[19] 中华人民共和国国家知识产权局

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

[12] 发明专利申请公布说明书

[21] 申请号 200680033162.9

[43] 公开日 2008 年 9 月 10 日

[11] 公开号 CN 101263480A

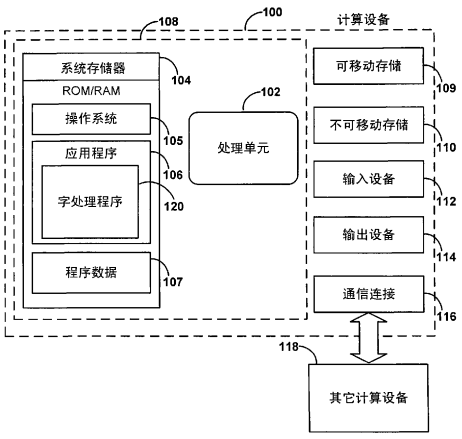
[22] 申请日 2006.9.7
[21] 申请号 200680033162.9
[30] 优先权
[32] 2005. 9. 9 [33] US [31] 60/715,986
[32] 2006. 1.13 [33] US [31] 11/332,468
[86] 国际申请 PCT/US2006/034974 2006.9.7
[87] 国际公布 WO2007/030683 英 2007.3.15
[85] 进入国家阶段日期 2008.3.10
[71] 申请人 微软公司
地址 美国华盛顿州
[72] 发明人 T·A·戴维斯 A·泰尔哈特
B·M·琼斯 M·斯维基
R·A·利特尔 S·库帕拉
D·巴拉克
[74] 专利代理机构 上海专利商标事务所有限公司
代理人 顾嘉运
权利要求书 3 页 说明书 20 页 附图 8 页

[54] 发明名称

应用程序之间 XML 数据的实时同步

[57] 摘要

将一或多个数据存储与文档内主呈现存储分开地保存，从而存储、关联并允许使用与计算机生成文档相关联的任意数据。 数据存储向数据存储中的各种数据片暴露应用程序编程接口(API)，以允许不同应用访问和操作一或多个数据片。 多个客户应用程序可访问和编辑同一文档数据片，并且解决对一给定数据片的任何有冲突的改变。 每个数据消费者可以接收或拒绝改变并能作出附加的副效应改变作为原始改变的结果。 这样，可以在数据消费者之间实时地同步数据。



1．一种具有计算机可执行指令用于同步与计算机生成的文档相关联的以及在各数据消费者之间共享的数据的计算机可读介质，包括：

将与文档相关联的结构化数据项目存储在数据存储中，其中所述数据存储独立于所述文档；

由第一数据消费者发起对所述数据存储内结构化数据的改变；

确定何时有对所述结构化数据项目的改变感兴趣的其他数据消费者；

将所述改变通知其他数据消费者，其中所述其他数据消费者可以执行下列至少一项：接受所述改变、拒绝所述改变；以及作为所述原始改变的结果发起副效应改变；以及

当接受所述改变时将所述改变提交给所述数据存储，而当拒绝所述改变时就把所述数据存储退回到发起所述改变之前的状态。

2．如权利要求 1 所述的计算机可读介质，还包括依照可扩展标记语言（XML）规范在所述数据存储内构造所述结构化数据项目。

3．如权利要求 2 所述的计算机可读介质，还包括向允许程序性地访问所述结构化数据项目的数据消费者暴露应用程序编程接口（API）；其中当所述文档打开时所述 API 提供访问所述数据存储内所述结构化数据项目的能力，并且其中所述 API 允许一个以上数据消费者同时访问相同的结构化数据项目；并且其中所述 API 向所述数据消费者提供注册以获取涉及所述结构化数据项目中的一个和多个的通知的能力。

4．如权利要求 3 所述的计算机可读介质，还包括使用由所述第一数据消费者改变的结构化数据项目的值来更新与所述其他数据消费者之一相关联的显示。

5．如权利要求 4 所述的计算机可读介质，其特征在于，所述副效应改变被放入队列以供稍后执行，其中所述稍后执行在每个所述数据消费者都有机会接受、拒绝和作出副效应改变之后发生。

6．如权利要求 4 所述的计算机可读介质，其特征在于，当接受时提交所述改变包括处理每个副效应并确定每个副效应是否被接受。

7．如权利要求 4 所述的计算机可读介质，还包括维护可用于将所述数据存储返回到发起所述改变之前的状态的撤销列表。

8．一种用于在数据消费者之间共享与计算机生成的文档相关联的数据的计算机实现的方法，包括：

发起对包含在数据存储内且与文档相关联的结构化数据项目的改变；其中所述结构化数据项目依照可扩展标记语言（XML）来结构化的，并且其中所述数据存储是独立于所述文档的；

将所述改变的通知提供给已注册的数据消费者；

将作为所述改变的结果的副效应改变放入队列以供稍后处理；

确定何时来自所述已注册数据消费者之一的响应是拒绝所述改变，以及当所述响应是拒绝时，将所述改变和来自所述数据存储中的所有数据消费者的任何副效应改变退回到最近已知良好状态；以及

确定何时每个所述已注册数据消费者都接受所述改变，以及当每个所述已注册数据消费者都接受所述改变时，使用由所述数据消费者之一通过 API 提供的 XML 模式文件验证所述改变，并且将所述改变提交给所述数据存储。

9．如权利要求 8 所述的计算机实现的方法，其特征在于，发起对包含在所述数据存储内的结构化数据项目的改变；包括暴露为所述数据消费者提供访问所述数据存储内的结构化数据项目的能力的应用程序编程接口（API），其中所述 API 允许一个以上数据消费者同时访问相同的结构化数据项目。

10．如权利要求 9 所述的计算机实现的方法，还包括响应所述通知接收副效应改变，并延迟对所述数据存储内的所述副效应改变的执行直至所述改变已被处理。

11．如权利要求 10 所述的计算机实现的方法，还包括将所述数据存储内的副效应改变放入队列中。

12．如权利要求 10 所述的计算机实现的方法，其特征在于，确定何时每个已注册的数据消费者接受所述改变包括确定何时所述改变被每个数据消费者接受以及何时每个所述副效应改变的被接受。

13．如权利要求 10 所述的计算机实现的方法，还包括维护用于将所述数据存储返回到所述最近已知良好状态的撤销列表。

14．一种用于在数据消费者之间同步与计算机生成的文档相关联的数据，包括：

内部数据消费者，被配置成创建和编辑所述文档并且被配置成与关联于所述文档的结构化数据项目相交互；

各外部数据消费者，被配置成与关联于所述文档的结构化数据项目相交互；以及

数据存储，被配置成独立于所述文档存储与所述文档相关联的结构化数据项目，其中所述数据存储包括：API 代理程序，被配置成与所述外部数据消费者和所述内部数据消费者交互并且被配置成接收所提出的改变且响应所述提出的改变将所提出的改变通知已注册的数据消费者，当所述提交的改变被接受时将所述改变提交给所述数据存储而当所述提出的改变不被接受时，确保所述数据存储处于有效状态，其中所述数据存储还包括改变存储，被配置成存储所述提出的改变和与所述提出的改变相关联的副效应改变；以及撤销存储，被配置成使所述数据存储退回所述有效状态。

15．如权利要求 14 所述的系统，还包括依照可扩展标记语言（XML）规范在所述数据存储内构造所述结构化数据项目。

16．如权利要求 15 所述的系统，其特征在于，所述数据存储还被配置成将所述副效应改变放入队列以供稍后执行，其中所述稍后执行被配置成在每个已注册的数据消费者都有机会接受、拒绝和作出副效应改变之后发生。

17．如权利要求 16 所述的系统，其特征在于，所述数据存储还被配置成处理每个副效应并确定每个副效应是否被接受。

18．如权利要求 16 所述的系统，其特征在于，所述数据存储被进一步配置成：

接收对应用于所述结构化数据项上的可扩展标记语言（XML）标记数据的改变；

读取对所述 XML 标记数据的改变所指向的、与所述结构化数据项目相关联的 XML 模式文件；

依照所述读取的 XML 模式文件，确定对所述 XML 标记数据做出的改变是否有效；以及

如果依照所述读取的 XML 模式文件对所述 XML 标记数据的改变是无效的，则不允许对所述 XML 标记数据的改变；而当依照所述读取的 XML 模式文件对所述 XML 标记数据的改变是有效的时，则提交所述改变。

19．如权利要求 16 所述的系统，其特征在于，所述数据存储包括各自包括撤销存储和改变存储的多个 XML 数据存储。

20．如权利要求 19 所述的系统，其特征在于，每个撤销存储被配置成与所述其他撤销存储通信。

应用程序之间 XML 数据的实时同步

背景

计算机用户已经逐渐习惯于用户友好的软件应用程序，它们帮助用户写、计算、组织、准备演示、发送与接收电子邮件、制作音乐等等。例如，字处理应用程序允许用户准备各种有用的文档。电子表格应用程序允许用户输入、操纵和组织数据。幻灯片演示应用程序允许用户创建包含文本、图片、数据或其它有用对象的幻灯片演示。

然而，由这类应用程序创建的文档（例如，字处理文档、电子表格、幻灯片演示文档）在存储/传输文档上下文所需要的任意元数据内容方面功能有限。例如，建立在字处理文档之上的解决方案可能要求存储描述文档各种状态的工作流数据，例如，先前的工作流核准状态（日期、时间、名称）、当前核准状态、完成前的未来工作流状态、文档作者的名称和办公地址、文档变化，等等。存储该信息的选项主要限于对文档变量或具有限制的现有定制对象链接和嵌入（OLE）文档性质的使用。例如，不能存储分层数据；字符长度有限，等等。有关这类方法的性质存储在单个存储中，例如 OLE 性质存储器，这意味着这些性质具有冲突的可能性。而且，这样存储的性质没有数据验证。这类应用程序及有关文档的用户难以将任意数据与文档存储在一起，而这却是许多用户的共同需求。

发明内容

提供本概述以便以简化形式介绍概念的精选，这些概念将在以下的详细描述中被进一步描述。本概述并不旨在标识要求保护的主题的关键特征或本质特征，也不旨在用于帮助确定要求保护的主题的范围。

将一或多个数据存储与文档内主呈现存储（primary presentation storage）分开地保存，从而存储、关联并允许使用与计算机生成文档相关联的任意数据。用于结构化与文档相关联的信息的数据，诸如文档元数据，被保存在其中保存不同数据片之间的关系的一个数据存储中。数据存储向数据存储中的各种数据片暴露应用程序编程接口（API），以允许不同数据消费者（data consumer）实时访问和操作一或

多个数据片。多个数据消费者可以同时访问和编辑同一文档数据片，并且解决对一给定数据片的任何有冲突的改变。每个数据消费者可以接收或拒绝这一改变以及作为原始改变的结果作出额外的副效应（side-effect）改变。这样，就可以在数据消费者之间实时地同步数据。

数据片可按照标记语言如可扩展标记语言（XML）来结构化。XML 模式可与每一数据片相关联，并且数据存储可基于与给定数据片相关联的 XML 模式来自动地验证数据的 XML 结构。这有助于防止允许非法的改变进入系统。

附图说明

图 1 示出了计算机的示例性计算体系结构；

图 2 是示出一个或多个客户应用程序以及一个或多个数据存储以及数据存储的内容之间的关系的框图；

图 3 示出了显示内部和外部数据消费者与 XML 数据存储之间的交互的系统图示；

图 4 示出了实时的同步的示例；

图 5 示出了两个客户和 XML 数据存储之间的交互；

图 6 显示了两个外部数据消费者和对 XML 数据存储的改变之间的交互；

图 7 显示了涉及多个副效应改变的进程；以及

图 8 示出了依照本发明的各方面显示最后执行调用程序的副效应的进程。

具体实施方式

现在参考其中相似的数字代表相似的元素的附图，描述本发明的各种方面。具体地，图 1 和相应的讨论旨在提供对在其中可以实现本发明实施例的合适计算环境的简要描述。

一般而言，程序模块包括例程、程序、组件、数据结构和其它类型的结构，它们执行特定的任务或者实现特定的抽象数据类型。也可使用其它计算机系统配置，包括手持式设备、多处理器系统、基于微处理器或可编程消费电子产品、小型机、大型机等等。也可使用分布式计算环境，其中任务由通过通信网络链接的远程处理设备执行。在分布式计算环境中，程序模块可被置于本地或远程的存储器设备中。

在通篇说明书和权利要求书中，下列术语采用与此处相关联的含义，除非该

术语的上下文另有指示。

术语“呈现”指文档的可见部分，诸如打印文档时会出现的文本和布局等。

术语“标签”指插入文档中描述 XML 文档内元素的字符。每个元素可以具有不多于两个的标签：开始标签和结束标签。也有可能具有空元素（没有内容），在此情形中允许一个标签。

术语“标记语言”或“ML”指用于文档内特殊代码的语言，用于规定文档的诸部分如何由应用程序来解释。在字处理文件中，标记语言规定文本被如何格式化或布局。

术语“元素”指 XML 文档的基本单元。元素可包含属性、其它元素、文本和用于 XML 文档的其它内容区域。

在标签之间的 XML 内容被认为是该元素的“子”（或后代）。因此，嵌入在元素的内容中的其它元素称为“子元素”或“子节点”或者元素。直接嵌入在元素内容中的文本被认为是该元素的“子文本节点”。元素内的子元素与文本一起构成元素的“内容”。

术语“属性”指设置为特定值且与元素相关联的附加性质。元素可具有任意数量的与其相关联的属性设置，包括一个也没有。属性用于将附加信息与将不再包含其它元素或者作为文本节点来对待的元素相关联。

术语“XPath”是运算符，它使用模式表达式(pattern expression)来标识 XML 文档中的节点。XPath 模式是用斜杠分开的子元素名称列表，它描述到通过 XML 文档的路径。模式“选择”匹配该路径的元素

术语“副效应改变”是指响应于另一改变作出的改变。

术语“文档”可以由描述在相关联的内容类型中定义的各种性质的任意 XML，以及可用于描述文档的实际表面内容的其他标记语言组成。

术语“XML 数据存储和/或数据存储”指诸如字处理文档，电子表格文档，幻灯片演示文档等的文档内的容器，它在文件打开时提供对存储在该文档中的数据（例如 XML 格式）的存储和修改的访问。下面参考图 2 提供 XML 数据存储的进一步定义。

参加图 1，用于实现本发明的一个示例性系统包括计算设备，诸如计算设备 100。在一个非常基本的配置中，计算设备 100 通常包括至少一个处理单元 102 和系统存储器 104。取决于计算设备的确切配置和类型，系统存储器 104 可以是易失性的（诸如 RAM）、非易失性的（诸如 ROM、闪存等）或是两者的某种组合。

系统存储器 104 通常包括操作系统 105、一个或多个应用程序 104，并且可以包括程序数据 107。在一个实施例中，应用程序 106 可以包括字处理程序 120。这一基本配置在图 1 中由虚线 108 中的那些组件示出。

计算设备 100 也可具有其它特征或功能性。例如，计算设备 100 也可含有附加的数据存储设备（可移动和/或不可移动），诸如磁盘、光盘或磁带。这样的额外存储在图 1 中由可移动存储 109 和不可移动存储 110 示出。计算机存储介质可包括易失性和非易失性、可移动和不可移动介质，它们以用于存储诸如计算机可读指令、数据结构、程序模块或其它数据这样的信息的任意方法或技术来实现。系统存储器 104、可移动存储 109 和不可移动存储 110 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、磁盘存储或其它磁性存储设备、或能用于存储所需信息且可以由计算设备 100 访问的任何其它介质。任何这样的计算机存储介质都可以是设备 100 的一部分。计算设备 100 也可以具有诸如键盘、鼠标、笔、语音输入设备、触摸输入设备等输入设备 112。也可以包括诸如显示器、扬声器、打印机等的输出设备 114。这些设备在本领域是公知的，因此不必在此详细讨论。

计算设备 100 也可以包含允许该系统与其它计算设备 118 诸如经网络通信的通信连接 116。通信连接 116 是通信介质的一个示例。通信介质通常可具体化为诸如载波或其它传输机制等已调制数据信号中的计算机可读指令、数据结构、程序模块或其它数据，并且包括任何信息传递介质。术语已调制的数据信号是指以在该信号中编码信息的方式来设置或改变其一个或多个特性的信号。作为示例，而非限制，通信介质包括有线介质，诸如有线网络或直接线连接，以及无线介质，诸如声学、RF、红外线和其它无线介质。如此处所用的术语计算机可读介质既包括存储介质又包括通信介质。

可在计算设备 100 的系统存储器 104 中存储多个程序模块和数据文件，包括适于控制联网个人计算机操作的操作系统 105，诸如来自华盛顿雷德蒙德的微软公司的 Windows 操作系统。系统存储器 104 也可存储一或多个程序模块，诸如字处理应用 120 以及下述的其它应用。字处理应用 120 用于提供创建、编辑和处理电子文档的功能性。

根据本发明的一个实施例，字处理应用 120 包括来自微软公司的 WORD 程序。然而应该了解，可利用来自其它制造者的字处理应用程序。字处理应用的例示仅为

了示例，而非不限制可产生并操作文档的其它类型的应用。例如，可以等效地应用能处理各种形式内容（例如，文本、图像、图片等）的其它应用程序 106，诸如电子表格应用程序，数据库应用程序，幻灯片演示应用程序，画图或计算机辅助应用程序等等。产生并操作各种不同类型文档的示例应用程序 106 包括来自微软公司的 OFFICE。

实施例可实现为计算机过程、计算系统、或者制造品，如计算机程序产品或计算机可读介质。计算机程序产品可以是计算机系统可读且对指令的计算机程序进行编码以执行计算机过程的计算机存储介质。计算机程序产品也可以是计算系统可读的且对指令计算机程序进行编码以执行计算机过程的载波传播信号。

图 2 是例示一或多个客户应用程序与一或多个数据存储之间的关系以及数据存储中的内容的框图。一般而言，在将一或多个数据存储与文档内主呈现存储分开保存，以存储、关联并允许使用与计算机生成文档相关联的跨多个数据消费者的任意数据。用于结构化与文档相关联的信息的数据，诸如文档元数据，被保存在其中保存不同数据片之间的关系的一个数据存储中。数据存储向数据存储中的各种数据片暴露应用程序编程接口（API），以允许不同应用访问和操作一或多个数据片。如在此使用的，术语“数据消费者”、“应用”和“过程”能互换地使用，除非上下文另有明确指示。

数据片可按照标记语言如可扩展标记语言（XML）来结构化。XML 模式可与每一数据片相关联，并且数据存储可基于与给定数据片相关联的 XML 模式来验证应用于数据的 XML 结构，以确保每个请求的有效性。数据存储可包含根据可扩展标记语言（XML）结构化的任何数量的任何数据项，例如元数据。因此，文档解决方案提供者可将任意元数据作为 XML 与给定文档存储在一起，并且让该信息在发生事件时由有权访问数据的给定解决方案来处理，诸如当数据从数据存储中移除或被加载到数据存储时和/或当用户打开/编辑/保存文档时。

当文档正在编辑的同时，还提供对 XML 形式数据的程序性访问。根据一个实施例，提供一种解决方案开发者熟悉的标准机制，通过该机制在打开文档时能以在程序上访问和修改数据。这一程序性的访问被设计用于模拟标准 XML API。通过应用编程接口为一或多个编辑客户应用（例如，文档编辑或创建应用和/或第三方应用插件解决方案等等）提供对数据的程序性访问。因此，多个客户应用可访问并编辑相同的文档数据片，并且解决任何对给定数据片的有冲突的改变。数据消费者可以响应任何给定改变作出副效应改变。例如，响应于将公司名称设置为

“Microsoft（微软）”，数据消费者可以将证券代码改成“MSFT”。此外，对数据的改变和任何相关联的副效应可由数据存储来“捆绑”，使得撤消一或多个改变将后退所有相关改变。这帮助减少对数据消费者本身开发的负担，以确保当用户例如通过压下撤消命令从文档表面启动原始改变撤消时所有的改变都得以后退。

标准 XML 模式（XSD）也可用于定义与文档元数据相关联的任何定制 XML 数据片的内容，以确保应用于该文档数据的 XML 数据是有效的。这些模式可附连到存储在文档中的 XML 数据的任何实例中，并且数据存储可被配置为不允许会导致该数据的 XML 结构变成无效（即，XML 标签与其内容相反）的任何对 XML 数据的改变。这帮助确保解决方案开发者能将特定的 XML 元数据片附连到文档，并且确保该 XML 数据将根据相关联的模式继续在结构上保持“正确”，无论哪个数据消费者（例如，插件）修改该数据。模式可以被存储在计算机可读介质中，诸如在文件中或在硬盘上。

现参考图 2，文档数据 220 包括 XML 结构数据以及表示文档的表面或呈现层视图的相关联文档数据。例如，文档数据 220 可包括 XML 结构（例如，标题标签，正文标签，结论标签）以及字处理文档、电子表格文档、幻灯片演示文档等的相关联的表面视图数据（例如，字词，语句，段落）。

数据存储 208 是文档数据储存库，用于存储一或多个与关联于给定文档的一或多个类型的数据相关联的结构化数据片。尽管仅示出一个数据存储器，但可使用不止一个数据存储。元数据 1 225（结构化的数据项）可包括 XML 结构和用于关联于文档的第一元数据片的相关联数据。例如，元数据 1 225 可包括 XML 结构数据（例如日期标签，名称标签等），它们列出文档作者，文档创建日期，文档最后改变/保存日期等等。元数据 2 230（结构化数据项）可包括 XML 结构数据（标签）以及表示关联于文档的第二元数据片的相关联元数据。元数据 1 和元数据 2 是为了示例，而非限制可保存在数据存储 208 中关联于给定文档的大量不同类型数据。例如，如在此所述的，任意数据可按有权访问文档数据的解决方案开发者或用户的需要，由一或多个软件应用来结构化并添加到文档。

模式文件 240、245 可选地可被附连到存储在数据存储 208 中的每一数据片，用于指示与应用于每一数据片 225、230 的可扩展标记语言（XML）相关联的句法和验证规则。XML 模式文件提供一种在 XML 环境中描述和验证数据的方法。模式文件陈述使用什么 XML 标记数据（包括元素和属性）来描述 XML 文档中的内容，并且模式文件定义 XML 标记句法，包括每一元素所允许的位置，在元素内允

许什么类型的内容以及在哪些元素可出现在其它元素中。模式文件的使用确保文档（或者单个的数据片）以一致且能预知的方式来结构化。模式文件 240、245 可由用户创建并且一般由相关联的标记语言诸如 XML 来支持。

该文档的模式化通过在数据存储层处拒绝违反给定模式文件的任何改变来允许数据存储提供“保证”文档的结构有效性的能力。按照一实施例，数据存储 208 使用模式验证模块 260 针对相关联的模式文件验证添加的 XML 结构或者对给定数据片的改变。例如，如果文档创建者或编辑者对给定数据片例如元数据 1 作出 XML 结构上的改变，其中编辑者添加或移除给定的 XML 标签，则数据存储 208 将使用模式验证模块针对相关联的模式文件来检查 XML 结构上的改变以确保改变的合法性。如果改变不是有效的，则向编辑者产生一个错误。如所理解的，对应用于给定数据片的 XML 结构的这种控制考虑了结构上的一致性和可预知性，这对于允许客户和第三方应用程序与相关联数据交互尤为重要。任何数据消费者可以提供可用于验证数据的模式。

数据存储 208 提供可由客户应用 205（例如，字处理应用，电子表格应用，幻灯片演示应用等）以及第三方应用 210、215 通过相应应用 205、210、210 的对象模型（OM）访问的一或多个应用编程接口（API）270。这些 API 允许客户应用和第三方应用将任何现有 XML 文件加载到给定文档的数据存储 208，因而确保该数据现在是文档的一部分并且将在其生存期中在该文档内传播（例如，通过打开/编辑/保存/重命名等）或者直到该数据从数据存储中移除为止。按照一个实施例，数据存储中的数据以其 XML 格式可用，甚至在给定数据片 225、230 的源应用被关闭或者不可用的情况下。即，可以经由一组 API 来访问给定数据片 225、230。如下所述，API 还允许客户和第三方应用对应用于数据项 225、230 的 XML 标记数据进行改变。

一旦 XML 数据 225、230 被加载到数据存储中以便与文档 220 相关联，则可使用数据存储接口将其作为标准 XML 来操纵，其中数据存储接口被设计为提供与现有 XML 编辑接口相似的方法，以便利用开发者的 XML 编程标准的现有知识。这允许用户对添加到用于文档的数据存储的 XML 数据进行标准 XML 操作，诸如添加元素和属性，移除元素和属性，改变现有元素/属性的值，以及读取相关联 XML 树的任何现有部分的值。使用这些 XML 标准操作，解决方案可以使用文档存储结构化的复杂元数据。例如，可编写第三方用 215，它通过读取添加到每一文档的数据存储器 208 的元数据 1 225，而从多个文档定位和提取文档作者名称和文档创建

日期。示例的第三方可以是一个为作出由给定组织创建的所有文档的文档作者名称和文档创建日期列表而编程的应用。按照本发明的实施例，第三方应用可使用应用于元数据 1 的 XML 结构来有效地定位和提取所需数据。例如，可编写第三方应用来语法分析元数据 1 文件的 XML 结构以定位 XML 标签，诸如<docauthor>和<doccreationdate>，用于获得和使用关联于这些标签的数据。如应了解的，上述内容仅是一或多个应用可通过数据存储 208 与关联于文档的结构化数据交互的许多方法中的一个示例。

此外，数据存储器 208 提供任意数量的 API 接口 270 给任何单独的 XML 数据片 220、225、230（也称为存储项），以使多个应用 205、210、215 能够与同一数据片一起工作。例如，若干解决方案，诸如客户应用（例如字处理应用）和第三方应用解决方案（例如上述应用），可与同一组文档性质（例如包含在元数据 2 230 文件中的性质）一起工作。使用数据存储 208，这些应用的每一个通过其自己的数据存储 API 接口 270 接收对所需 XML 数据 230 的独立访问，这些 API 接口 270 允许每一个应用通过其自己 OM 与数据通信而不必处理让多个数据消费者访问同一数据片的复杂性。

为了允许这多个数据消费应用 205、210、215 访问同一数据，数据存储 208 在 XML 数据的任意部分被另一应用改变时通知这些应用的每一个，使得给定应用可响应于该改变(既包括在内部对于其自己过程的又包括在外部由对同一数据的其它改变所引起的）。当一个应用请求对给定数据项的改变时，该请求被自动发送到所有其它应用，以允许其它应用决定如何或是否响应所请求的改变。按照一个实施例，这是通过允许每一个应用注册对其具有接口的 XML 数据的任意部分的“监听”来实现的，使得给定的应用解决方案/程序仅接收属于其自己逻辑的那些消息。例如，一种类型的应用 210 可能希望注册监听对给定 XML 数据的所有改变以便向第三方解决方案提供详细的商务逻辑能力，而另一类型的应用 215 可能希望仅监听对同一数据内的一两个特定的 XML 元素的改变，因为其逻辑不关心对 XML 数据的任何其它部分的改变。

按照该实施例，多个应用 205、210、215 可访问和编辑同一文档数据片，并且解决对给定数据片的任何冲突改变。例如，当一个应用所作的一个改变引起由另一应用所作的“副效应”改变时，可对任何给定的改变作出副效应。例如，第一应用 210 的任务是从关联于给定文档的一或多个数据项 225、230 提取公司名称以将这些名称翻译成相应的股票代码，如果可用，则用来编译有关给定文档的公司股票

代码的列表。如果第二应用 215 引起给定数据片中的给定公司名称添加或改变，例如将公司名称从“公司 ABC”改变为“公司 XYZ”，则第一应用可监听该改变，从而自动更新其股票代码列表以包括代替“公司 ABC”的“公司 XYZ”的股票代码。此外，这类改变及任何相关联的副效应可由数据存储 208 进行捆绑，使得撤消一或多个改变时后退所有相关改变。

图 3 示出了显示内部和外部数据消费者与 XML 数据存储之间交互的系统图。如所示的，系统 300 包括文档 315、数据存储 302，呈现层 304、每一个都包括错误存储和撤消存储的 XML 存储 1-N（306）、全局改变存储 308、可选全局撤消存储 310、耦合到内部数据消费者 1-N 314 的内部代理程序 312 以及耦合到外部数据消费者 1-N 318 的外部代理程序 316。

使用数据存储 302 和 XML 数据存储器 306，文档具有包含任何数量的任意数据项的能力（只要每个数据项遵循标准的 XML 句法）。任意元数据可作为 XML 存储在文档内并且该信息在文档由用户打开/编辑/保存时可自动往返。

如上所述，当文档正在被编辑时通过可使用的 API 提供对该数据的程序性访问，提供一种解决方案开发者所熟悉的标准机制，通过该机制在文档打开时可程序性地访问和修改该信息。按照一个实施例，该程序性访问被设计为模拟标准的 XML 接口。使用 API，当诸如字处理应用的应用正在运行时，可以添加/移除数据；可将数据填充到存储项（数据存储的一部分）内；使用标准的 XML 结构操纵数据；可将模式与数据存储中的任意 XML 数据相关联；一旦与数据存储项相关联，模式可被添加/移除/修改；以及可将 XML 改变作为事件发送给任何监听的客户。如所示的，API 包括外部代理程序 316 和内部代理程序 312，前者为外部数据消费者 318 而后者为任何内部数据消费者 314 提供与数据存储 302 交互的接口。

对数据存储 302 的操纵可实时发生。如上所述，数据存储 302 和 306 可包含一或多种类型的数据。例如，公司可能具有一个数据存储，这是他们正在用于存储他们想要存储在单一数据存储器内的所有不同类型的数据的数据存储，同时另一公司可能想要在不同数据存储内器存储不同类型的数据。

数据消费者（内部 314 和/或外部 318）可以注册涉及有关数据存储中的数据的动作的事件。例如，数据消费者可注册以接收在对一或多个数据存储作出任何类型的改变时的事件。另一数据消费者可针对发生于数据存储内某一元素或某组元素的改变进行注册。常规事件包括添加项目、改变项目和从数据存储之一移除项目。当事件发生时，已经注册过的各数据消费者可对改变作出反应，同时保持数据存储

器的状态一致。很多时候，数据消费者在作出改变时不执行任何动作。在其它时候，数据消费者会响应于事件而执行某（些）动作。例如，数据消费者可响应于改变而作出某些其它改变，诸如响应于题目改变而更新文档内的首部。数据消费者还可执行不影响文档的某些其它操作。例如，如果插入股票报价机代码，则数据消费者可检索关联于该股票代码的数据，即使所有检索到的数据并不显示在文档的呈现层内。数据消费者也可以使用其验证逻辑拒绝改变。例如，如果数据消费者 1 收到不接受的改变，则该数据消费者可向代理程序返回一个标志，表示不接受该改变。只要改变不被接受，则使得该改变连同任何副效应退回（roll back），使得该改变从未发生过。每一 XML 存储 306 可使用其撤消存储来撤消其已作出的改变。作为替代，可使用全局撤消存储 310 来撤消在各数据存储上作出的改变。假设存在对发生于文档性质的事件感兴趣的三个数据消费者，因此这些数据消费者的每一个已经注册以接收有关这些性质改变的事件。当作出改变时，数据存储确定已经注册的每一数据消费者并且按预定的顺序将改变通知它们每一个。每一数据消费者进而可响应于该改变而执行某个动作。如果该改变以及由已注册数据消费者作出的、作为该改变的结果的任何改变不被任一数据消费者接受，则涉及该初始改变的所有改变被撤消。

外部代理程序应用编程接口层 316 提供由外部数据消费者 318 对数据存储 302 的访问并且允许第三方客户与数据存储器 302 交互，就象与关联于应用的内部数据消费者与数据存储器交互一样。为标识目的，为数据存储 302 内的每一 XML 数据存储 306 提供唯一的 ID。这帮助定位 XML 数据存储 306。

在任何时候，数据消费者可添加用于验证数据存储内数据的模式。当添加了模式并且数据消费者试图改变任何数据时，数据存储确定使用提供的模式改变是否有意义。一旦模式被附连，代理程序就变为验证中的对象。

专门定义的 XML 模式可用于围绕在文档中的内容提供语义标记，所述文档诸如文字处理文档、电子表格文档等。这种强大的功能使开发者能创建解决方案，所述解决方案可以利用该专门的 XML 嵌入直接作用于他们的数据的结构和内容，而非要求他们的解决方案处理下层应用程序的呈现格式的各种复杂状况。

例如，如果用户要在 XML 不可用的应用程序中创建用于股票调查记录的封面页，那么提取有用数据（例如公司名称、证券报价机代码、证券评级）会需要使用与文档的呈现格式密切联系的应用程序的对象模型。这当然地意味着所得的解决方案逻辑也依赖于与文档的呈现格式，并且如果呈现改变就会遭受失败。例如，如果

代码期望报价机符号在第一个表格的第 3 列第 2 行中，那么添加新的行/列会破坏该逻辑。然而，对启用了 XML 的应用程序而言，该代码现在可以被链接到消费者自身的数据的结构，从而消除逻辑依赖呈现的必要性。该相同的逻辑会搜索<stockSymbol/>XML 节点的内容，并且无论它存在于文档中的任何位置都找到它以便编辑它，即使其上下文情景呈现已被彻底改变。

XML 模式通常包封若干类型的数据，包括：元数据（例如，用于存储/处理的作者数据）；正文数据（例如，被报道的公司）；以及列表数据（例如，证券价格历史）。然而，这些数据类型不是互斥的。实际上，它们通常是同一 XML 文档中大量重叠的区域。理想情况下，虽然该数据都是使用单一的 XML 模式来表达的，但是这些不同的数据“类型”的每个可以在特定地适用于该数据的最佳表达的环境中被编辑或进行其他操作。例如，一种形式似乎允许用户方便地编辑文档的元数据，而该文档正文则可经字处理应用程序编辑。这是实时发生的，使得用户能够同时填写文档和表格的部分。

数据存储也可以每次接收一个以上元素。假设作为一个特定的流的数据（XML）在某些情况下会有助于满足该模式。例如，假设所附模式表示如果证券数据存在，它必须具有至少两个公司。如果证券数据是一个接一个添加的，那么它会是无效的。

依照一个实施例，使用单次通过来验证数据。代替会导致对数据存储作出改变的二次通过，在数据被提交给数据存储之前执行验证。这有助于防止数据消费者将错误引入数据存储。。

如此处所述，现在可以独立于中央 XML 数据存储 302 中的任何特定应用程序文档来存储与文档相关联的 XML 数据。可以为一个 XML 数据的呈现/编辑创建多个环境。通过它们到 XML 数据存储中相同数据的连接，该数据的表达被自动地同步。这样，多个应用程序可以同时显示相同的下层 XML 数据。这意味着向用户提供了编辑应用程序中相同数据的能力，即“最佳工作工具（the best tool for job）”。例如，用于编辑元数据信息的形式、用于编辑数据自由形式部分的字处理文档表面等。这也意味着用户可以按需在多个应用程序中编辑该数据。如果在多个应用程序中出现相同的信息，那么用户可以基于它们当前编辑的上下文情况，按需在这些应用程序的任何应用程序中编辑该数据。

虽然现在每个应用程序同时访问与文档相关联的整个 XML 数据，但是每个应用程序可以各自作出是否显示和编辑该数据的任何部分的选择。这意味着每个应用

程序仅需要显示在该上下文情景中相关的数据部分。例如，可以在文档中显示所有的 XML 数据，而另一应用程序可能只对数据中一个 XML 节点的值感兴趣，由此只需显示数据的数据的这一部分，而无需担心‘带上’XML 结构的剩余部分才能确保上下文情景。

用户可以在显示相同 XML 信息的任何应用程序中编辑数据并且使得在引用数据该部分的所有位置中数据马上更新（连同任何适用的业务逻辑）。这种接收每个 XML 改变的实时消息的能力是有用的，因为它允许创建反映出在单个 XML 文档内各种编辑需求的重叠性质的编辑环境。

应用程序也可以共享错误信息。用户定义的内容错误集合可以被存储在每个数据存储中。例如，业务逻辑可以指示<startDate>节点必须具有<endDate>节点之前的值。为了使多个数据消费者能集体共享它们的错误，在每个 XML 存储中包括错误存储，用于存储 XML 中节点的列表+错误（由错误文本和名称组成）。正如 XML 改变一样，一个客户可以创建错误，而该错误改变接着被广播给每个客户。这样，多个应用程序可以依赖于将验证逻辑在该数据的所有表示中共享的单个实现。换而言之，在正显示 XML 数据的每个应用程序中无需复制相同的逻辑。

撤销存储可以是全局撤销存储 310 和/或撤销存储可以包括在每个 XML 存储中。每个数据消费者的改变请求可以被连接到一个撤销栈中，诸如撤销存储 310，它将每个改变与所有相关的改变组合起来，这样每个改变都能作为一整体撤销。这使所有的客户能请求‘撤销’整个的最近改变，保持整个文档处于已知的“良好”状态。

数据的同步不限于一组预定义的数据消费者的集合。换而言之，新的数据消费者可以在任何时候对任何 XML 数据注册以获取通知，并且马上能够如同所有其他客户一样编辑相同的数据。例如，最初只有外部数据消费者 1 和 2 可以共享数据。稍后，一个或多个数据消费者可以使用数据存储注册，并且开始共享数据。

一个数据消费者担当 XML 数据的“所有者”并且负责：维护 XML 数据的持久形式；向请求数据消费者提供数据的副本；从数据消费者接收对数据的改变请求；以及将改变的通知发送给经注册的数据消费者。依照一个实施例，数据存储担当所有者并处理所有的更新和对每个数据消费者的通知。XML 数据存储包括可用于诸如字处理应用程序、电子表格应用程序、幻灯片演示程序和其他数据消费者之类的不同应用程序的一组接口。各接口针对：获取期望的 XML 数据片；通知数据存储数据消费者想要对数据存储作出的改变；以及注册以便从 XML 数据存储接收有关

其他数据消费者对存储项目作出的改变的通知。

只要数据存储将改变通知数据消费者，那么数据消费者可以：什么也不做并接受改变；请求一个或多个副效应改变以及拒绝改变。副效应改变一般涉及添加响应于对数据存储作出的其他改变而起始改变的逻辑。例如，使用股票调查笔记的数据消费者可能希望在数据存储中的<stockSymbol/>节点改变时接收通知，并且响应于这一改变向 web 服务提交数据并更新数据存储内的<stockData/>子树。

为了撤销/取消，可以将副效应改变与原始改变捆在一起，并且用数据存储对它们进行不同地处理。对它们进行不同地处理是因为副效应改变是响应改变而被请求的，它自身尚未提交给 XML 数据存储。

如果数据消费者（314 和/或 318）请求对 XML 数据存储 302 的改变，可以出于不同的原因拒绝该改变，包括：改变是无效的（例如不合式的 XML）；改变被数据消费者中的某些逻辑拒绝等。

一些数据消费者可以将它们自身版本的数据保存在独立于 XML 数据存储 302 维护的存储器 320 中。维护该 XML 数据的多个副本会导致问题，包括副本可能没有同步（例如，在性质面板中的‘标题’与内联在文档中显示的‘标题’不相匹配）。为了解决该问题，在会话期间维护每一 XML 数据片的一个“主”副本。接着在会话期间由多个数据消费者使用该主副本。当会话结束时，可以更新数据的其他副本以便反应 XML 数据存储的当前状态。依照一个实施例，数据存储 302 可以被配置成合并来自不同数据存储的相同项目，接着保存每个副本并在稍后退出。当接收到对公共数据项目的请求时，数据存储 302 将这两个数据存储项目包裹在单个父节点中；创建导入与每个数据项目相关联的模式的合并的 XSD；并且将用于该存储项目的接口传递给数据消费者。

数据存储器 302 被配置成检测过多的递归，并且在检测到时，如果数据存储检测到响应于给定改变的副效应的循环就引起自动失败。依照一个实施例，深度超过 16 层或者总的副效应超过 1000 个的循环被认为是过多的。数据存储也可以被配置成当 XML 数据存储发现改变在结构上是无效的时，自动地拒绝任何改变及其副效应。这意味着如果客户请求结构化改变，并且发现该改变在结构上无效，那么数据存储将其自身恢复为最近已知的良好状态并生成可传递给其他数据消费者的错误。

每个数据消费者（内部 314/外部 318）也可以拒绝无效的改变。例如，数据消费者可以包括其自身的验证层。如果在数据消费者内请求的改变是无效的，那么用

其现有的验证层拒绝该改变，并且使其自身的数据存储退回，而无需通知 XML 数据存储。如果该改变是来自 XML 数据存储的，那么数据消费者返回由数据存储抛出的事件拒绝，而数据存储发起取消以返回其'最近已知良好'状态。

在由其他数据消费者请求对数据存储作出改变的情况下，如果数据存储具有与当前数据相关联的 XML 模式集合（305），那么该数据存储会试图验证那些改变。如果模式存在，那么数据存储就拒绝任何结构上的无效。

为了支持数据绑定，诸如内部数据消费者 1 314 之类的内部应用程序数据消费者处理 XML 数据存储内和文档表面 315 上的动作之间的交互。当用户编辑数据绑定字段时，该改变影响文档的内容（于是向应用程序的撤销栈添加一记录），但是也影响数据存储的 XML 内容（于是向数据存储的撤销栈添加一记录）。为了帮助确保表面和数据一直保持同步，应用程序的撤销栈（用户与之交互）能够将表面改变连同相应的 XML 数据存储撤销引用‘绑定’到一个撤销记录中，确保每个栈顶部项目的撤销能够保持应用程序和数据存储处于相同状态。

不同的替换可用于处理用户发起的撤销，包括：为包括主应用程序的每个数据消费者维护分开的撤销栈；共享全局撤销栈；以及基于当前的焦点限制数据消费者的撤销。

当使用全局撤销栈时，数据消费者直接将撤销请求传递给主应用程序，主应用程序接着将最后一个项目从其撤销栈取消（撤销会是相同的，而不管聚焦在应用程序帧中的哪里）。这意味着用某些普适记录将 XML 数据存储的所有改变都集中到主应用程序的撤销栈上（例如“撤销性质编辑”）。例如，如果在字处理应用程序中用户向<company/>(公司)字段键入“Microsoft Corp.（微软公司）”，那么该操作导致数据存储的撤销栈包括该动作。接着，如果用户要在字处理应用程序中按下撤销以便移除该文本，那么字处理应用程序撤销其撤销栈上最近的操作（并通知数据存储撤销其栈上最近的操作），这会导致其他客户作出该动作。相反地，如果用户接着在字处理应用程序中键入某些非绑定文本并在面板中按下撤销，那么其他客户会丢弃该请求直至主应用程序，这会从其撤销栈移除最近的动作（在这种情况下，对文档表面的编辑）。

当数据消费者拒绝 XML 数据存储发送的改变时，XML 数据会以“坏的”业务逻辑状态结束。例如，假设存在对经费报告执行检查的业务逻辑。逻辑包括检查行项目是否超过$100；如果是，那么数据消费者拒绝对<LineItemAmount/>（行项目量）的更新。如果不是，那么数据消费者使用新的行项目量更新总数。如果总数

超过$500，那么数据消费者拒绝对<reportTotal/>（报告总价）的更新。现在使用以上逻辑，假设用户输入$50 的发票行，使得总价高于$500，那么第一逻辑检查成功，但是第二逻辑检查拒绝对总价的更新。这意味着如果仅撤销最近的改变，那么就会有发票使得行项目的总和与总价不匹配。结果，依照一个实施例，撤销原始改变的所有副效应。

数据存储 302 用作允许将这些处理捆在一起以供撤销的处理机制。依照各实施例，使用处理‘拒绝’的三个不同的替换。首先，数据存储可以发出撤销改变以返回到有效状态（也称为“退回”）。第二，撤销和取消具有不一致（non-parity），而第三，客户都不能取消。

第一个替换是使得 XML 数据存储发出撤销改变，以回到有效状态。这必须撤销所有的操作返回触发商业逻辑错误的改变。在该替换中，XML 数据存储可以发出将‘撤销’标记设置为真的改变请求，并使得数据消费者对其自身执行这些改变；以及 XML 数据存储可以发出带有其‘撤销’标记设置的改变请求。以下是一个示例。

```
用户编辑客户 1 中的节点 A{
    (客户 1 作出内部逻辑)
    通知 XML 数据存储{
        存储更新
        通知客户 2 有关 A{
            改变 B{
                使其排队等候，返回 OK
            }
        }
        通知客户 1 有关 B{
            内部逻辑
                改变 C{
                    内部逻辑
                    通知 XML 数据存储{
                        使其排队等候，返回 OK
                    }
                    OAC
                }
```

```
        }
        通知客户 2 有关 C{
            改变 D{
                使其排队等候，返回 OK
                }
            }
        通知客户 1 有关 D{
            内部逻辑
                **拒绝**
            退回 DOM 改变到 D
            返回失败
        }
        通知客户 2 撤销 D
        通知客户 2 撤销 C
        通知客户 1 撤销 C - *在拒绝后退出时撤销*
        通知客户 1 撤销 B - *在拒绝后退出时撤销*
        通知客户 2 撤销 B
        通知客户 2 撤销 A
        返回失败给初始改变请求
    }
}
```

本质上，数据消费者接收两个 XML 数据存储调用（在退出父改变 A 时请求撤销改变 B 和 C）或者两个 XML 文档对象管理器放弃同步。

另一替换是保持取消和撤销之间当前的不一致，在使得数据消费者‘取消’当前改变会让解决方法处于无效业务逻辑状态的范围内。在这种情况下，XML 数据存储*仅*退回被拒绝/取消的改变，撤销该改变，但是接着就只是停止——而不撤销该存储的撤销栈上的任何进一步的改变。

数据消费者也可以维护它们自身唯一的撤销栈。然而，应用程序应该‘知道’当它们栈上顶端的动作与来自实时同步边界另一侧的请求操作相匹配时，就不将附加动作添加到它们的撤销栈上。例如，如果用户在文字处理应用程序中向<comopany/>字段键入“Microsoft Corp.”，那么该操作会被‘实时同步’并导致在每个注册的数据消费者的撤销栈上出现撤销动作。接着，如果用户要在字处理应用

程序中按下撤销以便移除刚输入的文本，那么字处理应用程序会撤销其撤销栈上最近的操作（并通知数据存储撤销其栈上的最近操作），但是这次数据消费者会看到该请求并从撤销站取消最近的动作（包括任何副效应），因为该存储的撤销栈必须有必要与数据存储相匹配。如果用户接着在数据消费者中按下撤销，那么数据消费者会从其撤销站移除最近的动作，并且当从数据存储得到撤销操作时，文字处理应用程序会注意到它与其撤销栈上的最近动作匹配并也将它移除。相反地，如果在用户在数据消费者中按下撤销之前，用户接着在文字处理应用程序中键入某些未绑定文本（向字处理应用程序添加撤销记录），那么客户会从其撤销栈移除最近的动作，并且存储会作出相同的动作，然而字处理应用程序则添加另一动作（因为在撤销栈上的最后一项不是最近的 XML 改变）。然而，字处理应用程序会存储该改变连同需要执行重做以返回其原始状态的事实。

另一替换涉及使得数据消费者彼此知晓。可以在数据消费者之间传递撤销控制。例如，如果用户在一个应用程序中编辑<company/>字段，这接着被广播到另一数据消费者，那么撤销栈会如同：

存储	应用程序	数据消费者 1
标记（传递给数据消费者 1）	公司→“Microsoft Corp.”＋存储撤销	公司→“Microsoft Corp.”

在这种情况下，XML 数据存储上的撤销记录不会主存该处理。相反，它会主存允许它将控制传递给客户以完成该撤销动作的标记。假设用户接着进入字处理应用程序并执行撤销：应用程序的画布会向回更新，并且接着将控制传递给数据存储。XML 数据存储本会尝试撤销最近的处理，但是却看到标记并将控制传递给数据消费者用于该撤销。数据消费者会执行到该存储的撤销请求，并接着将其作为撤销广播给所有其他的客户。这意味着如果用户在客户内执行了动作，之后是对字处理应用程序中不同字段的动作，那么撤销栈会如同：

存储	应用程序	数据消费者 1
标记（传递给进程 1）	公司->“Microsoft Corp.”＋存储撤销	公司->“Microsoft Corp.”
日期→“January 20,2005（2005 年 1 月 20 日）”	日期→“January 29,2005（2005 年 1 月 29 日）”＋存储撤销	标记（传递给主应用程序）

在这种情况下，如果用户的下一个动作是在主应用程序中撤销，那么主应用程序（和数据存储）会作出撤销动作，数据消费者会对其自身的 DOM 执行撤销并丢弃撤销标记。如果用户的下一个动作是在数据消费者中撤销，那么数据消费者会将把对该动作的控制交给主应用程序（因为顶部的动作仅是‘标记’），并且会执行相同的主应用程序撤销。

图 4 示出了实时同步的示例。在最简单的情况之一下，“实时同步”是指使得在一个位置（例如字处理器应用程序）出现的对 XML 数据的用户编辑实时地反映在另一位置/应用程序的 UI 中的能力。当编辑文档时，影响 XML 数据的改变被传递给对数据感兴趣的其他已注册的数据消费者，诸如字处理应用程序和性质面板。这有助于确保每个应用程序的 XML 数据的内容保持相同。

参考窗口 400，打开文档 415 以供编辑并显示性质面板 420。在性质（property）面板(405)和文档（410）中都显示了标题。假设窗口 400 中的标题（Title）405 从“Data Binding-Live Sync Integration”改变成“Foo Bar Biz”，如窗口 425 中的标题 435 和标题 440 中所示。一旦在性质面板标题 435 内更新标题，就将改变发送给字处理应用程序，这样使得它可以接受或拒绝改变。在该示例中，应用程序接受使用性质面板应用程序对标题作出的改变，并且文档内的标题 440 被更新。

图 5-8 示出了用于在数据消费者之间的 XML 数据的实时同步的进程。

当阅读对在此提供的例程的讨论时，应当了解，各种实施例的逻辑操作是作为(1)运行于计算机系统上的一系列计算机实现的动作或程序模块，以及/或者(2)计算机系统内互连的机器逻辑电路或电路模块来实现的该实现是取决于实现本发明的计算机系统的性能要求来选择的。因此，所例示的并且构成在此所述的实施例的逻辑操作被不同地表示为操作、结构性设备、动作或模块。这些操作、结构性设备、动作和模块可用软件、固件、专用数字逻辑以及它们的任何组合来实现。

图 5 示出了两个客户和 XML 数据存储之间的交互。

数据存储 520 接收使用应用程序 510 对节点 A 作出的用户编辑。数据消费者 1 530 从数据存储 520 接收对节点 A 的改变的通知。作为对节点 A 的改变的结果，数据消费者 1 请求对节点 B 的副效应改变。数据存储 520 使得副效应改变 B 进入队列以供稍后执行。一旦使得来自数据消费者 1 的所有副效应改变进入队列，数据存储 520 就通知数据消费者 2（540）节点 A 的改变。数据消费者 2 请求对节点 C 的副效应改变。作为响应，数据存储使节点 C 改变进入队列以供稍后执行。此时，

完成了与节点 A 有关的处理，但是仍然有对节点 B 和节点 C 的未决副效应改变。由于数据消费者 1 请求对节点 B 的改变，数据存储将有关对 B 提出的改变的通知发送给数据消费者 2。数据消费者 2 不作出任何改变作为响应，并接受这一改变。类似地，数据消费者 2 要求对节点 C 的改变，因此数据存储将节点 C 的改变的通知发送给数据消费者 1。数据消费者 1 接受这一改变。在该示例中，所有改变被所有感兴趣的数据消费者接受。因此，数据存储提交对数据存储的改变。简而言之，数据存储允许每个数据消费者响应改变作出任何改变，数据存储以接受到这些改变的顺序来执行和通知这些改变，由此允许连续地通知它们。

图 6 显示了两个外部数据消费者和对 XML 数据存储的改变之间的交互。

在原始改变是由诸如图 5 中所示的创建文档的应用程序之类的内部客户生成的情况下，数据消费者作出些许改变。该示例示出来自多个客户的改变的两条基本规则。第一条规则是顶级改变在深度上首先发生。第二条规则涉及使得副效应改变进入队列。

第一条规则是指改变的副效应在任何新的改变可能发生之前发生。考虑以下示例，其中数据消费者 1 执行函数以作出两个改变。第一个改变是对节点 A 做出的，而第二个改变是对节点 B 做出的。

当数据消费者 1（630）请求对节点 A 的改变时，该通知在 XML 数据存储 620 接收到改变节点 A 的请求之后被发送给数据消费者 2（640）。作为响应，数据消费者 2 请求对节点 C 的副效应改变，这被数据存储放入队列以供稍后执行。数据消费者 1 接着请求对节点 B 的改变，这也被放入队列，因为对节点 A 的改变还未完成。数据消费者 2 接受对节点 A 的改变，并且作为响应，数据存储执行排队的对 C 的副效应改变。数据消费者 1 接收对节点 C 的改变的通知，并可以响应于这一改变。在该示例中，数据消费者 1 接受这一改变。数据存储接着执行数据消费者 1 请求的排队的对 B 的副效应改变。数据消费者 2 从它会对其作出响应的数据存储中接收改变 B 的通知。数据消费者 2 接受这一改变并且各改变被提交给数据存储。

在这种情况下，数据消费者 2 首先响应对 A 的改变，这样使得以下两行代码触发两个唯一的改变：

```
…
doc.CustomXMLParts(1).SelectSingleNode().AddNode(“foo”,”bar”)
doc. CustomXMLParts (1).SelectSingleNode().AddNode(“foo2”,”bar”)
…
```

第二条规则是指由于请求改变的副效应而将其被放入队列，因此在第一个改变的副效应发生之前队列内会放入多个改变的事实。

图 7 显示了涉及多个副效应改变的过程。以下示例示出了以下改变。假设当数据消费者 1（730）接收到节点 A 改变的通知时，希望执行对节点 B 和 C 的副效应改变。响应于对节点 B 的改变的通知，数据消费者 2（740）希望执行对节点 D、E 和 F 的副效应改变。

参见图 7，可以看出作为对节点 A 的改变的结果，数据消费者 1 使得其所有副效应改变在与节点 B 的改变有关的副效应之前执行。因此，对节点 C 的改变会在对节点 D、E 和 F 的改变之前发生。

图 8 示出了显示最后执行调用程序的副效应的进程。在这种情况下仍可以看出由数据消费者 1 自身生成的副效应会在所有其他的客户有机会看到改变并生成它们自身的副效应之后发生。这是必须的副效应，因为 XML 数据存储要在它能够将成功条件返回给调用程序并允许调用程序提供使用其来作出副效应的事件之前将改变通知每个客户（以确保每个客户有机会接受/拒绝该改变）。可以预知的是请求改变的数据消费者应该是最后的，以便听取 XML 数据存储的所有客户是接受还是拒绝这一改变。同样，这有助于确保在两个改变冲突且是非结构化的情况下，调用程序的改变会获胜，这也是通常期望的结果。

以上的说明书、示例和数据提供了对本发明的组分的制造和使用的完整描述。

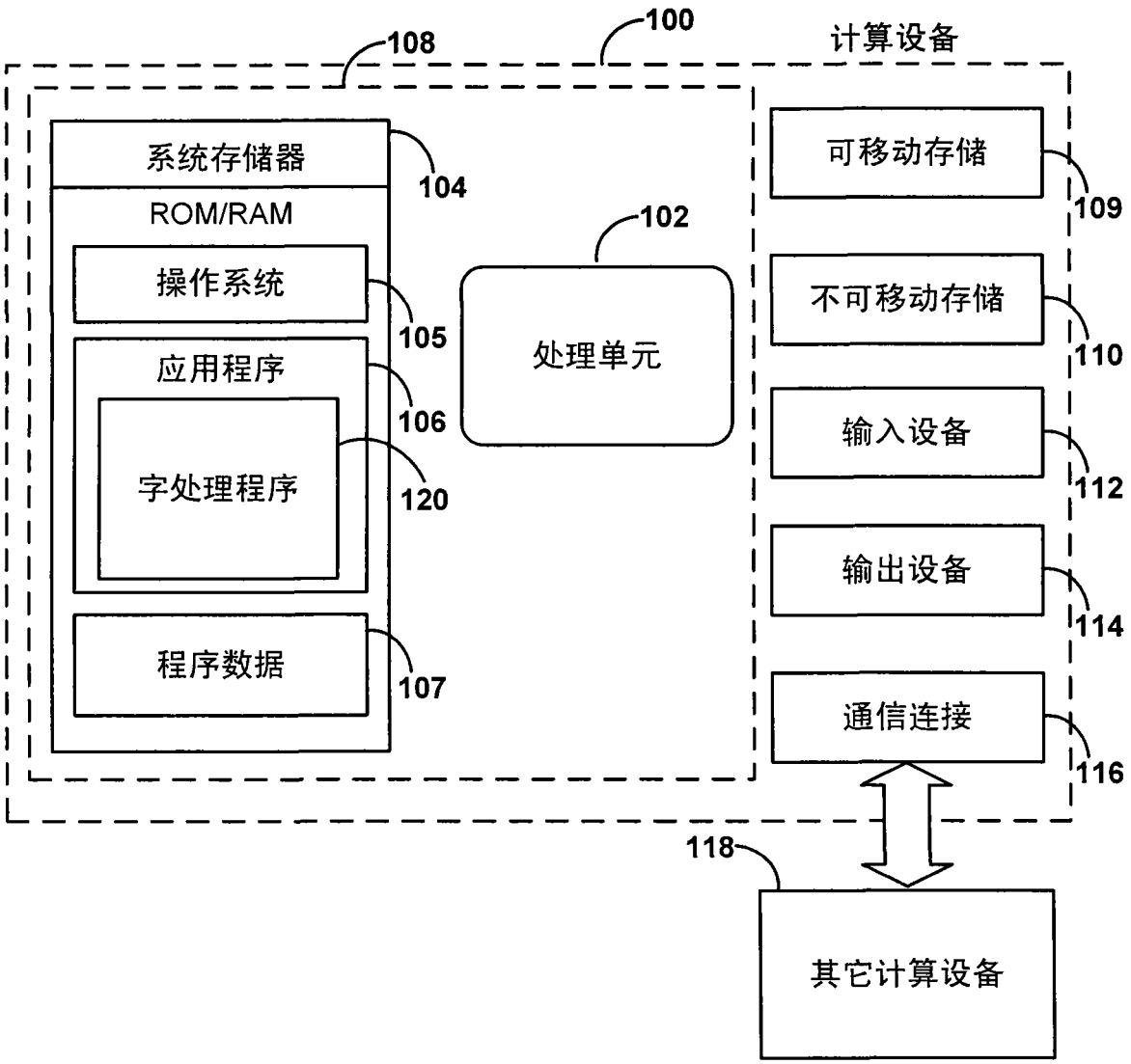

100
计算设备
108
系统存储器
104
ROM/RAM
操作系统
105
应用程序
106
字处理程序
120
程序数据
107
102
处理单元
可移动存储
109
不可移动存储
110
输入设备
112
输出设备
114
通信连接
116
118
其它计算设备

图 1

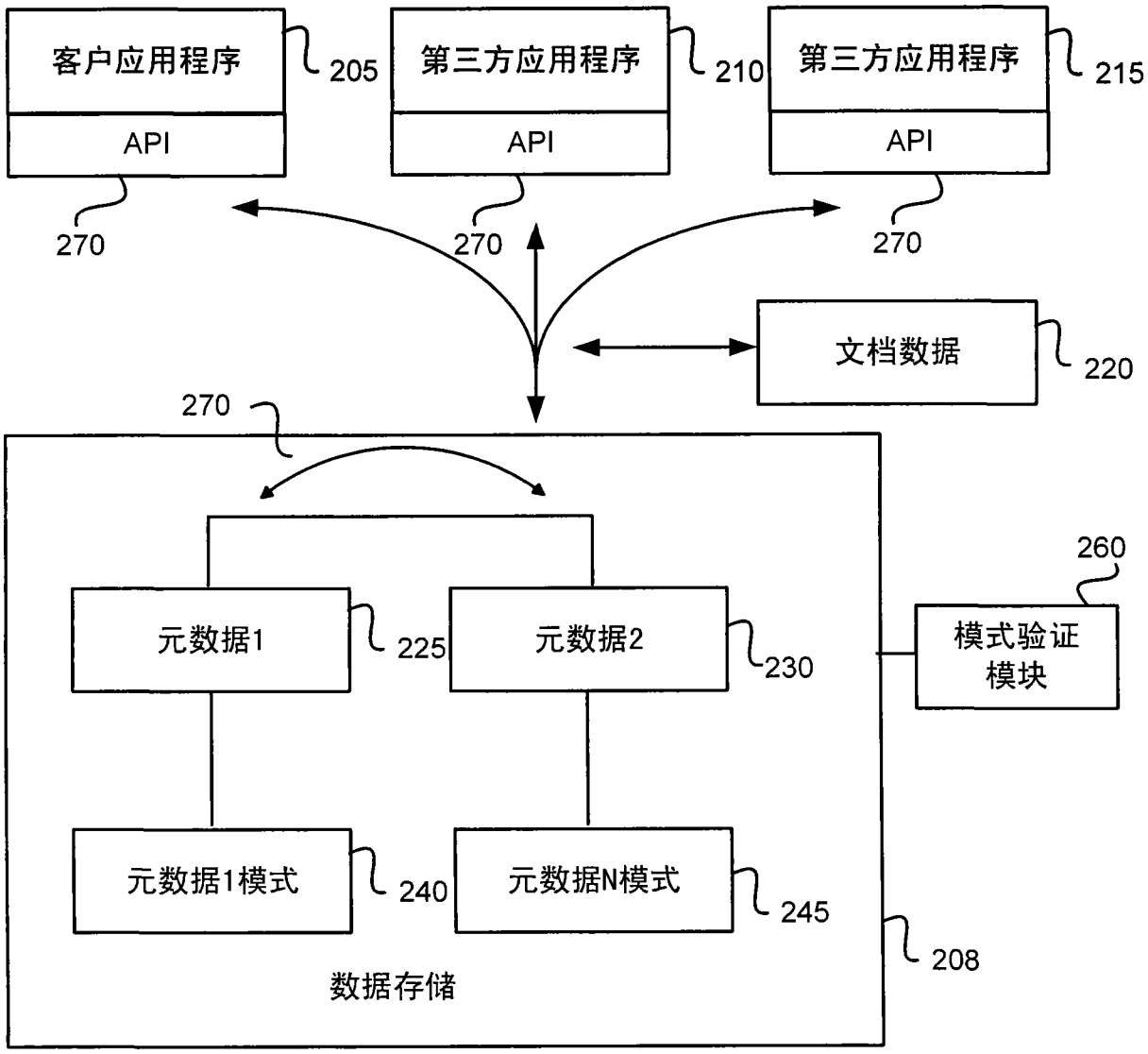

客户应用程序
205
第三方应用程序
210
第三方应用程序
215
API
API
API
270
270
270
文档数据
220
270
260
元数据1
225
元数据2
230
模式验证
模块
元数据1模式
240
元数据N模式
245
208
数据存储

图 2

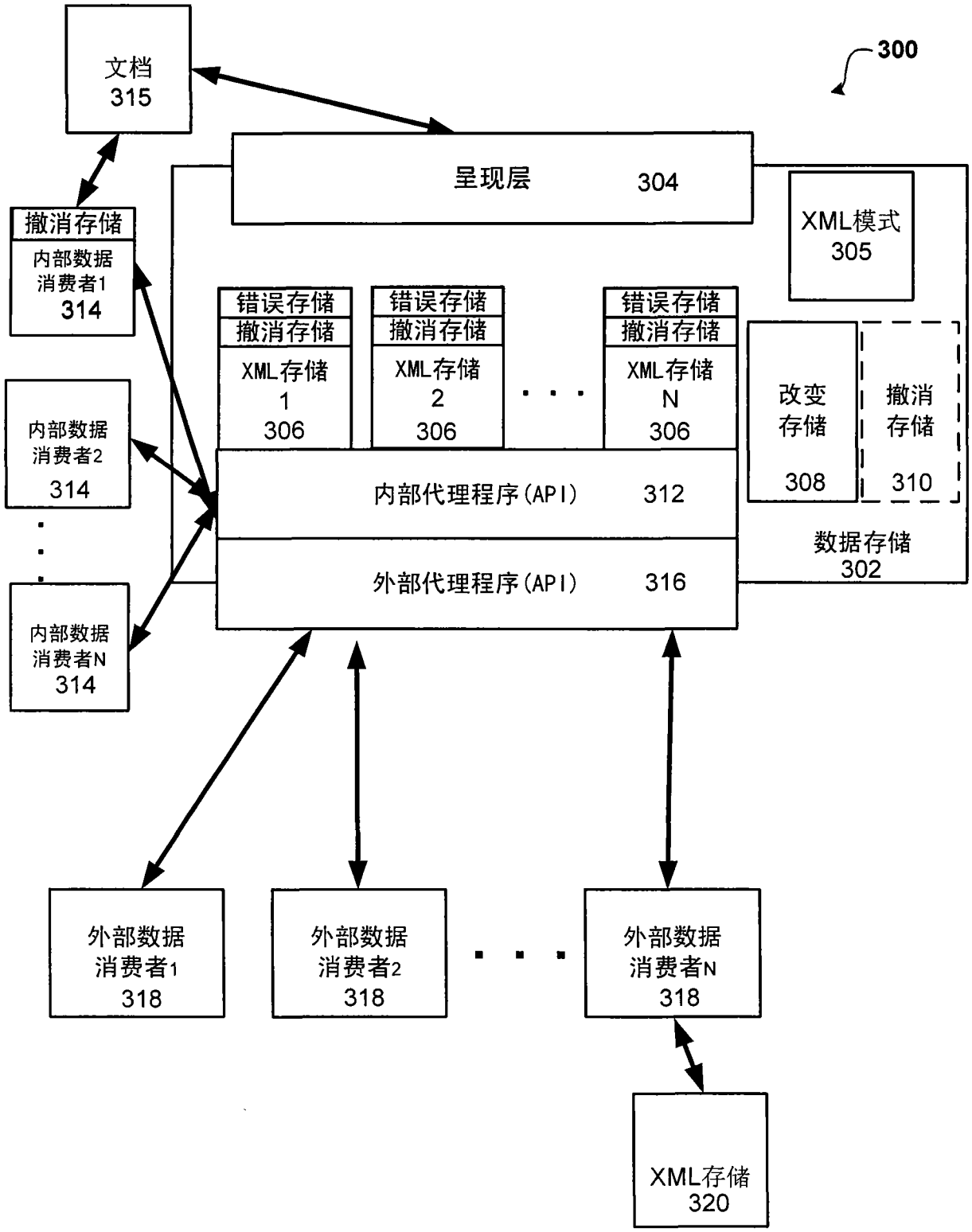

文档
315
300
呈现层
304
撤消存储
内部数据
消费者1
314
XML模式
305
错误存储
撤消存储
XML存储
1
306
错误存储
撤消存储
XML存储
2
306
错误存储
撤消存储
XML存储
N
306
改变
存储
308
撤消
存储
310
内部数据
消费者2
314
内部代理程序(API)
312
数据存储
302
外部代理程序(API)
316
内部数据
消费者N
314
外部数据
消费者1
318
外部数据
消费者2
318
外部数据
消费者N
318
XML存储
320

图 3

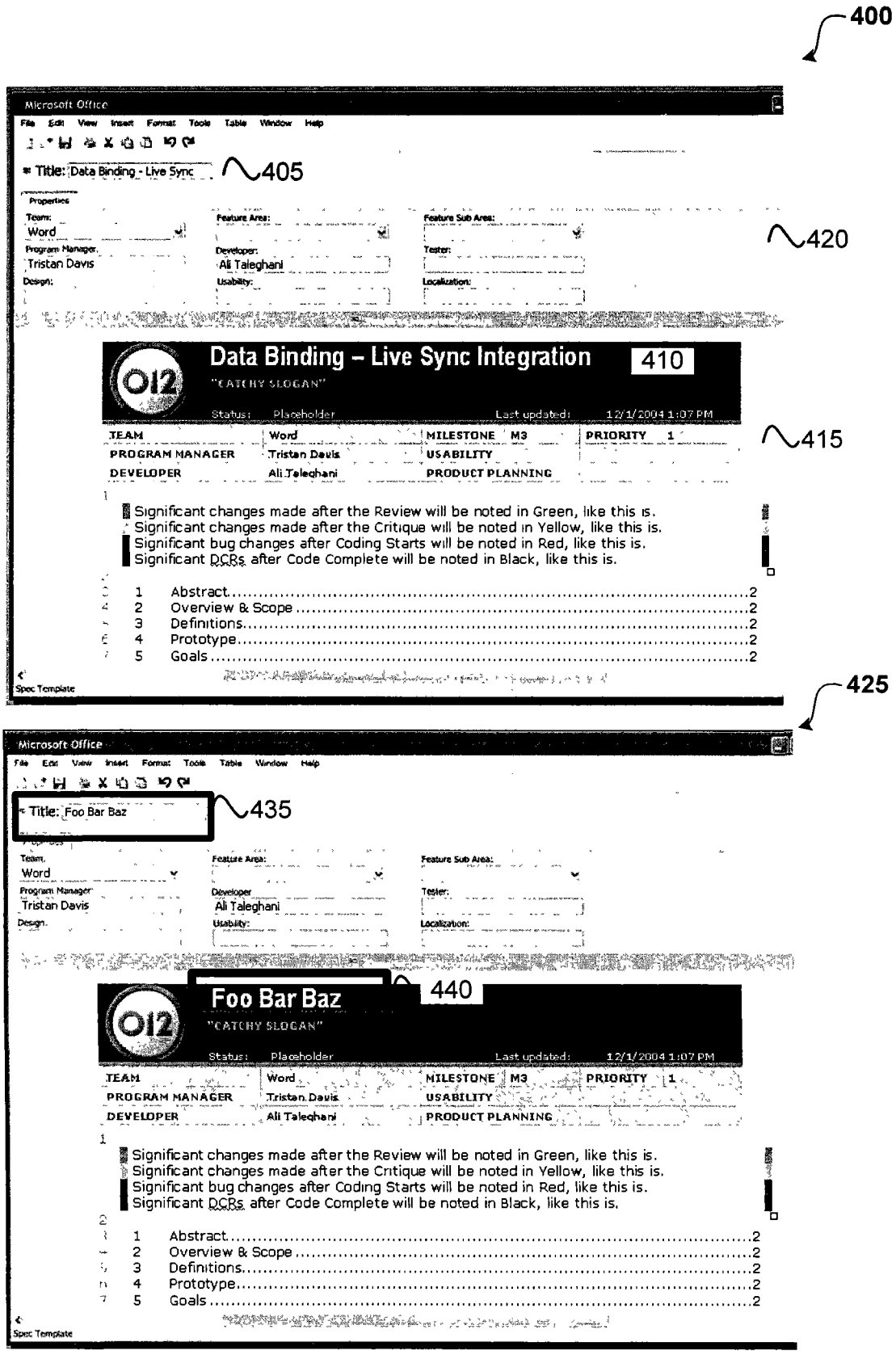

400
Microsoft Office
Title: Data Binding - Live Sync
405
Word
Tristan Davis
Ali Taleghani
420
Data Binding – Live Sync Integration
410
"CATCHY SLOGAN"
Status: Placeholder
Last updated: 12/1/2004 1:07 PM
TEAM
Word
MILESTONE M3
PRIORITY 1
PROGRAM MANAGER
Tristan Davis
USABILITY
DEVELOPER
Ali Taleghani
PRODUCT PLANNING
415
Significant changes made after the Review will be noted in Green, like this is.
Significant changes made after the Critique will be noted in Yellow, like this is.
Significant bug changes after Coding Starts will be noted in Red, like this is.
Significant DCRs after Code Complete will be noted in Black, like this is.
1 Abstract....2
2 Overview & Scope....2
3 Definitions....2
4 Prototype....2
5 Goals....2
Spec Template
425
Microsoft Office
Title: Foo Bar Baz
435
Foo Bar Baz
440
"CATCHY SLOGAN"
Status: Placeholder
Last updated: 12/1/2004 1:07 PM
TEAM
Word
MILESTONE M3
PRIORITY 1
PROGRAM MANAGER
Tristan Davis
USABILITY
DEVELOPER
Ali Taleghani
PRODUCT PLANNING
Significant changes made after the Review will be noted in Green, like this is.
Significant changes made after the Critique will be noted in Yellow, like this is.
Significant bug changes after Coding Starts will be noted in Red, like this is.
Significant DCRs after Code Complete will be noted in Black, like this is.
1 Abstract....2
2 Overview & Scope....2
3 Definitions....2
4 Prototype....2
5 Goals....2
Spec Template

图 4

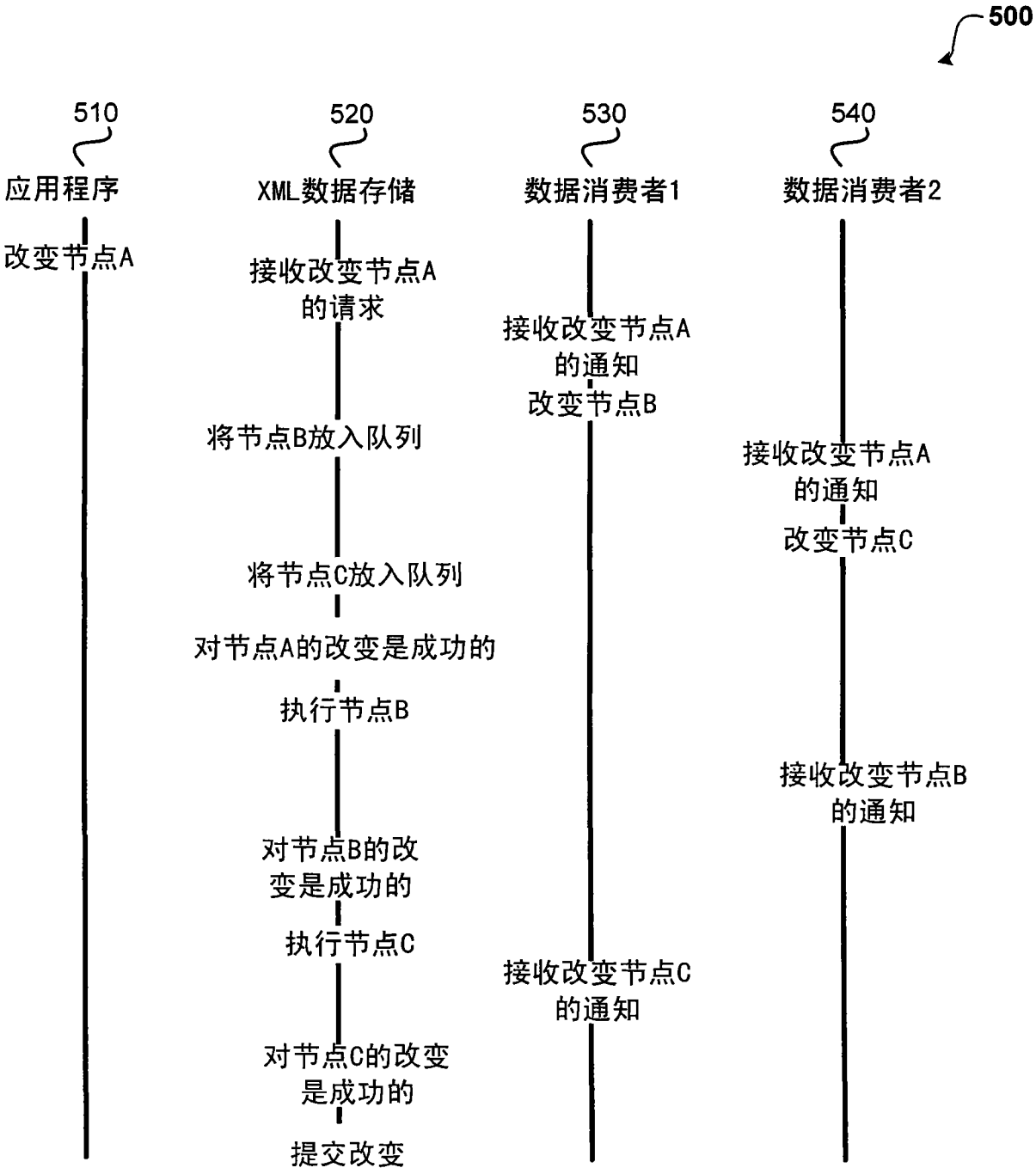

500
510
520
530
540
应用程序
XML数据存储
数据消费者1
数据消费者2
改变节点A
接收改变节点A
的请求
接收改变节点A
的通知
改变节点B
将节点B放入队列
接收改变节点A
的通知
改变节点C
将节点C放入队列
对节点A的改变是成功的
执行节点B
接收改变节点B
的通知
对节点B的改
变是成功的
执行节点C
接收改变节点C
的通知
对节点C的改变
是成功的
提交改变

图 5

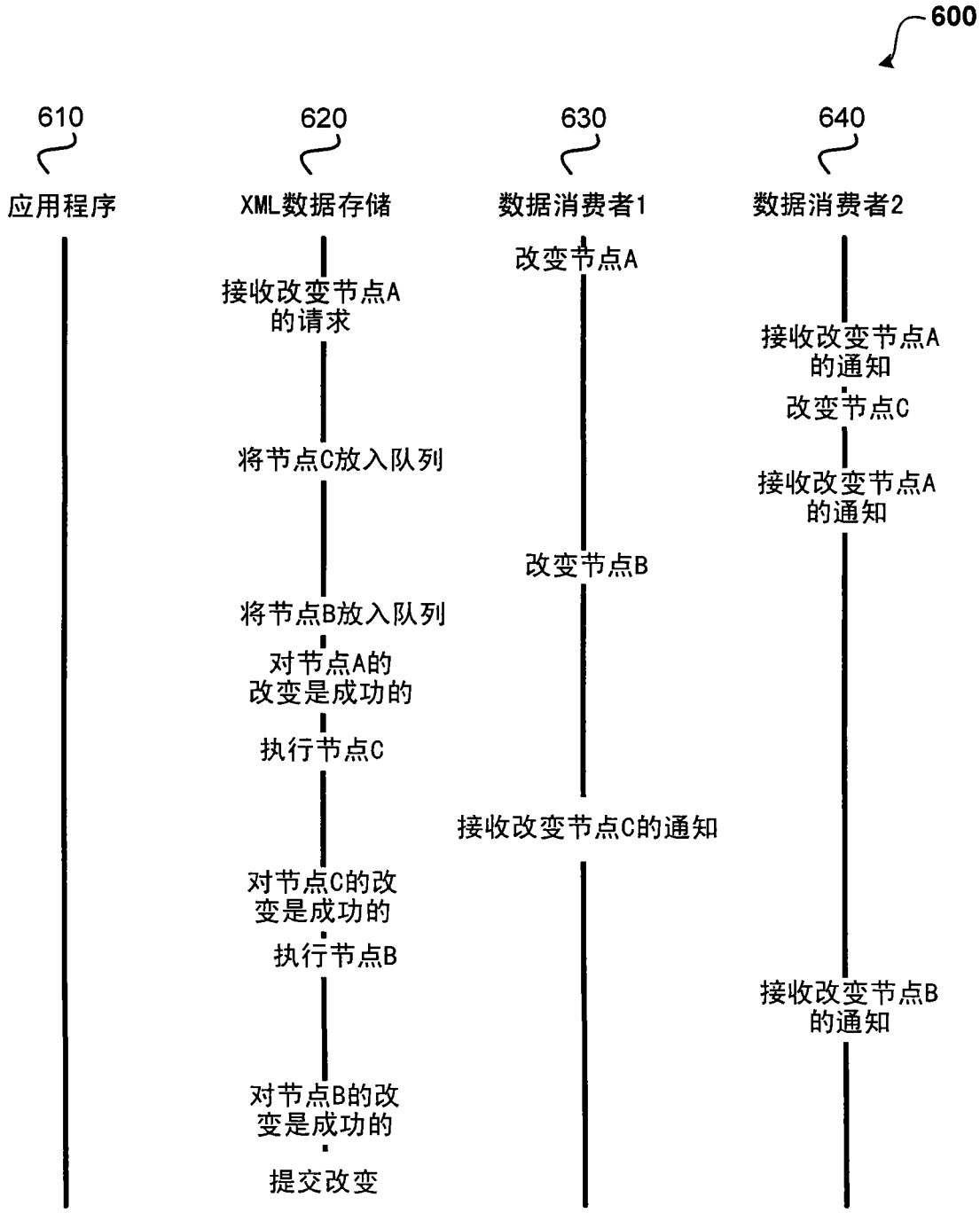

600
610
620
630
640
应用程序
XML数据存储
数据消费者1
数据消费者2
改变节点A
接收改变节点A的请求
接收改变节点A的通知
改变节点C
将节点C放入队列
接收改变节点A的通知
改变节点B
将节点B放入队列
对节点A的改变是成功的
执行节点C
接收改变节点C的通知
对节点C的改变是成功的
执行节点B
接收改变节点B的通知
对节点B的改变是成功的
提交改变

图 6

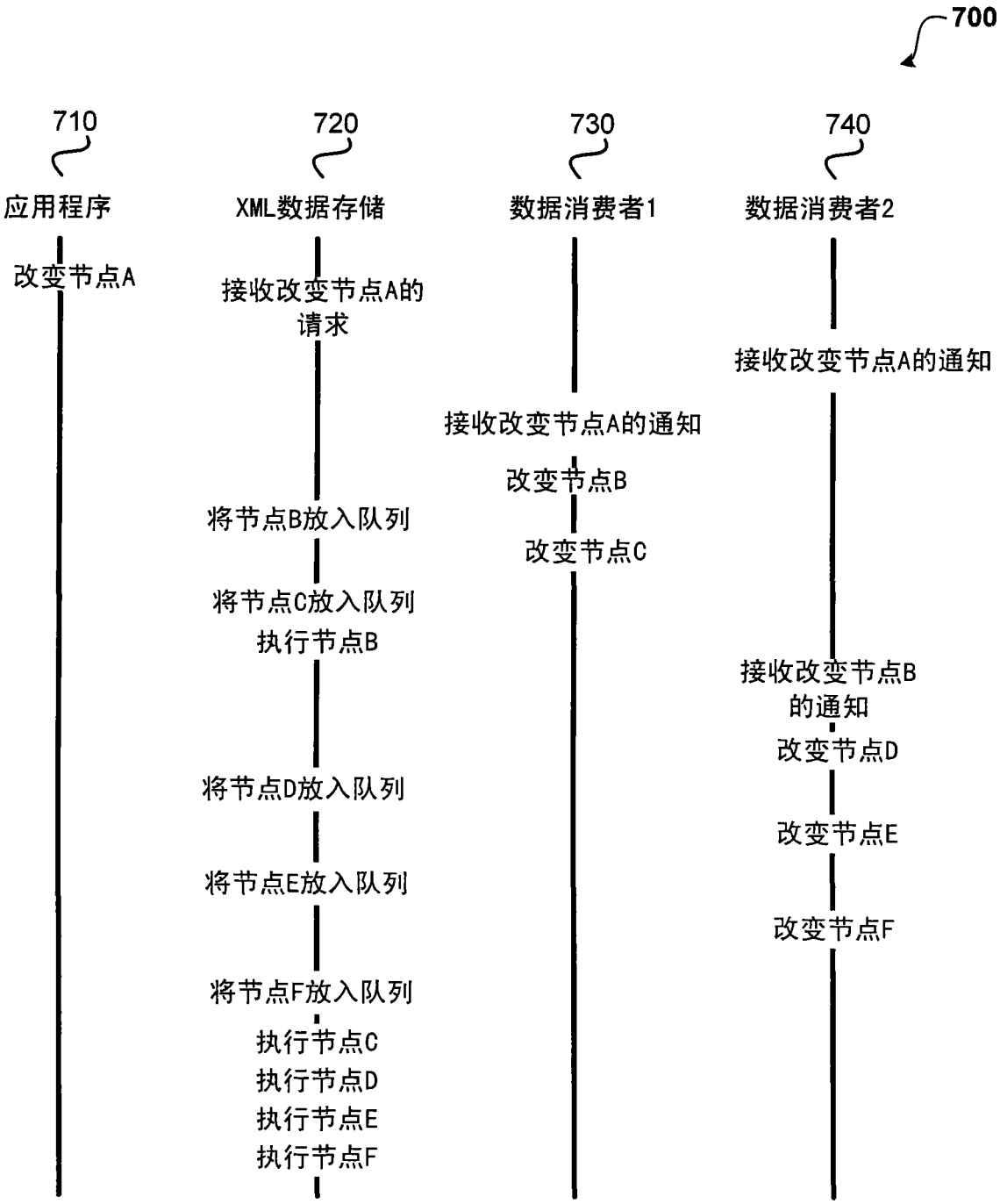

700
710
720
730
740
应用程序
XML数据存储
数据消费者1
数据消费者2
改变节点A
接收改变节点A的请求
接收改变节点A的通知
接收改变节点A的通知
改变节点B
将节点B放入队列
改变节点C
将节点C放入队列
执行节点B
接收改变节点B的通知
改变节点D
将节点D放入队列
改变节点E
将节点E放入队列
改变节点F
将节点F放入队列
执行节点C
执行节点D
执行节点E
执行节点F

图 7

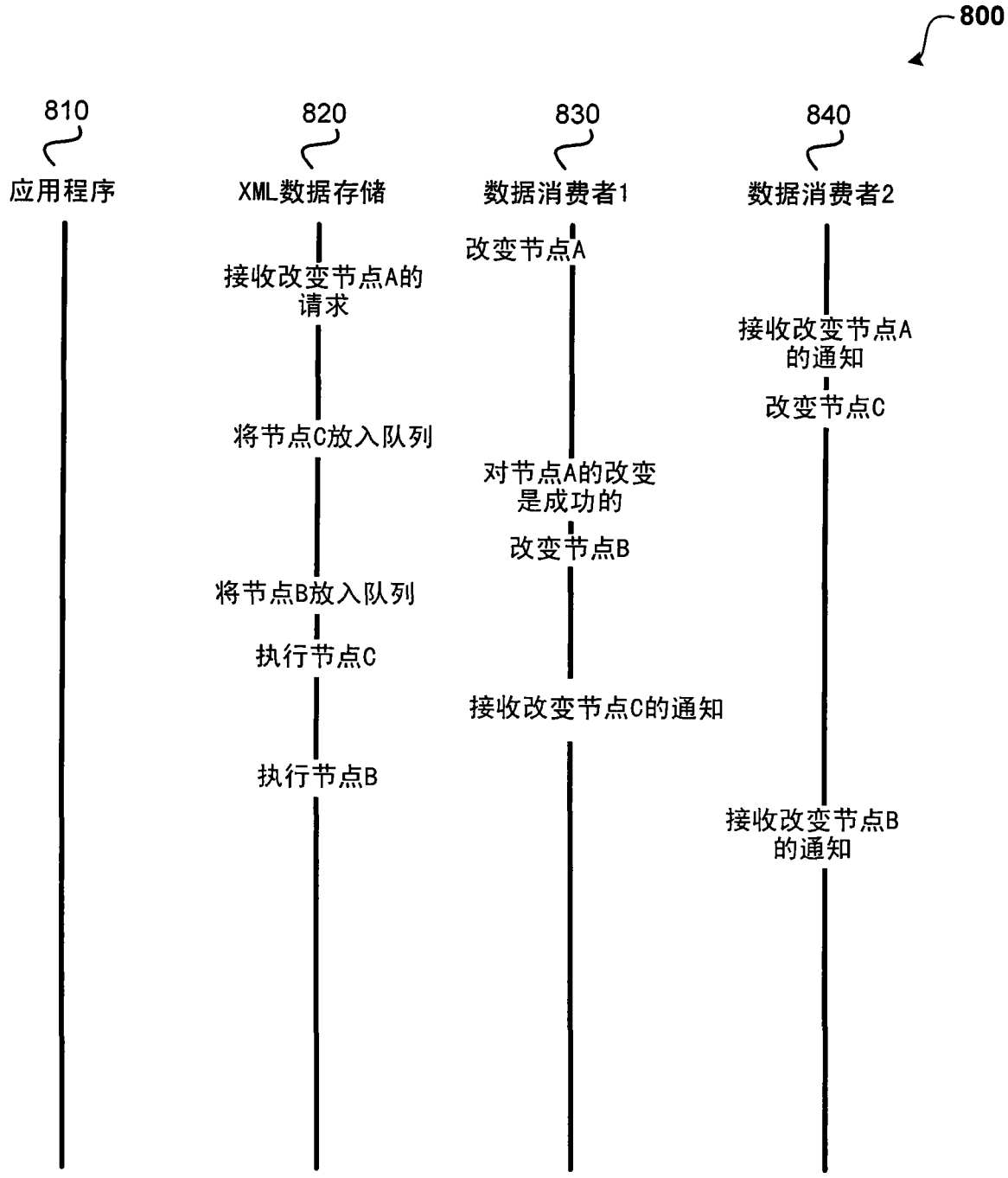

800
810
820
830
840
应用程序
XML数据存储
数据消费者1
数据消费者2
改变节点A
接收改变节点A的请求
接收改变节点A的通知
改变节点C
将节点C放入队列
对节点A的改变是成功的
改变节点B
将节点B放入队列
执行节点C
接收改变节点C的通知
执行节点B
接收改变节点B的通知

图 8