(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102999545 A
(43)申请公布日 2013.03.27

(21)申请号 201210335264.3
(22)申请日 2012.09.11
(30)优先权数据
61/533,744 2011.09.12 US
13/350,604 2012.01.13 US
(51)Int.Cl.
G06F 17/30(2006.01)

(71)申请人 微软公司
地址 美国华盛顿州
(72)发明人 J·D·卡普兰 S·法克斯
R·克里希纳瓦斯米 R·M·拜尔斯
R·A·拜因顿
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 陈斌

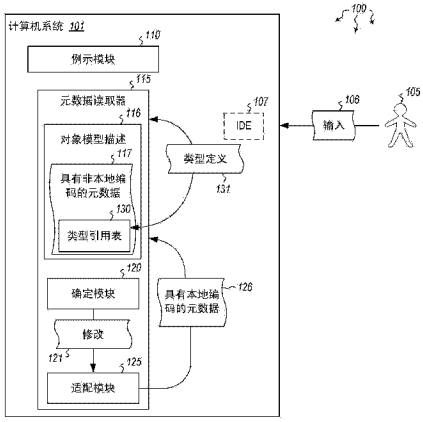
权利要求书 2 页 说明书 7 页 附图 8 页

(54)发明名称

高效提供相同类型的多个元数据表示

(57)摘要

本发明涉及高效提供相同类型的多个元数据表示。实施例涉及动态适配元数据供本地数据编码使用，以及高效地修改对象模型类型引用。在一个情形中，计算机系统在对象模型描述上例示元数据读取器，来访问该对象模型描述中的元数据的各个部分。元数据读取器被配置成读取本地元数据，其中本地元数据是以元数据读取器预期的编码表示的元数据。元数据读取器确定所访问的元数据是以非本地编码被编码，并然后确定要执行哪些元数据修改来将非本地编码变换成本地编码。计算机系统然后根据所确定的修改将对象模型的元数据从非本地编码动态地适配到本地编码。如此，对象模型可由本地运行时读取。



CN 102999545 A

1. 一种在包括多个计算系统的计算机联网环境中、在包括至少一个处理器和存储器的计算机系统处提供用于动态适配元数据来供本地数据编码使用的计算机实现的方法，所述方法包括：

在对象模型描述上例示元数据读取器来访问所述对象模型描述中的元数据的一个或多个部分的动作，所述元数据读取器被配置成读取本地元数据，所述本地元数据包括以所述元数据读取器所预期的编码表示的元数据；

所述元数据读取器确定所访问的元数据是以非本地编码被编码的动作；

所述元数据读取器确定哪些元数据修改要被执行来将非本地编码变换成本地编码的动作；以及

根据所确定的修改将所述对象模型的元数据从非本地编码动态地适配到本地编码的动作，使得所述对象模型可由本地运行时读取。

2. 如权利要求 1 所述的方法，其特征在于，所述动态适配允许用户读取采用多种不同非本地编码的元数据。

3. 如权利要求 1 所述的方法，其特征在于，所述动态适配在元数据导入－读取级发生。

4. 如权利要求 3 所述的方法，其特征在于，在所述元数据导入－读取级的动态适配防止运行时级的一个或多个后续适配。

5. 如权利要求 1 所述的方法，其特征在于，应用一可见性修改器来使得被动态适配的元数据内部或公共可见。

6. 如权利要求 1 所述的方法，其特征在于，一个或多个数据类型被重写以符合本地编码。

7. 如权利要求 1 所述的方法，其特征在于，元数据的本地编码和非本地编码被创建在同一元数据文件中。

8. 如权利要求 7 所述的方法，其特征在于，所述元数据读取器被配置成确定哪个用户正在请求元数据文件，以及基于该确定将元数据文件的合适编码传递给用户。

9. 如权利要求 7 所述的方法，其特征在于，对于各用户，所述编码是完整的且一致的。

10. 如权利要求 7 所述的方法，其特征在于，仅对于本地和非本地用户被不同编码的那些元数据实体以本地和非本地编码被编码。

11. 如权利要求 10 所述的方法，其特征在于，所述用户指定哪个类型的编码要被提供给他们。

12. 一种在包括多个计算系统的计算机联网环境中、在包括至少一个处理器和存储器的计算机系统处用于高效修改对象模型类型引用的计算机实现的方法，所述方法包括：

例示类型引用表的动作，所述类型引用表被配置成接收类型定义请求以及使用可由元数据读取器读取的元数据中存储的一个或多个类型引用权标来提供合适的类型定义作为响应；

所例示的类型引用表接收来自类型请求者对指定类型的类型定义请求的动作；

在所述类型引用表的元数据中实现类型引用权标，以基于哪个类型曾被请求以及哪个类型适合于所述元数据读取器的当前消费者来为所述元数据读取器动态地替换一个或多个类型的动作；以及

基于由所述元数据中的所述类型引用权标提供的被动态替换的类型来向类型请求者

提供所请求的类型定义的动作。

13. 如权利要求 12 所述的方法，其特征在于，非本地类型引用被本地类型引用原地替换。

14. 如权利要求 12 所述的方法，其特征在于，动态替换向元数据读取器提供了非本地类型的标准化视图。

15. 一种计算机系统，包括：

一个或多个处理器；

系统存储器；

存储有计算机可执行指令的一个或多个计算机可读存储介质，所述计算机可执行指令在由所述一个或多个处理器执行时使得所述计算系统执行一种用于动态适配元数据供本地数据编码使用的方法，所述方法包括：

在对象模型描述上例示元数据读取器来访问该对象模型描述中的元数据

的一个或多个部分的动作，所述元数据读取器被配置成读取本地元数据，所

述本地元数据包括以所述元数据读取器所预期的编码表示的元数据；

所述元数据读取器确定所访问的元数据是以非本地编码被编码的动作；

所述元数据读取器确定哪些元数据修改要被执行来将非本地编码变换成本地编码的动作；

根据所确定的修改将所述对象模型的元数据从非本地编码动态地适配到本地编码的动作，使得所述对象模型可由本地运行时读取，其中对元数据的本地和非本地编码被创建在同一元数据文件中；

所述元数据读取器确定哪个用户正在请求所述元数据文件的动作；以及

基于所述确定，将所述元数据文件的合适的编码传递给所述用户的动作。

高效提供相同类型的多个元数据表示

技术领域

[0001] 本发明涉及计算机技术，尤其涉及高效提供相同类型的多个元数据表示的技术。

背景技术

[0002] 计算机已变得高度集成于工作、家庭、移动设备以及许多其他地方中。计算机可快速且高效地处理大量信息。被设计成在计算机系统上运行的软件应用允许用户执行包括商业应用、学校作业、娱乐等等在内的各种各样的功能。软件应用通常被设计成执行特定的任务，诸如用于草拟文档的文字处理器应用或者用于发送、接收和组织电子邮件的电子邮件程序。

[0003] 在某些情况下，软件应用被设计成实现各种形式的元数据。该元数据可由不同的元数据类型来表示。基于用户期望看到的哪个元数据类型，不同类型可被用于不同的用户。在这些情形中，多个不同的元数据文件通常被存储并用于提供各种元数据类型。这一系列元数据文件的存储和实现可能在检索和应用正确的元数据文件时导致低效率。

发明内容

[0004] 本文所述的实施例涉及动态适配元数据供本地(native)数据编码使用，以及高效地修改对象模型类型引用。在一个实施例中，计算机系统在对象模型描述上例示(instantiate)元数据读取器，来访问该对象模型描述中的元数据的各个部分。元数据读取器被配置成读取本地元数据，其中本地元数据是以元数据读取器预期的编码表示的元数据。元数据读取器确定所访问的元数据是以非本地编码被编码，并然后确定要执行哪些元数据修改来将非本地编码变换成本地编码。计算机系统然后根据所确定的修改将对象模型的元数据从非本地编码动态地适配到本地编码。如此，对象模型可由本地运行时读取。

[0005] 在另一实施例中，计算机系统例示一类型引用表，该类型引用表被配置成接收类型定义请求以及作为响应来提供合适的类型定义。该类型引用表包括可由元数据读取器读取的元数据。所例示的类型引用表接收来自类型请求者对指定类型的类型定义请求。计算机系统在类型引用表中实现类型引用，以基于哪个类型曾被请求以及哪个类型适合于元数据读取器的当前消费者来动态地替换元数据的各种类型。计算机系统然后基于被动态替换的类型来向类型请求者提供所请求的类型定义。

[0006] 提供本发明内容以便以简化形式介绍将在以下详细描述中进一步描述的一些概念。本发明内容并非旨在标识所要求保护的主题的关键特征或必要特征，也不旨在用于帮助确定所要求保护的主题的范围。

[0007] 附加的特征和优点将在以下的描述中被阐述，并且部分地可通过该描述而对本领域技术人员显而易见，或者可通过对本文中的教示的实践来习得。本发明的实施例的特征和优点可以通过在所附权利要求中特别指出的手段和组合来被实现并获得。本发明的实施例的特征将从以下描述和所附权利要求书中变得完全显而易见，或者可通过如下所述对本发明的实践而习得。

附图说明

[0008] 为了进一步阐明本发明的各实施例的以上和其他优点和特征，将参考附图来呈现本发明的各实施例的更具体的描述。可以理解，这些附图只描绘本发明的典型实施例，因此将不被认为是对其范围的限制。本发明的实施例将通过使用附图用附加特征和细节来描述和解释，附图中：

[0009] 图 1 示出本发明的实施例可在其中操作的计算机架构，本发明的实施例包括动态地适配元数据供本地数据编码使用以及高效地修改对象模型类型引用。

[0010] 图 2 示出用于动态地适配元数据供本地数据编码使用的示例方法的流程图。

[0011] 图 3 示出用于高效地修改对象模型类型引用的示例方法的流程图。

[0012] 图 4 示出外部元数据被转换成本地元数据的实施例。

[0013] 图 5 示出使用类型引用来编码元数据的实施例。

[0014] 图 6 示出通过对类型引用的原地修改将外部类型系统转换成本地类型系统的实施例。

[0015] 图 7 示出元数据容器的实施例。

[0016] 图 8 示出提供元数据容器的视图的实施例。

[0017] 图 9 示出提供元数据容器的视图的替代实施例。

具体实施方式

[0018] 本文所述的实施例涉及动态适配元数据供本地数据编码使用，以及高效地修改对象模型类型引用。在一个实施例中，计算机系统在对象模型描述上例示元数据读取器，来访问该对象模型描述中的元数据的各个部分。元数据读取器被配置成读取本地元数据，其中本地元数据是以元数据读取器预期的编码表示的元数据。元数据读取器确定所访问的元数据是以非本地编码被编码，并然后确定要执行哪些元数据修改来将非本地编码变换成本地编码。计算机系统然后根据所确定的修改将对象模型的元数据从非本地编码动态地适配到本地编码。如此，对象模型可由本地运行时读取。

[0019] 在另一实施例中，计算机系统例示一类型引用表，该类型引用表被配置成接收类型定义请求以及作为响应来提供合适的类型定义。该类型引用表包括可由元数据读取器读取的元数据。所例示的类型引用表接收来自类型请求者对指定类型的类型定义请求。计算机系统在类型引用表中实现类型引用，以基于哪个类型曾被请求以及哪个类型适合于元数据读取器的当前消费者来动态地替换元数据的各种类型。计算机系统然后基于被动态替换的类型来向类型请求者提供所请求的类型定义。

[0020] 以下讨论现涉及可以执行的多种方法以及方法动作。应当注意，虽然这些方法动作可能是按一定次序讨论的，或者是在流程图中被描绘为是按照特定顺序进行的，然而并非必然需要特定的次序，除非特别声明，或者是在一个动作被执行之前因为该动作依赖于另一动作的完成而需要的情况。

[0021] 本发明的各实施例可包括或利用专用或通用计算机，该专用或通用计算机包括诸如例如一个或多个处理器和系统存储器等计算机硬件，如以下更详细讨论的。本发明范围内的各实施例还包括用于承载或存储计算机可执行指令和 / 或数据结构的物理和其他计

算机可读介质。这样的计算机可读介质可以是可由通用或专用计算机系统访问的任何可用介质。以数据形式存储有计算机可执行指令的计算机可读介质是计算机存储介质。承载计算机可执行指令的计算机可读介质是传输介质。由此，作为示例而非限制，本发明的各实施例可包括至少两种显著不同的计算机可读介质：计算机存储介质和传输介质。

[0022] 计算机存储介质（设备）包括 RAM、ROM、EEPROM、CD-ROM、基于 RAM 的固态驱动器（SSD）、闪存、相变存储器（PCM）、或其它类型的存储器、或者其他光盘存储、磁盘存储或其他磁存储设备、或可用于以计算机可执行指令、数据或数据结构形式存储所期望的程序代码装置且可被通用或专用计算机访问的任何其他介质。

[0023] “网络”被定义成允许在计算机系统和 / 或模块和 / 或其他电子设备之间传输电子数据的一个或多个数据链路和 / 或数据交换机。当信息通过网络（硬连线、无线、或者硬连线或无线的组合）被传输或提供给计算机时，该计算机将该连接适当地视为传输介质。传输介质可以包括如下的网络：所述网络可以用于以计算机可执行指令形式或以数据结构形式运送数据或所期望的程序代码装置，并且可以被通用或专用计算机访问。上述的组合也应被包括在计算机可读介质的范围内。

[0024] 此外，在到达各种计算机系统组件之后，计算机可执行指令或数据结构形式的程序代码装置可从传输介质自动传输到计算机存储介质（或反之亦然）。例如，通过网络或数据链路接收到的计算机可执行指令或数据结构可被缓冲在网络接口模块（例如，网络接口卡或“NIC”）内的 RAM 中，然后最终被传输给计算机系统 RAM 和 / 或计算机系统处的较不易失性的计算机存储介质。因而，应当理解，计算机存储介质可被包括在还利用（或甚至主要利用）传输介质的计算机系统组件中。

[0025] 计算机可执行（或计算机可解释）指令例如包括致使通用计算机、专用计算机、或专用处理设备执行某个功能或某组功能的指令。计算机可执行指令可以是例如二进制代码、诸如汇编语言之类的中间格式指令、或甚至源代码。尽管用结构特征和 / 或方法动作专用的语言描述了本主题，但可以理解，所附权利要求书中定义的主题不必限于上述特征或动作。相反，上述特征和动作是作为实现权利要求的示例形式而公开的。

[0026] 本领域的技术人员将理解，本发明可以在具有许多类型的计算机系统配置的网络计算环境中实践，这些计算机系统配置包括个人计算机、台式计算机、膝上型计算机、消息处理器、手持式设备、多处理器系统、基于微处理器的或可编程消费电子设备、网络 PC、小型计算机、大型计算机、移动电话、PDA、寻呼机、路由器、交换机等等。本发明还可在其中通过网络链接（或者通过硬连线数据链路、无线数据链路，或者通过硬连线和无线数据链路的组合）的本地和远程计算机系统每个都执行任务的分布式系统环境（例如，云计算、云服务等）中实践。在分布式系统环境中，程序模块可位于本地和远程存储器存储设备中。

[0027] 图 1 示出了可在其中采用本发明的原理的计算机架构 100。计算机架构 100 包括计算机系统 101。计算机系统 101 可以是任何类型的本地或分布式计算机系统，包括云计算系统。计算机系统可包括用于执行各种不同任务的各种不同模块。计算机系统使用各种不同接口来允许用户与系统的交互。在某些实施例中，计算机系统可提供允许开发者开发软件程序的集成开发环境（IDE）107。从而，在某些情况下，用户 105 可以是开发者，并可发出输入 106 来与 IDE 交互。IDE 可提供对各种不同编程构造和其他元素的访问。在某些情况中，这可包括元数据读取器 115。

[0028] 本文所述的某些实施例提供了单个元数据文件，该元数据文件向不同元数据消费者呈现单个类型的不同视图。无需强制元数据读取器 115 在元数据被读取时在进行中做出这些改变，就可呈现该不同视图。当在对象模型描述 116 上例示元数据读取器时，读取器可检测元数据的编码并确定可能需要什么修改使得元数据对于读取器的消费者来说看上去自然。读取器在适配器（如适配模块 125）中包装元数据，适配器修改元数据的编码使得元数据读取器的消费者无需修改即可在外部（即本文中的“非本地”）编码（如 117）上操作。实施例还描述了对元数据的编码以允许以高效方式进行转换。如本文所使用的，“本地”编码指的是采用元数据读取器所预期的格式的元数据编码。“非本地”或外部编码指的是不是元数据读取器所预期的任何元数据编码。

[0029] 在修改是显著的情况下，可对单个类型创建元数据的多个副本。这些副本中的每一个都包括区别特征，使得元数据读取器确定哪个视图要被呈现给哪个元数据消费者成为可能。一旦做出了确定，则所确定的视图就被提供给各消费者。元数据消费者可将标识他们的优选视图类型的选项传递给读取器。元数据读取器然后可使用它来选择呈现哪个视图类型。然后向该消费者隐藏其余视图。

[0030] 本文所述的某些实施例包括以下方面 :1）元数据读取器对将外部对象模型编码适配成本地对象模型编码所需的修改的自动检测，2）修改对象模型所引用的类型以改为来自不同对象模型的类型，3）对元数据编码以允许两个不同对象模型的类型系统之间的高效转换，4）修改元数据编码使得它符合元数据读取器的消费者所预期的元数据编码标准，5）使得元数据消费者能够无需修改而透明地消费变化的元数据编码，以及 6）包含单个实体的多个视图供不同消费者直接使用的单个元数据文件。这些视图对于每个消费者来说是完整的且一致的，且仅那些对于每个消费者需要不同视图的实体具有多个视图（即这并不简单地是包含元数据文件的三个副本的 zip 文件）。元数据消费者可因此容易地指定他们希望哪个类型的视图。元数据读取器还可能够基于消费者的选择来选择完整且一致的元数据视图。

[0031] 当在对象模型描述 116 上例示元数据适配器 125 时，该适配器正被期望该对象模型采用特定的本地编码的客户端消费。如果正被读取的对象模型不采用该本地编码，则元数据读取器可检测 120 需要哪些修改 121 来将外部编码变换成本地编码 126。然后将例示一适配器来在客户端读取元数据时修改元数据，以便允许客户端以各种外部编码 117 透明地读取元数据。其他元数据读取器仅允许以一个特定的格式来读取，而不是允许在两个格式之间动态地转换。

[0032] 由于元数据读取器 115 的客户端使用该读取器来理解对象模型，元数据适配器可透明地对非本地元数据做出许多类的修改，以便允许本地客户端直接理解它。这包括将外部类型系统所引用的类型修改成本地类型系统所使用的类型，以及将元数据标志修改为本地系统中的等同标志。

[0033] 对象模型描述 116 可被编码来以最小的运行时处理时间和最小的空间要求来提供高效转换。本文所包括的实施例描述了这一编码，它允许外部类型系统被原地转换成元数据读取器的本地类型系统。在该编码中，对对象模型中正在使用的类型的引用是通过类型引用进行的。在某些情况中，其他元数据编码可基于权标和定义的相对位置来利用类型引用权标和类型定义权标的组合（例如，如果它们都在同一元数据文件中，则选择类型定义权标，而如果它们在不同的元数据文件中，则选择类型引用权标）。在本文的实施例中，即使

类型定义权标可用，由于该权标在与定义相同的元数据文件中，类型引用权标被使用。使用类型引用权标而不是类型定义权标允许本文的系统高效地重定向类型，而如果它们是通过类型定义权标而不是类型引用权标来引用的，则代价高。

[0034] 当使用类型引用来编码非本地对象模型使用的类型时，适配器 125 通过原地转换类型引用的目标来在类型系统之间转换，同时使得对该引用的其他使用在元数据中不被修改。这允许将对外部类型的每个使用重定向到本地类型，而不要求对元数据表的完全扫描和更新。

[0035] 此外，元数据容器中所描述的大多数类型具有可在不同消费者之间共享的单个视图，或仅需要如上所述的最小适配。然而，也存在可能对于每个消费者使用不同视图的一类类型。其他实现通常具有整个元数据文件的多个分开的副本（每个视图一个完整副本），或者可能早就尝试根据消费者来在进行中适配或重写元数据。多副本方案增加了复杂性，因为它强制元数据部署者管理两个分开的文件并可能损害性能，因为存在对两个消费者具有共享表示的类型的不需要的副本。在进行中适配或重写仅对于非常简单的变换来说执行顺利，而任何显著的变换将显著减慢读取，且可能甚至要求在存储器或盘上可能临时生成元数据的完整的新副本。

[0036] 本文所述的实施例包括单个元数据容器，该元数据容器对视图应对所有消费者相同的类型保持元数据的单个定义。元数据容器还保持视图不同（或需要不同）的类型的多个副本。元数据读取器然后向每个消费者提供该容器的不同视图，且仅向该消费者示出具有这些类型的类型的通用视图，以及对其余类型的消费者专用视图。下面，分别针对图 2 和图 3 的方法 200 和 300，进一步解释这些概念。

[0037] 考虑到以上描述的系统和架构，参考图 2 和图 3 的流程图将更好地理解根据所公开的主题实现的方法。为了解释简明起见，这些方法被示出和描述为一系列框。然而，应该理解和了解，所要求保护的主题不受框的次序的限制，因为一些框可按不同的次序进行和 / 或与此处所描绘和描述的其他框同时进行。此外，并非全部所示的框都是实现下面所述的方法所必需的。

[0038] 图 2 示出用于动态地适配元数据供本地数据编码使用的方法 200 的流程图。现在将频繁参考环境 100 的组件和数据来描述方法 200。

[0039] 方法 200 包括在对象模型描述上例示元数据读取器来访问该对象模型描述中的元数据的一个或多个部分的动作，该元数据读取器被配置成读取本地元数据，该本地元数据包括以该元数据读取器所预期的编码表示的元数据（动作 210）。例如，例示模块 110 可在对象模型描述 116 上例示元数据读取器 115。元数据读取器可被例示来访问对象模型描述中的元数据的各部分。元数据可以是本地类型（126）或外部、非本地类型（117）。元数据读取器被配置成读取以本地格式编码的元数据，但可能不能够读取以各种不同的非本地格式中的任何一种编码的元数据。元数据读取器从而确定所访问的元数据是以非本地编码被编码（动作 220），并进一步确定要执行哪些元数据修改来将非本地编码变换成本地编码（动作 230）。

[0040] 修改可包括将非本地编码引用的类型修改（或重编码）成本地编码使用的类型。修改还可包括将非本地编码的元数据标志修改成本地编码的元数据标志。如此，在修改之后，元数据读取器将能够识别元数据标志。还可单独地或除上述修改之外执行其他修改。在某

些情况中，数据类型可被重写以符合本地编码。这些数据类型可被动态地重写以使得它们可由元数据读取器读取或以其他方式访问。

[0041] 方法 200 还包括根据所确定的修改将对象模型的元数据从非本地编码动态地适配到本地编码的动作，使得对象模型可由本地运行时读取(动作 240)。例如，适配模块 125 可使用由模块 120 确定的所确定的修改 121 来将非本地元数据 117 动态适配到本地编码 126。在应用了修改之后，对象模型 116 于是可由本地运行时读取。动态适配从而允许用户读取采用多种不同的非本地编码的元数据。不管元数据是以哪个非本地编码被编码的，所述的系统将使用一个或多个不同修改来把该元数据适配成可由元数据读取器读取的本地编码。

[0042] 在某些实施例中，动态适配可在元数据导入－读取级发生。通过在导入－读取级适配元数据，可防止在运行时的后续适配。在某些情况中，可应用可见性修改器来使得被动态适配的元数据内部或公共可见。该可见性修改器可由开发者 105 按需在元数据中改变。

[0043] 图 3 示出用于高效地修改对象模型类型引用的方法 300 的流程图。现在将频繁参考环境 100 的组件和数据来描述方法 300。

[0044] 方法 300 包括例示类型引用表的动作，该类型引用表被配置成接收类型定义请求以及使用可由元数据读取器读取的元数据中存储的一个或多个类型引用权标来提供合适的类型定义作为响应(动作 310)。例如，例示模块 110 可例示类型引用表 130，类型引用表 130 接收类型定义请求以及使用元数据 117 中存储的各种类型引用权标来提供类型定义 131 作为响应。该类型引用表 130 包括可由元数据读取器 115 读取的元数据。所例示的类型引用表接收来自类型请求者对指定类型的类型定义请求(动作 320)。此后，在类型引用表 130 的元数据中实现类型引用权标，以基于哪个类型曾被请求以及哪个类型可由元数据读取器读取来为元数据读取器 115 动态地替换一个或多个类型(动作 330)。因此，使用类型引用表按请求以及按需动态地替换类型。非本地类型引用被本地类型引用原地替换。这些动态替换向元数据读取器提供了非本地类型的标准化视图。

[0045] 方法 300 还包括基于由元数据中的类型引用权标提供的被动态替换的类型来向类型请求者提供所请求的类型定义的动作(动作 340)。从而，类型引用表 130 可基于动态替换的类型来向类型请求者提供类型定义 131。类型引用表可被例示来确保经类型引用接收类型请求者的请求。类型是在类型引用表中被替换的。以此方式，用户请求保持不变，而本地类型定义替代非本地定义被使用。

[0046] 元数据的本地和非本地编码都可被创建在同一元数据文件中。从而，该同一元数据文件可包括不止一个编码。在单个元数据文件具有多个编码的情况下，元数据读取器可确定哪个用户正在请求该元数据文件，以及基于该确定可将该元数据文件的合适的编码传递给用户。因此，如果用户请求了本地编码，则本地编码的本地视图将被发送给用户。以此方式，对于用户来说，编码是完整的且一致的。为了提供高效执行，仅那些被不同类型的用户访问的元数据文件被以本地和非本地编码两者编码。如上所述，动态适配可包括各种不同的修改方法，包括将非本地编码引用的类型修改成本地编码使用的类型，以及将非本地编码的元数据标志修改成本地编码的元数据标志。从而，各种类型的非本地元数据可被动态地适配来供本地数据编码使用。

[0047] 附加实施例在图 4-9 中描述。例如，在图 4 中，描绘了本地客户端、元数据读取器、

适配器、以及外部对象模型之间的交互。以外部编码(454)接收的元数据可被发送到元数据适配模块 453,来被适配成可由元数据读取器 452 读取的本地编码。本地元数据客户端 450(在 API 边界 451 的另一侧)可与元数据读取器交互(例如使用 API)来访问本地编码的元数据。

[0048] 图 5 示出在使用多个类型时使用类型引用的元数据编码。例如,类型消费者 510 可包括类型 1 (511A)和一个或多个其他类型(类型 n (511N))。这些类型可通过类型引用(506A)被链接到外部类型定义(如 502A)。从而,类型 1 (511A)可通过类型应用 506A 被链接到外部类型定义 1(502A)。类似地,其他类型(类型 n)可通过类型引用 506N 被链接到外部类型定义 502N。外部类型定义被存储在类型定义表 501 中,而类型引用被存储在类型引用表 505 中。

[0049] 图 6 示出通过对类型引用的原地修改外部类型系统到本地类型系统的转换。因此,尽管类型引用 506A 和 506N 先前指向外部类型定义 502A 和 502N(在图 5 中),但是类型引用可被修改来指向不同的、本地类型定义。从而,类型引用 506A 可被修改成指向本地类型定义 626A,而类型引用 506N 可被修改成指向本地类型定义 626N。如此,外部类型系统可被转换成本地类型系统。

[0050] 图 7 示出在元数据容器内,要求多个元数据视图的每个实体可具有单独的视图副本。这些副本然后引用用于不需要不同视图的实体的共享视图。因此,元数据容器 705 中的实体 A 可具有两个不同的视图 :第一视图 710A 和第二视图 710B。而且,尽管实体 A 具有两个视图,实体 B 和 C 各自具有它们自己的通用元数据视图(分别是 711 和 712)。图 8 进一步示出元数据读取器可自动向第一元数据消费者 701A 给出与其所希望的视图一致的元数据容器 705 中实体的视图。因此,因为消费者 701A 优选元数据视图 710A,所以元数据读取器 702 用视图 710A 示出实体 A,以及用通用视图示出实体 B 和 C (711 和 712)。图 9 示出元数据读取器 702 自动向元数据消费者 701B 给出与其所希望的视图一致的元数据容器中实体的视图。因此,因为消费者 701B 优选元数据视图 710B,所以元数据读取器 702 用视图 710B 示出实体 A,以及用通用视图示出实体 B 和 C (711 和 712)。

[0051] 因此,提供了动态地适配元数据供本地数据编码使用的方法、系统和计算机程序产品。而且,提供了高效地修改对象模型类型引用的方法、系统和计算机程序产品。

[0052] 本发明可具体化为其它具体形式而不背离其精神或本质特征。所描述的实施例在所有方面都应被认为仅是说明性而非限制性的。因此,本发明的范围由所附权利要求书而非前述描述指示。落入权利要求书的等效方案的含义和范围内的所有改变被权利要求书的范围所涵盖。

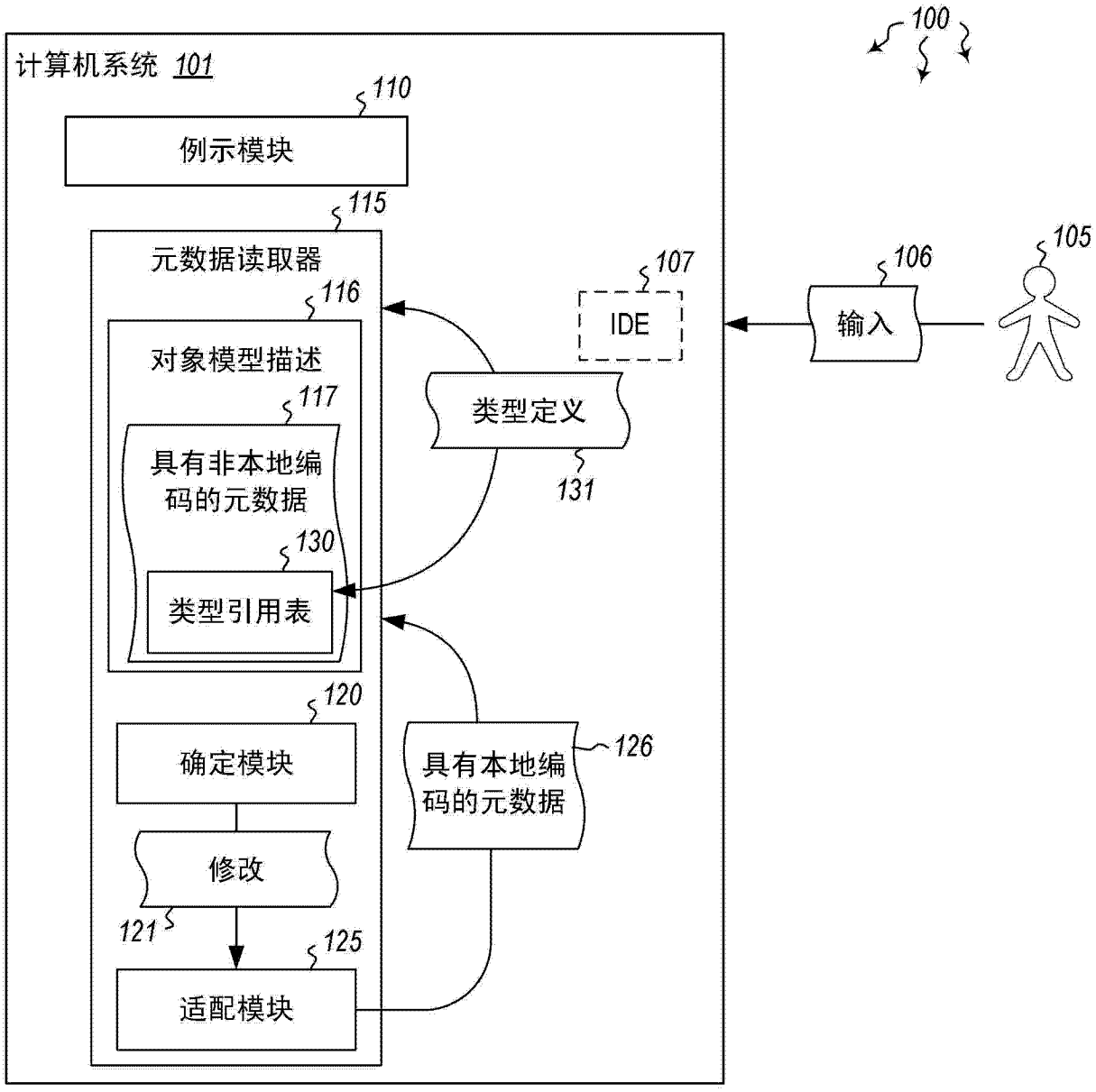

100
计算机系统 101
110
例示模块
115
元数据读取器
116
对象模型描述
117
具有非本地编码的元数据
130
类型引用表
120
确定模块
修改
121
125
适配模块
107
IDE
类型定义
131
106
输入
105
具有本地编码的元数据
126

图 1

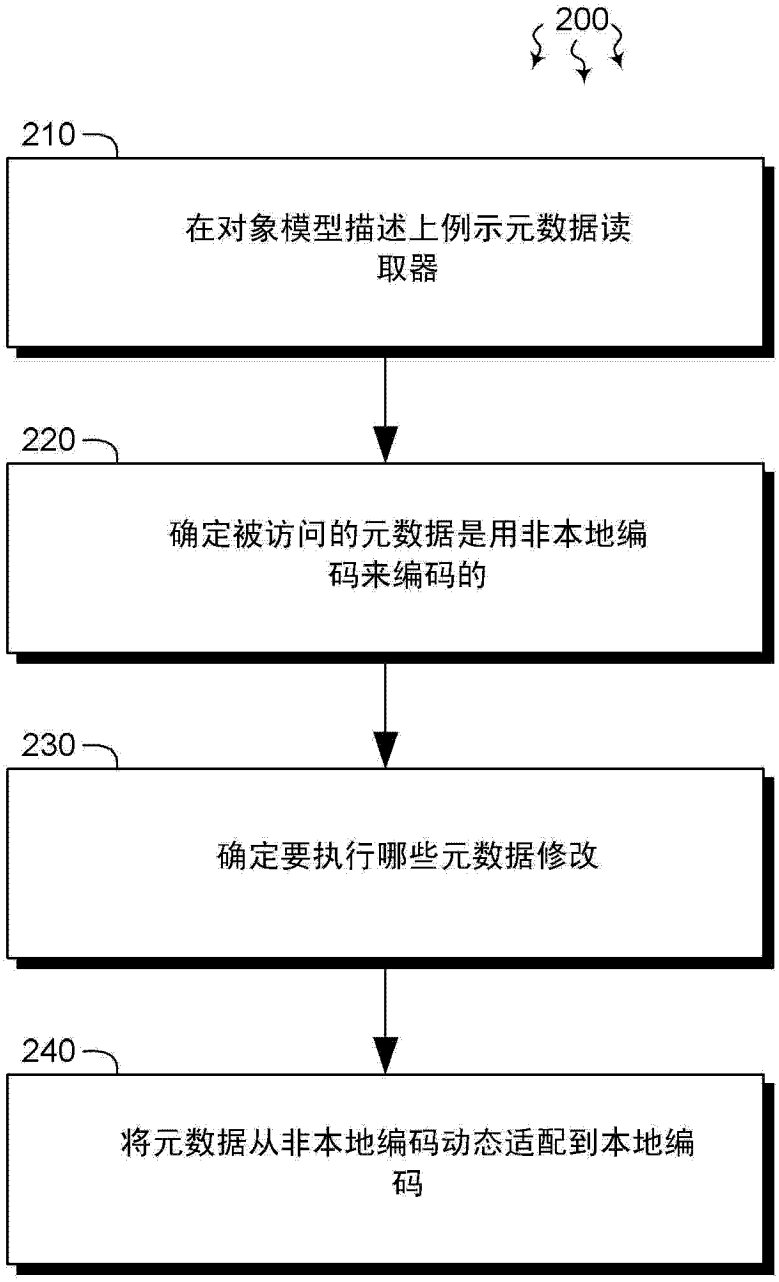

200
210
在对象模型描述上例示元数据读取器
220
确定被访问的元数据是用非本地编码来编码的
230
确定要执行哪些元数据修改
240
将元数据从非本地编码动态适配到本地编码

图 2

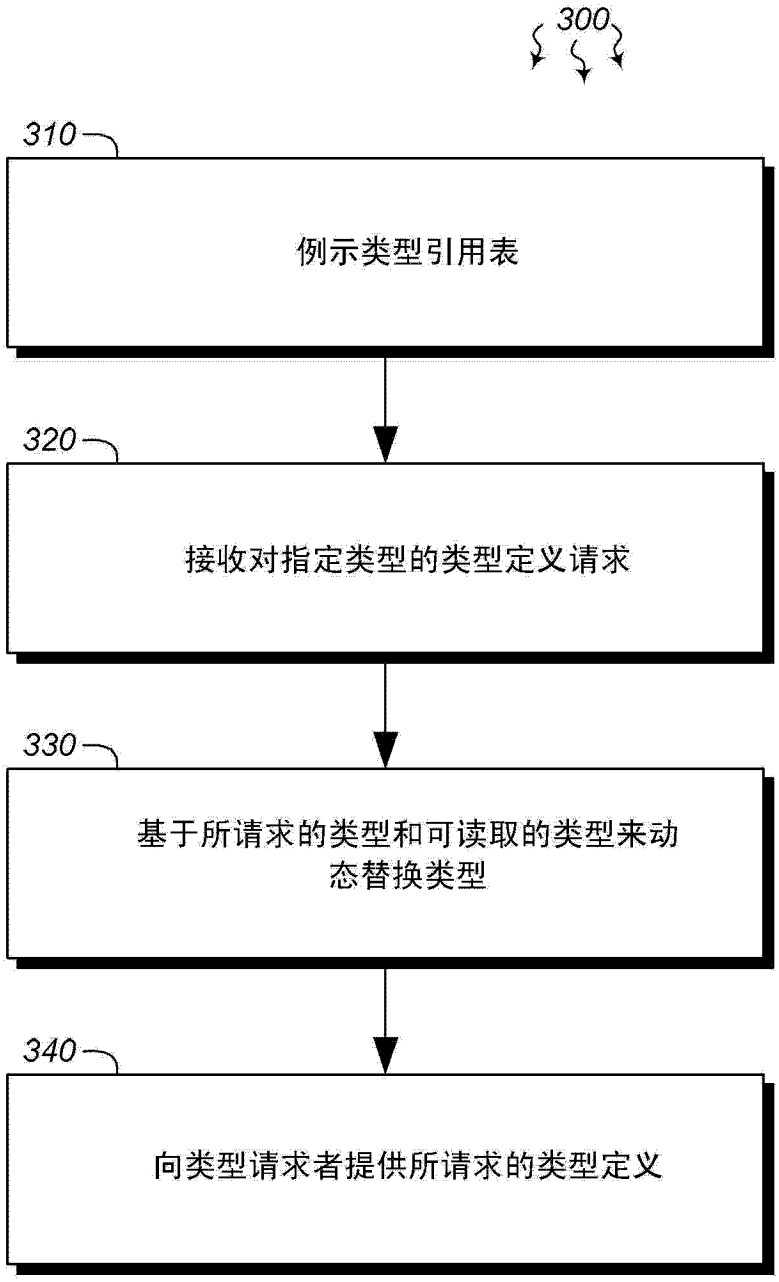

300
310
例示类型引用表
320
接收对指定类型的类型定义请求
330
基于所请求的类型和可读取的类型来动态替换类型
340
向类型请求者提供所请求的类型定义

图 3

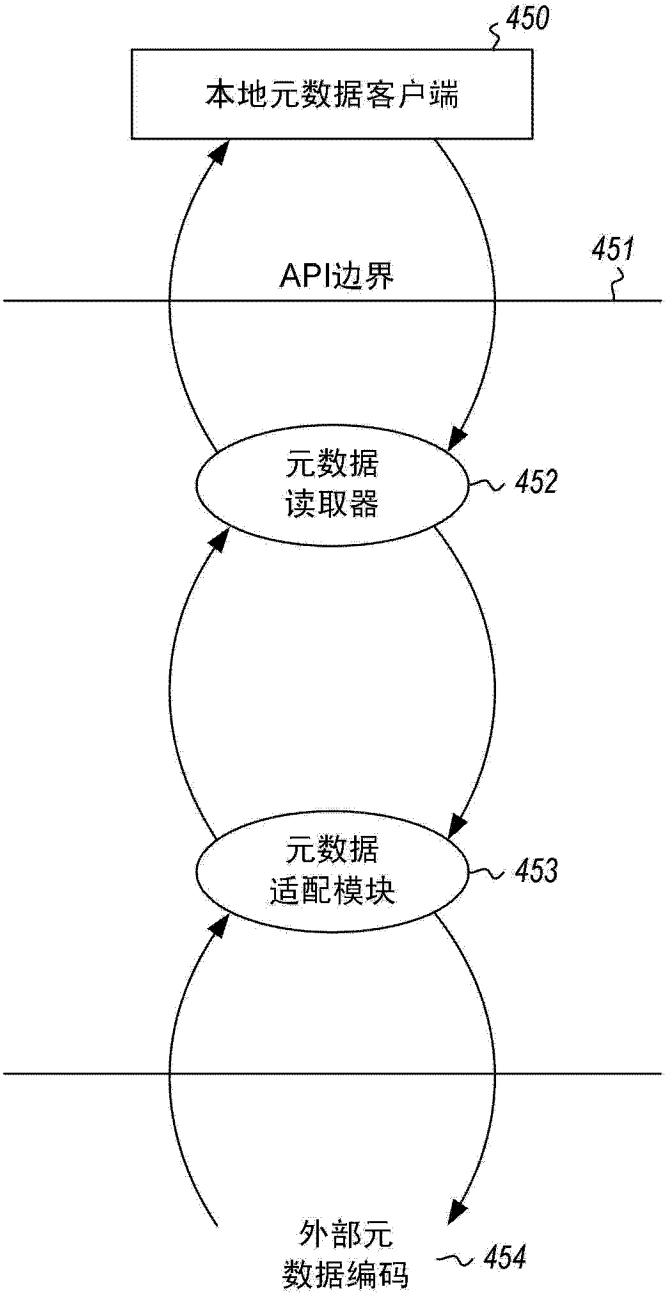

450
本地元数据客户端
451
API边界
元数据
读取器
452
元数据
适配模块
453
外部元
数据编码
454

图 4

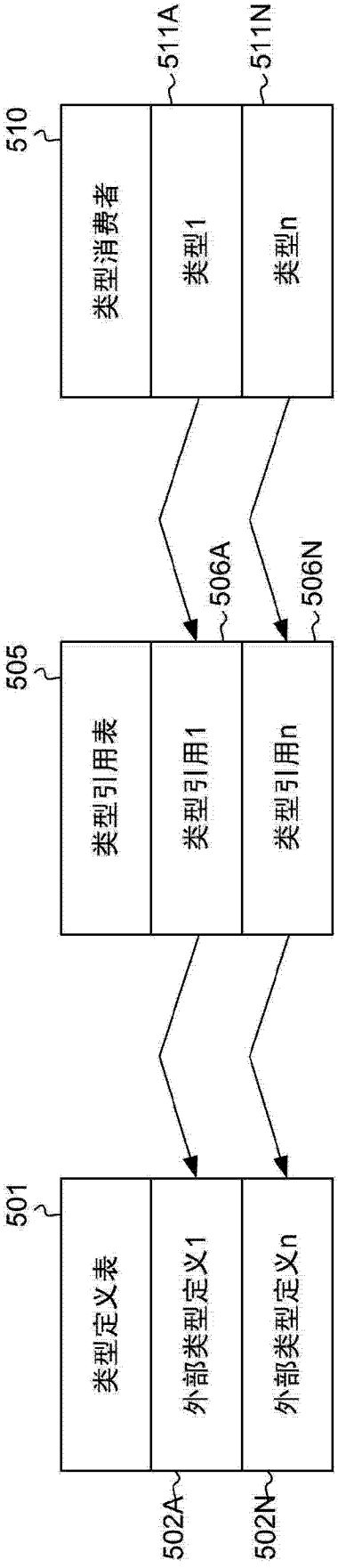

511A
511N
510
类型消费者
类型1
类型n
506A
506N
505
类型引用表
类型引用1
类型引用n
501
类型定义表
外部类型定义1
外部类型定义n
502A
502N

图 5

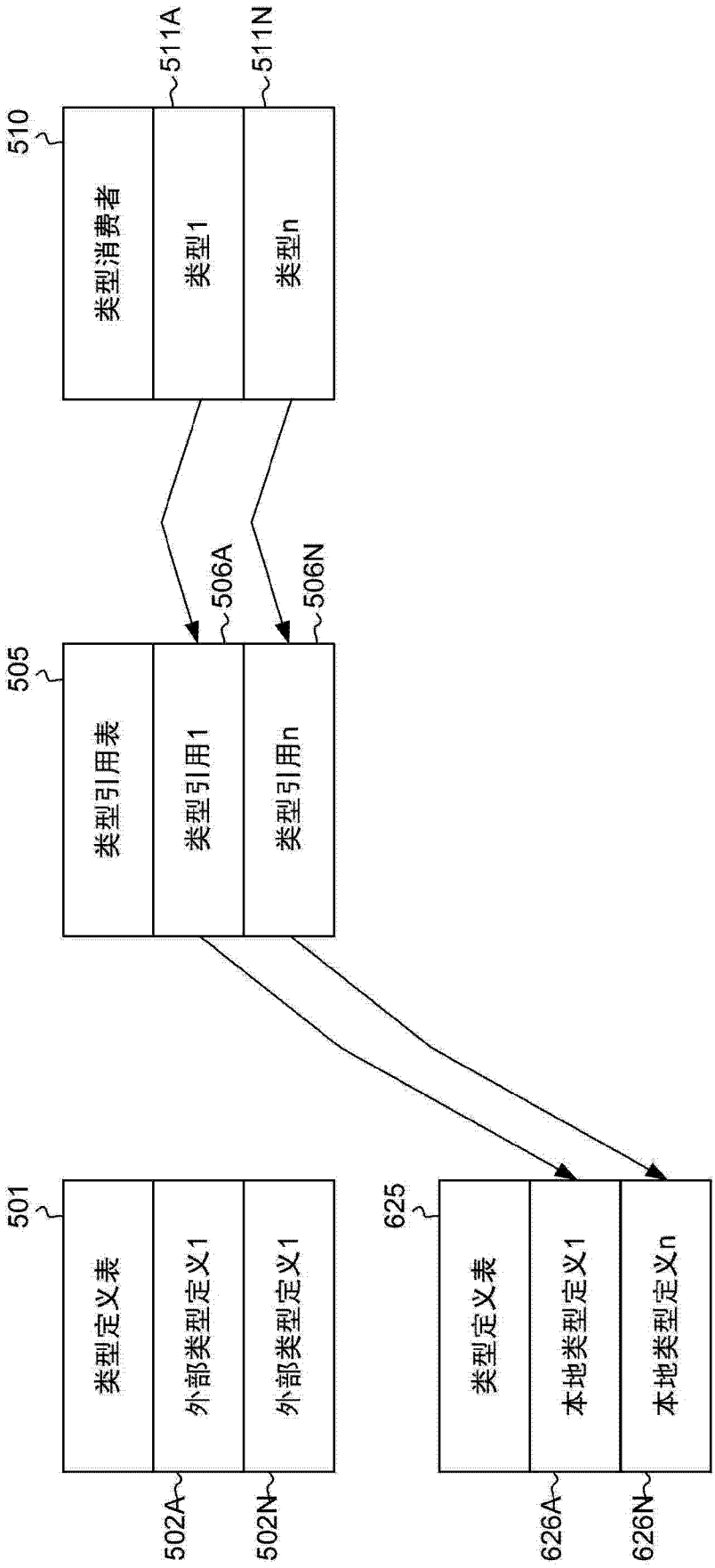

510
类型消费者
511A
类型1
511N
类型n
505
类型引用表
506A
类型引用1
506N
类型引用n
501
类型定义表
外部类型定义1
外部类型定义1
502A
502N
625
类型定义表
本地类型定义1
本地类型定义n
626A
626N

图 6

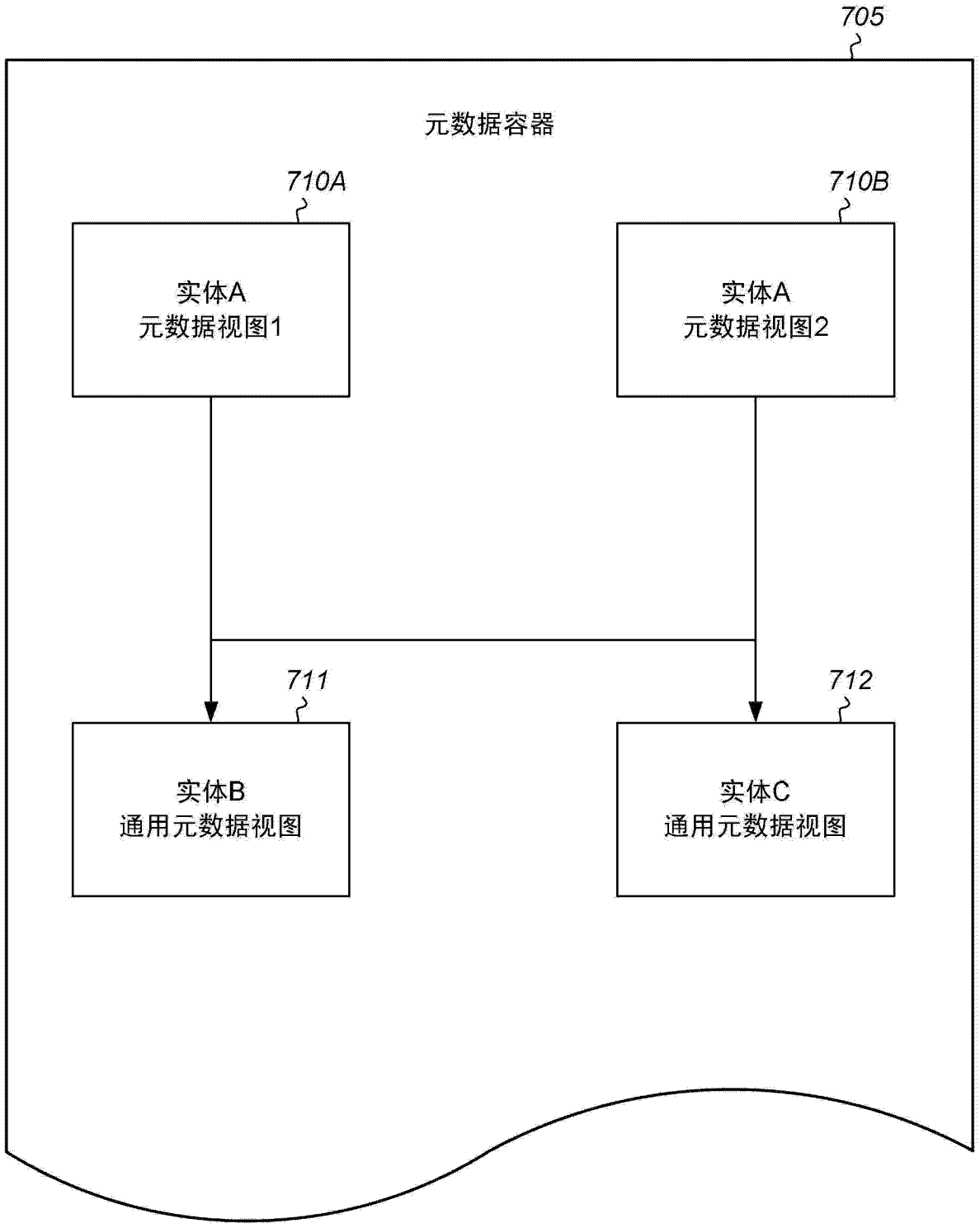

705
元数据容器
710A
实体A
元数据视图1
710B
实体A
元数据视图2
711
实体B
通用元数据视图
712
实体C
通用元数据视图

图 7

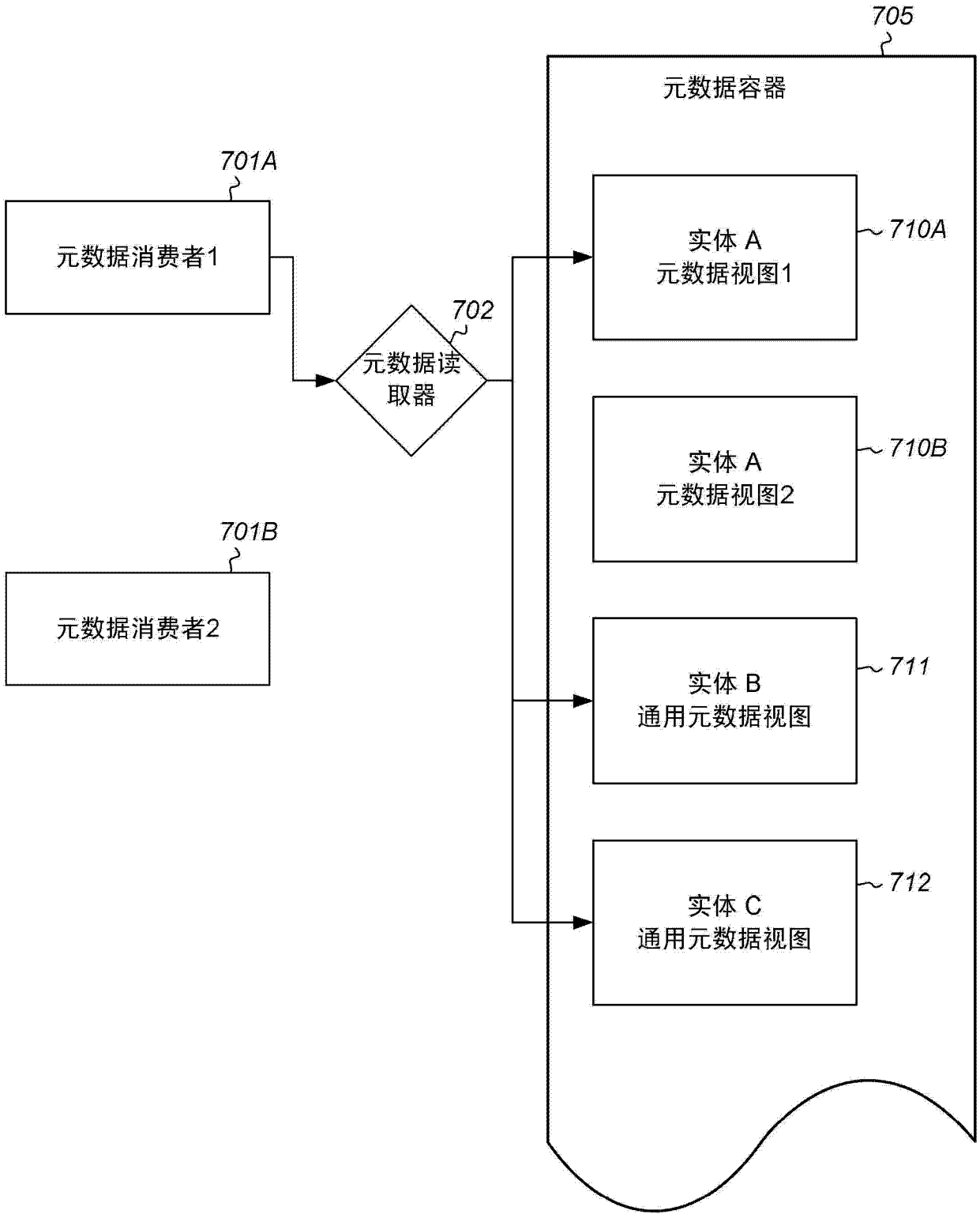

705
元数据容器
701A
元数据消费者1
702
元数据读取器
实体 A
元数据视图1
710A
实体 A
元数据视图2
710B
701B
元数据消费者2
实体 B
通用元数据视图
711
实体 C
通用元数据视图
712

图 8

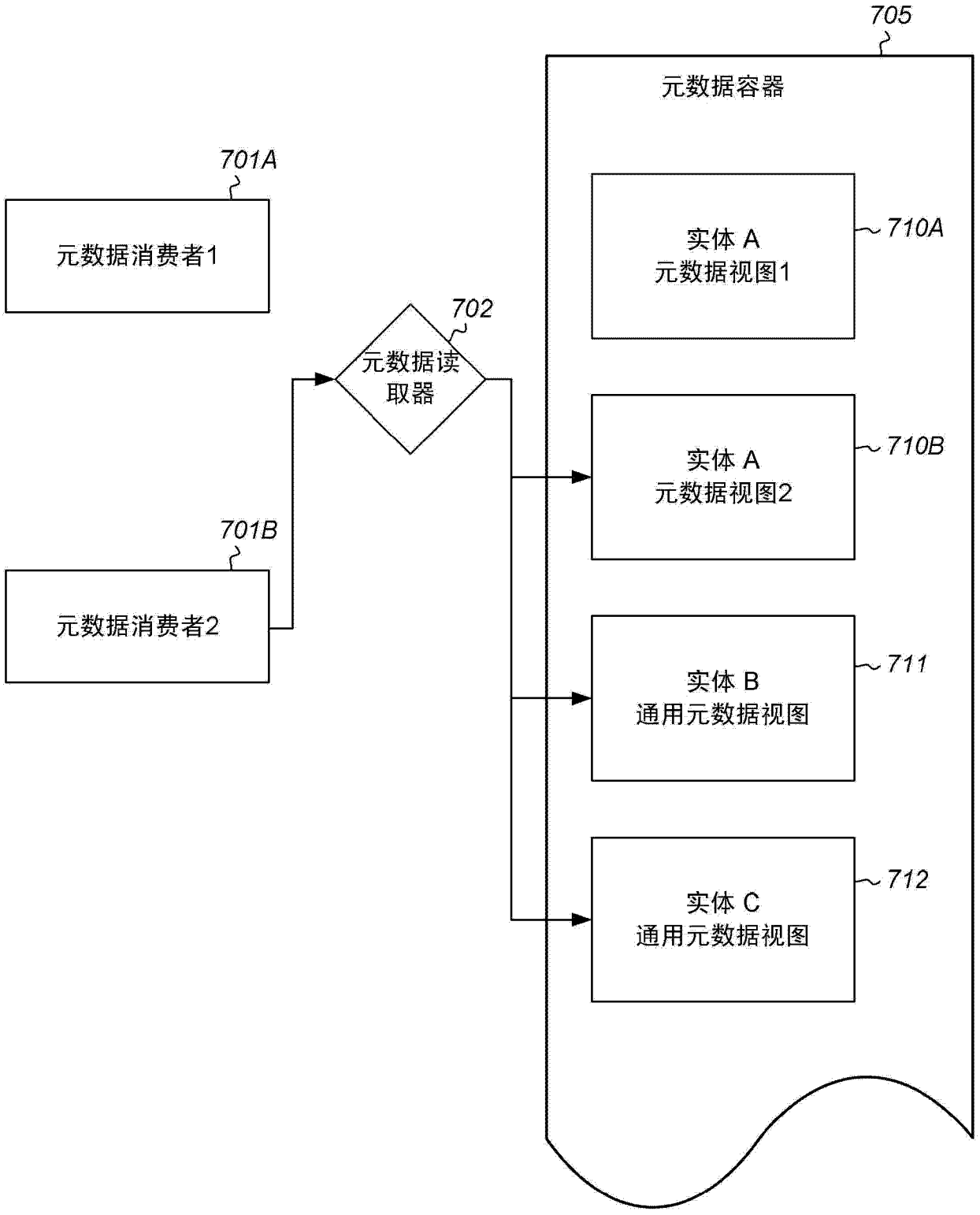

705
元数据容器
701A
元数据消费者1
702
元数据读取器
701B
元数据消费者2
实体 A
元数据视图1
710A
实体 A
元数据视图2
710B
实体 B
通用元数据视图
711
实体 C
通用元数据视图
712

图 9