[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/30 (2006.01)
G06Q 30/00 (2006.01)

[12] 发明专利说明书

专利号 ZL 200410058914.X

[45] 授权公告日 2008 年 11 月 26 日

[11] 授权公告号 CN 100437567C

[22] 申请日 2004.7.19
[21] 申请号 200410058914.X
[30] 优先权
[32] 2003.9.26 [33] US [31] 10/670,276
[73] 专利权人 微软公司
地址 美国华盛顿州
[72] 发明人 G·B·齐考德罗夫 R·Z·杰森
E·A·里尔
[56] 参考文献
US6408292B1 2002.6.18
US2002/0184255A1 2002.12.5
US6477525B1 2002.11.5
CN1339134A 2002.3.6
CN1438596A 2003.8.27
审查员 从 珊

[74] 专利代理机构 上海专利商标事务所有限公司
代理人 陈 斌

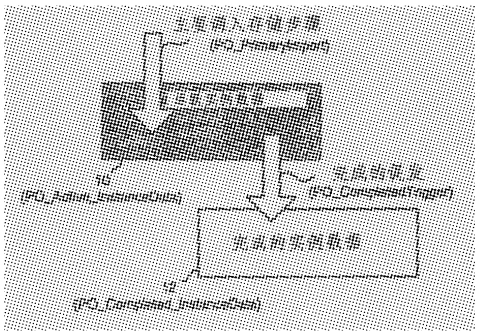
权利要求书 7 页 说明书 17 页 附图 7 页

[54] 发明名称

维护有关活动的多个实例的信息的方法

[57] 摘要

解决了与维护有关活动的多个实例的信息的问题。为对应于一个组织机构活动的活跃实例的数据和对应于其活动的非活跃实例的数据分别维护数据库表。可以为对应于活动的非活跃实例的数据维护多个数据库表。在另一方面，处理来自活跃实例表以及一个或多个非活跃实例表的数据以生成组合分析数据。



1．一种维护有关一个活动的多个实例的信息的方法，每个实例有一个活跃状态或一个非活跃状态，在活跃状态中，该实例是未完成的并且有关该实例的信息将被修改，在非活跃状态中，该实例是完成的并且有关该实例的信息将不会被修改，其特征在于，所述方法包括：

为处于活跃状态的多个实例中的每一个在第一数据库表中创建一条记录，每条记录包含针对多个数据类型中的每一个的字段，在每个活跃实例记录的一个或多个字段中，有用来指示活跃状态的值；

对于达到完成的多个实例的记录，给所述一个或多个字段赋值指示处于非活跃状态；

从所述第一数据库表中删除其在所述一个或多个字段中有指示其处于非活跃状态的值的实例的记录；以及

为从第一数据库表中删除的记录在第二数据库表中创建一个相应的记录。

2．如权利要求 1 所述的方法，其特征在于，所述第二数据库表的记录被创建后不做更新。

3．如权利要求 1 所述的方法，其特征在于，第一数据库表记录中的数据在删除时被复制到对应的第二数据库表记录中。

4．如权利要求 1 所述的方法，其特征在于，第一数据库表记录中的全部数据在删除时被完全复制到对应的第二数据库表记录中。

5．如权利要求 1 所述的方法，其特征在于，所述第一数据库表中只包含处于活跃状态的实例的记录。

6．如权利要求 1 所述的方法，其特征在于，所述一个或多个字段包括一个标志，如果实例是活跃的，则该标志取第一值，如果实例是非活跃的，则该标志取第二值。

7．如权利要求 1 所述的方法，其特征在于，所述一个或多个字段包括一个包含非活跃实例记录的实例完成时间的字段。

8．如权利要求 1 所述的方法，其特征在于，还包括：

创建包括所述第一数据库表和第二数据库表的视图。

9．如权利要求 1 所述的方法，其特征在于，还包括：

创建第三数据库表；

停止在所述第二数据库表中创建记录；以及

对于每个在创建第三数据库表后从第一数据库表中删除的记录，在第三数据库表中创建相应的记录。

10．如权利要求 9 所述的方法，其特征在于，所述创建第三数据库表包括：在预设时间段过后创建第三数据库表。

11．如权利要求 9 所述的方法，其特征在于，还包括：

删除所述第二数据库表。

12．如权利要求 9 所述的方法，其特征在于，还包括：

重命名所述第二数据库表。

13．如权利要求 9 所述的方法，其特征在于，还包括：

创建后续数据库表；

在后续表创建后，终止在之前创建的表中创建记录；以及

对于在创建最近一个表之后但在创建另一后续表之前从第一数据库表中删除的每一个记录，为其在所述最近创建的表中创建相应的记录。

14．如权利要求 13 所述的方法，其特征在于，还包括：

在保持所述第二数据库表一段选择的时间后，删除所述第二数据库表，在保持所述第三数据库表一段选择的时间后，删除所述第三数据库表，在每个后续的数据库表已经被保持一段选择的时间后，删除所述每个后续的数据库表。

15．如权利要求 13 所述的方法，其特征在于，还包括：

创建包括未删除表的视图。

16．如权利要求 13 所述的方法，其特征在于，所述创建一个后续的数据库表包括将先前创建的表重命名。

17．如权利要求 1 所述的方法，其特征在于，还包括：

根据所述第一数据库表和第二数据库表中的数据生成分析数据。

18．如权利要求 17 所述的方法，其特征在于，所述生成分析数据还包括：

为第一数据库表中的记录生成第一在线分析处理 OLAP 方块，

为第二数据库表中的记录生成第二在线分析处理 OLAP 方块，

将所述第一方块和所述第二方块组合成虚拟在线分析处理 OLAP 方块。

19．如权利要求 18 所述的方法，其特征在于，所述生成第二在线分析处理 OLAP 方块包括从所述第二数据库表中获取记录，还包括：

对于第二数据库表中的每个记录赋予一个唯一的递增的标识符值；

存储被获取以生成所述第二在线分析处理OLAP方块的最后一个记录的递增标识符值；

随后从第二数据库表中获取其它的记录，所述其它记录不被处理来生成第二在线分析处理 OLAP 方块；以及

基于所述其它记录更新第二在线分析处理 OLAP 方块。

20．如权利要求 19 所述的方法，其特征在于：

所述生成第二在线分析处理OLAP方块包括将第二数据库表记录的数据输入到一个星型模式中、并且在生成所述第二在线分析处理 OLAP 方块之后保存上述星型模式；以及

所述更新第二在线分析处理 OLAP 方块包括修改所述已保存的星型模式、使用来自所述其它第二数据库表记录的数据、并根据修改后的星型模式重新生成第二在线分析处理 OLAP 方块。

21.如权利要求 18 所述的方法，其特征在于，还包括：

确定一后续的未被处理以形成第二在线分析处理OLAP方块的非活跃实例记录集；以及

基于所述后续的集，更新第二在线分析处理 OLAP 方块。

22．如权利要求 21 所述的方法，其特征在于，还包括：

针对活跃实例数据库记录的后续集，生成后续的第一在线分析处理 OLAP 方块；以及

将所述后续的第一在线分析处理OLAP方块和已更新的第二在线分析处理 OLAP 方块组合成后续虚拟在线分析处理 OLAP 方块，其中：

所述活跃实例数据库记录的后续集包括未被处理以生成第一在线分析处理 OLAP 方块的活跃实例记录，且

所述非活跃实例数据库记录的后续集包括这样一些记录，这些记录与在所述第一在线分析处理 OLAP 方块被生成时处于活跃状态的实例相关，而相关的活跃实例记录已被处理用以生成所述第一在线分析处理 OLAP 方块。

23．一种维护有关一个活动的多个实例的信息的计算机系统，所述计算机系统包括：

用于对于一个活动的多个实例中的每个实例，在第一数据库表中为其创建一条记录的装置，其中：

每个实例有一个活跃状态或一个非活跃状态，在活跃状态中，该实例是未完成的并且有关该实例的信息将被修改，在非活跃状态中，该实例是完成的并且有关该实例的信息将不会被修改，

所述第一数据库表中的记录为处于活跃状态的实例创建，且

所述第一数据库表中的每条记录包含针对多个数据数据类型中的每一个的字段，每个活跃实例记录中的一个或多个字段具有指示活跃状态的值；

用于对于达到完成的多个实例的记录，给所述一个或多个字段赋于指示非活跃状态的值的装置；

用于从第一数据库表中删除多个在其一个或多个字段中的值指示其处于非活跃状态的实例的记录的装置；以及

用于对于从第一数据库表中删除的记录，在第二数据库表中创建相应的记录的装置。

24．如权利要求23所述的计算机系统，其特征在于，所述第二数据库表中的记录被创建后不被更新。

25．如权利要求23所述的计算机系统，其特征在于，第一数据库表记录中的数据在其被删除时被复制到对应的第二数据库表记录中。

26．如权利要求23所述的计算机系统，其特征在于，第一张表中记录的全部数据在被删除时被完全复制到对应的第二数据库表记录中。

27．如权利要求23所述的计算机系统，其特征在于，所述第一数据库表只包含处于活跃状态的实例的记录。

28．如权利要求23所述的计算机系统，其特征在于，所述一个或多个字段包括一个标志，如果实例是活跃的，则该标志取第一值，如果实例是非活跃的，则该标志取第二值。

29．如权利要求23所述的计算机系统，其特征在于，所述一个或多个字段包括一个包含非活跃实例记录的实例完成时间的字段。

30．如权利要求23所述的计算机系统，其特征在于，还包括：

用于创建包括所述第一数据库表和第二数据库表的视图的装置。

31．如权利要求23所述的计算机系统，其特征在于，还包括：

用于创建第三数据库表的装置；

用于终止在第二数据库表中创建记录的装置；以及

用于对于在创建第三数据库表后从第一数据库表中删除的每个记录，在第

三数据库表中创建相应的记录的装置。

32．如权利要求31所述的计算机系统，其特征在于，所述用于创建第三数据库表的装置包括：用于在预设的时间段过后创建第三数据库表的装置。

33．如权利要求31所述的计算机系统，其特征在于，还包括：

用于删除第二数据库表的装置。

34．如权利要求31所述的计算机系统，其特征在于，还包括：

用于将第二数据库表重命名的装置。

35．如权利要求31所述的计算机系统，其特征在于，还包括：

用于创建后续数据库表的装置；

用于在后续表创建后，终止在之前创建的表中创建记录的装置；以及

用于对于在创建最近一个表之后但在创建另一后续表之前从第一数据库表中删除的每个记录，为其在所述最近创建的表中创建一个相应的记录的装置。

36．如权利要求35所述的计算机系统，其特征在于，还包括：

用于在保持所述第二数据库表一段选择的时间后，删除所述第二数据库表的装置；

用于在保持所述第三数据库表一段选择的时间后，删除所述第三数据库表的装置；

以及用于在每个后续的数据库表已经被保持一段选择的时间后，删除所述每个后续的数据库表的装置。

37．如权利要求35所述的计算机系统，其特征在于，还包括：

用于创建一个包括未删除表的视图的装置。

38．如权利要求35所述的计算机系统，其特征在于，所述用于创建后续数据库表的装置包括用于将先前创建的表重命名的装置。

39．如权利要求23所述的计算机系统，其特征在于，还包括：

用于基于第一和第二数据库表中的数据生成分析数据的装置。

40．如权利要求39所述的计算机系统，其特征在于，所述用于生成分析数据的装置还包括：

用于为第一数据库表中的记录生成第一在线分析处理OLAP方块的装置，

用于为第二数据库表中的记录生成第二在线分析处理OLAP方块的装置，

用于将所述第一第二方块组合成虚拟在线分析处理OLAP方块的装置。

41．如权利要求40的计算机系统，其特征在于，所述用于生成第二个在线分析处理OLAP方块的装置包括用于从第二数据库表中获取记录的装置，还包括：

用于对于第二数据库表中的每个记录赋予一个唯一的递增的标识符值的装置；

用于存储被获取以生成所述第二在线分析处理OLAP方块的最后一个记录的递增标识符值的装置；

用于随后从第二数据库表中获取其它的记录，所述其它记录不被处理来生成第二在线分析处理OLAP方块的装置；以及

用于基于所述其它记录更新第二在线分析处理OLAP方块的装置。

42．如权利要求41所述的计算机系统，其特征在于：

所述用于生成第二在线分析处理OLAP方块的装置包括用于将第二数据库表的记录数据输入一个星型模式中、并且在生成该第二在线分析处理OLAP方块之后保存所述星型模式的装置，以及

所述用于更新第二在线分析处理OLAP方块的装置包括用于修改已存储的星型模式、使用来自所述其它第二数据库表记录的数据、及基于所修改的星型模式重新生成第二在线分析处理OLAP方块的装置。

43．一种用于递增地生成活动实例的分析数据的方法，每个实例具有一个处于其中时有关该实例的信息将被修改的活跃状态，或者一个处于其中时有关该实例的信息将不被修改的非活跃状态，所述方法包括：

通过处理与处于活跃状态的实例相关的最初的数据库记录集，生成第一在线分析处理OLAP方块；

通过处理与处于非活跃状态的实例相关的最初的数据库记录集，生成第二在线分析处理OLAP方块；以及

将所述第一和第二方块组合成虚拟在线分析处理OLAP方块。

44．如权利要求43所述的方法，其特征在于，还包括：

确定一后续的未被处理以形成第二在线分析处理OLAP方块的非活跃实例记录集；以及

基于所述后续的集，更新第二在线分析处理OLAP方块。

45．如权利要求44所述的方法，其特征在于，还包括：

针对活跃实例数据库记录的后续集，生成后续的第一在线分析处理OLAP

方块；以及

将所述后续的第一在线分析处理OLAP方块和已更新的第二在线分析处理OLAP方块组合成后续虚拟在线分析处理OLAP方块，其中：

所述活跃实例数据库记录的后续集包括未被处理以生成第一在线分析处理OLAP方块的活跃实例记录，且

所述非活跃实例数据库记录的后续集包括这样一些记录，这些记录与在所述第一在线分析处理OLAP方块被生成时处于活跃状态的实例相关，而相关的活跃实例记录已被处理用以生成所述第一在线分析处理OLAP方块。

维护有关活动的多个实例的信息的方法

(1)技术领域

本发明涉及监测企业或其他组织机构工作流程的方法和计算机系统。更具体地，本发明涉及用于查看有关一个活动的多个实例的信息并用于维护该信息的方法。

(2)背景技术

计算机——尤其是计算机数据库应用程序——被企业和其他机构用来监测和记录有关组织活动信息。通常，机构会有各式各样必须执行的处理过程和活动，并且这些过程和活动会经常性的重现。确实，对一个组织来说，在任何特定时间有众多处于不同完成阶段的活动的实例是很普遍的现象。例如，一个企业可能会根据从客户处收到的订单来出售货物。一个所关心的活动可能是完成那些顾客订货单；每个订货单代表该活动的一个单独的实例。在某个特定的时间，该企业也许有多个处于不同完成阶段的活动的实例（即，来自多个客户的多个订单）。在另一个例子中，一个金融机构可能根据顾客的申请来贷放资金。一个所关心的活动可能是要把贷款申请处理完成（即，批准或拒绝），每一个申请表示这种活动的一个独立的实例。在任何特定的时间，可能会有多个处于不同处理阶段的贷款申请实例。再如另外一个例子，一个负责发行许可证的政府部门，可能有多个处于不同处理阶段的许可申请。

为了监控一个活动的众多实例，很多组织把关于那些活动实例的信息存储在一个数据库程序中。特别地，可以针对每个活动实例创建一条记录或其它数据对象。然后，记录的一个单独字段或其它部分将被建立，以便保存对每个实例都通用的某种信息类型的值。用前面的一个例子进行说明，卖方企

业可以为每个客户订单分别创建一条数据库记录。在每条记录中，独立的字段可包括收到订单的时间、收到订单的地点、订购的货物、发货时间等等。数据库程序的这种用法常常概念化为表。活动的每个实例占用表的一行（或元组）。多个实例公共的每一信息类型被单独赋予表中一列。

通过在数据库表中为某个活动的每个实例放置相关数据，使利用不同途径来分析这些数据成为可能。然而，随着越来越多记录的积累，数据库的有用性会降低。对于一个大型企业，譬如每天会收到几十万甚至数百万份订单的销售方，记录的数目会达到数十万，甚至数百万条。每次检索数据库时，就需要一定的时间去搜索磁盘驱动器或其他存储驱动器。同样的，当新建记录和更新现有记录时，每新建或更新一条记录都需要一些时间。随着记录数目的增长，寻找一条特定记录所需的时间越来越长。在一个拥有成百（或成千）用户和数十万（或数百万）条数据库记录的企业或组织机构中，访问数据库系统的等待时间将会很长，而且系统磁盘空间可能会用完。

图 1－3 提供了这个问题的更详细的说明，并且在首选实施例的详细描述中，提供了一个相关的例子。图 1 是一个流程图，假设了一个批发企业处理客户订货单的过程，这是一家依据客户订单来销售货物的企业。为了方便起见，从现在开始我们称之为“A 企业”。在方框 1 中，A 企业收到一个订货单，然后创建了相应的数据库记录；同时输入收到这个订单的时间。在方框 2 和 3 中，在产品订货数量和订货人所在城市的记录字段中输入更多的数据。在方框 4 中，判断是否接受此订货单。如果不接受，则在方框 5 中填写相应的记录字段，并记录拒绝时间。如果接受此订货单，则结果会被记录下来。如果此订货单被批准，则在方框 6 和 7 中将输入更多的信息。在方框 8 中装箱，则在另一个字段中记入装箱时间。在方框 9 和 10 中将输入更多的数据，例如，集装箱类型和货运方式。发货后，在方框 11 中将输入发货时间。

图 2 是一张数据库表，它描述 A 企业中订货单实例数据库的一部分。每个订单在一个独立的行中，并且每个列对应一个订单的一种数据。为了简单起见，图 2 仅仅显示了流程图 1 中所搜集的一部分信息对应的列。在这个例子中，某些字段包含 NULL 值，表示一个特定事件的值未知，因为在相关的订货单中，这个值是未公布。此数据库中一个典型的查询可能是“上周提交的订

单中，金额在1000美元以上的尚未接受或拒绝的有哪些？”这样的查询可以通过对这个表进行SQL（结构化查询语言）查询来实现。

附录A显示了一段代码创建图2中的表的SQL代码的实现（“create table PO_InstanceData”），以及用于更新图2的表中各行的存储过程（“create procedure PO_PrimaryImport”）。此存储过程接受一个唯一地确定了与一个活动实例的一条记录的PONum变元（在这个例子中，就是一个特定的订货单）和用于表中每一列的参数。在开始更新或插入记录之前，存储过程的多个参数可以将调用所需的名称－值对累积在存储器中（基于应用程序收集数据时使用的事务逻辑）。由于锁定和操作数据库记录的处理过程所需要的额外系统开销，所以在一个命令中对记录进行越多的更新（或插入）操作，效率就越高。存储过程首先发出一条“update”语句，替换一条记录中值不是NULL的那些列。这时它假定表中存在一些关于订货单（PO）的数据。要调用函数“coalesce”，以返回第一个非空（Non－NULL）变元。如果没有任何记录被更新（@@rowcount=0），这就是第一条关于此订货单的信息，会插入一条带所有变元参数的记录（即使它们的值为空（NULL））。

当相对较少的用户或程序线程试图写入一个表，并且记录也比较少时，诸如附录 A 所示的存储过程是令人满意的。不幸的是，正如图 3 中所示，随着表中的记录越来越多，性能会随着时间不断降低。实线表示写入性能，或者说每秒能写入的记录数目，它快速下降到很低的水平。相反的，平均磁盘I/O队列长度（虚线）很快增长到超出可接受的限度。这种性能的下降，应归因于表大小的增长。当记录数目较小时，性能受数据库服务器处理事务的速度的限制，而这个速度又取决于服务器的中央处理单元的能力。随着时间的增加，记录数目也在增长（例如，越来越多的接收和处理过的订货单），性能迅速下降。例如，附录 A 中，存储过程中更新语句的第一次执行，会导致从磁盘中将表的一部分（图 2）读入 RAM（或者其他内存系统）。只要记录总数较小，服务器能够在系统内存中缓存所需的大部分或者全部数据。如果随后的更新需要操作早已缓存在系统内存中的记录，则服务器不需要再去读磁盘。然而，当记录的数目超过了内存的容量，每个操作都可能请求一个读磁盘的物理操作。根据所用的硬件，在磁盘被读取的时候，其它对于这张表的查询

（或者更新）可能会被阻止。最终，对所有用户来说，反应时间慢到了不可接受的地步。如果要在这个表上执行更多复杂的数据操作，这个问题将更加严重。这些操作包括，在线分析处理（OLAP）和 OLAP 方块的建立。

(3)发明内容

本发明针对与维护一个活动的多个实例的信息有关的上述及其它难题。在本发明的一个方面，为对应于一个组织多个活动的多个活跃实例的数据和对应于非活跃的实例的数据独立维护数据库表。在另一方面，可以为对应于一个活动的非活跃实例的数据维护多个数据库表。在再一方面，处理从活跃实例的表中得到的数据以及从一个或多个非活跃实例的表中得到的数据，以产生组合分析数据。

在一个实施例中，本发明包含一个用于维护关于一项活动的多个实例的信息的方法。该活动的每个实例有一个活跃的状态或者非活跃的状态，在活跃的状态中实例信息将被修改，在非活跃的状态中，实例信息将不被修改。该方法包括：为处于活跃状态的多个实例中每一个，在第一数据库表中建立一条记录；每条记录为多种数据类型中的每一种包含一个字段，每一个活跃实例的一个或多个字段拥有一个指示活跃状态的值。该方法还包括，对于非活跃状态的多个实例的每个记录，给这一个或多个字段赋于指示非活跃状态的值。该方法还包括从第一表中删除多个实例的记录，这些记录的一个或多个字段指示其处于非活跃状态；以及为从第一表中删除的每个记录在第二数据库表中建立相应的记录。

在另一实施例中，本发明的方法包括建立第三和后续的数据库表，以及在后继表创建后，终止在先前建立表中创建记录。对于在建立了最近的表之后但在建立另一后续的表之前从第一表中删除的每个记录，都在最近的建立的一个表中建立相应的记录。在另外一个实施例中，本发明的方法包括为第一表中的记录产生第一在线分析处理（OLAP）方块，为第二表中的数据生成第二 OLAP，并且将第一第二方块组合成一个虚拟 OLAP 方块。

结合附图阅读以下较佳实施例的详细说明，本发明的以上及其他特性和优点会更清楚并容易完全理解。

(4)附图说明

图 1 是假想的企业处理客户订货单的流程图。

图 2 是表示假想企业的数据库一部分的表。

图 3 是显示数据库系统性能随着时间而下降的图。

图 4 是显示依照发明的至少一个实施例用于活跃和已完成实例数据独立表的方框图。

图 5 是依照发明的至少一个实施例包含依然在处理过程中的活动的部分数据库表的例子实施例。

图 6 是依照发明的至少一个实施例包含已完成活动的部分数据库表的例子实施例。

图 7 是依照发明的另一个实施例显示活动和已完成实例数据的独立表的方框图实施例。

图 8 是依照发明的至少一个实施例对活跃的和已完成的实例数据进行组合 OLAP 分析的数据处理流程图实施例。

(5)具体实施方式

本发明可以方便地与美国专利申请序列号. 10/157,968 中描述的方法、设备和系统结合使用。该专利申请的发明名称是“Support for Real-Time Queries Concerning Current State, Data and History of a Process”，于 2002 年 5 月 31 日提交，其内容通过引用包括于此。

本发明的说明中将会参考到结构化查询语言（SQL）指令和其它数据分析特性，包括由华盛顿州雷蒙德的微软公司所提供的 SQL SERVER™ 2000 关系数据库管理系统（RDBMS）软件和相关的在线分析处理（OLAP）服务软件中的。尽管在此描述的是在实现本发明中实施例时可以使用的 SQL 指令的一些方面，但是对于本领域技术人员而言，一旦阅读了本发明提供的描述，则可用其他的指令、编程算法和过程实现本发明是显而易见的。对 SQL SERVER™ 2000 RDBMS 软件和相关的 OLAP 服务软件的简单说明可以从多种途径得到，包括 Karen Delaney（2001 微软出版社）的《Inside Microsoft® SQL SERVER™ 2000》，

以及可以从http://www.microsoft.com/sql/techinfo/productdoc/2000/处得到的Microsoft® SQL SERVER™ 2000 Books Online。本发明并非只限于使用SQL SERVER™ 2000 RDBMS软件及相关的OLAP服务软件来实现，它可以使用其它种类的RDBMS和OLAP软件来实现。

本发明说明中也将参考运行于服务器并且由一个或多个客户端访问的RDBMS软件（例如前述的SQL SERVER™ 2000软件）。这种配置是人们熟知的，例如在先前引入的美国专利申请10/157,968中有描述。然而，客户机/服务器配置只是实现本发明的方式中的一个例子。本发明也可以在其他物理系统配置中实现。

通过维护不同的表：对应于组织活动中的活跃实例数据的表、以及有限数目的非活跃实例的表，本发明可以解决上文所讨论的很多问题。例如，在很多组织机构中，最重要的活动是那些当前悬而未决的，或者最近刚刚完成的。据联系图1－3讨论的假想的A企业的例子，A企业的经理们最感兴趣的是依然在处理中的订货单，也就是那些仍然没有被送出去的货物。那些经理们同样感兴趣的是那些在相对较近时期里已完成的订货单（即，在最近几个月中货物已交货的订货单）。虽然A企业是为了描述本发明而建立的一个假想例子，但是在真实的企业中，仍在处理的实例或最近已完成实例是很常见的。例如，在很多企业中，大部分的送货投诉和大部分的费用收取问题都是在发货之后短时期内发生的。尽管出于某些目的也许需要更早些填具的订单数据，但是这些需要相对来说很少。类似地，很多其他类型的企业和组织最关心的是依然没有完成的活动的实例，以及一定数量的最近完成的活动实例。

因此，A企业在不同的数据库表中维护活跃的和最近已完成的订单数据。通过将表的内容限制在活跃的订单和最近才完成的订单，能保持较小的数据量。这样一来，当更新或以其它方式存取活跃实例数据表时，系统性能不会像图3中那样下降，而且存取已完成实例表时性能也不会下降，因为在此表中只插入记录，不会更新记录。和图3中的例子不同，整体性能随着时间的流逝也保持稳定。图4的方框图举例说明了这个概念。表10保存了涉及活跃订货单的数据记录。在这个例子中，如果订货单的货物依然没有送达客户，则此订货单是活跃的。组织的活跃活动实例在其它上下文环境下会有不同的

定义。一旦一个订货单完成，该订货单的数据将被移动到表 12。在此例中，当定购货物被送达客户时，该订货单被完成。如同活跃实例一样，一个组织的已完成实例在其它上下文环境下可有不同的定义。

每收到一个新订单，就在表 10 中建立一条记录。表 10 在图 4 中用方框形式表示，图 5 更详细地显示了表 10 中的一部分内容。与图 2 中的表类似，表 10 中一个记录（例如，行）对应一个订货单，每个字段（列）中的数据类型各不相同。在这个例子中，“PONum”是订货单编号，“City”是发出订货单的客户所在城市，“Quantity”是定购的货物数量。“ShipTime”是所定购的货物装运日期和时间。“DeliveryTime”是交货日期和时间。尚未装运的订单的 ShipTime 字段中是一个<NULL>条目。图 5 中的一些记录，显示了还未装箱的货物的订单（PONum 8680 和 8685－87）。当这些订货单的货物装运后，“ShipTime”字段将被更新。与图 2 中的表不同，表 10 中仅限于记录那些当前活跃的订货单数据，也就是那些货物尚未发出的订单。因此，每条记录的“DeliveryTime”字段是一个 NULL 条目。表 10 还有个“IsCompleted”标志的字段。IsCompleted 的值为 0 时，表示对应的订货单还没有完成。在一些实施例中，IsCompleted 是一个“system”标志，不显示给查询此表的用户。

当一个订货单完成后，该订货单记录被从活跃实例数据表 10 中删除，并且在已完成实例表 12 中为该订货单建立一条新的记录（图 4）。这个新的记录是表 10 中所删除记录的复本，但是会用新数据覆盖 DeliveryTime 字段的 NULL 值，并且没有 IsCompleted 字段。图 6 详细的显示了表 12 的其中一部分。与图 2 和图 5 中的表类似，表 12 中一个记录对应一个已完成的订货单，不相同数据类型各有一个字段，包括前面提到的订货单号 PONum、收到时间 RecvTime、城市 City、数量 Quantity、发运时间 ShipTime 和交货时间 DeliveryTime。在这个例子中我们定义在定购的货物发货时订货单完成，所以表 12 中 DeliveryTime 字段在每一条记录上都是非 NULL 的值。表 12 还有一个字段记录号“RecordID”。如下面将详细介绍的，表 12 中每增加一条已完成订货单的记录，就动态递增生成这个值。通过与图 6 中的 PONum 和 RecordID 字段进行比较可以看出，订货单完成的顺序，不一定会完全按照分配的订货单编号的顺序。因此，RecordID 字段提供了一种机制，可以按照创建表 12 中的记录

的次序生成索引。这确保了表的大小不会影响插入的性能，从而避免了随时间推移性能下降的问题。

因为表 10 中仅包括当前活跃的订货单的数据，所以表的大小始终比较小。尽管随着业务量的波动表的大小可能会有所起伏，但是不会无限增大。随着更多的订货单从活跃 / 未完成状态成转变成非活跃 / 已完成状态，已完成实例数据表 12 将会变大。但是，已完成实例数据表 12 的增长，比起包含了所有活跃和非活跃实例的单个数据表（如图 2 中的表）的增长，简直是不足挂怀。因为记录只是被插入表 12，之后并不更新。所以不需要在每次存取表 12 时，搜索某一条记录。换言之，因为记录不用被更新，所以，表 12 中插入一条记录之前，不必寻找特定的记录。

附录 B 和 C 包含了一段用于创建和更新图 4-6 中的表的 SQL 代码的示例。附录 B 的第一条语句（“create table PO_Active_InstanceData”）创建了用于活跃实例的表 10，并且建立 PONum、RecvTime、City、Quantity、ShipTime、DeliveryTime 和 IsCompleted 列。将 PONum 指定为主键。换句话说，表 10 中的每行都是由订货单标号即 PONum 来唯一地标识。类似地，下一条语句(“create table PO_Completed_InstanceData”）创建了用于已完成实例的表 12，并且建立 PONum、RecvTime、City、Quantity、ShipTime、DeliveryTime 和 RecordID 列。不过在表 12 中，RecordID 被作为主键，它的值通过 SQL“identity”属性自动递增获得。换句话说，随着向名为 PO_Completed_InstanceData 的表 12 中每增加一条新的记录，数据库服务器自动将前一个 RecordID 的值加 1，然后将得到的值插入新增记录中。

下一条语句（“create procedure PO_PrimaryImport”）创建一个名为 PO_PrimaryImport 的存储过程，它被用于在表 10 中创建新记录或者更新现有的记录。存储过程 PO_PrimaryImport 有 5 个变元，对应于表 10 中一行上的列。例如，如果收到图 5 中的订货单 8680，客户机将向数据库服务器发出下面的存储过程调用：

```
PO_PrimaryImport ( 8680, 08/25/2003, 17:19, Redmond, 270, , , )
```

如果之前订货单 8681 的 PONum、RecvTime、City 和 Qunantity 的数据已经输入，下面的存储过程调用将把装运时间（ShipTime）更新成 08/26/2003 0910：

PO_PrimaryImport （ 8681, , , , 08/26/2003 0910, , ）

为了更新订货单 8682 的记录以反映发货时间（DeliveyTime）为 08/26/2003 晚上 12 点整，并将订货单标记为已完成状态（IsCompleted=1），可发出下面的调用：

PO_PrimaryImport （ 8682, , , , 08/26/2003 1200, 1）

需要指出，操作人员将上述任何信息输入客户机时，不一定通过键入一个上述命令来完成。例如，用户可以通过一个图形用户界面来输入信息，在客户机 / 服务器上运行的一层或多层中间软件可以用正确的语法生成这个存储过程调用。再如，订单可能通过因特网自动接收，然后 web 服务器软件能生成必要的 SQL 命令。

存储过程 PO_PrimaryImport 接收这个存储过程调用中传递的值，然后把这些值分别赋给一个或多个本地变量@PONum、@RecvTime、@City、@Quantity、@ShipTime、@DeliveryTime 和@IsCompleted。接着，存储过程试图通过“insert”语句将那些本地变量的值插入表 PO_Active_InstanceData（表 10）。然而，并不是将那些本地变量作为新记录插入表 10，而是触发附录 C 中的触发器（“PO_CompletedTrigger”）。

参考附录 C，在声明了本地变量@PONum 和@IsCompleted 之后，触发器把从“插入的”系统表中得到的值赋给这些变量。插入表是数据库服务器自动生成，并在 RAM 或其它系统存储器中临时存储那些在前面执行存储过程 PO_PrimaryImport 的“insert”语句时受到影响的行的副本。在这种情况下，插入表包含存储过程 PO_PrimaryImport 传递的变元的副本。换句话说，其中包含了用户当前试图插入或更新到表 10 中的值。

触发器首先查看被传递的 IsCompleted 位的值是否等于 1，根据 IsCompleted

位可以判断，所传递的 PONum 值所对应的记录是完成的。如果 IsCompleted 位等于 1，触发器就把已完成订货单记录的所有值插入到表 12 PO_Completed_InstanceData 的新记录中。表 12 中新记录的 PONum 值是从插入表（“select inserted.PONum”） 中得到的。函数“coalesce”用于获取表 12 中新记录的 RecvTime、City、Quantity、ShipTime 和 DeliveryTime 的值。特别地，（聚结）coalesce 函数将从它的变元中返回第一个非 NULL 表达式。例如，“coalesce (inserted.RecvTime, po.RecvTime)”提供了表 12 中新记录的 RecvTime 字段的值。触发器首先去验查插入表中 RecvTime 值是否为非 NULL。如果是，这个值就成为表 12 中新记录的 RecvTime 值。如果插入表中 RecvTime 值是 NULL,那么触发器就从表 10 的某个记录中获取 RecvTime 的值，用户试图使用对存储过程 PO_PrimaryImport 的调用来插入或更新这个记录。代码“from inserted left join PO_Active_InstanceData po on inserted.PONum=po.PONum”可以保证 coalesce 函数有一个变元是 Non-NULL（非空的）。准确地说，触发器的这部分代码载明，每个 coalesce 函数调用的变元值将从下面的集合中得到：这个集合包含（插入）“inserted”表中的所有行，外加 O_Active_InstanceData 表（表 10）中的所有行，这两个表中 PONum 值是相同的。例如，如果调用存储过程 PO_PrimaryImport 来传递其所有变元的非 NULL 值（即，当订货单被第一次输入系统的时候，此订货单的货物已经发货），则表 12 中新记录的 RecvTime 值将从插入表中得到。但是，如果调用此存储过程 PO_PrimaryImport 是为了更新表 10 中已经存在的记录，并且此记录中已经存在 RecvTime 值（即，在 PO_PrimaryImport 存储过程调用中不传递 RecvTime 值），那么表 12 中新记录的值将从表 10 的现存记录中得到。

在获得 RecvTime 的值之后，触发器用类似的方法获得表 12 中新记录的 City、Quantity、ShipTime 和 DeliveryTime 字段的值。正如前面所讨论的，服务器自动生成 RecordID 字段的值。然后，触发器删除表 10 中已完成的订货单记录（“delete from PO_Active_InstanceData where PONum=@PONum”）。此时，该触发器终止（“return”）。

如果在对触发了触发器的 PO_PrimaryImport 存储过程的调用中传递的

IsCompleted 值是 0，那么触发器不在表 12 中建立新记录，而是设法更新由 PO_PrimaryImport 存储过程调用中所指定的表 10 中的记录（“update PO_Active_InstanceData”）。正如在表 12 中新建记录的触发器一样，coalesce 函数被用来为表 10 中所更新的记录提供一个值，这个值可从插入表，或者表 10 现存记录中得到。但是，在这种情况下，coalesce 函数的变元值由代码“from PO_Active_InstanceData po join inserted on po.PONum=inserted.PONum”提供。准确地说，触发器的这部分代码指明，每个 coalesce 函数调用的变元值，将从下面的集合中得到：这个集合包含 PO_Active_InstanceData 表（表 10）和插入表中的所有行，这些行应具有相同的 PONum 值。

如果存储过程 PO_PrimaryImport（附录 B）被调用以在表 10 中增加一个未完成的新记录（也就是，插入一条尚未发货的订货单的新记录），则触发器中的“update PO_Active_InstanceData”代码，并没有更新什么值。在此情况下，在表 10 中不存在 PONum 值和插入表的 PONum 值相等的记录，从而在表 10 中没有记录需要更新。如果表 10 中没有记录要更新，或者没有记录要插入表 12，那么触发器将通过系统函数@@rowcount 来检测。准确的说，如果前面的 update 语句没有影响到任何行，那么函数@@rowcount 返回 0 值。如果函数@@rowcount 返回 0，那么触发器的一部分代码“insert PO_Active_InstanceData select * from inserted”将用根据（插入）“inserted”表中得到的值在表 10 中插入一条新记录。

在本发明的另一个实施例中，已完成实例数据表的数据量是有限的。如上所示，该表（图 4 中的表 12）的增长不及表 2 这样的表那么在意。然而表 4 也会随着时间不断增长。即使表中没有记录被更新（也就是，记录只是被插入），该表也会最终变得太大，以至于超过系统磁盘容量，或者对于此表中数据的查询，所需要的时间长到无法接受。如前所示，A 企业的经理已经确定，只有对近期内完成的订货单才需要已完成订货数据。但是，删除表 12 中的记录可能很耗时。在很多软件环境中，必须逐行删除；在每个记录上必须获得行锁，然后行被标识成已删除。实际上，删除一行和插入或更新一行所需的时间完全一样。

因此，如图 7 所示，为已完成实例数据建立了多个表。在此实施例中，附

录 B 和附录 C 中的代码像前述那样操作，它从表 10 中删除已完成订货单记录，并在表 12 中建立相应的新记录。然而，不允许表 12 无限制的增长。在一定时间后（例如，一个月），表 12 将被重命名为唯一的分区名，且不会再有更多的记录被添加到这个已改名的表中。此后，会新建一个空的表 12，它使用已改名的表从前的名称，即 PO_Complete_InstanceData。从这时起，触发器在该新表中插入记录。又过了一个月（或者其它预先指定的时间间隔）之后，这个表也被重命名，然后再创建一个表 12。当一个已改名的表被保留了指定的时间段之后（例如，六个月），整个表被删除。与删除单个记录不同，删除整个表非常快。在一个实施例中，使用 SQL 命令“drop table”去删除表。当删除一个已完成的实例数据表时，该表中的数据不一定会丢失。例如，在调用删除表“drop table”的函数之前，将表中数据传输到磁带或其它类型的存储媒体中存档。在其它的实施例中，不是在固定的间隔后建立新的表 12，而是在表 12 达到一定的大小时，建立新的表 12。

因此，活跃实例和已完成实例表中的所有数据，都可以方便的查看和查询（也就是，单次查询不一定会涉及到每个独立的表）。活跃和已完成实例表（或者所需要的部分表的小组）能被组合成一个分区视图。在至少一个实施例中，可以使用 SQL 语句“union all select * ...”将多个表组合起来。每次在删除一个已完成实例数据表（或者，每创建一个新的已完成实例数据表时）时，都可以重新创建这个由表组合而成的视图。

在本发明的另一方面，活跃的和已完成的实例的表中的数据被进一步处理，以便提供更多的分析数据。举例来说，A 企业可能希望对当前活跃的订货单数据以及当前已完成订货单数据作联合分析。例如，A 企业希望为这些组合的数据生成一个或多个 OLAP 方块。此外，尽管 A 企业是为了说明本发明而假想的企业，但是真实的组织也需要生成反映该组织活动中活跃的和已完成的实例的 OLAP 方块。

从图 8 可以看出，在至少一个实施例中的一个数据处理流程。它是对活跃和已完成实例数据的组合 OLAP 分析的处理。定期地执行存储过程（“BeginDataProcessing”）（例如，每天晚上）。该过程首先建立活跃实例数据表（表 10）的一个副本。因为此表比较小，所以该副本能很快建立。

然后该副本中的数据被传送到数据转化服务器（DTS）（未示出），以便被放置到一个星型模式中，然后把这些活跃实例数据完全处理成 OLAP 方块 30。存储过程 BeginDataProcessing 也从一个递增窗口中获取一部分已完成实例数据。特别的，存储过程 BeginDataProcessing 每次获取已完成实例数据时，数据库服务器将保存最后获取的记录的 RecordID 值。如前所述，这个值是在创建记录时，由服务器以递增的方式赋值的。通过参照以前运行过程 BeginDataProcessing 的时候存储的最后获取的活跃实例数据记录的 RecordID 值，此次运行存储过程时，仅获取上次运行之后建立的记录。这个递增的窗口作为一个特殊视图来实现，该视图建立在分区视图之上，这个分区视图包含多个用于活跃和已完成实例数据的表/分区块。这样一来，该递增窗口能包含多个分区的数据。例如，如果过程 BeginDataProcessing 每周执行一次，那么在上次执行存储过程 BeginDataProcessing 后可能有新的已完成实例数据表建立，而且可能需要处理多个已完成实例数据表的数据。在至少一个实施例中，存储过程 BeginDataProcessing 同时获得活跃实例数据表的副本和已完成实例数据表的记录的副本。否则，在过程 BeginDataProcessing 复制之后，但在获取已完成实例数据表记录之前，订货单可能被从活跃实例数据表中移除。如果发生这种情况，那么相同的的订货单将会被处理两次，从而破坏了数据分析的准确性。

存储过程 BeginDataProcessing 把自前次 BeginDataProcessing 执行之后建立的增量已完成实例数据记录传送给 DTS。然后 DTS 将该递增数据放入一个已包含了从上一次已完成实例数据处理来的数据的星型模式。然后，已完成实例数据的星型模式（它现在包含最近的已完成实例数据记录的递增数据）被用于更新已完成实例 OLAP 方块 32。类似于已完成实例星型模式，OLAP 方块 32 包含了前面几次会话中处理过的记录的信息。然后 OLAP 方块 30 和 OLAP 方块 32 被组合成一个虚拟 OLAP 方块 34。随着时间的推移，与 OLAP 方块 32 中包含其信息的已完成记录数会变得越来越大。然而，通过递增地处理已完成实例数据，并且将处理结果与之前处理过的已完成实例数据进行组合，能在相当短的时间内生成 OLAP 方块 32（以及虚拟 OLAP 方块 34）。换言之，可以避免重复处理已完成实例数据。

虚拟 OLAP 方块 34 向用户提供了一个业务的“快照”，它包含了历史的和进行中的（即，活跃的）的订货单的信息。可以安排在晚上或其它较闲的时间内运行处理方块 30-34 的数据转换服务（DTS）软件包。

尽管本发明的说明使用的是一个假想的企业，但是应注意，该发明并不局限于某一特定类型的企业、组织或活动。事实上，本发明并不限于以已完成实例数据的使用时间为依据来保留已完成实例数据的实现，别的组织为了能快速存取非活跃实例数据，可能使用一些其它标准，而不是维护最近月份所完成的订货单的数据。仅仅作为一个例子，一个调查公司可能希望快速地操作若干批次数据，但是可能很少操作其它批次的数据。因此，虽然介绍了运用该发明的具体例子，但是本领域技术人员将明白，上述系统和技术的很多变化形式、排列将落入所附权利要求书中阐明的本发明的精神和范围。这些和其它改变属于所附属权利要求所定义的本发明的范围。

附录 A

```
create table PO_InstanceData
(
      PONum int primary key,
      RecvTime datetime null,
      City nvarchar(50) null,
      Quantity int null,
      ShipTime datetime null,
      DeliveryTime datetime null
)
go

create procedure PO_PrimaryImport
(
      @PONum int,
      @RecvTime datetime=null,
      @City nvarchar(50)=null,
      @Quantity int=null,
      @ShipTime datetime=null,
      @DeliveryTime datetime=null
)
as
begin

update PO_InstanceData
set
      RecvTime=coalesce(@RecvTime,RecvTime),
      City=coalesce(@City,City),
      Quantity=coalesce(@Quantity,Quantity),
      ShipTime=coalesce(@ShipTime,ShipTime),
      DeliveryTime=coalesce(@DeliveryTime,DeliveryTime)
where PONum=@PONum

if @@rowcount=0
insert PO_InstanceData values
  (@PONum,@RecvTime,@City,@Quantity,@ShipTime,@DeliveryTime)

end
```

附录 B

```
create table PO_Active_InstanceData
(
      PONum int primary key,
      RecvTime datetime null,
      City nvarchar(50) null,
      Quantity int null,
      ShipTime datetime null,
      DeliveryTime datetime null,
      IsCompleted bit
)
create table PO_Completed_InstanceData
(
      RecordID int primary key identity,
      PONum int,
      RecvTime datetime null,
      City nvarchar(50) null,
      Quantity int null,
      ShipTime datetime null,
      DeliveryTime datetime null
)
go

create procedure PO_PrimaryImport
(
      @PONum int,
      @RecvTime datetime=null,
      @City nvarchar(50)=null,
      @Quantity int=null,
      @ShipTime datetime=null,
      @DeliveryTime datetime=null,
      @IsCompleted bit=0
)
as
insert PO_Active_InstanceData values (
      @PONum,
      @RecvTime,
      @City,
      @Quantity,
      @ShipTime,
      @DeliveryTime,
      @IsCompleted)
go
```

附录 C

```
create trigger PO_CompletedTrigger on PO_Active_InstanceData
instead of insert
as
begin
     declare @PONum int
     declare @IsCompleted bit
     select @PONum=PONum,@IsCompleted=IsCompleted from
inserted

     if (@IsCompleted=1)
     begin
          insert PO_Completed_InstanceData
(PONum,RecvTime,City,Quantity,ShipTime,DeliveryTime)
          select
               inserted.PONum,
               coalesce(inserted.RecvTime,po.RecvTime),
               coalesce(inserted.City,po.City),
               coalesce(inserted.Quantity,po.Quantity),
               coalesce(inserted.ShipTime,po.ShipTime),
               coalesce(inserted.DeliveryTime,po.DeliveryTime)
          from inserted left join PO_Active_InstanceData po on
inserted.PONum=po.PONum

          delete from PO_Active_InstanceData where
PONum=@PONum
          return
     end

     update PO_Active_InstanceData
     set
          RecvTime=coalesce(inserted.RecvTime,po.RecvTime),
          City=coalesce(inserted.City,po.City),
          Quantity=coalesce(inserted.Quantity,po.Quantity),
          ShipTime=coalesce(inserted.ShipTime,po.ShipTime),

     DeliveryTime=coalesce(inserted.DeliveryTime,po.DeliveryTi
me)
     from PO_Active_InstanceData po join inserted on
po.PONum=inserted.PONum

     if @@rowcount=0
     insert PO_Active_InstanceData select * from inserted
end
go
```

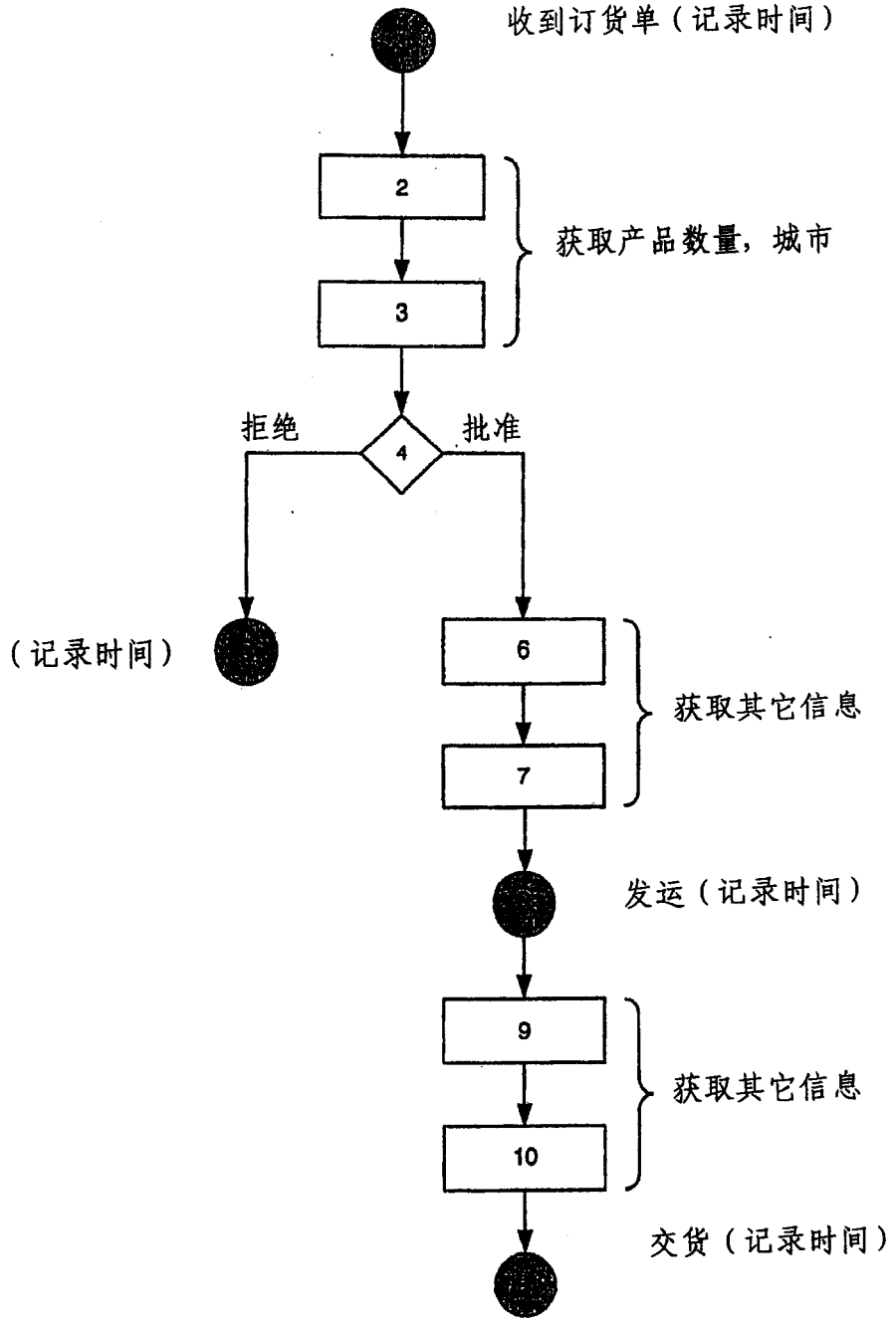

收到订货单（记录时间）
2
3
获取产品数量，城市
拒绝
4
批准
（记录时间）
6
7
获取其它信息
发运（记录时间）
9
10
获取其它信息
交货（记录时间）

图 1

订货号	收到时间	城市	数量	发运时间	交货时间
123	8:00am	Seattle	150	8:24am	12:45pm
124	8:30am	Seattle	234	8:45am	1:20pm
125	8:35am	Redmond	87	9:05am	2:30pm
126	8:45am	Seattle	450	9:20am	3:10pm
127	8:55am	Redmond	200	9:30am	<NULL>
128	8:57am	Seattle	340	9:20am	3:05pm
129	9:12am	Seattle	120	9:45am	<NULL>
130	9:30am	Redmond	25	10:15am	<NULL>
131	9:45	Seattle	250	10:35am	<NULL>
132	10:00am	Redmond	100	<NULL>	<NULL>
133	10:15am	Seattle	230	<NULL>	<NULL>
134	10:25am	Redmond	45	<NULL>	<NULL>
...					

图 2

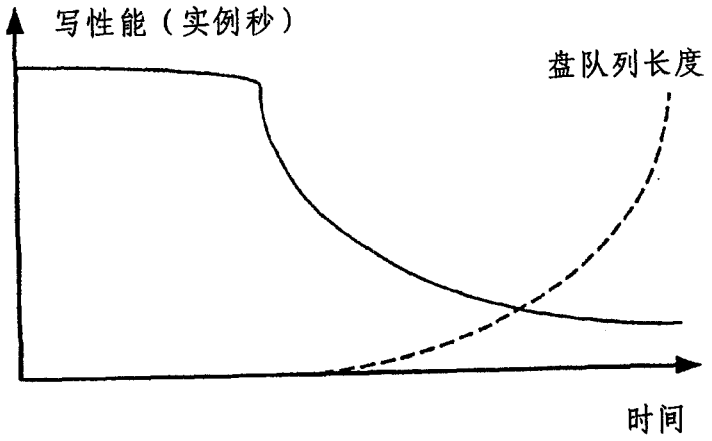


图 3

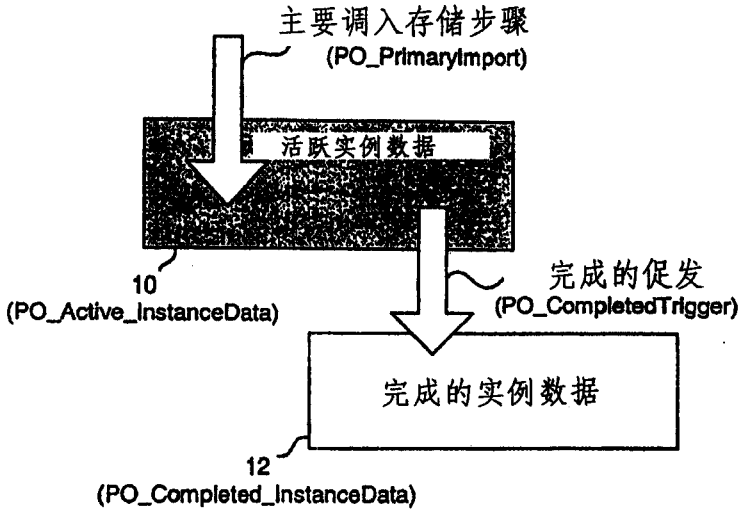
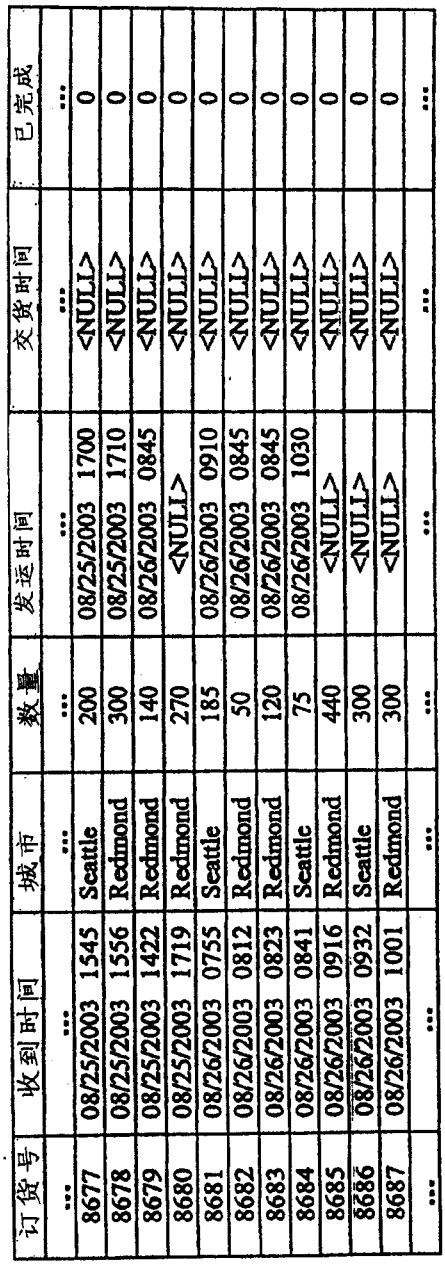

主要调入存储步骤
(PO_PrimaryImport)
活跃实例数据
10
(PO_Active_InstanceData)
完成的促发
(PO_CompletedTrigger)
完成的实例数据
12
(PO_Completed_InstanceData)

图 4



订货号	收到时间	城市	数量	发运时间	交货时间	已完成
...	...	...	...	...	...	...
8677	08/25/2003 1545	Seattle	200	08/25/2003 1700	<NULL>	0
8678	08/25/2003 1556	Redmond	300	08/25/2003 1710	<NULL>	0
8679	08/25/2003 1422	Redmond	140	08/26/2003 0845	<NULL>	0
8680	08/25/2003 1719	Redmond	270	<NULL>	<NULL>	0
8681	08/26/2003 0755	Seattle	185	08/26/2003 0910	<NULL>	0
8682	08/26/2003 0812	Redmond	50	08/26/2003 0845	<NULL>	0
8683	08/26/2003 0823	Redmond	120	08/26/2003 0845	<NULL>	0
8684	08/26/2003 0841	Seattle	75	08/26/2003 1030	<NULL>	0
8685	08/26/2003 0916	Redmond	440	<NULL>	<NULL>	0
8686	08/26/2003 0932	Seattle	300	<NULL>	<NULL>	0
8687	08/26/2003 1001	Redmond	300	<NULL>	<NULL>	0
...	...	...	...	...	...	...

10

图 5

订货号	收到时间	城市	数量	发运时间	交货时间	记录号
...	...	...	...	...	...	
7822	08/05/2003 0905	Redmond	100	08/05/2003 1100	08/05/2003 1330	7719465084
7823	08/05/2003 0935	Redmond	200	08/05/2003 1100	08/05/2003 1402	7719465085
7844	08/05/2003 1123	Seattle	220	08/05/2003 1315	08/05/2003 1551	7719465086
7835	08/05/2003 1014	Seattle	70	08/05/2003 1315	08/05/2003 1430	7719465087
7845	08/05/2003 1132	Seattle	40	08/05/2003 1315	08/05/2003 1615	7719465088
7847	08/05/2003 1145	Redmond	310	08/05/2003 1422	08/05/2003 1710	7719465089
7829	08/05/2003 0944	Seattle	250	08/06/2003 0845	08/06/2003 1120	7719465090
7870	08/05/2003 1610	Redmond	200	08/06/2003 0930	08/06/2003 1030	7719465091
7871	08/05/2003 1615	Redmond	10	08/06/2003 0930	08/06/2003 1054	7719465092
7879	08/05/2003 1704	Seattle	280	08/06/2003 0915	08/06/2003 1200	7719465093
7891	08/06/2003 0755	Seattle	275	08/06/2003 0845	08/06/2003 1000	7719465094
...	...	...	...	...	...	

12

图 6

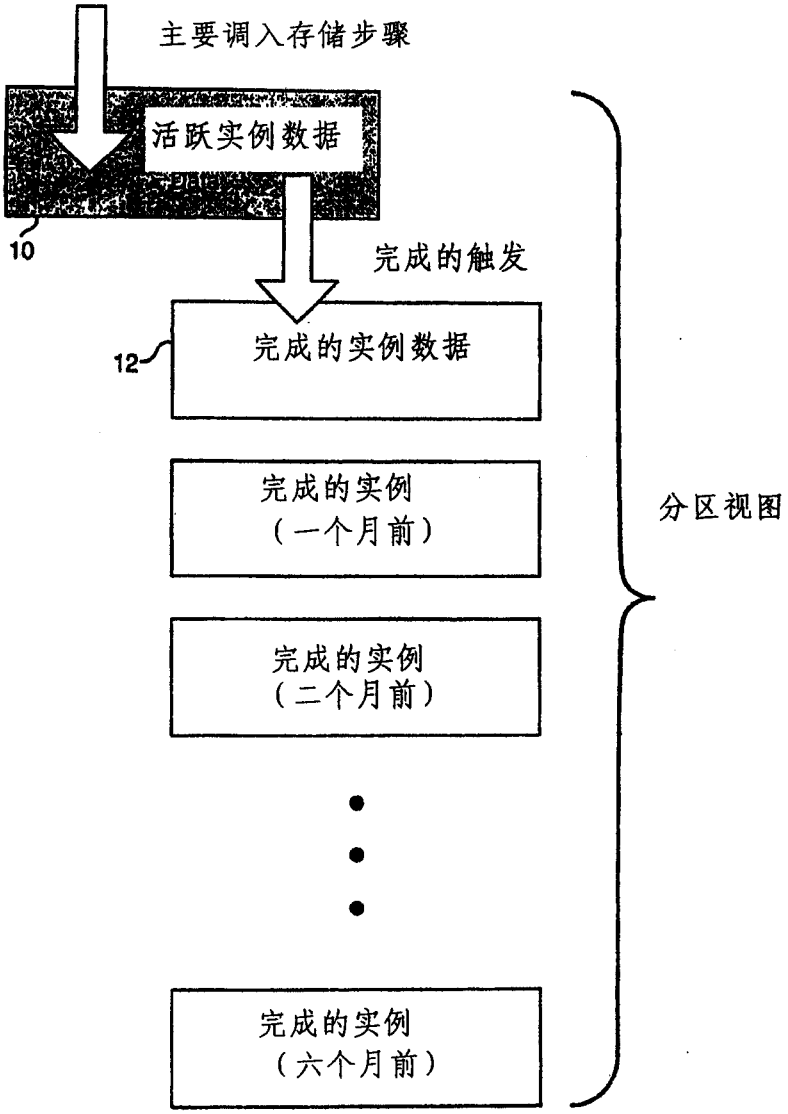

主要调入存储步骤
活跃实例数据
10
完成的触发
12
完成的实例数据
完成的实例
（一个月前）
完成的实例
（二个月前）
完成的实例
（六个月前）
分区视图

图 7

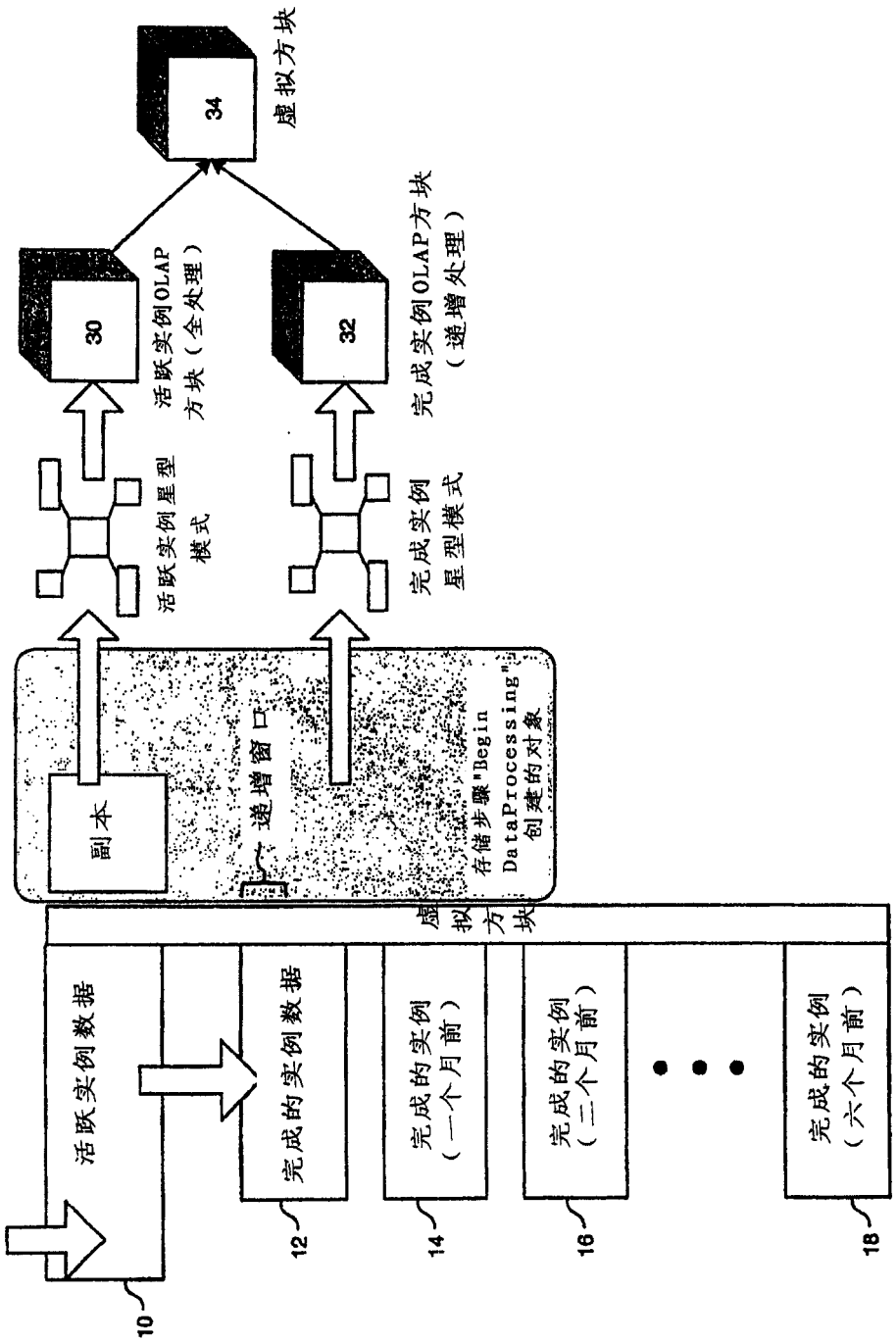

活跃实例数据
10
副本
递增窗口
"DataProcessing"存储步骤"Begin"创建的对象
虚拟方块
活跃实例星型模式
活跃实例OLAP方块（全处理）
30
完成实例星型模式
完成实例OLAP方块（递增处理）
32
34
虚拟方块
完成的实例数据
12
完成的实例（一个月前）
14
完成的实例（二个月前）
16
完成的实例（六个月前）
18

图 8