



(12) 发明专利

(10) 授权公告号 CN 101268439 B

(45) 授权公告日 2012.04.25

(21) 申请号 200680029606.1

(51) Int. Cl.

(22) 申请日 2006.07.20

G06F 7/00(2006.01)

(30) 优先权数据

11/207,482 2005.08.19 US

(56) 对比文件

(85) PCT申请进入国家阶段日

2008.02.13

US 6694337 B1, 2004.02.17, 说明书第2栏
第26行至第4栏第10行、说明书附图1-2.

(86) PCT申请的申请数据

PCT/US2006/028346 2006.07.20

US 2005/0149582 A1, 2005.07.07, 说明书第
[0009]-[0017]段.

(87) PCT申请的公布数据

W02007/024376 EN 2007.03.01

US 5555404 A, 1996.09.10, 说明书第5栏33
行至第8栏13行.

(73) 专利权人 微软公司

同上.

地址 美国华盛顿州

CN 1564990 A, 2005.01.12, 全文.

(72) 发明人 R·H·格伯 B·S·拉曼

审查员 胡雅娟

J·R·汉密尔顿 J·F·路德曼

M·M·克里什纳 S·H·史密斯

S·阿什威

(74) 专利代理机构 上海专利商标事务所有限公
司 31100

代理人 顾嘉运

权利要求书 3 页 说明书 15 页 附图 16 页

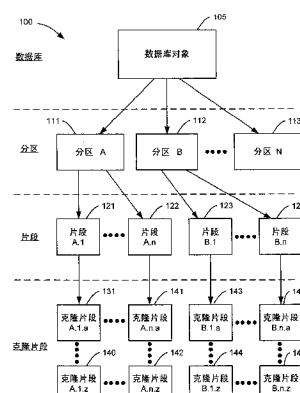
(54) 发明名称

数据库片段克隆和管理

(57) 摘要

提供数据库片段克隆和管理的机制与技术。

诸如表、最原始数据或索引的数据库对象被划分成片段。每一片段被克隆以创建克隆片段，在操作上它们彼此是完全相同的。一个或多个克隆片段可被指定为主克隆片段用于执行数据操作，或者可被指定为次克隆片段用作备份。对每一片段的更新在主克隆片段上实现，且随后从主克隆片段传播到相应的次克隆片段。



1. 一种用于管理克隆的数据库片断的方法，包括：

标识与数据库对象相关联的主克隆片段，所述主克隆片段被选择用于与所述数据库对象相关联的数据库操作；

标识与所述主克隆片段相应的次克隆片段，所述次克隆片段相对于所述主克隆片段是过时的；以及

通过执行数据库操作来用所述主克隆片段中的数据刷新所述过时的次克隆片段；

其中，所述方法还包括：

响应于在更新期间所述过时的次克隆片段离线，保存所述主克隆片段中的更新；以及

在所述过时的次克隆片段返回到在线时，用所述保存的更新来刷新所述过时的次克隆片段。

2. 如权利要求 1 所述的方法，其特征在于，所述用于刷新所述过时的次克隆片段的数据库操作包括增量地刷新所述过时的次克隆片段的操作的批处理。

3. 如权利要求 2 所述的方法，其特征在于，所述用于刷新所述过时的次克隆片段的操作的批处理是结构化查询语言 (SQL) 操作。

4. 如权利要求 1 所述的方法，其特征在于，还包括响应于在所述主克隆片段上执行的更新，在所述过时的次克隆片段正被刷新的同时更新所述次克隆片段。

5. 如权利要求 4 所述的方法，其特征在于，所述更新所述次克隆片段的步骤还包括更新所述过时的次克隆片段的最新行，但不更新所述过时的次克隆片段的任何非最新行。

6. 如权利要求 1 所述的方法，其特征在于，所述用所述保存的更新来刷新所述过时的次克隆片段的步骤还包括：

标识与包括在所述主克隆片段中的第一记录相关联的第一更新标识符；

标识与所述过时的次克隆片段中相应于所述第一记录的第二记录相关联的第二更新标识符；以及

至少部分基于对所述第一更新标识符与所述第二更新标识符的比较，确定具有相同克隆记录标识符的第一记录和第二记录是否一致。

7. 如权利要求 6 所述的方法，其特征在于，所述用所述保存的更新来刷新所述过时的次克隆片段的步骤还包括如果所述第一和第二记录被确定为不一致，

用所述第一记录中的数据刷新所述第二记录；以及

将所述第一更新标识符的值指派给所述第二更新标识符。

8. 如权利要求 7 所述的方法，其特征在于，还包括对包括在所述次克隆片段中的每一记录执行所述确定所述第一和第二记录为不一致、用所述第一记录中的数据刷新所述第二记录和将所述第一更新标识符的值指派给所述第二更新标识符的步骤。

9. 如权利要求 1 所述的方法，其特征在于，所述用所述保存的更新来刷新所述过时的次克隆片段的步骤还包括在读确认模式中在所述次克隆片段上运行预刷新操作。

10. 如权利要求 9 所述的方法，其特征在于，所述预刷新操作在相应于所述主克隆片段的其它次克隆片段上并行地运行。

11. 如权利要求 1 所述的方法，其特征在于，所述刷新所述次克隆片段的数据库操作包括在所述次克隆片段中插入、删除和更新记录的数据操纵语言 (DML) 操作。

12. 如权利要求 1 所述的方法，其特征在于，所述主克隆片段是克隆数据片段或克隆索

引片段中的至少之一。

13. 如权利要求 1 所述的方法,其特征在于,所述过时的次克隆片段是克隆索引片段,并且所述过时的次克隆片段是由多个克隆数据片段刷新的,每一所述克隆数据片段与所述过时的次克隆片段中至少一个条目相关联。

14. 一种用于为数据库提供克隆片段的方法,包括:

标识数据库对象的片段;

从所述标识的片段生成相应的克隆片段;

将所述克隆片段分配给一组计算设备;以及

将至少一个所述克隆片段指定为主克隆片段,以被配置为用于与所述数据库对象相关联的操作,而将其它相应的克隆片段指定为次克隆片段;

确定对所述数据库对象的所述被标识的片段的更新;

对所述主克隆片段执行所述更新;以及

将所述更新传播并应用于其它相应的次克隆片段;

其中,所述方法还包括:

响应于在更新期间所述次克隆片段离线,保存所述主克隆片段中的更新;以及

在所述次克隆片段返回到在线时,用所述保存的更新来刷新所述次克隆片段。

15. 如权利要求 14 所述的方法,其特征在于,还包括:

将所述次克隆片段之一标识为不包括所述更新的过时的次克隆片段;以及

用所述主克隆片段中的数据刷新所述被标识的过时的次克隆片段。

16. 如权利要求 15 所述的方法,其特征在于,所述用所述保存的更新来刷新所述次克隆片段的步骤是使用标准 SQL 数据库操作执行的。

17. 如权利要求 15 所述的方法,其特征在于,还包括:

确定对所述被标识的片段的另一更新;

在所述主克隆片段上执行所述另一更新;以及

仅在所述另一更新之前所述过时的次克隆片段中的记录对于所述主克隆片段中的相应记录为最新的情况下,选择性地在所述过时的次克隆片段上执行所述另一更新。

18. 如权利要求 17 所述的方法,其特征在于,所述选择性地执行步骤包括在所述已被刷新的过时的次克隆片段的一部分上执行所述另一更新。

19. 如权利要求 18 所述的方法,其特征在于,还包括:

响应于确定所述过时的次克隆片段已经刷新,立即将所述被刷新的次克隆片段指定为符合成为主克隆片段的要求。

20. 如权利要求 15 所述的方法,其特征在于,所述用所述保存的更新来刷新所述次克隆片段的步骤被实现为工作负载,以将少量记录锁定一段短时间间隔并与其它用户工作负载并发地运行。

21. 如权利要求 14 所述的方法,其特征在于,还包括:

将多个所述次克隆片段标识为不包括所述更新的过时的次克隆片段;以及

用所述主克隆片段中的数据并行地刷新所述被标识的多个过时的次克隆片段。

22. 如权利要求 21 所述的方法,其特征在于,所述多个过时的次克隆片段相应于共同的主克隆片段、不同的主克隆片段或不同的数据库对象中的至少一个。

23. 如权利要求 15 所述的方法,其特征在于,所述克隆片段被生成为对数据库操作实质上相同。

24. 一种用于交互计算设备的系统中的计算设备的方法,所述计算设备包括与数据库对象相关联的数据,所述方法包括下列步骤:

标识所述数据库对象的一部分用于更新;

使用第一数据库操作更新与所述数据库对象的所述被标识部分相关联的主克隆片段;

标识与所述主克隆片段相关联的次克隆片段;以及

至少部分地基于包括在所述主克隆片段中的数据,使用与所述第一数据库操作相关联的第二数据库操作更新所述次克隆片段;

其中,所述方法还包括下述步骤:

响应于在更新期间所述次克隆片段离线,保存所述主克隆片段中的更新;以及
在所述次克隆片段返回到在线时,用所述保存的更新来刷新所述次克隆片段。

25. 如权利要求 24 所述的方法,其特征在于,所述方法还包括下列步骤:

确定所述次克隆片段相对于所述主克隆片段是过时的;

使用所述主克隆片段中的数据刷新所述次克隆片段;以及

在所述次克隆片段正被刷新时,选择性地更新所述次克隆片段。

26. 一种设备可执行方法,包括:

监视与数据库对象相关联的克隆片段,所述克隆片段包括主克隆片段与次克隆片段;

确定至少一个所述克隆片段已经变为不可用;

增加与所述数据库对象相关联的更新标识符;

在事务上持久化所述更新标识符;

标识与所述数据库对象相关联的 DML 语句,所述 DML 语句包括对所述数据库对象的更新;

将所述持久化的更新标识符与所述 DML 语句相关联;

在受所述 DML 语句影响的所述主克隆片段上实现所述更新;

在受所述更新影响的所述主克隆片段中的每一行内存与所述 DML 语句相关联的所述更新标识符;

将所述更新与所述更新标识符传播至与所述受影响的主克隆片段相关联的所述次克隆片段;

在所述次克隆片段上存储所述更新标识符;

其中,所述方法还包括:

响应于在更新期间所述次克隆片段离线,保存所述主克隆片段中的更新;以及
在所述次克隆片段返回到在线时,用所述保存的更新来刷新所述次克隆片段。

数据库片段克隆和管理

背景技术

[0001] 数据库系统一般用于高效地管理为可访问性而组织的信息。为确保可访问性，系统可包括数据库的备份拷贝，以防原始拷贝损坏或丢失。用于数据库备份的一种通用技术是周期性地将整个数据库拷贝到计算机可读介质上。例如，信息系统管理员可在每周末将数据库拷贝到盘或带上。尽管数据库的拷贝可用这种方式来保存，但拷贝整个数据库是耗时的，并且备份间隔之间的活动如果没有通过其它手段跟踪就会丢失。

[0002] 数据库备份的另一技术涉及在不同计算机器上保存同一数据库的拷贝。如果数据库之一变成不可用，则在另一计算设备上的数据库的另一拷贝仍可访问。这类故障备份能确保数据库的可用性。然而，实时同步不同机器中整个数据库的多个拷贝是一个复杂而昂贵的过程。

[0003] 在本领域中仍未找找出一种能在没有不适当的复杂性或者没有数据的不必要丢失的情况下确保数据库可用性的一种有效方法。

发明内容

[0004] 下面提供本发明的简单概要，以便给读者提供基本的理解。本概要不是本发明的详尽概观，并且它不标识本发明的关键 / 重要元素或描绘本发明的范围。其唯一目的是以简化形式提供某些概念作为稍后提供的更具体的描述的一个序言。

[0005] 本示例提供用于数据库片段 (fragment) 克隆和管理的机制与技术。数据库对象，诸如表、行集合、索引或表或索引的分区，被分割成片段。注意，行集合被视为表中的行或者索引中的条目的集合。术语行和记录实际上被视为相同。因而，行集合也等价于记录集。每个片段被克隆以创建克隆片段，在实质上，它们在操作上是彼此相同的。克隆片段之一可被指定为主克隆片段，用于执行数据库操作，而一个或多个克隆片段可被指定为次克隆片段，其目的是用作主片段克隆的备份。对每个片段的更新是在主克隆片段上实现的，随后从主克隆片段传播到相应的次克隆片段。

[0006] 克隆片段能进入离线状态，变得不能更新。这种离线的不可用的克隆片段被定义为过时的克隆片段，因为在这种克隆片段中的数据在事务上对于相对应的主克隆片段不再是最新的。当过时的克隆片段返回在线状态时，克隆片段用包括在主克隆片段中的数据来刷新。在被刷新的同时，克隆片段可继续被更新。在刷新过程完成后，克隆片段立即变成符合被指定为主克隆片段的候选要求。

[0007] 许多伴随的特征将通过以下结合附图考虑的详细描述参考而变得更容易理解。

附图说明

[0008] 本描述将通过结合附图阅读的下列详细描述而被更好地理解，其中：

[0009] 图 1 示出由数据库系统管理的数据库对象的示例克隆数据结构。

[0010] 图 2 示出数据库对象的示例克隆片段。

[0011] 图 3 示出更新数据库对象的克隆片段的示例操作。

- [0012] 图 4 示出刷新与数据库对象相关联的过时的克隆片段的示例操作。
- [0013] 图 5 示出标识克隆片段的示例数据结构。
- [0014] 图 6 示出标识数据库对象的克隆片段中的记录的示例数据结构。
- [0015] 图 7 示出更新次克隆索引片段与次克隆数据片段的示例过程。
- [0016] 图 8 示出当刷新过时的克隆片段时的状态过程。
- [0017] 图 9 是具有克隆的数据库片段的示例数据库系统。
- [0018] 图 10 示出更新数据库对象的示例过程。
- [0019] 图 11 示出刷新数据库对象的过时的克隆片段的示例刷新过程。
- [0020] 图 12 示出修改克隆更新标识符 (CUID) 的示例过程。
- [0021] 图 13 示出指派根据图 12 所示的过程所分配的 CUID 值的示例过程。
- [0022] 图 14 示出 CUID 值的示例传播路径。
- [0023] 图 15 示出实现所述系统与方法的示例计算机设备。
- [0024] 图 16 示出一示例过程, 用于传播与应用更新于一记录, 并将旧的 CUID 值从主克隆片段传播到所有次克隆数据片段, 也传播到包括该记录的拷贝的所有克隆索引片段。
- [0025] 图 17 示出作为图 11 所示的刷新过程的一部分的示例过程, 用于使过时的克隆成为当前的克隆。
- [0026] 相同的参考数字用于指定附图中相同的部分。

具体实施方式

[0027] 下面结合附图提供的详细描述旨在作为对当前示例的描述, 而不是要表示可构造或使用当前示例的唯一形式。本说明书阐述示例的功能以及一系列用于构造和操作示例的步骤。然而, 相同或等价功能和序列可通过不同示例来完成。

[0028] 尽管当前示例在此被描述和例示为在数据库片段克隆与管理系统中实现, 但所述系统是作为示例而非限制来提供的。如本领域的技术人员将明了的, 当前示例适合应用于各种不同类型的数据库片段克隆和管理系统。

[0029] 图 1 示出由数据库系统管理的数据库对象 105 的示例克隆数据结构 100。数据库是信息的集合, 这些信息以使数据库中所需数据片能被快速地选择和 / 或更新的方式来组织。数据库对象可以是整个数据库, 也可以是数据库的任意部分。例如, 数据库对象 105 可以是整个表、索引、行的集合 (例如行集合) 等等。

[0030] 数据库对象 105 可划分为分区 111-113。通常, 数据库对象 105 是为了便利或性能原因而被划分的。例如, 数据库对象 105 可包括与多个年份相关联的数据。数据库对象 105 可划分成分区 111-113, 其中每个分区与一特定的年份相关联。数据库对象 105 的划分分区是一个可选的步骤, 可以在实际的实现中实现, 也可以不实现。

[0031] 数据库对象 105 的每个分区 111-113 (或整个未划分分区的对象 105) 一般划分成片段, 如片段 121-124。片段 121-124 是由数据库系统在操作的基础上划分的数据库对象 105 的各部分。例如, 片段 121-124 可被指派给不同的计算设备, 使得与数据库对象 105 相关联的查询可由并行工作的计算设备通过片段 121-124 来执行。

[0032] 数据库对象 105 中的片段进而被克隆以创建克隆片段。如图 1 所示, 每一逻辑地划分的片段 121-124 被克隆, 以产生示例克隆片段 131、140-146。通常, 数据库对象 (或数

据库对象的一个分区)的片段是通过将这样的对象分离成行(对于表)或索引条目(对于索引)的分立集合来创建的。在表的键上或索引上进行散列是完成这种分离的一个基础。行的集合有时称为行集合或者记录的集合。克隆片段将结合图2更详细地讨论。简言之,当正确地更新时,与数据库对象105的一特定片段相关联的克隆片段在操作上是相同的,因此克隆片段可方便地由数据库系统使用。克隆片段的使用使得一个特定片段的两个或多个拷贝能够随时使用,诸如保持高水平的数据可用性,加速查询和其它数据库操作,执行负载平衡,等等。例如,为保持高水平的数据可用性,至少一个克隆片段可用作另一用于数据库操作的克隆片段的备份。为加速搜索,多个操作上相同的克隆片段可同时用于数据库查询。为执行负载平衡,不同但操作上相同的克隆片段的拷贝可基于工作负载状况而被激活。

[0033] 图2示出数据库对象的示例克隆片段131-139。如图所示,克隆片段131-139可视为三个组151-153。每组克隆片段与数据库对象的特定片段有关。在每组中的克隆片段在操作上彼此相同地被创建。因而,当被适当更新时,每一克隆片段131-139能用于可应用于相应片段151-153的数据库操作。

[0034] 在一实施例中,克隆片段131-139可配置为提供高水平的数据可用性。在此实施例中,来自151-153的每一组克隆片段可被指定为数据库操作的主克隆片段。该组中的其它克隆片段是用作方便可用的备份的次克隆片段。在图2中,克隆片段131,135和139示为主克隆片段,而其余的克隆片段被指定为次克隆片段。

[0035] 为提供高水平的数据可用性,组中每一克隆片段可包括在不同设备中,使得如果一个设备故障,另一设备中的次克隆片段能非常快地代替故障设备中的克隆片段作为主克隆片段。例如,克隆片段131-133可各自被包括在独立的设备中,使得如果包括主克隆片段131的设备发生故障,次克隆片段132-133的任一个可被指定为主克隆片段。

[0036] 管理克隆片段的数据库系统可在克隆片段上执行各种操作。这些操作一般使用标准的数据库操作来执行,诸如数据操纵语言(DML)语句或其它结构化查询语言(SQL)语句。更新和刷新克隆片段的示例操作将结合图3和4更详细地讨论。在一个示例实现中,操作可包括:

- [0037] 1. 创建克隆片段
- [0038] 克隆片段可创建为与数据库中的正常表或索引行集合没有区别。
- [0039] 2. 删除克隆片段
- [0040] 克隆能象数据库中的行集合一样被删除。
- [0041] 3. 完全初始化克隆片段的数据
- [0042] 克隆片段能被彻底从零开始初始化,以包含一个被载入克隆片段的新行集合。
- [0043] 4. 传播数据变化到克隆片段
- [0044] 主克隆片段的变化被传播到一个或多个次克隆片段。传播随着对主克隆片段的更新在同一事务上下文中发生。
- [0045] 5. 刷新过时的克隆片段
- [0046] 当克隆片段已经离线或者尚未从主克隆片段接收到更新的事务传播时,它被定义为过时的克隆片段。过时的克隆片段也能描述为陈旧的克隆片段。使过时的克隆回到在事务上与主片段克隆一致的过程称为刷新。
- [0047] 6. 读克隆片段

[0048] 克隆片段可以为了数据检索（表访问）或为了查找（索引访问）而被读取，就象正常表或索引被读取和访问一样。在此实现中，用户工作负载只从主克隆片段读取。使用该限制以简化避免系统中不必要的死锁的机制。然而，如果死锁不是问题，或者通过给定系统中的其它手段能够避免，则该限制可放宽。

[0049] 7. 更新克隆片段

[0050] 用户工作负载更新主克隆片段，并且数据库系统将这些变化传播和应用于相应于同一事务内的该主片段的次克隆。传播变化指将实质上相同的被应用于主克隆的 DML 操作应用于一个次克隆。

[0051] 图 3 示出更新数据库对象的克隆片段 131-139 的示例操作 300。数据库更新可包括任何类型的修改，诸如添加、删除和改变数据。当与数据库对象相关联的变化被确定时，数据库对象中需要更新的片段被标识。相应于被标识的片段的主克隆片段被更新。如图所示，操作 301-303 是分别更新主克隆片段 131、135 和 139 的操作。在主克隆片段 131、135 和 139 已经更新后，这些更新随后被传播到相应的次克隆片段。在图 3 中，操作 311-313 是更新相应于经更新的主克隆片段的次克隆片段的操作。

[0052] 通常，操作 301-303 和 313-313 是经由插入，更新和删除的标准数据库操作来实现的，它们实现 DML 语句语义。为实现一致性，更新主克隆片段的操作与更新相应于该主克隆片段的次片段的操作可配置为一个原子的操作集合。

[0053] 图 4 示出刷新与数据库对象相关联的过时的克隆片段 133 的示例操作 400。克隆片段是过时的，如果它已经不能执行与相应于该过时片段的数据库对象相关联的被传播的更新操作。例如，克隆片段 133 可能已经因各种原因而不能更新，如设备故障，丢失连接，系统更新等等。为了可用，必须通过刷新过程使次克隆片段 133 回到与主克隆片段 131 在事务上一致的状态。

[0054] 当克隆片段 133 返回在线并且变得能够执行更新操作时，克隆片段 133 基于包括在当前的主克隆片段（它包括所有当前和过去的更新）中的数据被刷新。例如，在图 4 中，用包括在主克隆片段 131 中的数据刷新克隆片段 133 来执行克隆刷新操作 413。如此，克隆片段 133 就用在克隆片段 133 不可用时发生的更新来刷新了。为了效率，克隆刷新操作 413 能经由 SQL 删除操作移除过时的行和 SQL 插入操作添加从主片段拷贝的新行来实现。这样一个实现使刷新操作能够经由与数据库系统中正常的 SQL 语句执行所使用的执行路径相似的执行路径来管理而不增加额外的过程。

[0055] 主克隆片段 131 的更新操作 401 在数据库对象的更新影响到相应于主克隆片段 131 的对象的一部分时发生。更新随后被传播到与主克隆片段 131 相关联的次克隆片段。如图 4 所示，次克隆片段 132 由与主克隆片段 131 相关联的操作 411 更新。更新克隆片段的需求可在刷新该克隆片段的同时发生。例如，对克隆片段 133 的更新可在克隆刷新操作 413 正在执行的同时发生。更新操作 412 能在克隆刷新操作 413 正在执行的同时在克隆片段 133 上执行。通常，用户的更新操作 412 是比克隆刷新操作 413 优先级更高的数据库操作。更新操作 412 和克隆刷新操作 413 可用各种方法和相对于彼此的时序来实现。例如，当必须更新时，在克隆片段 133 上的更新操作 413 可在克隆刷新 413 继续之前完成。而且，克隆刷新 413 被设计为同时最小化由针对任何给定的更新操作 412 进行锁定而引起的任何延迟的频率和时段。克隆刷新通过作为一系列小的事务批处理的运行来完成，这些批处理

占用并保持锁定很短的时间间隔。操作 412 只更新已经由操作 413 刷新的过时克隆中的行或条目，或者更新在过时克隆离线时主克隆中保持不变的过时克隆中的行或条目。即，防止操作 412 更新过时克隆中过时的行。不允许传播的更新操作在过时克隆中的不一致行看上去象是最新。该限制在其中整行用更新 412 来传播的实现中可放宽。然而，性能原因倾向于不鼓励为所有 DML 操作传播整行的操作 412，无论是否只有少数列值有变化。克隆刷新操作 413 的责任是把新的最新的行引入过时的克隆。

[0056] 图 4 所示的刷新过时克隆片段的操作 400 使数据库系统能够将过时的克隆片段刷新回与相应的主片段在事务上一致，同时使用标准数据库操作保持该主克隆的在线状态。使用标准的数据库操作，诸如那些表示 DML 语句的操作，允许将克隆刷新作为数据库系统上额外的但标准的 SQL 语句负载来管理，而无需特殊的过程。操作 400 还允许刷新过时的克隆片段而不必专有地锁定整个片段也不必延长刷新过程从而需要追上刷新过程期间的更新，因为那些更新在刷新操作 413 进行的同时被继续应用。重要的是，操作 400 还使过时的次克隆片段能够有效地被更新，从而快速变得能够符合成为主克隆片段的要求。

[0057] 在一示例实现中，克隆刷新操作 413 包括刷新操作的多个小的批处理。使用小的刷新操作批处理，避免大时间段地阻塞并发用户工作负载。因而，刷新过程以增量方式执行，允许过程与用户工作负载同时在线存在。

[0058] 图 4 所示的克隆刷新操作 413 的多个实例能针对任何数据库对象的相同或不同片段内的过时克隆片段同时运行。即，这样的刷新操作是彼此独立的。

[0059] 图 5 示出标识克隆片段 500 的示例数据结构 511-514。数据结构 500 用于标识相应于数据库对象的克隆片段。在此例中，数据结构标识相应于与数据库相关联的表的克隆片段 500。如图 5 所示，克隆片段 500 的标识符可包括表标识符 511，分区标识符 512，片段标识符 513 和克隆标识符 514。表标识符 511 标识克隆片段 500 相对应的数据库对象（在此情形下为表）。数据库对象可为了便利或性能原因而被划分成分区以分离数据。分区标识符 512 标识克隆片段 500 相对应的分区。

[0060] 管理数据库对象的数据库系统被配置为自动地将数据库对象分离到片段中并克隆这些片段。片段标识符 513 标识克隆片段 500 相对应的数据库对象的特定片段。克隆标识符 514 标识与数据库对象的特定片段相关联的多个克隆片段中的克隆片段 500。

[0061] 图 6 示出标识数据库对象的克隆片段中的记录 600 的示例性数据结构 611-612。克隆片段可包括数据库的记录（或索引条目）。例如，克隆片段可包括具体化为与数据库相关联的表的一部分中的行的记录。记录 600 中的数据结构 611-612 使克隆片段中的记录 600 和该记录的更新状态能够被标识。

[0062] 如图 6 所示，记录 600 可包括克隆记录标识符 611 和克隆更新标识符 612。克隆记录标识符 611 唯一地标识相应的表或索引内的记录 600。索引记录中的克隆记录标识符用来自相应的克隆数据片段的更新来传播。用户定义的唯一键可用作克隆记录标识符或者可添加系统定义的唯一键（或补充到用户定义的键），使得复合键足以唯一地用作克隆记录标识符。例如，片段标识符可为了提供附加的唯一性而添加到键。克隆更新标识符标识记录 600 的更新状态。在此示例实现中，克隆记录标识符 611 应当在给定表的上下文中是唯一的。随着时间过去对于一个给定的数据库对象，克隆更新标识符 612 仅必须相对于具有同一克隆记录标识符 611 的其它记录的更新状态，唯一地标识一个记录的更新状态。

[0063] 在一实施例中,数据库系统包括主克隆片段和次克隆片段,且克隆片段包括行。克隆记录标识符 611 和克隆更新标识符 612 可作为列包括在克隆片段的行中。克隆记录标识符 611 和克隆更新标识符 612 的列可包括在次克隆片段中,并且在次克隆片段在事务上是一致的时候,这些列可包括与它们在主克隆片段上所包括的相同的值。因而,克隆记录标识符 611 允许在主与次克隆片段之间映射行,而克隆更新标识符 612 允许验证这些行是否一致。

[0064] 过时的克隆片段的刷新过程使用克隆记录标识符 (CRID) 和克隆更新标识符 (CUID) 来确定在克隆片段中的特定记录是否应当刷新。例如,过时的克隆片段中的记录可包括:

[0065] CRID = x 和 CUID = y

[0066] 与过时的克隆片段相关联的主克隆片段中相应的记录可包括:

[0067] CRID = x 和 CUID = z

[0068] 刷新过程使用 CRID 来标识过时的克隆片段中的每一记录并且在主克隆片段中定位相应的记录,如果它存在的话。刷新过程随后将过时的克隆片段中的每一这样的记录与主克隆片段中相应的记录比较,如果它存在的话,来确定过时的克隆片段中的记录是否应当被更新。

[0069] 数据库对象的片段可以是任何类型的片段,诸如数据片段,索引片段,等等。对于索引片段,索引中的每一行通过将它的数据片段标识符作为索引键的一部分来存储以指明该行来自何处。索引本身不必知道基表片段的克隆中的哪一个是当前的主克隆。因为对于给定的片段存在一个主克隆片段,因此索引记录为访问目的可以在任何给定时间映射到特定的片段。因而,索引片段可包括分离的次行集合(或索引条目)的共存集合,它们指向表的不同片段。只要每个这样的次行集合与同一索引片段中的其它行集合分开来处理,则上述假设和技术继续有效。对于实现,锁定相应于任何特定基表片段的索引片段中的行,应当不导致对索引片段中其它行(它们不相应用于该基表片段)的任何锁定。

[0070] 索引片段可以与数据片段相似地克隆。为了更新传播,克隆索引片段作为基表片段的相同集合上的附加索引来处理。即,对所有克隆索引片段的更新直接从其上建立克隆索引片段的基表的主克隆数据片段传播。尽管有可能将更新从一个索引片段传播到另一个,但这个过程可引入一个附加的潜伏步骤。如果相应于给定数据片段中记录的给定索引片段中的索引条目集合作为次行集合来处理,则相应的主行集合实质上与基表片段相同。为了更新传播,索引片段的克隆中的每个次行集合因而独立于该同一克隆内的所有其它次行集合。

[0071] 图 7 示出更新克隆索引片段和克隆数据片段的示例过程。如上所述,数据片段的最新(或当前)的次克隆片段实质上与给定数据片段的主克隆片段相同。同样,索引片段的最新次克隆片段实质上与该索引片段的主克隆片段相同。允许克隆索引片段包含指向在多于一个数据片段中的记录的索引条目。因此,克隆索引片段可从多于一个克隆数据片段接收被传播的更新。

[0072] 图 7 中的示例示出索引片段 I1,它包含引用两个不同数据片段 D1 和 D2 中的记录的索引条目。在此例中,主克隆索引片段 I1.a 705 和次克隆索引片段 I1.b 706 都直接从主克隆数据片段 D1.a 702 和主克隆数据片段 D2.a 703 两者接收被传播的更新。

[0073] 数据片段与索引片段之间的映射代表一种被称为物理数据库设计的标准数据库操作。对于表的任何给定的被索引列，列值可基于索引的划分分区与划分片段定义来离散地映射到一个或多个索引片段。表和索引的划分分区定义可指定将记录分成集合的任何传统的数据库方法，例如，按照值的范围，值的散列或者经由循环指派（round-robin assignments）。系统定义的划分片段定义是经由在表或索引的键上进行散列来完成的。然而，如果基于散列的划分片段以非常不成比例的数据偏斜方式来用行填充片段，则可以使用将行循环指派给片段的方法。

[0074] 当克隆索引片段变得过时且需要刷新时，有两种可能的方法来完成刷新。如果主克隆索引片段存在，则过时的次克隆索引片段可直接从该主克隆索引片段来刷新。或者，过时的次克隆索引片段可直接从被索引的表的主克隆数据片段集合来刷新。在后一情形中，只有在主克隆数据片段中的那些记录被用于其键值映射到正被刷新的索引片段的刷新。

[0075] 在一示例实现中，克隆更新标识符（CUID）值可在每个表的基础上保存。即，对于更新给定主克隆片段的 DML 语句，从专用于该片段的表的元数据中读取 CUID 值。由该语句更新的记录被指派该 CUID 值。当对这些记录的更新传播到次克隆片段时，所指派的新 CUID 也被传播。

[0076] 在此示例实现中，随着时间过去，CUID 值对于给定表是唯一的。当前的 CUID 值在描述给定表的元数据中事务上是持久的。

[0077] 在 CUID 值命中从最大值回绕到最小值的点的时候，可使用各种技术让 CUID 值复位到一个一致的最小值。给定 CUID 值的字节范围（例如约 6 个字节），这种情形将在历史的时间尺度上发生。一种用于复位 CUID 值的简单技术是在表上加专用锁，并复位该表的主克隆片段中所有记录的 CUID 值为最小 CUID 值。然后，正常的更新传播机制将有效地复位次克隆片段中的 CUID。

[0078] 表的 CUID 值每次在表的片段变成离线时增加。该实现将在下面结合图 12 讨论。

[0079] 其它 CUID 指派策略是可能的，但各自具有影响克隆刷新的基本技术的可能。例如，事务 ID 可用作 CUID 值的基础，但要求某种程度不利的克隆刷新技术，以便确保更新传播和正确地克隆刷新交互。明确地，在 WaitForPropagation（等待传播）步骤（下面描述）中，基本克隆刷新技术将不得不等待所有当前活动的事务完成，而不是只等待所有当前 DML 语句完成。

[0080] 一般而言，如果管理 CUID 值的技术能够满足下列要求，则可以实现：

[0081] a) 对于在片段内使用的给定 CRID 值，在事务上一致的次克隆片段中由该 CRID 值标识的所有记录必须具有与在主克隆片段中由该 CRID 值标识的记录中的 CUID 值相同的 CUID 值。

[0082] b) 对于给定的片段，如果 CRID 存在于过时的次克隆片段（在刷新前）中，也存在于主克隆片段中，则 CUID 仅当该行在主克隆片段中自次克隆片段变成过时起未更新过时才相同。

[0083] 下面示出更新克隆片段的示例 DML 语句。示例 DML 语句包括：Sales. 2004. 1. a（克隆记录标识符（CRID）:5 和 3）中使用 CUID 值为 3 更新的两个记录。

[0084] 示例 Sales. 2004. 1. a 主克隆片段示于表 1。

[0085]

CRID	CUID	其它列
1	2
5	<u>3</u>
7	1
3	<u>3</u>
8	1

[0086] 表 1. 主克隆片段 Sales. 2004. 1. a

[0087] 在主克隆片段 Sales. 2004. 1. a 更新后, 这些更新被传播并应用到次克隆片段 Sales. 2004. 1. b, 如表 2 所示。

[0088]

CRID	CUID	其它列
1	2
5	<u>2</u> <u>3</u>
7	1
3	<u>+</u> <u>3</u>
8	1

[0089] 表 2. 主克隆片段 Sales. 2004. 1. b

[0090] 在给定时刻, 可用片段具有一个主克隆片段 (克隆刷新的潜在源) 和 N 个次克隆 (刷新的潜在目标)。其行在事务上与主克隆片段中相应行不一致的次克隆片段是陈旧或过时的克隆片段。过时克隆片段内的行 (记录或索引条目) 可以处于以下示例状态之一。术语“主行”指主克隆片段中的行。过时的克隆片段内的行可以是 :

[0091] 1. 一致

[0092] 主行在过时的次克隆片段离线时没有变化。或者, 刷新过程已经更新了过时的次克隆片段中的行, 它与相应的主行一致。次克隆片段中一致的行相应于主克隆片段中具有相同 CRID 和 CUID 值的行。

[0093] 2. 不一致

[0094] 主行在过时的次克隆片段离线时被更新。在过时的次克隆片段中不一致的行相应于主克隆片段中具有相同 CRID 值的行, 但具有不同的 CUID 值。

[0095] 3. 缺少

[0096] 在过时的次克隆片段离线时, 创建了新主行。过时的次克隆片段中缺少的行描述这样的情形 : 存在具有一 CRID 值的主行, 在过时的次克隆片段中无具有该相同 CRID 值的相

应行。

[0097] 4. 额外

[0098] 在过时的次克隆片段离线时,主行被删除。过时的次克隆片段中额外的行描述这样的情形:过时的次克隆片段中存在具有一 CRID 值的行,主克隆片段中无具有该相同 CRID 值的相应行。

[0099] 在刷新技术的早期阶段,过时克隆的不一致行的 CRID 捕捉到一个临时文件 (StaleCloneCRIDs (过时克隆 CRID)) 中。该临时文件具有一个列 Stale_CRID,其值标识需要刷新的记录,例如不一致、缺少或额外的记录。在临时文件 (StaleCloneCRIDs) 的一实现中,可添加一个附加的列 Batch_ID (批处理 ID),并用一个值初始化它,该值简化了以许多小的递增的批处理操作方式访问 StaleCloneCRIDs 内分立的行集合的过程。

[0100] 图 8 示出在刷新过时克隆片段时的状态过程。如图 8 所示,次克隆片段 805 从离线状态进行到预刷新 (PreRefresh) 状态。当过时克隆在预刷新状态时,可为了优化刷新操作的性能而对过时克隆应用一些操作。接着,次克隆片段 805 视为在线并且被设置在刷新中 (InRefresh) 状态。过时克隆片段在被刷新时保持处于刷新中状态。在此状态中,次克隆片段 805 可按从主克隆片段 803 传播的变化来更新。

[0101] 在刷新过程已经执行后,次克隆片段 805 进入当前状态。一旦次克隆片段 805 进入当前状态,它就符合变成主克隆片段的要求了。

[0102] 图 11 示出刷新数据库对象的过时克隆片段的示例刷新过程 1100。

[0103] 0. StaleCloneOnline (过时克隆在线) 1150 :

[0104] 在框 1102,当过时克隆是克隆刷新的目标时(例如,当它在离线之后变成可由系统访问时),将它设置在预刷新状态。接着,在框 1104,在克隆片段处于预刷新状态时,执行预扫描 (pre-pass) 优化。这样的优化被设计用于优化刷新过程 1100 的其余部分的后续性能。

[0105] 1. EnableUpdatePropagation (启用更新传播) 1160 :

[0106] 在框 1106,将过时克隆片段标记为正处于刷新中状态。然后在框 1108,启用从主克隆片段到过时的克隆片段的更新传播。将所传播的更新应用于过时的克隆片段可以是选择性的。例如,所传播更新的应用仅在那些记录已经被刷新或者可以确定为不需要刷新的时候执行。即,所传播的更新仅应用于其 CUID 值对于给定的 CUID 值符合主克隆片段中相应记录的 CUID 值的记录。在另一示例实现中,如果整行值由更新传播,则更新可以在克隆片段正在被刷新时被传播并应用于过时克隆片段的任何记录。

[0107] 2. WaitForPropagation (等待传播) 1110 :

[0108] 在下一框 1110,刷新过程 1100 等待直到主克隆片段上的所有活动的 DML 语句已经开始向过时克隆片段传播更新为止。(或者使用一个等价的机制来确保对于主克隆片段的所有后续 DML 被一致地传播到过时的克隆片段。) 框 1110 可以通过等待所有当前执行的 DML 语句结束来完成,由此保证任何新激活的 DML 语句传播更新至所有在线克隆片段,包括处于刷新中状态的过时克隆片段。

[0109] 3. BuildStaleCloneCRIDs (建立过时克隆 CRID) 1112 :

[0110] 在框 1112,在读确认 (ReadCommitted) 模式中,创建并初始化 StaleCloneCRIDs 临时表。保证 StaleCloneCRIDs 覆盖过时克隆片段中所有非最新的行(不一致,缺少和额外

的行),但由于运行在读确认隔离模式中,StaleCloneCRIDs 偶然会包括实际上一致的 CRID 行。在填充 StaleCloneCRIDs 临时表的一实现中,可在主克隆片段和过时克隆片段的相应克隆记录标识符 (CRID) 列上使用 SQL 外连接操作,其中相应的克隆更新标识符 (CUID) 列值不匹配或者为空 (null)。下面的伪代码提供一个这样的外连接的示例。注意在此例中,将附加的 Batch_Id 列添加到 StaleCloneCRIDs 表,以提供一个可用于创建给定大小的刷新批处理的示例性基础。

```
[0111]   SELECT
[0112]     (case when(p.crid 不为空)then p.crid else c.crid end)as crid,
[0113]     IDENTITY(int,1,1)AS Batch_Id
[0114]   into
[0115]     StaleCloneCRIDs
[0116]   FROM
[0117]     (选择来自 primary_fragment_clone( 主片段克隆 ) 的 crid、cuid)as p
[0118]     FULL OUTER JOIN
[0119]       (选择来自 stale_fragment_clone( 过时片段克隆 ) 的 crid、cuid)as c
[0120]     ON(p.crid = c.crid)
[0121]   where
[0122]     p.cuid<>c.cuid OR
[0123]     (p.crid 为空)OR
[0124]     (c.crid 为空)
```

[0125] 4. MakeCurrent(使成为当前)1170 :

[0126] 图 17 示出了使过时克隆成为当前克隆的示例过程 1170。过程 1170 是图 11 所示的示例刷新过程 1100 的一部分。

[0127] 在框 1114, 初始化 StaleCloneCRIDs 临时表中 Stale_CRIDs(过时 CRID) 的第一个小的批处理 (BatchOfStaleCRIDs(过时 CRID 的批处理))。在框 1114 的一示例中, BatchOfStaleCRIDs 定义为 StaleCloneCRIDs 临时表中其 Batch_Id 列值在参数值 @start_Id 与 @end_Id 之间的行的 CRID。初始的实际 @start_Id value 被设置为 StaleCloneCRIDs 临时表中最小的 Batch_Id 列值 (例如,由 IDENTITY(身份) 功能用种子值 1 来初始化的 Batch_Id)。初始的实际的 @end_Id 值随后被设置为所需的 BatchSize(批处理大小) 的值。BatchSize 值可以变化以完成不同的刷新目标。较小的 BatchSize 值能提供与正在进行的用户工作负载的较大并发性。较大的 BatchSize 值有可能放弃较好刷新性能。在判决框 1116, 进行检查以确定 BatchOfStaleCRIDs 是否为空。如果 BatchOfStaleCRIDs 为空, 则刷新过程 1100 进行到框 1118, 并将被刷新的克隆片段标记为正在当前 (Current) 状态。处于当前状态的次克隆片段符合成为主克隆片段的要求。然后, 过程 1100 在框 1120 退出。

[0128] 如果在判决框 1116, 发现 BatchOfStaleCRIDs 不为空, 则 :

- [0129] a. 在框 1122, 在重复读 (Repeated Read) 模式中开始新的事务。
- [0130] b. 在框 1124, 在主克隆片段中由 BatchOfStaleCRIDs 标识的行上设置读锁。该步骤不是功能所要求的, 但对于避免与所传播的更新发生可能的死锁的目的是重要的。
- [0131] c. 在框 1126, 在过时的克隆片段中由 BatchOfStaleCRIDs 中的 CRID 所标识的行

被删除。下列伪代码描述从过时的克隆片段中删除陈旧的行的一个示例实现。@start_Id 和 @end_Id 的实际值是早先在 MakeCurrent 过程 1170 (在框 1114 或框 1132) 中所设置的值。

```
[0132] delete from stale_fragment_clone  
[0133] where exists (select*from StaleCloneCRIDs as s  
[0134]         where stale_fragment_clone.crid = s.crid and  
[0135]         (s.Batch_Id 在 @start_Id 和 @end_Id 之间))
```

[0136] d. 在框 1128, 主克隆中由 BatchOfStaleCRIDs 标识的行被插入到过时的克隆片段中。下面是从主片段将新行插入过时的克隆片段的一个示例实现的示例伪代码描述。@start_Id 和 @end_Id 的实际值是早先在 MakeCurrent 过程 1170 (在框 1114 或框 1132) 中设置的值。

```
[0137] insert into stale_fragment_clone  
[0138] select*from primary_fragment_clone as p  
[0139] where exists  
[0140]         (select*from StaleCloneCRIDs as s  
[0141]         where p.crid = s.crid and  
[0142]         (s.Batcd_Id 在 @start_Id 和 @end_Id 之间))
```

[0143] e. 在框 1130, 事务被确认。

[0144] f. 在框 1132, 来自 StaleCloneCRIDs 临时表的下一批 CRID 被标识为下一个 BatchOfStaleCRIDs。@start_Id 和 @end_Id 所使用的实际值各自增加所需 BatchSize 值。

[0145] g. 刷新过程 1100 随后回到框 1116。

[0146] 如果在 BatchOfStaleC 则 Ds 的执行过程中发生错误, 则事务中止并且刷新过程 1100 退出。

[0147] 如果刷新过程 1100 没有成功结束, 则过时的克隆片段移到预刷新状态 (或者移到离线状态, 如果对过时克隆片段的访问彻底丢失的话)。

[0148] 示例的克隆片段刷新技术进行连续的正向过程, 同时经由小而短的事务来最小化与现有工作负载的冲突。上述克隆刷新技术也可应用于克隆索引片段, 就如应用于克隆数据片段一样。在一示例实现中, 过时的次克隆索引片段可以从相应的主克隆索引片段来刷新。

[0149] 如果基表片段的所有克隆数据片段变成离线, 则每一索引片段中相应于该片段的行也实际上是离线的。当基表主克隆片段回到离线时, 刷新技术在任何过时的表克隆数据片段上操作, 也在任何过时的克隆索引片段上操作。在这点上, 更新操作被传播到正被刷新的过时的次克隆索引与数据片段。同样, 如果索引片段的所有克隆片段变成离线, 则使用该索引片段的标识符访问路径将是离线的。过时的克隆索引片段在它再次变成可访问时经由克隆刷新而恢复。在一示例实现中, 过时的克隆索引片段直接使用基表片段的行来刷新。然而, 过时的次克隆索引片段也可从相应的主克隆索引片段来刷新。

[0150] 上述数据库片段克隆和管理机制的性能可以通过下面讨论的示例优化来提高:

[0151] a) 克隆刷新的预刷新步骤

[0152] 在运行克隆刷新技术之前, 可以在完全处于读确认模式的情况下运行相同的刷新

技术（图 11 针对过程 1100 所述）。（例如，当在预刷新步骤期间运行时，过程 1100 的框 1122 在读确认模式中开始事务。）在缺少许多并行的更新时，该预扫描将清除克隆片段中的绝大多数行，留下少量不一致的行，并且有可能再引入少量不一致的行。这个思想是在完成该预扫描之后，在可重复读模式中的严格的刷新阶段（图 17 中的过程 1170）将运行得快得多。较少的行将被锁定（由于较少的批处理）一个总的较短的时间段，因而增加了系统的并发性。

[0153] 当主克隆片段具有多个需要刷新的次克隆片段时，在所有次克隆片段上的刷新扫描（pass）可以彼此独立地运行。并行的程度可以基于克隆片段多快地回到在线来控制，并针对强加于系统上的并发性命中的多少来折衷。

[0154] b) 对于次克隆片段的故障的事务弹性

[0155] 当对次克隆片段的传播的更新因系统原因而失败时，克隆片段将被标记为离线。正在进行的事务的效果被直接保存在主克隆片段的行中。对于保存先前事务的效果的目的，不需要附加的未决工作队列。为便于错误处理，其更新正被传播的语句可被重新运行（roll back）。但如果对一个故障的次克隆片段的更新传播的失败对于语句的执行没有其它副作用，则这是没必要的。该语句的更新将已被应用于并保存在主克隆片段中。

[0156] c) 对于主克隆片段的故障的事务弹性

[0157] 当主克隆片段故障时，如果有一个在线且一致的次克隆片段可被立即指定为新主克隆片段，则访问该克隆片段的用户事务不必中止。然而，任何当前访问该发生故障的主克隆片段的语句将被重新运行。

[0158] d) 克隆刷新弹性

[0159] 如果对次克隆片段中的一个记录的克隆刷新更新因系统原因而失败，则克隆刷新通过可从开始重启，重试当前活动的 MakeCurrent 批处理或者重做 StaleCloneCRIDs 的整个 MakeCurrent 处理过程。

[0160] 图 9 是带有克隆的数据库片段的示例数据库系统 900。数据库系统 900 一般包括多个计算设备 911–914，用于分配工作负载和为访问数据库对象以及负载平衡提供可靠的可访问性。在示例数据库中的对象被划分成克隆片段，它们在计算设备 911–914 之间划分。如此，计算设备 911–914 能并发地在示例数据库的对象的不同部分上执行操作。计算设备 911–914 可用任何类型的数据库系统来配置，如 SQL 服务器。

[0161] 如图所示，计算设备 911–914 可包括主克隆片段 923–926 和次克隆片段 934–937。在一实施例中，主克隆片段 923–926 由计算设备 911–914 用于执行有关生产性的任务，如添加、删除或修改数据库对象、查询、报表等等。次克隆片段 934–937 用作主克隆片段 923–926 的备份。

[0162] 通常，在每一计算设备中的主克隆片段不相等于该设备中的次克隆片段。如果计算设备 911–914 之一发生故障，则另一计算设备能使用相等于由该故障设备管理的主片段的次克隆片段来接管操作。如此，数据库系统 900 能确保数据库对象的片段保持可用，只要至少该片段的至少一个克隆保持可访问即可。

[0163] 在另一实施例中，计算设备 911–914 可包括重叠的主克隆片段。如此，多个计算设备可在数据库对象的相同部分上执行操作，如查询。

[0164] 图 10 示出更新数据库对象的示例过程 1000。数据库对象可包括整个数据库的数

据或数据库的一部分,如表,索引等等。更新可涉及一或多个片段的修改。在框 1001,数据库对象的受更新影响的片段被标识,并且它们形成名为 SetOfFragments(片段的集合)的集合,即用于过程 1000 的一个本地数据结构。在此集合中的每一元素最初被标记为未处理(这不改变实际的片段,只改变本地数据结构中的元素)。过程 1000 继续至判决框 1002,其中作出集合 SetOfFragments 中是否仍存在未处理的元素的判断。如果否,则过程 1000 继续至终止框 1012,其中过程终止。

[0165] 返回到判决框 1002,如果集合 SetOfFragments 中仍有未处理的元素,则过程 1000 继续至框 1003,其中未处理的元素是从集合 SetOfFragments 获得的,并且这代表应当被更新的下一个片段。在框 1004,确定相应的主克隆片段,并且主克隆片段内需要更新的记录的集合被标识,形成名为 SetOfRecords(记录的集合)的集合,即用于过程 1000 的一个本地数据结构。在该集合中的每一元素最初被标记为未处理(这不改变实际的记录,只改变本地数据结构中的元素)。在判决框 1005,作出集合 SetOfRecords 中是否仍存在任何未处理的元素的判断。如果否,过程 100 继续至框 1006,其中集合 SetOfFragments 中的当前元素被标记为已处理(这不改变实际的片段,只改变本地数据结构中的元素)。过程 1000 随后回到判决框 1002。

[0166] 返回到判决框 1005,如果在集合 SetOfRecords 中仍存在未处理的元素,则过程 1000 继续至框 1007,其中从集合 SetOfRecords 获得一个未处理的元素。该元素代表应当更新的下一个记录。在框 1008,获得该记录的 CUID。(CUID 值的维护与指派在本文其它地方参考图 12 和 13 进一步描述。)

[0167] 过程 1000 继续至框 1009,其中用新值以及新 CUID 更新该记录。在框 1010,旧的 CUID 以及被更新的记录(包含新 CUID)被传播至所有次克隆数据片段,以及传播至包含该记录的拷贝的所有主和次克隆索引片段。图 16 更详细地描述该框。

[0168] 在更新已被传播和应用之后,过程 1000 继续至框 1011,其中集合 SetOfRecords 中的当前元素被标记为已处理(这不改变实际记录,只改变本地数据结构中的元素)。过程 1000 随后回到判决框 1005。

[0169] 图 12 示出修改克隆更新标识符(CUID)的示例过程 1200。过程 1200 可针对每一数据库对象实现,如数据库的表。在框 1203,过程 1200 等待与表相关联的克隆片段的配置的变化。在判决框 1205,作出一个克隆片段已经变成不可用的判断。如果克隆片段全部可用,则过程 1200 返回至框 1203 的等待状态。

[0170] 返回到判决框 1205,如果一个克隆片段已经变成不可用,则过程 1200 移到框 1207,其中表的 CUID 值被增加。在框 1209,表的新 CUID 值在事务上被持久化。

[0171] 图 13 示出一个示例过程 1300,用于指派根据图 12 所示的过程 1200 分配的 CUID 值。数据库系统可被配置为执行过程 1300 用于在数据库上操作的语句。在框 1302,过程准备对数据库执行一个新的语句。在判决框 1304,作出新语句是否为 DML 语句的判断,它一般包括诸如在数据库中插入,删除和修改项的操作。如果语句不是 DML 语句,则过程 1300 前进到框 1320,其中对数据库执行非 DML 语句。然后,过程回到框 1302。

[0172] 返回判决框 1304,如果语句是 DML 语句,则过程 1300 在框 1306 继续,其中读取正被更新的数据库表的 CUID 值。在框 1308,将该 CUID 值与该 DML 语句相关联。在框 1310,CUID 值存储到由 DML 语句更新的每一主克隆片段的每一行中。在框 1312, CUID 值被存储

到受 DML 语句的传播的更新影响的克隆片段的每一行或索引条目中。CUID 值可通过对克隆数据和索引片段的更新来传播。CUID 值的示例传播路径将结合图 14 讨论。当 CUID 值已经被传播时, 过程 1300 返回至框 1302。

[0173] 图 14 示出 CUID 值的示例传播路径。在图 14 中, 主克隆数据片段 1403 已经实现了 DML 语句中所包括的更新。如上所述, 与 DML 语句相关联的 CUID 值存储在由主克隆数据片段 1403 中由 DML 语句更新的每一行中。主克隆数据片段 1403 可通过路径 1411 连同对次克隆数据片段 1405 的更新传播 CUID 值。具体地, 与 DML 语句相关联的 CUID 值被指派给受更新传播影响的每一行。在一实现中, CUID 值从主克隆数据片段分别通过路径 1412 和 1413 传播到主克隆索引片段 1407 和次克隆索引片段 1409。在另一实现中, 代替用路径 1413 传播 CUID 值, CUID 值从主克隆索引片段 1407 通过路径 1415 传播至次克隆索引片段 1409。

[0174] 图 16 示出一个示例过程 1600, 用于将经更新的记录和旧的 CUID 值从主克隆片段传播至所有次克隆数据片段, 以及传播至包含该记录的拷贝的所有克隆索引片段。这些接收传播的更新的克隆片段可以处于当前状态, 或者处于刷新中状态。过程 1600 一般形成在本文另一部分描述的一个较大更新过程的一个步骤。在框 1601, 包含该记录的拷贝的所有次克隆数据片段以及克隆索引片段 (包括主或次两者) 被标识。这些克隆片段形成名为 SetOfSecondaries (次 (克隆片段) 的集合) 的集合, 即用于过程 1600 的一个本地数据结构。在该集合中的每一元素最初被标记为未处理 (这不改变实际的克隆片段, 只改变本地数据结构中的元素)。过程 1600 继续至判决框 1602, 其中作出在集合 SetOfSecondaries 中是否仍有任何未处理的元素的判断。如果否, 则过程 1600 继续到终止框 1603, 其中过程终止, 并且包括过程 1600 为一个步骤的任何较大过程, 随后继续至该较大过程的下一步骤。

[0175] 返回到判决框 1602, 如果在集合 SetOfSecondaries 中仍有未处理的元素, 则过程 1600 继续至框 1604, 其中从集合 SetOfSecondaries 获得一个未处理的元素。该元素代表应当被更新的下一个次克隆片段。该次克隆片段可以是次克隆数据片段, 主克隆索引片段或者次克隆索引片段。在判决框 1605, 作出在次克隆片段中是否存在具有与主克隆片段的记录相同的 CRID 的记录并且其 CUID 要符合主克隆片段的旧的 CUID 值的判断。如果是, 则过程 1600 继续至框 1606, 其中用新值和新 CUID 更新该记录。过程 1600 随后继续至框 1609。

[0176] 返回到判决框 1605, 如果没有找到匹配的记录, 则过程 1600 继续至判决框 1607, 其中作出次克隆片段是否处于刷新中状态的判断。如果否, 则在框 1608, 次克隆片段被标记为离线, 因为它不再与主克隆片段一致。过程 1600 随后继续至框 1609。

[0177] 返回到判决框 1607, 如果次克隆片段处于刷新中状态, 则过程 1600 继续至框 1609, 其中集合 SetOfSecondaries 中的当前元素被标记为已处理 (这不改变实际的克隆片段, 只改变本地数据结构中的元素)。过程 1600 随后回到判决框 1602。

[0178] 图 15 示出实现所述系统和方法的示例计算机设备 1500。在其大多数基本配置中, 计算设备 1500 一般包括至少一个中央处理单元 (CPU) 1505 和存储器 1510。

[0179] 取决于计算设备的确切配置和类型, 存储器 1510 可以是易失性 (如 RAM), 非易失性 (如 ROM, 闪存等) 或两者的组合。另外, 计算设备 1500 也可具有另外的特征 / 功能。例如, 计算设备 1500 可包括多个 CPU。所述方法可用任何方式由计算设备 1500 中的任何处理单元来执行。例如, 所述过程可由多个 CPU 并行地执行。

[0180] 计算设备 1500 也可包括另外的存储 (可移动和 / 或不可移动), 包括但不限于羊

绒或光盘或带。这样的另外的存储在图 15 中由存储 1515 例示。计算机存储介质包括用存储信息如计算机可读指令、数据结构、程序模块或其它数据的任何方法或技术实现的易失性与非易失性、可移动和不可移动的介质。存储器 1510 和存储 1515 都是计算机存储介质的示例。计算机存储介质包括但不限于, RAM, ROM, EEPROM, 闪存或其它存储器技术, CD-ROM, 数字多功能盘 (DVD) 或其它光存储, 磁带盒, 磁带, 磁盘存储或其它磁存储设备, 或可用于存储信息并可由计算设备 1500 访问的任何其它介质。任何这样的计算机存储介质可以是计算设备 1500 的一部分。

[0181] 计算设备 1500 也可包含允许该设备与其它设备通信的通信设备 1540。通信设备 1540 是通信介质的一个示例。通信介质一般体现为在经调制的数据信号如载波或其它传输机制中的计算机可读指令、数据结构、程序模块或其它数据，并包括任何信息传递介质。术语“已调制的数据信号”指一种信号，其一或多个特征以将信息编码在该信号中的方式被设置或改变。作为示例，且非限制，通信介质包括有线的介质，如有线网络或直接线接连接，以及无线介质，如声音、RF、红外以及其它无线介质。在此使用的术语计算机可读介质或设备可读介质包括计算机存储介质和通信介质两者。所述方法可用任何形式如数据、计算机可执行指令等编码在任何计算机可读介质中。

[0182] 计算设备 1500 也可具有输入设备 1535 如键盘、鼠标、笔、语音输入设备、触摸输入设备等。输出设备 1530 如显示器、扬声器、打印机等也可包括。所有这些设备在本领域是公知的并且不必详细讨论。

[0183] 本领域的技术人员将认识到，用于存储程序指令的存储设备可分布在网络上。例如，远程计算机可存储描述为软件的过程的示例。本地或终端计算机可访问远程计算机并下软件的一部分或全部来运行程序。或者，本地计算机可按需下载软件的片断，或者在本地终端处执行一些软件指令而在远程计算机（或计算机网络）执行一些软件指令。本领域的技术人员将认识到，通过利用本领域技术人员已知的常规技术，软件指令的全部或一部分可由专用电路如 DSP、可编程逻辑阵列等来携带。

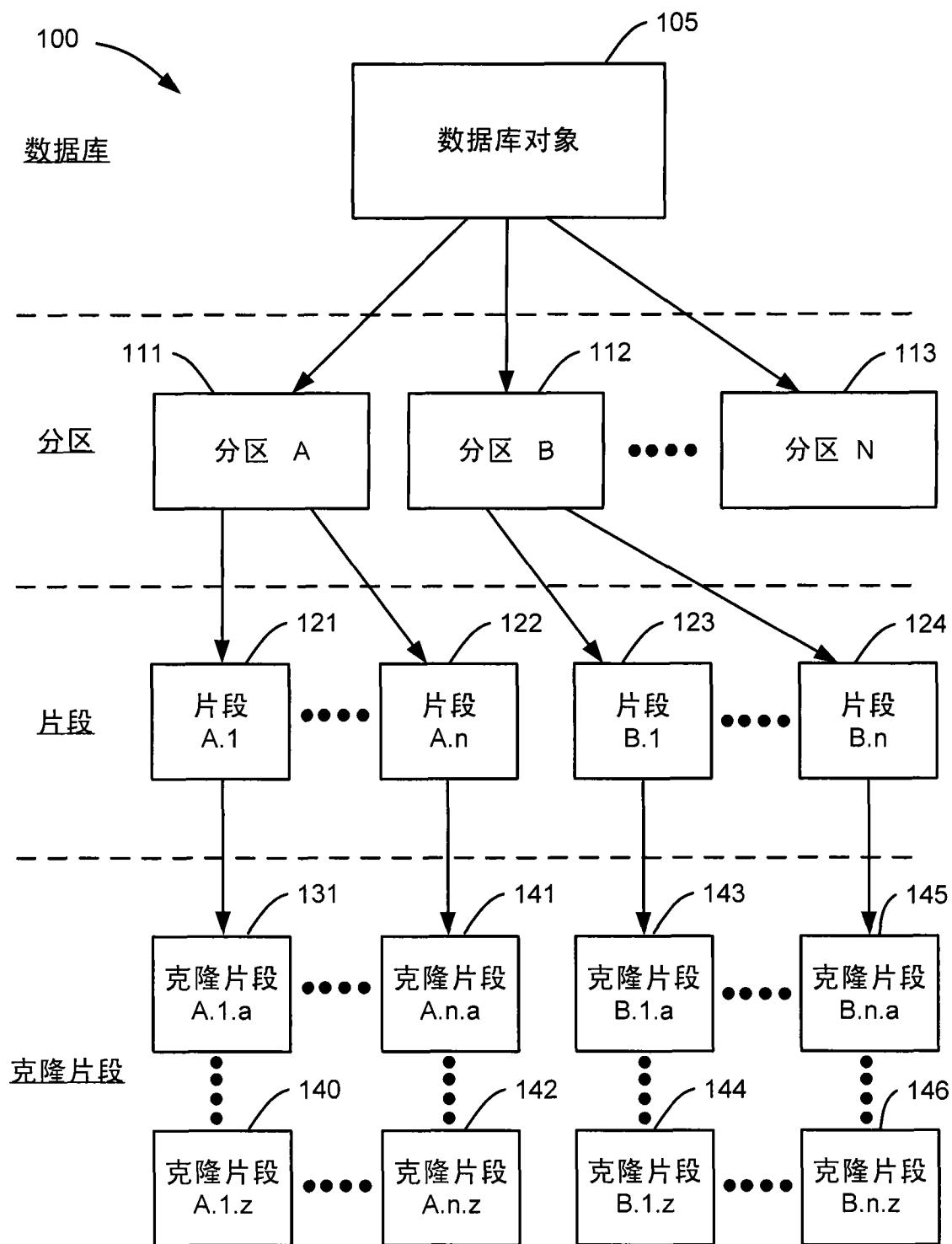


图 1

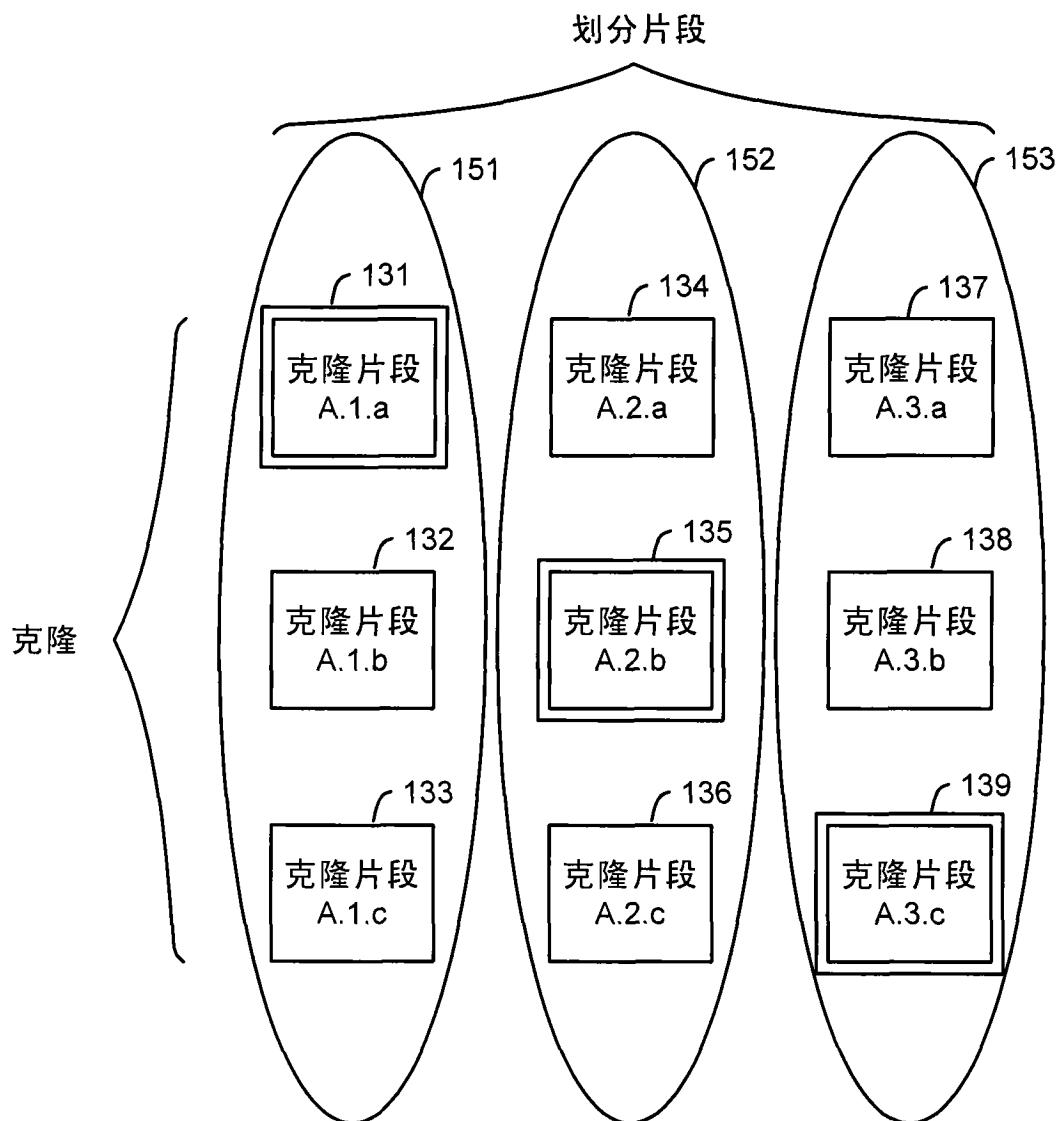


图 2

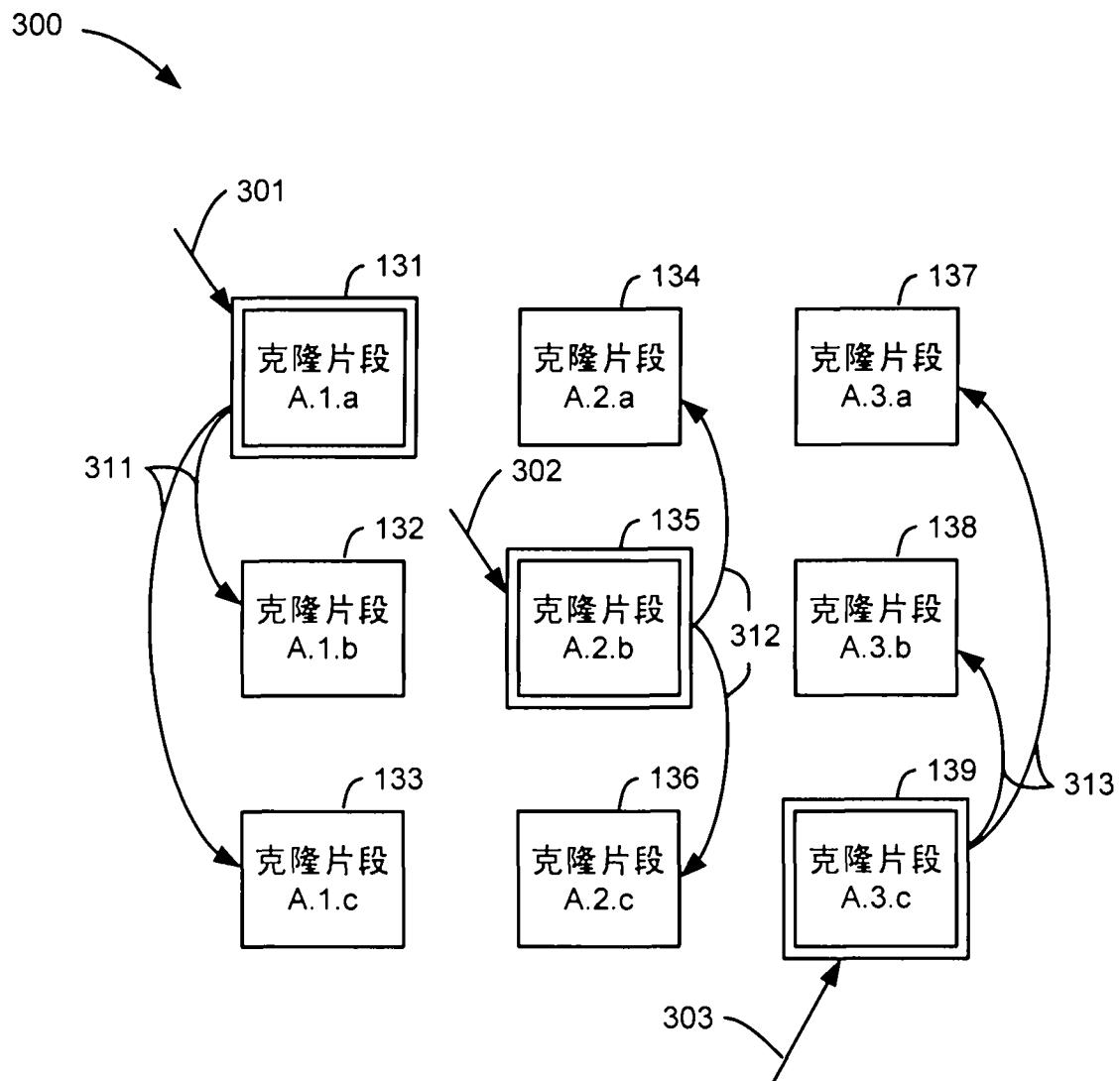


图 3

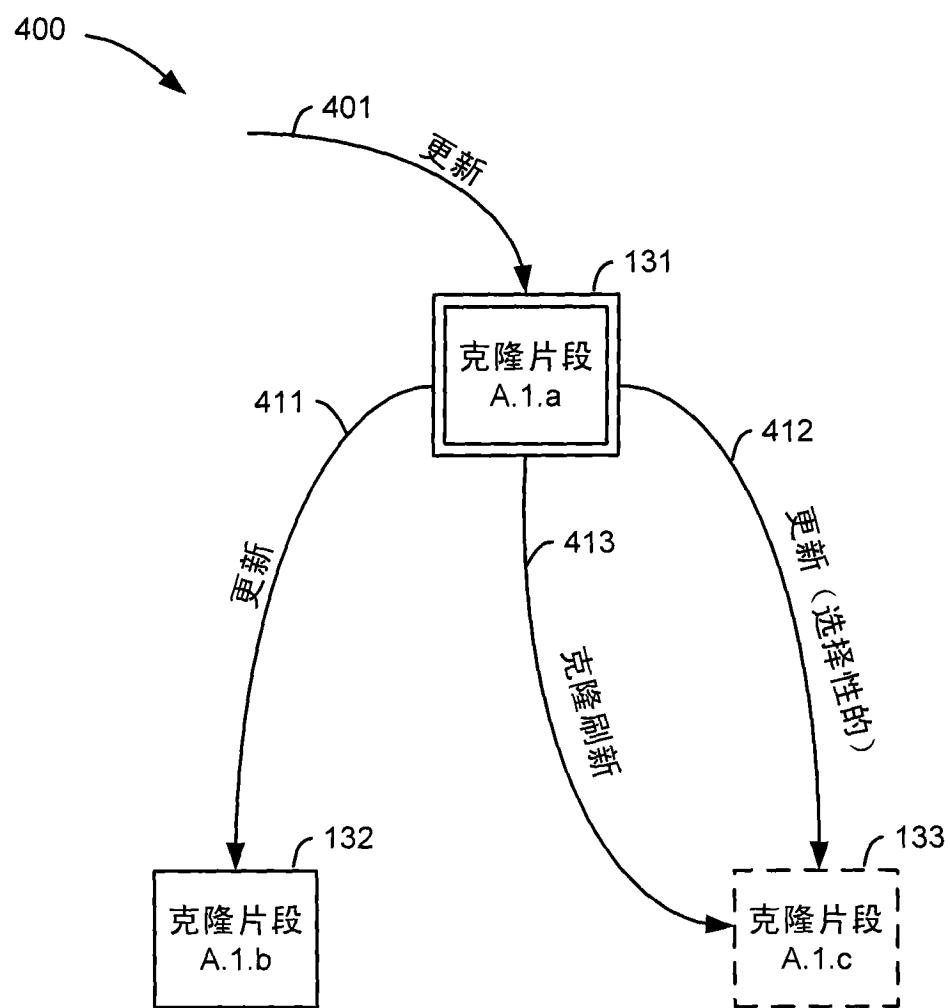


图 4

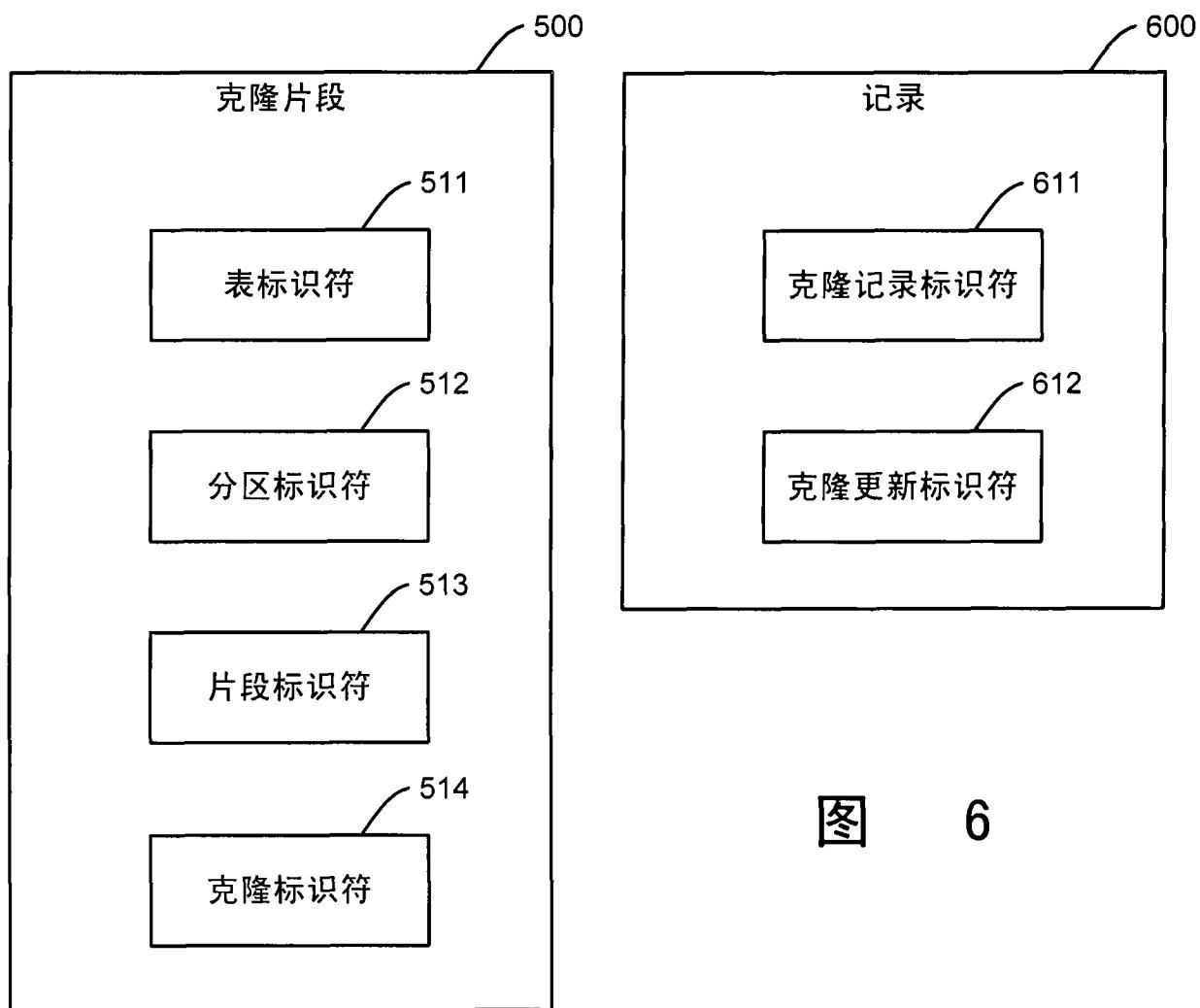


图 5

图 6

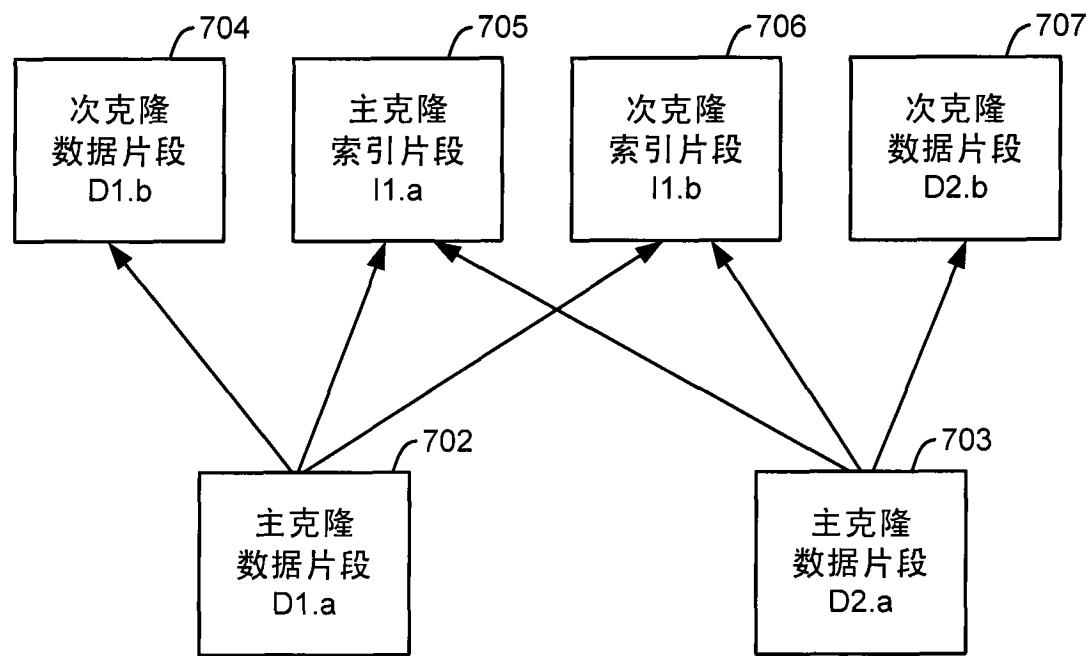


图 7

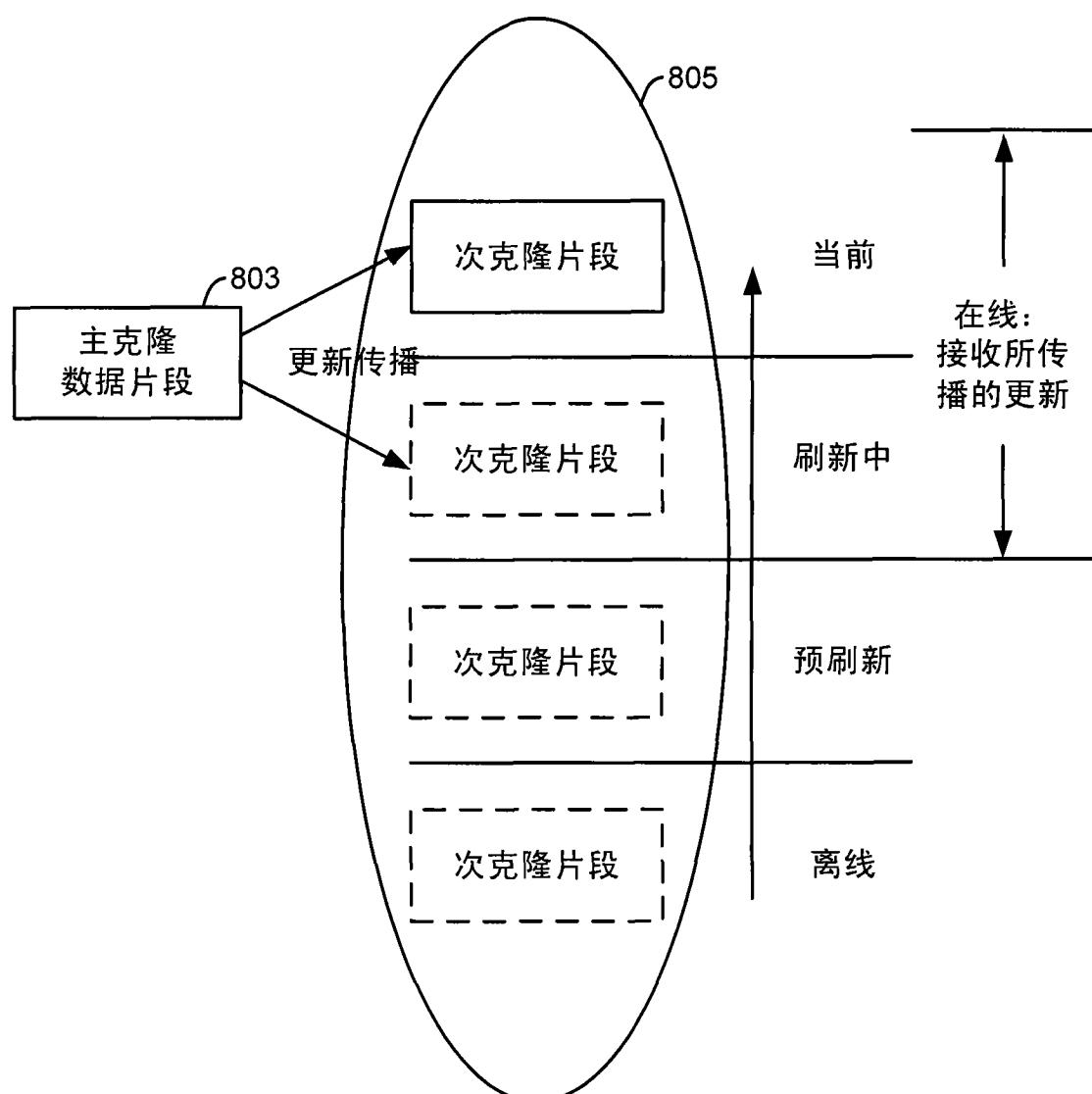


图 8

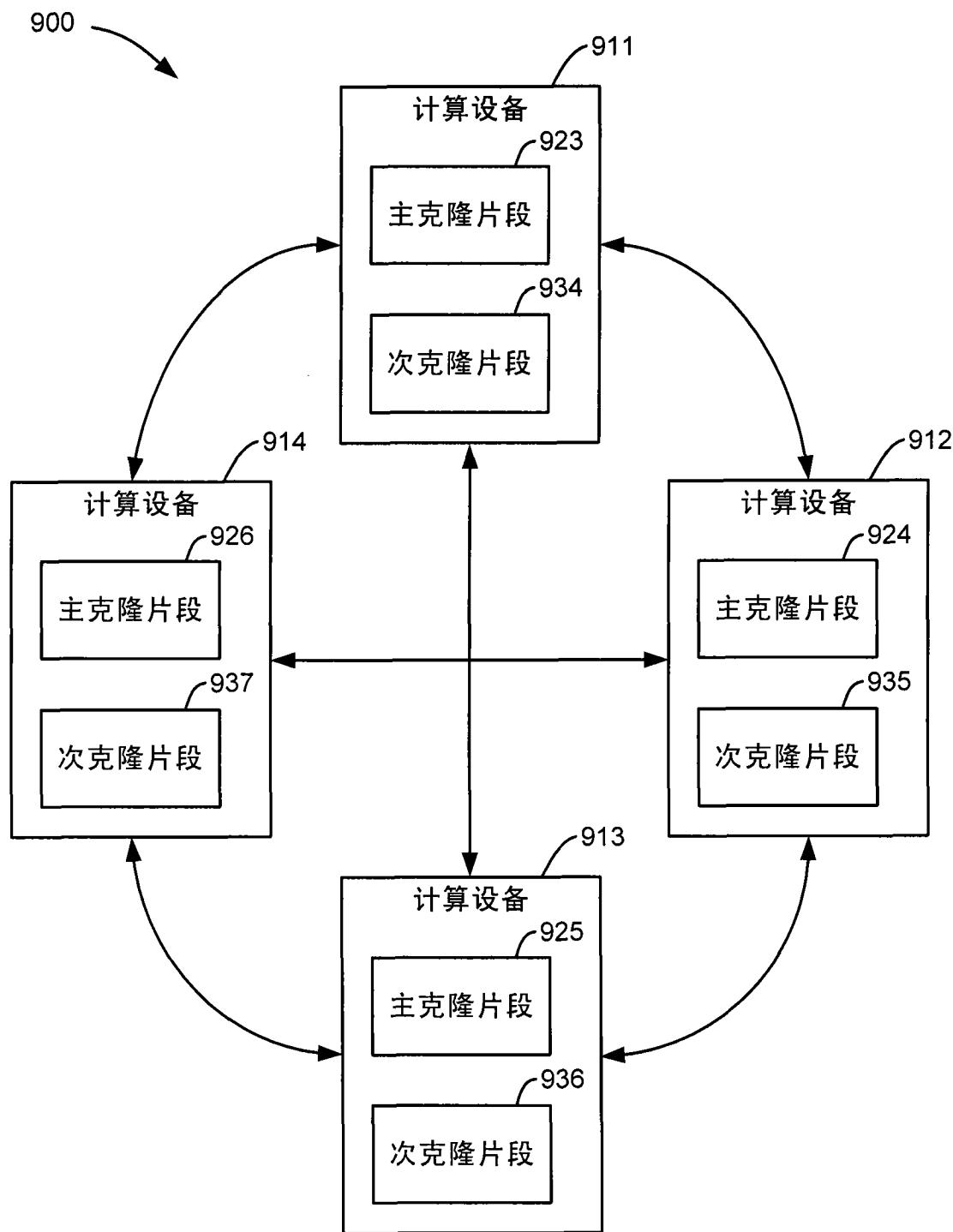


图 9

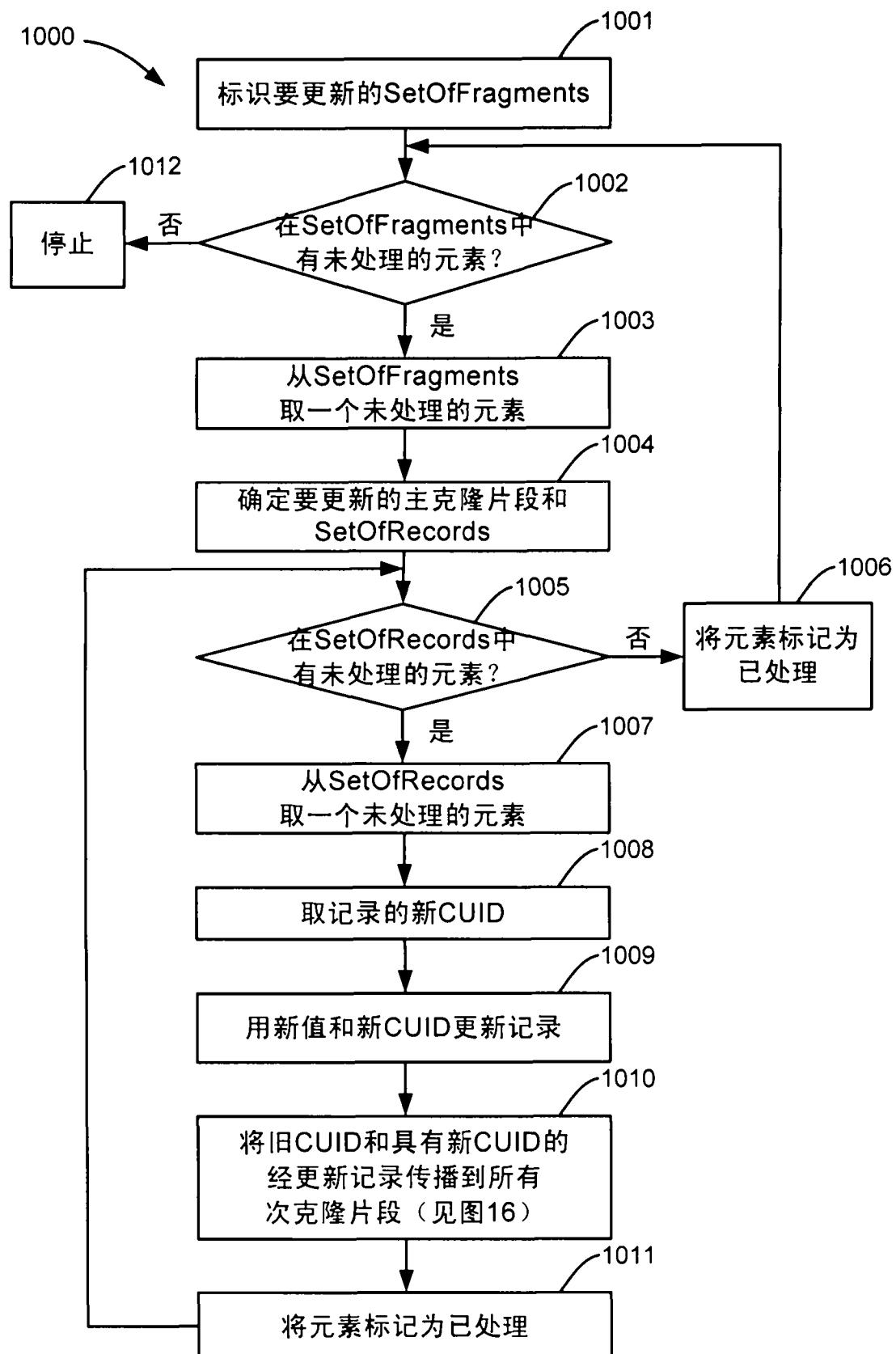


图 10

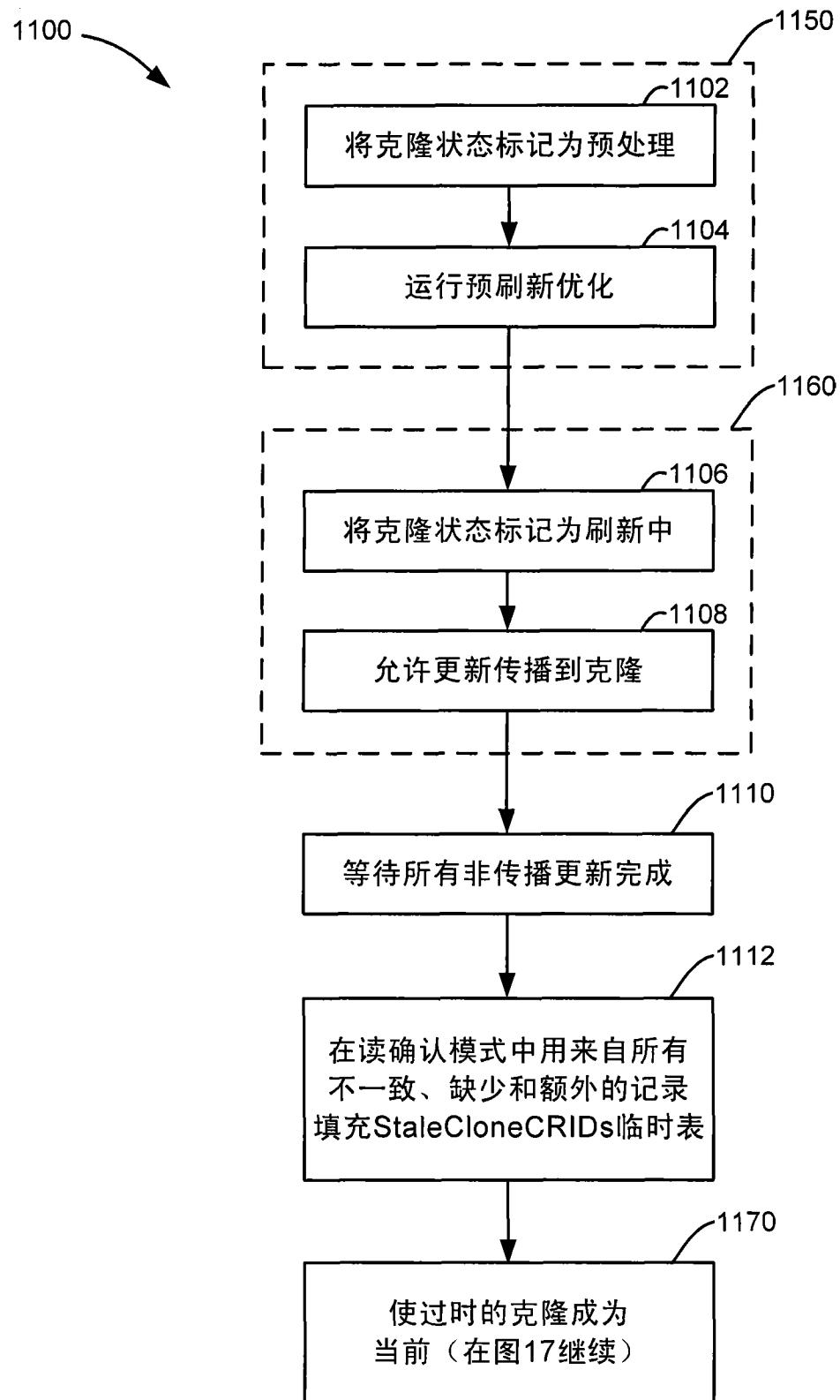


图 11

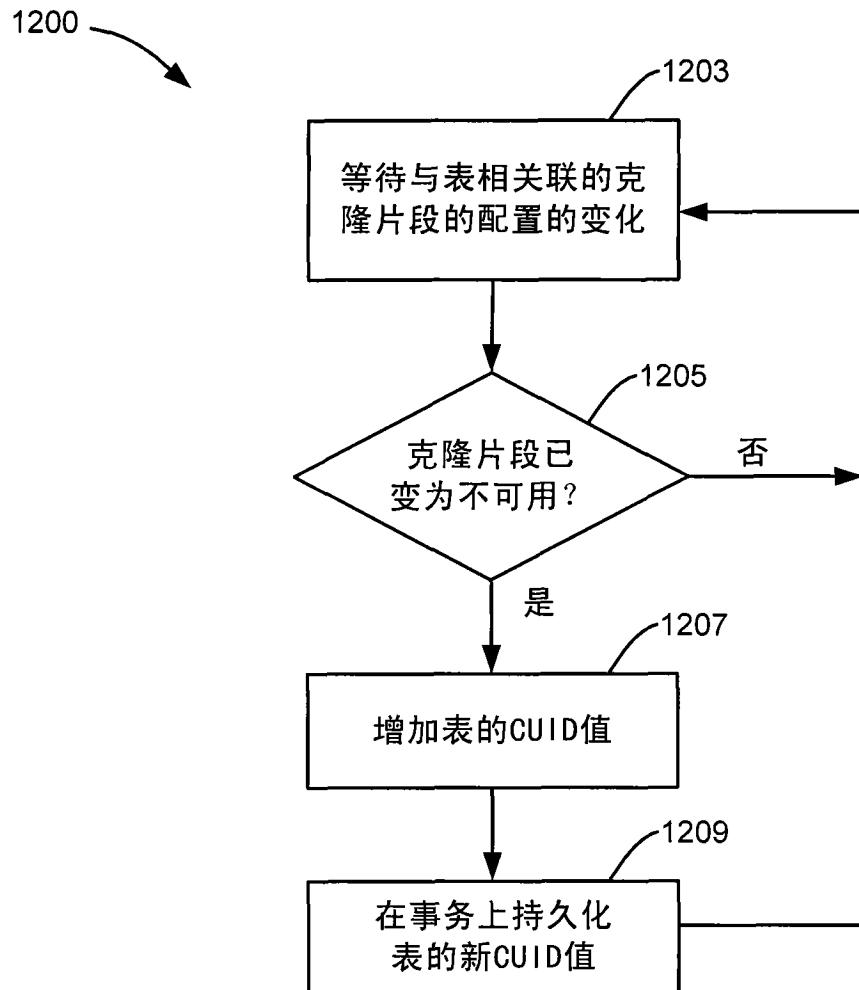


图 12

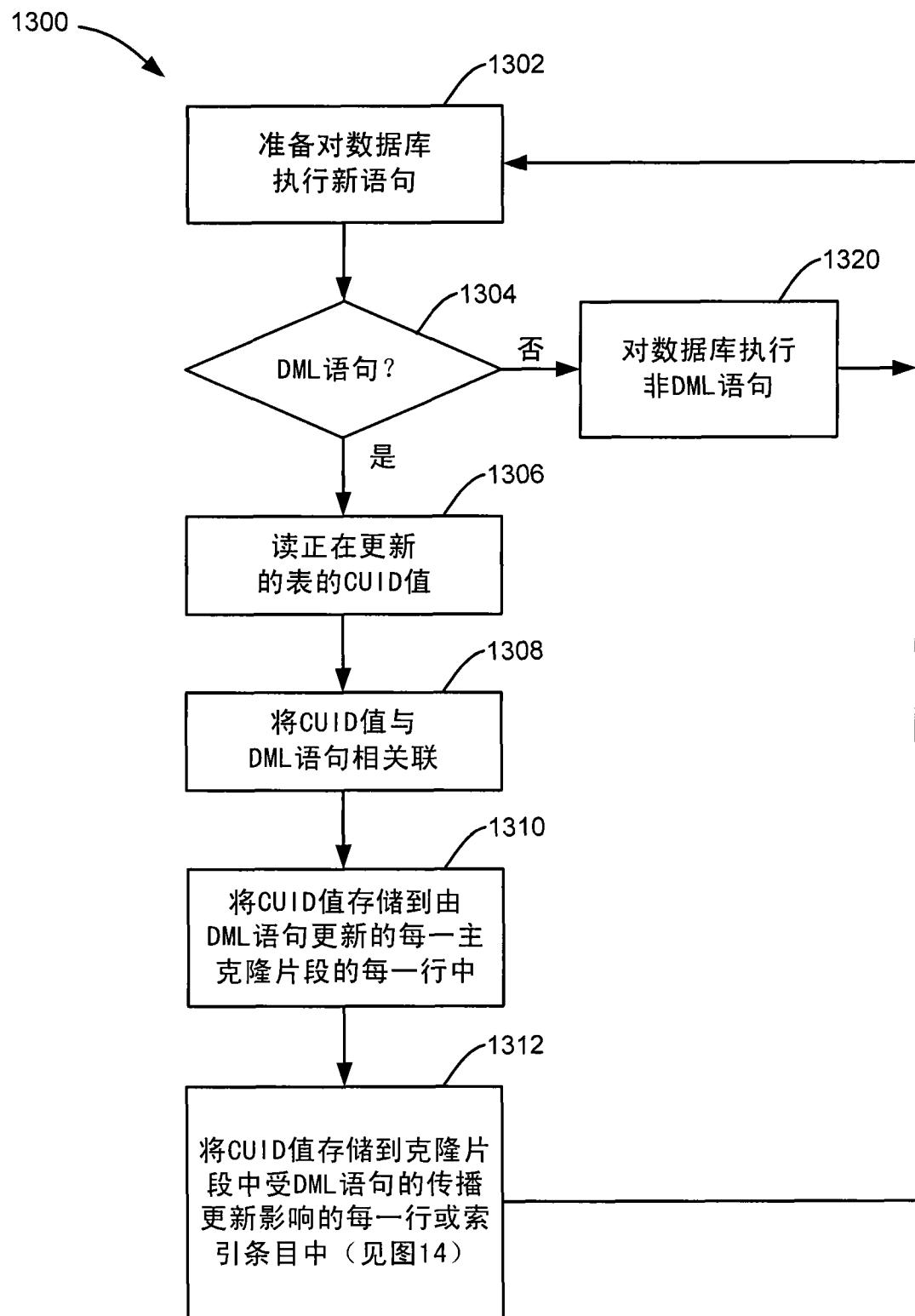


图 13

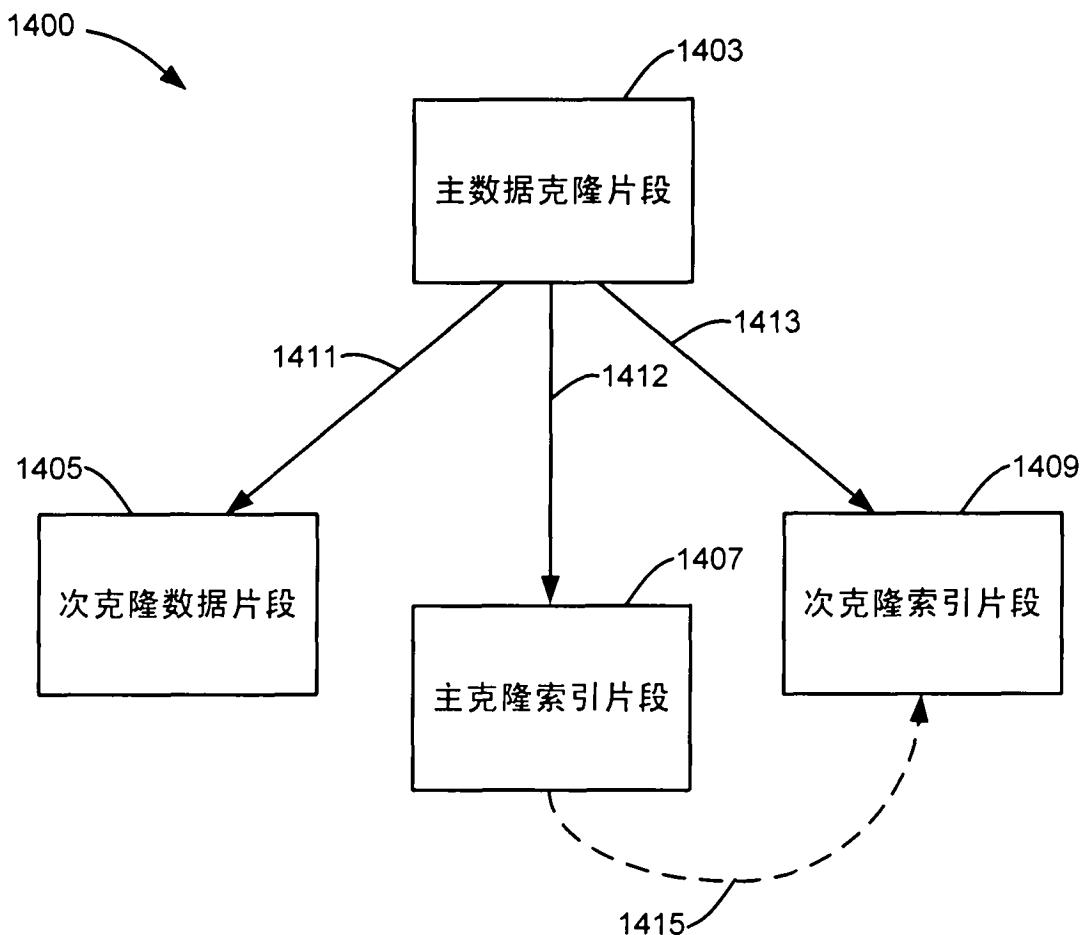


图 14

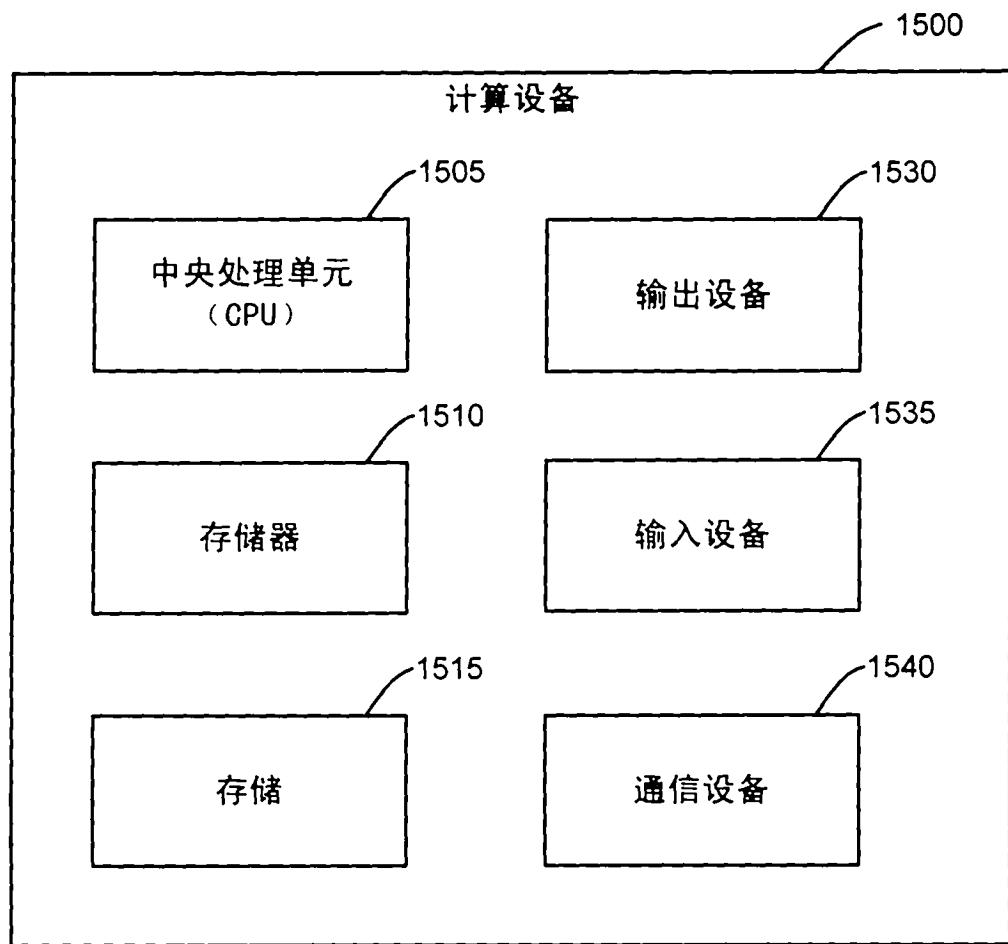


图 15

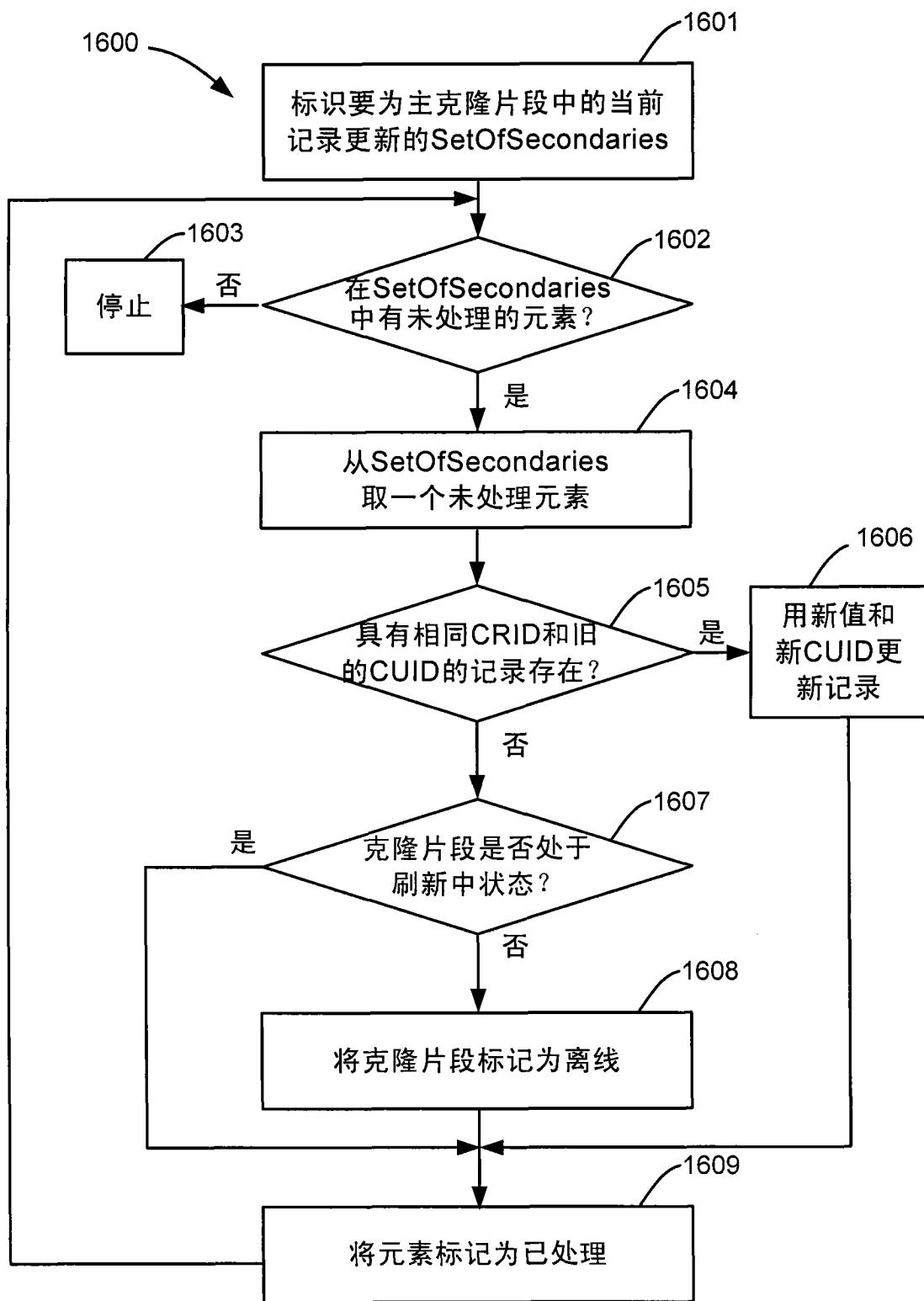


图 16

从图11继续

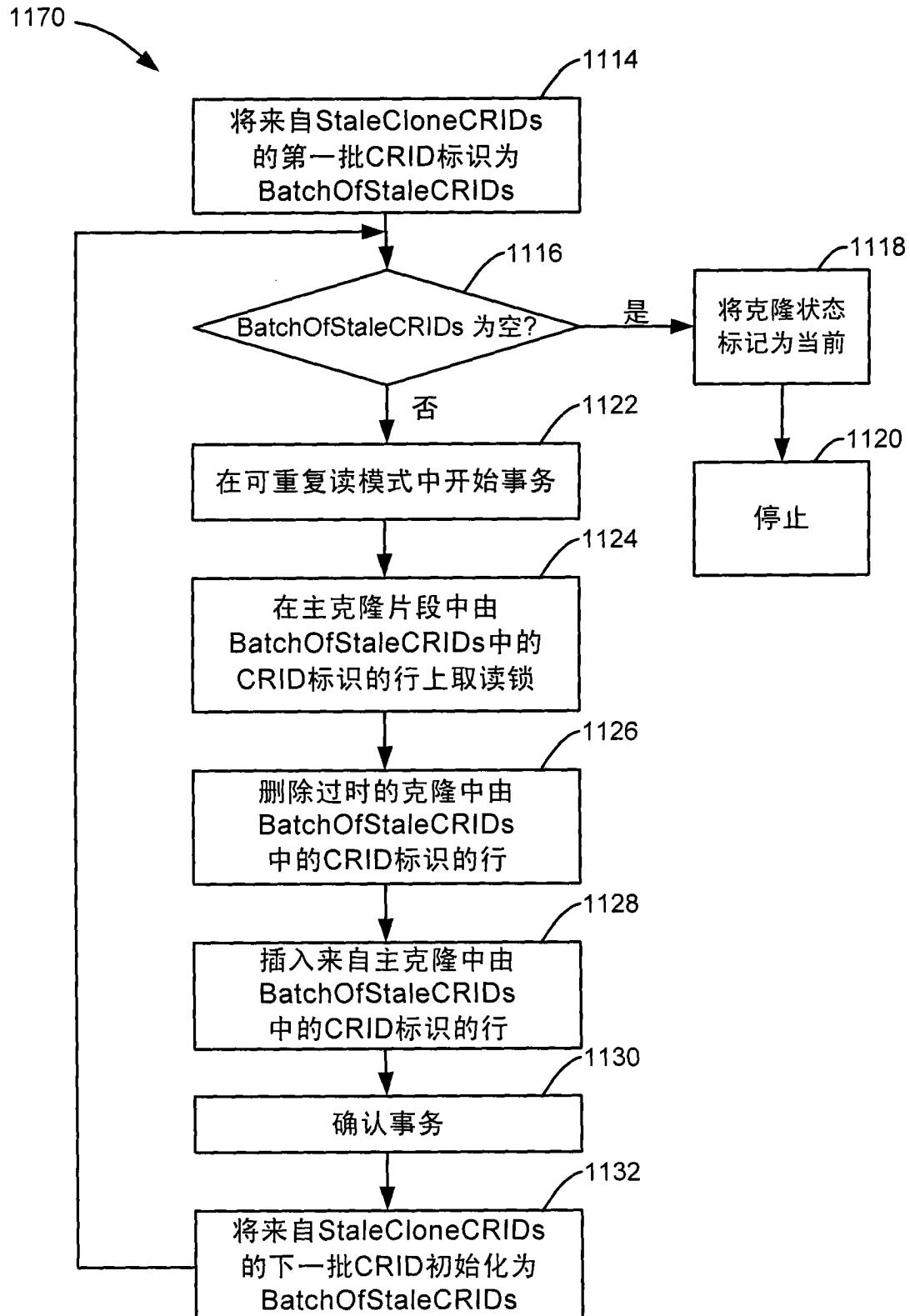


图 17