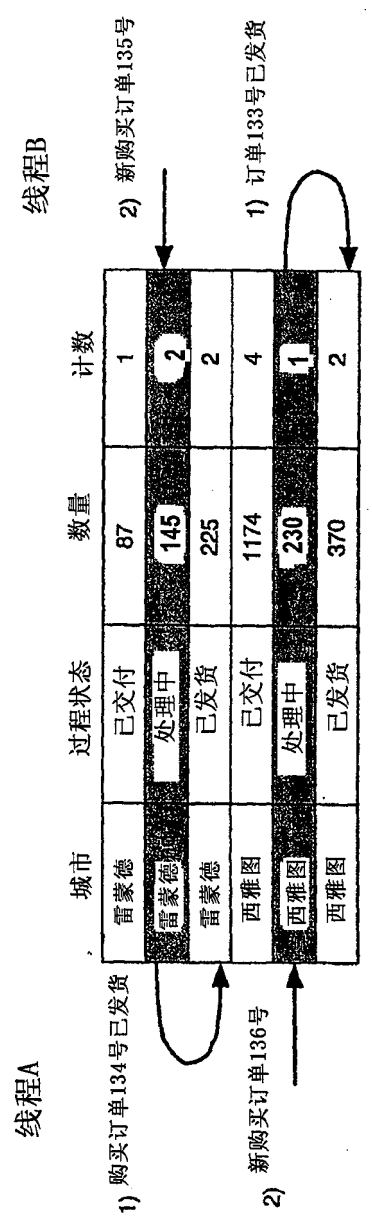

线程B
2) 新购买订单135号
1) 订单133号已发货
城市
过程状态
数量
计数
雷蒙德
已交付
87
1
雷蒙德
处理中
145
2
雷蒙德
已发货
225
2
西雅图
已交付
1174
4
西雅图
处理中
230
1
西雅图
已发货
370
2
线程A
1) 购买订单134号已发货
2) 新购买订单136号

图 6

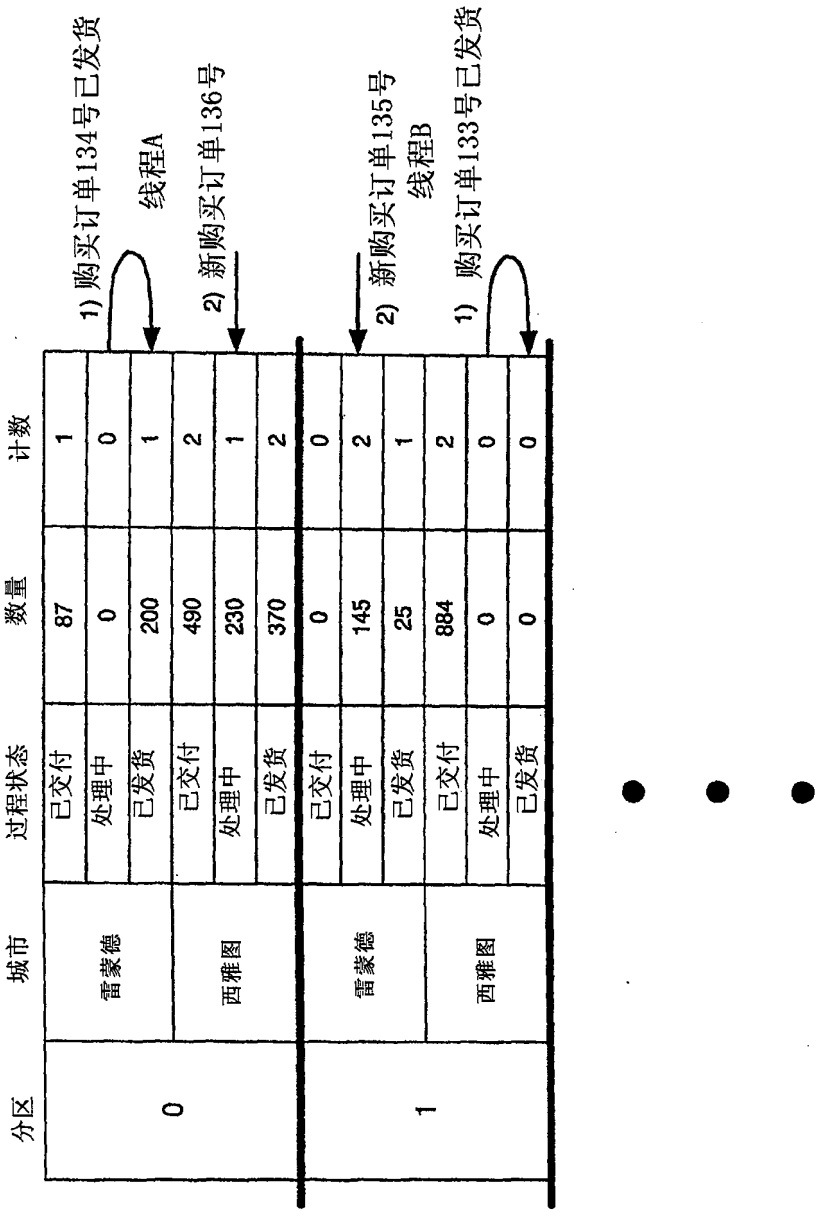

分区
城市
过程状态
数量
计数
0
雷蒙德
已交付
87
1
处理中
0
0
已发货
200
1
西雅图
已交付
490
2
处理中
230
1
已发货
370
2
1
雷蒙德
已交付
0
0
处理中
145
2
已发货
25
1
西雅图
已交付
884
2
处理中
0
0
已发货
0
0
1) 购买订单134号已发货
线程A
2) 新购买订单136号
2) 新购买订单135号
线程B
1) 购买订单133号已发货

图 7

分区ID	城市	过程状态	数量	计数
0	雷蒙德	已交付	...	...
0	雷蒙德	处理中	...	...
0	雷蒙德	已发货	...	...
0	西雅图	已交付	...	...
0	西雅图	处理中	...	...
0	西雅图	已发货	...	...
1	雷蒙德	已交付	...	...
1	雷蒙德	处理中	...	...
1	雷蒙德	已发货	...	...
1	西雅图	已交付	...	...
1	西雅图	处理中	...	...
1	西雅图	已发货	...	...
2	雷蒙德	已交付	...	...
2	雷蒙德	处理中	...	...
2	雷蒙德	已发货	...	...
2	西雅图	已交付	...	...
2	西雅图	处理中	...	...
2	西雅图	已发货	...	...
3	雷蒙德	已交付	...	...
3	雷蒙德	处理中	...	...
...	...	...	...	...

图 8

城市	过程状态	数量	计数
雷蒙德	已交付	87	1
雷蒙德	处理中	145	2
雷蒙德	已发货	225	2
西雅图	已交付	1174	4
西雅图	处理中	230	1
西雅图	已发货	370	2

图 9

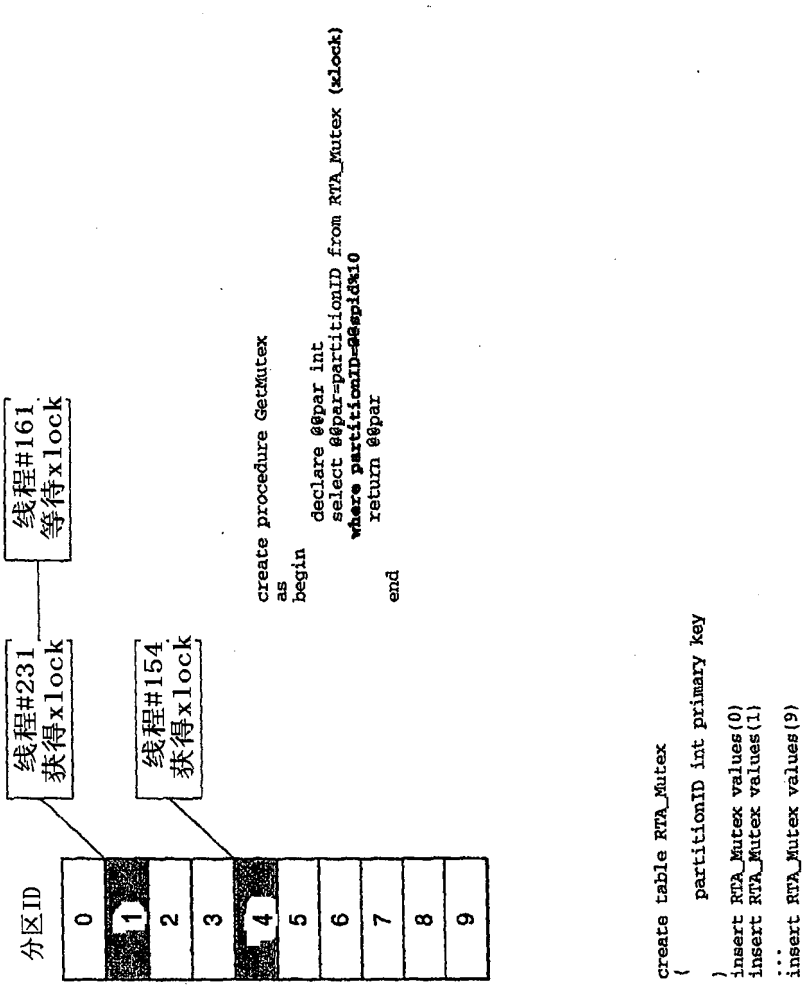


图 10

时刻	城市	过程状态	数量	计数
8	雷蒙德	已交付	87	1
		已发货	200	1
	西雅图	已交付	1174	4
9	雷蒙德	已发货	25	1
	西雅图	已发货	370	2
10	雷蒙德	处理中	145	2
	西雅图	处理中	230	1

图 11

时间片	时刻	城市	过程状态	数量	计数
<NULL>	8	雷蒙德	已发货	200	1
	9	雷蒙德	已发货	25	1
		西雅图	已发货	370	2
	10	雷蒙德	处理中	145	2
		西雅图	处理中	230	1
1月20日，下午12点	12	西雅图	已交付	150	1
1月20日，下午1点	1	西雅图	已交付	234	1
1月20日，下午2点	2	雷蒙德	已交付	87	1
1月20日，下午3点	3	西雅图	已交付	790	2

图 12

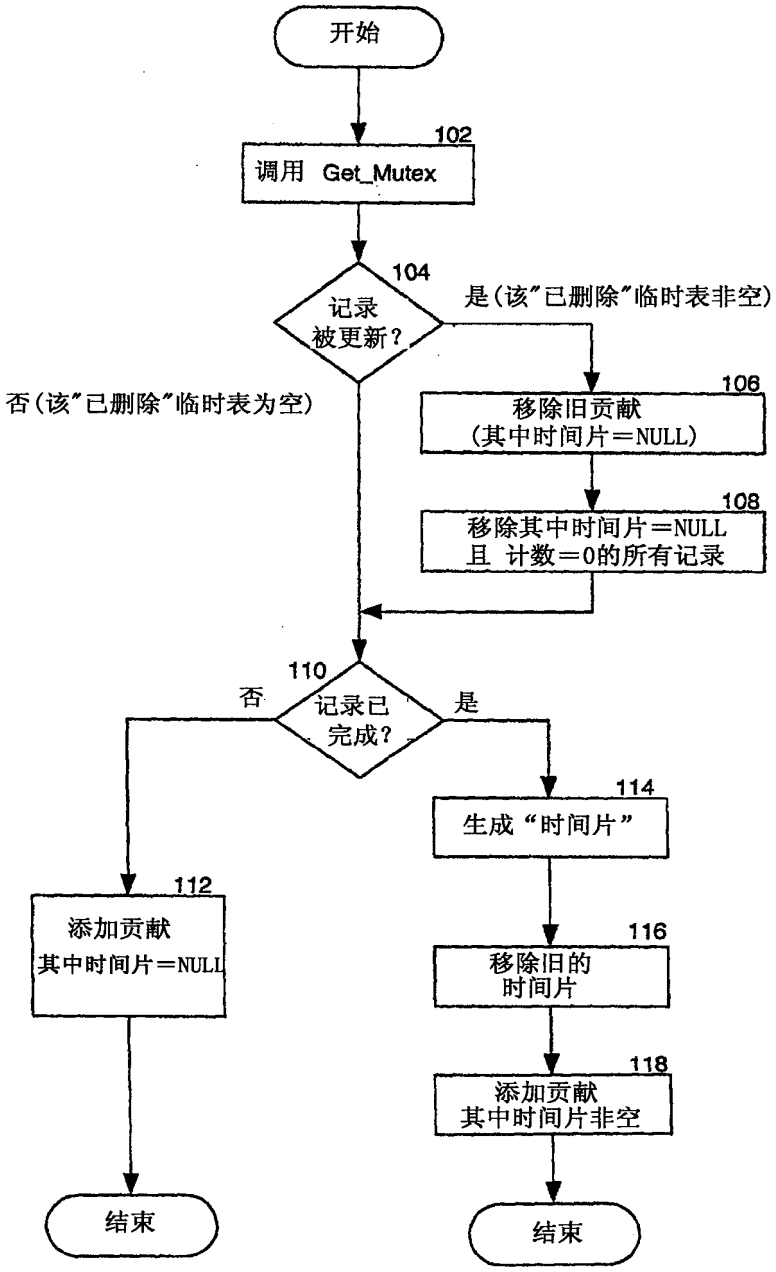

开始
102
调用 Get_Mutex
104
记录
被更新?
是(该"已删除"临时表非空)
否(该"已删除"临时表为空)
106
移除旧贡献
(其中时间片=NULL)
108
移除其中时间片=NULL
且 计数=0的所有记录
110
记录已
完成?
否
是
112
添加贡献
其中时间片=NULL
结束
114
生成“时间片”
116
移除旧的
时间片
118
添加贡献
其中时间片非空
结束

图 13