[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 200580028882.1

[43] 公开日 2007 年 9 月 19 日

[11] 公开号 CN 101040280A

[22] 申请日 2005.8.31
[21] 申请号 200580028882.1
[30] 优先权
[32] 2004. 8.31 [33] US [31] 60/606,301
[86] 国际申请 PCT/US2005/030897 2005.8.31
[87] 国际公布 WO2006/026636 英 2006.3.9
[85] 进入国家阶段日期 2007.2.27
[71] 申请人 国际商业机器公司
地址 美国纽约
[72] 发明人 穆哈梅梅·伯兹亚内
布赖恩·德尔勒特
戴维·M.·康特
鲍里斯·克里洛夫
奥克奇欧·G.·奥斯尼
卡斯奥·D.·桑托斯
查尔斯·K.·尚克
马克·R.·萨克 张 宏
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 李德山

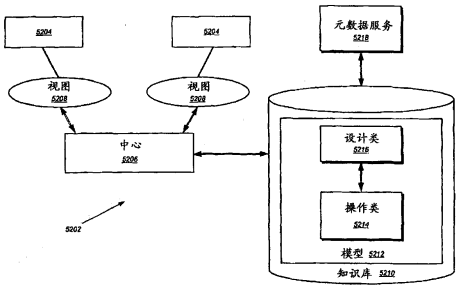
权利要求书 5 页 说明书 42 页 附图 26 页

[54] 发明名称

元数据管理

[57] 摘要

连同企业计算环境中的数据集成管理(5202)和使用元数据。集成的、依赖平台的元数据管理(5202)方法允许全企业访问数据集成服务(104)和基础数据，并且便于数据集成环境中工具和作用的重新使用和重新设计。为管理元数据提供了若干工具(5204)，包括维持版本化的元数据模型(5212)，在设计周期中可以分支排列与合并，并跨越全企业动态实施。依赖平台的方法便于变化的使用，包括在异类硬件和软件计算环境中的实施。



1. 一种方法，包括:

向知识库登记元数据模型;

将第一存储机制与所述元数据模型的一个或多个设计属性相关联；以及

将第二存储机制与所述元数据模型的一个或多个操作属性相关联，其中所述第二存储机制为所述元数据模型的所述一个或多个操作属性中的至少一个存储时间戳。

2. 根据权利要求 1 的方法，其中，所述第一存储机制是版本化的存储机制，它存储所述元数据模型的所述一个或多个设计属性中的至少一个的一个或多个版本。

3. 根据权利要求 1 的方法，进一步包括注解所述元数据模型的所述一个或多个设计属性和所述一个或多个操作属性，以便将它们与或者所述第一存储机制或者所述第二存储机制相关联。

4. 根据权利要求 1 的方法，进一步包括提供包结构，以在所述第一存储机制和所述第二存储机制之间分配所述元数据模型的所述一个或多个设计属性和所述一个或多个操作属性。

5. 根据权利要求 1 的方法，进一步包括提供与所述元数据模型相关联的清单，以在所述第一存储机制和所述第二存储机制之间分配所述元数据模型的所述一个或多个设计属性和所述一个或多个操作属性。

6. 根据权利要求 1 的方法，进一步包括将所述操作属性登记为第一模型以及将所述设计属性登记为第二模型。

7. 根据权利要求 1 的方法，其中，所述元数据模型可以跨越所述一个或多个操作属性和所述一个或多个设计属性查询。

8. 根据权利要求 1 的方法，进一步包括向所述元数据模型登记一个或多个映射，所述一个或多个映射描述所述元数据模型与一个或多个其他元数据模型的关系。

9. 一种系统，包括：

知识库，包含登记的元数据模型；

所述知识库内的第一存储机制，所述第一存储机制与所述元数据模型的一个或多个设计属性相关联；以及

所述知识库内的第二存储机制，所述第二存储机制与所述元数据模型的一个或多个操作属性相关联，所述第二存储机制用于为所述元数据模型的所述一个或多个操作属性中的至少一个存储时间戳。

10. 根据权利要求 9 的系统，其中，所述第一存储机制是版本化的存储机制，它存储所述元数据模型的所述一个或多个设计属性中的至少一个的一个或多个版本。

11. 根据权利要求 9 的系统，进一步包括若干注解，将所述元数据模型的所述一个或多个设计属性和所述元数据模型的所述一个或多个操作属性与或者所述第一存储机制或者所述第二存储机制相关联。

12. 根据权利要求 9 的系统，进一步包括包结构，以在所述第一存储机制和所述第二存储机制之间分配所述元数据模型的所述一个或多个设计属性和所述一个或多个操作属性。

13. 根据权利要求 9 的系统，进一步包括与所述元数据模型相关联的清单，以在所述第一存储机制和所述第二存储机制之间分配所述元数据模型的所述一个或多个设计属性和所述一个或多个操作属性。

14. 根据权利要求 9 的系统，其中，所述操作属性登记为第一模型，所述设计属性登记为第二模型。

15. 根据权利要求 9 的系统，其中，所述元数据模型可以跨越所述一个或多个操作属性和所述一个或多个设计属性查询。

16. 根据权利要求 9 的系统，进一步包括向所述元数据模型登记的一个或多个映射，所述一个或多个映射描述所述元数据模型与一个或多个其他元数据模型的关系。

17. 一种包括计算机可用介质的计算机程序产品，所述计算机可用介质包括计算机可读程序代码，其中在一台或多台计算机上执行所述计算机可读程序代码时，使所述一台或多台计算机执行下列操作：

向知识库登记元数据模型；

将第一存储机制与所述元数据模型的一个或多个设计属性相关联；以及

将第二存储机制与所述元数据模型的一个或多个操作属性相关联，其中所述第二存储机制为所述元数据模型的所述一个或多个操作属性中的至少一个存储时间戳。

18. 一种管理元数据的方法，包括：

将面向对象的元数据模型组织为包括操作属性的操作模型和包括设计属性的设计模型；

在操作知识库中存储所述操作模型；以及

在通用知识库中存储所述设计模型。

19. 根据权利要求 18 的方法，进一步包括对所述操作模型元数据的至少一项加注时间戳。

20. 根据权利要求 18 的方法，其中，所述通用知识库支持多于一种版本的所述设计模型。

21. 根据权利要求 18 的方法，进一步包括为用户与所述元数据模型的交互提供用户环境。

22. 根据权利要求 21 的方法，其中，所述用户环境包括编辑所述模型的工作空间。

23. 根据权利要求 22 的方法，其中，所述工作空间是用户独占的。

24. 根据权利要求 21 的方法，其中，所述工作空间支持元数据事例的版本化。

25. 根据权利要求 18 的方法，进一步包括动态地比较所述通用知识库中的所述设计模型的一个或多个不同版本。

26. 根据权利要求 18 的方法，其中，所述通用知识库支持所述设计模型版本的分支。

27. 根据权利要求 18 的方法，进一步包括协调所述设计模型的多个版本。

28. 根据权利要求 18 的方法，进一步包括通过面向消息的服务异步地调用所述元数据模型而在元数据服务中使用所述元数据模型。

29. 根据权利要求 18 的方法，进一步包括并发地执行使用所述元数据模型的服务。

30. 一种管理元数据的系统，包括:

面向对象的元数据模型，包括具有所述元数据模型的一个或多个操作属性的操作模型和具有所述元数据模型的一个或多个设计属性的设计模型;

存储所述操作模型的操作知识库; 以及

存储所述设计模型的通用知识库。

31. 一种方法，包括:

以第一模型固有的术语表示查询;

使用描述所述第一模型与第二模型之间一个或多个关系的映射信息，将所述查询转换为所述第二模型固有的术语; 以及

将所述查询转换为固有的数据源格式。

32. 根据权利要求 31 的方法，其中，所述映射信息可以被查询。

33. 根据权利要求 31 的方法，其中，所述第一模型是视图，所述第二模型是中心。

34. 根据权利要求 31 的方法，其中，所述方法在企业计算系统中执行。

35. 根据权利要求 31 的方法，其中，所述方法在数据集成系统中执行。

36. 一种包括计算机可用介质计算机程序产品，所述计算机可用介质包括计算机可读程序代码，其中在一台或多台计算机上执行所述计算机可读程序代码时，使所述一台或多台计算机执行下列操作:

登记第一模型;

识别第二模型和所述第一模型的至少一个属性到所述第二模型的映射; 以及

保持所述第一模型的所述至少一个属性到所述第二模型的所述

映射。

37. 根据权利要求 36 的计算机程序产品，进一步包括:

识别所述第一模型的未映射到所述第二模型的至少一个其他属性；以及

保持所述第一模型的所述至少一个其他属性。

38. 根据权利要求 36 的计算机程序产品，其中，所述第一模型包括多个类。

39. 根据权利要求 36 的计算机程序产品，其中，所述第二模型包括多个类。

40. 根据权利要求 36 的计算机程序产品，其中，在一台或多台计算机上执行所述计算机可读程序代码时，使所述一台或多台计算机提供存储机制，用于保持作为反身存储机制的所述第一模型的所述至少一个属性到所述第二模型的所述映射。

41. 根据权利要求 36 的计算机程序产品，其中，在一台或多台计算机上执行所述计算机可读程序代码时，使所述一台或多台计算机定义模式，用于表示关系数据库中的元数据模型，并使用所述模式保持所述第一模型的所述至少一个属性到所述第二模型的所述映射。

42. 根据权利要求 41 的计算机程序产品，其中，在一台或多台计算机上执行所述计算机可读程序代码时，使所述一台或多台计算机通过改变所述模式而修改所述第一模型。

43. 根据权利要求 41 的计算机程序产品，其中，在一台或多台计算机上执行所述计算机可读程序代码时，使所述一台或多台计算机通过改变所述关系数据库中的一个或多个属性而修改所述第一模型。

44. 根据权利要求 36 的计算机程序产品，其中，在一台或多台计算机上执行所述计算机可读程序代码时，使所述一台或多台计算机通过改变所述映射而修改所述第一模型。

45. 根据权利要求 36 的计算机程序产品，其中，所述第一模型和所述第二模型是元数据模型。

元数据管理

相关申请的交叉引用

本申请要求 2004 年 8 月 31 日提交的标题为“Metadata Management” 编号为 60/606,301 的美国临时申请的优先权。

技术领域

本发明涉及信息技术领域，更确切地说，涉及数据集成系统领域。

背景技术

计算机应用程序的出现使得许多商业过程迅速得多，高效得多；不过，使用不同数据结构、通讯协议、语言和平台的不同计算机应用程序的激增已经导致典型商业企业的信息技术基础设施极其复杂。典型企业内的不同商业过程可以使用完全不同的计算机应用程序，每种计算机应用程序的开发和优化都是针对具体的商业过程，而不是针对整个企业。例如，商家可能有某个具体的计算机应用程序用于跟踪应付帐款，完全不同的一个应用程序用于记录与客户的交往。事实上，即使同一商业过程也可能使用不止一个计算机应用程序，比如企业保留集中式的客户交往数据库，而员工们也保留他们自己的交往信息，比如在个人信息管理器中。

尽管专用计算机应用程序提供了定制解决方案的优点，但是此类激增导致效率低下，比如遍及企业的多次重复登录和操作相同的数据，或者企业未能捕捉与一个过程相关联的数据，而企业执行另一个过程时本来能够受益于该数据。例如，假若应付帐款过程与供应链和定购过程分离，企业就可能从客户接受和填写定单，而其信用历史本来可以使企业拒绝该定单。许多其他实例表明，企业会从对其横跨多种计算机应用程序的全部数据的一致访问中受益。

许多公司已经意识到并涉及了对跨越商业企业中不同应用程序的数据共享的需要。因此，企业应用程序集成即 EAI，已经显现为涉及全异源数据的基于消息的策略。随着计算机应用程序复杂度和数量的增加，EAI 努力遭遇了许多挑战，其范围从需要应对不同协议到需要涉及日益增大的数据容量和事务处理数目以及日益增加的更快捷数据集成的欲望。对 EAI 已经采取了多种方法，包括最小公分母方法、原子方法以及桥式方法。不过，EAI 以各个应用程序之间通讯为基础。作为明显的缺点，随着平台和应用程序的线性增加，EAI 解决方案的复杂度呈几何增长。

尽管数据集成系统针对企业需求提供了有用的工具，不过典型情况下，这样的系统部署为定制的解决方案。它们的开发周期长，并可能需要精心的技术培训，以容纳商业结构和信息需求中的变化。仍然存在着对数据集成系统工具的需要，以容许在变化的商业环境中使用、重用和修改功能。更确切地说，仍然存在着对更加灵活的元数据管理工具的需要，以便用于企业计算环境中的数据集成。

发明内容

本文提供的方法和系统用于管理和使用与企业计算环境中数据集成有关的元数据。集成的、不依赖平台的元数据管理方法可以容许在全企业范围内访问数据集成服务和基础数据，并便利数据集成环境中的工具和作业的再使用和再设计。提供的工具用于管理元数据，包括维护版本化的若干元数据模型，在设计周期中可以将它们分支和合并，以及跨越全企业动态地实施。不依赖平台的方法可以便利各种各样的用途，包括在多机种的硬件和软件计算环境中的实施方案。

一方面，本文介绍的方法包括：以第一模型固有的术语表示查询；使用描述所述第一模型与第二模型之间一个或多个关系的映射信息，将所述查询转换为所述第二模型固有的术语；以及将所述查询转换为固有的数据源格式。另一方面，系统包括以第一模型固有的术语表示查询的装置；使用描述所述第一模型与第二模型之间一个或多个关系

的映射信息，将所述查询转换为所述第二模型固有的术语的映射模型；以及将所述查询转换为固有的数据源格式的装置，在所述格式中对所述数据源执行所述查询。

所述映射信息可以被查询。所述映射信息可能在转换步骤期间可用。所述第一模型可以是视图。所述第二模型可以是中心。数据源可以是数据库。所述数据库可以为一个或多个数据源存储元数据。所述数据库可以存储表示企业元数据的持久模型。所述数据库可以是关系数据库和/或文件。所述方法可以在企业计算系统中执行，所述系统也可以在企业计算系统之内。所述方法可以在数据集成系统中执行，所述系统也可以在数据集成系统之内。所述第一模型固有的术语可以包括外部客户固有的语法。所述第一模型可以是用户界面的视图。所述方法可以进一步包括在所述用户界面中显示查询的结果，所述系统也可以包括用于显示查询结果的用户界面。所述第一模型可以是用于服务的视图。所述服务可以包括数据集成系统服务。所述服务可以包括远程工具和/或实时集成服务。所述第一模型和所述第二模型至少其一可以是知识库中存储的元数据模型。所述方法可以进一步包括以转换工具将所述查询的结果转换为所述第一模型，所述系统也可以包括对应的转换工具。所述转换工具可以存储在知识库中。

另一方面，本文介绍的方法可以包括：向知识库登记元数据模型；将第一存储机制与元数据模型的一个或多个设计属性相关联；以及将第二存储机制与元数据模型的一个或多个操作属性相关联，其中第二存储机制为所述元数据模型的所述一个或多个操作属性中的至少一个存储时间戳。

在所述方法中，所述第一存储机制可以是版本化的存储机制，它存储所述元数据模型的所述一个或多个设计属性中的至少一个的一个或多个版本。所述方法可以进一步包括注解元数据模型的一个或多个设计属性和一个或多个操作属性，将它们与或者第一存储机制或者第二存储机制相关联。所述方法可以进一步包括提供包结构，以定位第一存储机制和第二存储机制之间元数据模型的一个或多个设计属性和

一个或多个操作属性。所述方法可以进一步包括提供与所述元数据模型相关联的清单，以定位第一存储机制和第二存储机制之间元数据模型的一个或多个设计属性和一个或多个操作属性。所述方法可以进一步包括将操作属性登记为第一模型和将设计属性登记为第二模型。所述元数据模型可以跨越一个或多个操作属性或一个或多个设计属性查询。所述方法可以进一步包括向所述元数据模型登记一个或多个映射，所述一个或多个映射描述所述元数据模型与一个或多个其他元数据模型的关系。

在另一方面，系统可以包括：包含登记的元数据模型的知识库；所述知识库内的第一存储机制，所述第一存储机制与元数据模型的一个或多个设计属性相关联；以及所述知识库内的第二存储机制，所述第二存储机制与元数据模型的一个或多个操作属性相关联，第二存储机制用于为所述元数据模型的所述一个或多个操作属性中的至少一个存储时间戳。

所述第一存储机制可以是版本化的存储机制，它存储所述元数据模型的所述一个或多个设计属性中的至少一个的一个或多个版本。所述系统可以包括注解，将元数据模型的一个或多个设计属性和元数据模型的一个或多个操作属性与或者第一存储机制或者第二存储机制相关联。所述系统可以包括包结构，以定位第一存储机制和第二存储机制之间元数据模型的一个或多个设计属性和一个或多个操作属性。所述系统可以包括与所述元数据模型相关联的清单，以定位第一存储机制和第二存储机制之间元数据模型的一个或多个设计属性和一个或多个操作属性。操作属性可以登记为第一模型，设计属性登记为第二模型。所述元数据模型可以跨越一个或多个操作属性或一个或多个设计属性查询。所述系统可以进一步包括向元数据模型登记的一个或多个映射，所述一个或多个映射描述所述元数据模型与一个或多个其他元数据模型的关系。

在另一方面，保持模型的方法包括：登记第一模型；识别第二模型和所述第一模型的至少一个属性到所述第二模型的映射；以及保持

第一模型的所述至少一个属性到所述第二模型的映射。

所述方法可以包括识别第一模型的未映射到第二模型的至少一个其他属性；以及保持第一模型的所述至少一个其他属性。第一模型可以包括多个类，第二模型也可以包括多个类。所述方法可以包括提供存储机制，用于保持作为反身存储机制的第一模型的所述至少一个属性到第二模型的映射。所述方法可以进一步包括定义模式，用于表示关系数据库中的元数据模型，并使用所述模式保持第一模型的所述至少一个属性到第二模型的映射。所述方法可以进一步包括通过改变所述模式、改变关系数据库中的一个或多个属性以及/或者改变所述映射而修改第一模型。第一模型和第二模型可以是元数据模型。

在另一方面，作为保持模型的系统可以包括：第一模型的至少一个属性到第二模型的映射;以及登记所述第一模型的知识库，所述知识库被配置为保持第一模型的所述至少一个属性到第二模型的映射。

第一模型的至少一个其他属性未被映射到第二模型，所述知识库被配置为保持第一模型的所述至少一个其他属性。第一模型和/或第二模型每个都可以包括多个类。所述系统可以进一步包括存储机制，保持第一模型的所述至少一个属性到第二模型的映射，所述存储机制包括反身存储机制。所述系统可以进一步包括表示关系数据库中元数据模型的模式，所述模式保持第一模型的所述至少一个属性到第二模型的映射。通过改变所述模式、改变关系数据库中的一个或多个属性以及/或者改变所述映射而修改第一模型。第一模型和第二模型可以是元数据模型。

在另一方面，模型驱动的元数据转换架构可以包括：多个转换引擎，使用一个或多个模型到模型的映射在一个或多个模型之间进行转换；以及转换登记处，用于动态地选择所述多个转换引擎之一。

转换引擎可以包括编译语言引擎、解释语言引擎或解释映射引擎中的一个或多个。模型到模型的映射可以在星型架构中的中心与一个或多个视图之间。一个或多个模型到模型的映射可以是用户可配置的。在已经部署了若干对应模型之后可以配置模型到模型的映射之一。在

中心与多个相同视图之间进行转换的多个转换引擎中可以重复模型到模型的映射之一。在中心与多个不同视图之间进行转换的多个转换引擎中可以实现不同的模型到模型的映射。

在另一方面，在模型之间转换元数据的方法包括：接收在第一模型与第二模型之间转换元数据的请求；检索模型到模型的映射，其特征在于在所述第一模型与所述第二模型之间转换；以及使用所述模型到模型的映射将元数据从所述第一模型转换到所述第二模型。

模型到模型的映射可以包括用于由转换引擎进行转换的编译语言、解释语言或映射的一个或多个。模型到模型的映射可以在星型架构中的中心与视图之间。所述方法可以进一步包括提供用户界面，用于配置模型到模型的映射。所述方法可以进一步包括在登记处存储模型到模型的映射以便动态访问。所述方法可以进一步包括在已经部署了第一模型和第二模型至少其一之后配置模型到模型的映射。在中心与多个相同视图之间进行转换的多个转换引擎可以并发地使用模型到模型的映射。所述方法可以进一步包括登记多个不同的模型到模型的映射，其中在中心与多个不同视图之间进行转换的多个转换引擎并发地使用不同的模型到模型的映射。

在一方面，本文公开的管理元数据的方法包括：将面向对象的元数据模型组织为包括操作属性的操作模型和包括设计属性的设计模型；在操作知识库中存储操作模型；以及在通用知识库中存储设计模型。

所述方法可以进一步包括对操作模型元数据的至少一项加注时间戳。通用知识库可以支持设计模型的不止一种版本。所述方法可以进一步包括为用户与模型的交互提供元数据环境。用户环境可以包括编辑模型的工作空间。工作空间可以是用户独占的和/或共享的。元数据环境可以包括组空间。组空间可以支持元数据事例的版本化。元数据环境可以在用户计算机上本地驻留，也可以驻留在用户计算机可访问的远程服务器上。所述方法可以包括动态地比较通用知识库中设计模型的一个或多个不同版本。通用知识库可以支持设计模型版本的分

支。所述方法可以包括协调设计模型的多个版本，以及/或者动态地协调设计模型的多个版本。所述方法可以包括通过面向消息的服务异步地调用元数据模型而在元数据服务中使用元数据模型，以及/或者通过应用程序接口同步地调用元数据模型而在元数据服务中使用元数据模型。所述方法可以包括并发地执行使用元数据模型的服务，以及/或者使用并行机制执行使用该模型的服务。

本文介绍的管理元数据的系统可以包括：面向对象的元数据模型，包括具有一个或多个元数据模型操作属性的操作模型和具有一个或多个元数据模型设计属性的设计模型；存储操作模型的操作知识库；以及存储设计模型的通用知识库。

对操作模型元数据的至少一项可以加注时间戳。通用知识库可以支持设计模型的不止一种版本。本系统可以包括用户与模型交互的元数据环境。用户环境可以包括编辑模型的工作空间。工作空间可以是用户独占的或共享的。元数据环境可以包括组空间。组空间可以支持元数据事例的版本化。元数据环境可以在用户计算机上本地驻留，也可以驻留在远程服务器上。通用知识库可以支持动态比较设计模型的一个或多个不同版本。通用知识库可以支持设计模型版本的分支。通用知识库可以支持协调设计模型的多个版本。通用知识库可以支持动态地协调设计模型的多个版本。所述系统可以包括通过面向消息的服务异步地调用元数据模型而使用元数据模型的元数据服务，以及/或者通过应用程序接口同步地调用元数据模型而使用元数据模型的元数据服务。所述元数据模型可以用于以并发或并行中至少一种方式执行的服务。

本文公开的协调元数据的方法可以包括：将协调区属性与元数据对象相关联，协调区属性识别以公共协调规则集为特征的协调区；根据所述公共协调规则集协调多个元数据对象的事例，以提供协调区内的元数据对象协调事例。

所述方法可以包括定义第二协调区，用于协调元数据对象协调事例与元数据对象的一个或多个附加事例。协调区可以包括多个元数据

对象的事例。所述方法可以进一步包括将匹配类型与协调区属性相关联，匹配类型定义了对元数据对象事例的处理。所述方法可以进一步包括将标识与元数据对象的事例相关联，该标识唯一地识别在协调区内元数据对象的事例。所述方法可以进一步包括为元数据对象提供协调谱系。该协调谱系可以描述通过一个或多个协调区的路径、识别一个或多个数据源、识别一个或多个协调规则以及/或者包括元数据对象事例的历史。

在另一方面，本文公开的协调元数据的系统可以包括：以公共协调规则集为特征的协调区；多个元数据对象的事例，包括将所述多个元数据对象事例的每一个都与所述协调区相关联的协调区属性；以及协调引擎，对于多个元数据对象事例与之相关联的协调区，根据公共协调规则集，通过协调所述多个元数据对象事例，在协调区内产生元数据对象的协调事例。

所述系统可以包括第二协调区，用于协调元数据对象协调事例与元数据对象的一个或多个附加事例。协调区可以包括多个元数据对象的事例。匹配类型可以定义对协调区内元数据对象事例的处理。与元数据对象的每个事例相关联的标识可以唯一地识别在协调区内元数据对象的事例。为元数据对象可以提供协调谱系。该协调谱系可以描述通过一个或多个协调区的路径、识别一个或多个数据源、识别一个或多个协调规则以及/或者包括元数据对象事例的历史。

在另一方面，为数据集成系统提供并发元数据服务的方法可以包括：将元数据服务分为对象流；识别一串对象，它们具有根据元数据对这些对象主要是内部的引用；在多个处理器的单一处理器上执行该串对象；识别该串对象之外的至少一个对象；以及在所述多处理器的另一处理器上执行所述至少一个对象。

这些对象可以包括至少一个元数据模型。这些处理器在物理上分开的硬件上。所述服务可以包括解决元数据冲突的协调过程。这些对象可以包括元数据导入。使用数据依赖性的图可以识别主要是内部的引用。所述服务可以组织为用于并发的管道。所述管道至少可以包括

识别对象阶段、取出候选者阶段、协调阶段、合并阶段和存储阶段。

在其他方面，计算机程序产品可以包括计算机可用介质，它包括计算机可读程序代码，其中在一台或多台计算机上执行所述计算机可读程序代码时使所述一台或多台计算机执行任何一个或多个以上方法。

本文所用的“国际商用机器”或“IBM”应当是指纽约 Armook 的国际商用机器公司。

本文所用的“数据源”或“数据目标”意在具有与这些术语一致的最广泛的可能含义，应当包括数据库、多个数据库、知识库信息管理器、队列、消息服务、知识库、数据设施、数据存储设施、数据提供商、网站、服务器、计算机、计算机存储设施、CD、DVD、移动存储设施、中心存储设施、硬盘、多路协作数据存储设施、RAM、ROM、闪存、存储卡、临时存储设施、永久存储设施、磁带、本机连接的计算设施、远程连接的计算设施、无线设施、有线设施、移动设施、中心设施、网络浏览器、客户、便携式电脑、个人数字助理(“PDA”)、电话、便携式电话、移动电话、信息平台、分析设施、处理设施、商务企业系统或处理数据所在的其他设施，或者为存储数据或其他信息而提供的其他设施，以及以上任何系统中所用的、保持结构化或非结构化数据的任何文件或文件类型，或者任何流式的、消息化的、事件驱动的或另外来源的数据，以及上述的任何组合，除非另外指明了特定含义或者短语语境有另外的需要。存储机制是任何物理的或逻辑的设备、资源或设施，有能力担任数据源或数据目标，或者以其他方式以可检索的形式存储数据。

“Enterprise Java Bean(EJB)”应当包括用于 J2EE 平台的服务器方组件架构。EJB 支持分布式、事务处理式、安全便携式 Java 应用程序的快速简便开发。EJB 支持允许并发消息消费的容器式架构，并且提供对分布式事务处理的支持，所以使用 J2EE 架构的数据库更新、信息处理以及到企业系统的连接能够在同一环境中进行。

“JMS”应当意味着 Java Message Service，它是基于 Java 的

J2EE 企业架构中的企业消息服务。“JCA”应当意味着以下更具体介绍的 J2EE 平台的 J2EE Connector Architecture。应当承认，尽管 EJB、JMS 和 JCA 是现代分布式事务处理环境中常用的软件工具，但是提供类似功能的任何平台、系统或架构都可以用于本文介绍的数据集成系统。

本文所用的“实时”应当包括商业交易或商务的近似持续时间阶段，并且应当包括在商业操作或商业处理期间发生的过程或服务，与脱机情况（比如晚间的批量处理操作）相反。根据商业处理的持续时间，实时可能包括几秒、不到一秒、几分、几个小时或甚至几天。

本文所用的“商业处理”、“商业逻辑”以及“商业交易”应当包括商家能够执行的任何方法、服务、操作、过程或事务处理，包括但不限于销售、营销、实施、库存管理、定价、产品设计、专业服务、财务服务、管理、金融、保险业、分析、订合同、信息技术服务、数据存储、数据挖掘、信息传递、选定货运路线、调度、通讯、投资、事务处理、提议、促销、广告、出价、工程、制造、供应链管理、人力资源管理、数据处理、数据集成、工作流管理、软件生产、硬件生产、新产品开发、研究、开发、策略功能、质量控制和保证、包装、后勤、客户关系管理、应付回扣和退货、客户支持、产品维护、电话营销、协会交流、投资者关系等等，不一而足。

本文所用的“面向服务的架构（SOA）”应当包括形成商业企业基础设施部分的服务。在 SOA 中，服务能够变为应用程序开发和部署的构件，容许应用程序的快速开发和避免冗余代码。每种服务都可以实施一组商业逻辑即能够由周围环境约束的商业规则，比如服务数据输入的来源或服务数据输出的目标。在以下说明中提供了 SOA 的多种事例。

本文使用的“元数据”应当包括对正在处理的数据产生上下文关系的数据、有关该数据的数据、关于相关信息上下文关系的信息、关于数据源的信息、关于数据位置的信息、关于数据意义的信息、关于数据寿命的信息、关于数据标题的信息、关于数据单位的信息、关于

数据字段的信息以及/或者关于该数据上下文关系有关的任何其他信息的信息。

本文使用的“WSDL”即“Web服务描述语言”包括XML格式，所描述的网络服务（往往是web服务）作为在一组端点上操作包含或者面向文档或者面向过程的信息的消息。所述操作和消息被抽象地描述，然后绑定到具体的网络协议和消息格式，以定义端点。有关的具体端点被组合为抽象端点（服务）。WSDL可扩充，因此无论使用何种消息格式或网络协议进行通讯，都容许描述端点和它们的消息。

附图说明

图1是示意图，商业企业具有多个商务过程，其中每一个都可能包括多个不同的计算机应用程序和数据源；

图2是示意图，显示了跨越商业企业的多个商务过程的数据集成；

图3是示意图，显示的架构用于为商业企业的多个数据源提供数据集成；

图4显示了元数据管理系统的架构；

图5显示了通过视图模型和数据模型查询数据库的通讯；

图6显示了为视图模型转换查询结果而正被访问的转换引擎；

图7显示了为外部服务转换查询结果而正被访问的转换引擎；

图8显示了静态模型的映射；

图9显示了可扩充模型的映射；

图10显示了模型映射的组合；

图11描绘的架构向外部元数据揭示多个内部服务；

图12描绘了映射模型驱动的元数据转换；

图13显示了与元数据环境的交互；

图14显示了存储多个元数据版本的通用知识库；

图15描绘了客户动态地比较版本化知识库中的若干元数据版本；

图16A显示了元数据协调的过程；

图16B描绘了跨越若干协调区的分段协调过程；

图 17 描绘了版本化元数据对象的协调;

图 18 显示了在元数据过程中使用并发的实例;

图 19 是从用户界面 6702 到元数据数据库 6712 的查询过程中涉及实体的图示;

图 20 显示了从元数据模型扩充元数据数据库过程中涉及的实体;

图 21 显示了从工具访问知识库过程中涉及的实体;

图 22 显示了工具访问版本化和非版本化元数据模型过程中涉及的实体;

图 23 显示了用户界面访问通用知识库中多版本元数据过程中涉及的实体;

图 24 显示了若干版本的元数据的协调过程中涉及的实体;

图 25 显示了在协调过程中使用并发涉及的实体。

具体实施方式

在以下全部讨论中，相同的要素编号意指相同的要素，除非另外明确地指出。

本文公开的发明能够采用的形式为完全硬件实施例、完全软件实施例或者既含有硬件单元又含有软件单元的实施例。在优选实施例中，本发明以软件实施，其中包括但是不限于固件、驻留软件、微代码等。

不仅如此，本发明也能够采用计算机程序产品的形式，从提供程序代码的计算机可用即计算机可读介质上可以访问，程序代码与计算机或任何指令执行系统有关或者由其使用。为了本说明书的目的，计算机可用即计算机可读介质可以是能够包含、存储、交流、传播或输送与指令执行系统、装置或设备有关或者由其使用的程序的任何装置。

该介质可以是电子的、磁性的、光学的、电磁性的、红外线的或半导体的系统（或装置或设备），也可以是传播介质。计算机可读介质的实例包括半导体或固态存储器、磁带、可拆卸计算机磁盘、随机存取存储器（RAM）、只读存储器（ROM）、刚性磁盘和光盘。光盘的流行实例包括小型盘—只读存储器（CD-ROM）、小型盘—读/

写（CD-R/W）和 DVD。

用于存储和/或执行程序代码的数据处理系统将包括通过系统总线直接地或间接地连接到若干存储器单元的至少一台处理器。存储器单元可以包括在实际执行程序代码期间所采用的本机存储器、海量存储器和高速缓冲存储器，后者提供至少存储某种程序代码的临时存储，以便减少在执行期间必须从海量存储器检索代码的次数。

输入/输出即 I/O 设备（包括但是不限于键盘、显示器、定点设备等）可以直接地连接到系统，也可以通过插入的 I/O 控制器连接到系统。

网络适配器也可以连接到系统，以使得数据处理系统能够变为通过插入的私有或公共网络连接到其他数据处理系统或者远程打印机或存储设备。调制解调器、电缆调制解调器和以太网卡仅仅是网络适配器的一些当前可用的类型。

图 1 表示了便于商业企业多种数据集成的平台 100。该平台包括多个商务过程，其中每一个都可以包括多个不同计算机应用程序和数据源。该平台可以包括几个数据源 102，它们可以是比如以上介绍的那些数据源。这些数据源可以包括来自天南海北的种类繁多的数据类型。例如，数据源包括的系统可以来自若干提供商，比如 Sybase、Microsoft、Informix、Oracle、Inlomover、EMC、Trillium、First Logic、Siebel、PeopleSoft、IBM、Apache 或 Netscape。数据源 102 包括的系统可以使用若干数据库产品或标准，比如 IMS、DB2、ADABAS、VSAM、MD 系列、UDB、XML、复合平面文件或 FTP 文件。数据源 102 可以包括若干应用程序创建或使用的文件，比如 Microsoft Outlook、Microsoft Word、Microsoft Excel、Microsoft Access，以及标准格式的文件，比如 ASCII、CSV、GIF、TIF、PNG、PDF 等。数据源 102 可以出自多个位置，它们也可以位于中心。从数据源 102 提供的数据可以以多种形式出现，并且具有彼此可以兼容或不兼容的不同格式。

数据目标在本说明书的后面讨论。一般来说，这些数据目标可以

是以上指出的任何数据源 102。典型情况下，这种名称差异表示数据系统在数据集成过程中是提供数据还是接收数据。不过应当承认，这种区别并不力图传达数据源和数据目标之间在能力上的任何差异（除非另外特别指出），因为在常规数据集成系统中，数据源可以接收数据，数据目标也可以提供数据。

图 1 中所展示的平台也包括数据集成系统 104。例如该据集成系统可以便于从数据源 102 收集数据，作为数据集成系统 104 接收查询或检索命令的结果。数据集成系统 104 可以向一个或多个数据源 102 发送命令，以使得数据源向数据集成系统 104 提供数据。由于收到的数据可以有多种格式，包括变化的元数据，数据集成系统可以重新配置收到的数据，以便后面在集成处理中能够联合。以下将更详细地介绍数据集成系统 104 可以执行的功能。

平台 100 也包括几个检索系统 108。检索系统 108 可以包括数据库或处理平台，用于进一步处理从数据集成系统 104 传递的数据。例如，数据集成系统 104 可以净化、联合、变换或以其他方式处理它从数据源 102 收到的数据，以使得检索系统 108 能够使用处理后的数据产生对商家有用的报告 110。报告 110 可以用于报告数据联合、回答复杂的查询、回答简单的查询，或者形成对商家和用户有用的其他报告，并且可以包括原始数据、表格、图表、图形，以及来自检索系统 108 的数据的任何其他表达。

平台 100 也可以包括数据库即数据库管理系统 112。数据库 112 可以用于暂时地、临时地或者永久地或长期地存储信息。例如，数据集成系统 104 可以从一个或多个数据源 102 收集数据，并且把数据变换为彼此兼容或彼此可兼容联合的形式。一旦数据被变换，数据集成系统 104 就可以在数据库 112 中以分解的形式、联合的形式或其他形式存储数据，以用于后来检索。

图 2 是示意图，显示了跨越商业企业的多个实体和商务过程的数据集成。在所展示的实施例中，数据集成系统 104 便利了用户界面系统 202 和数据源 102 之间的信息流。数据集成系统 104 可以从界面系

统 202 接收查询，其中查询需要提取并有可能变换在一个或多个数据源 102 中驻留的数据。界面系统 202 可以包括与数据集成系统 104 通讯的任何设备或程序，比如便携式或台式计算机上运行的网络浏览器、蜂窝网电话、个人数字助理（“PDA”）、联网的平台及其附着的设备，或者有可能与数据集成系统 104 接口的任何其他设备或系统。

例如，用户可以操作 PDA 通过 WiFi 或者 Wireless Access Protocol/Wireless Markup Language（“WAP/WML”）接口向数据集成系统 104 提出信息请求。数据集成系统 104 可以接收该请求并且产生任何所需要的查询，从网站或其他数据源 102 比如 FTP 文件地点访问信息。可以提取来自数据源 102 的数据并且将其转换为与提出请求的接口系统 202（在这个实例中是 PDA）兼容的格式，然后与该接口系统 202 通讯，使用户观察和操作。在另一个实施例中，所述数据可能先前已经从数据源提取了并存储在分开的数据库 112 中，它可能是由数据集成系统 104 所使用的数据仓库或其他数据设施。该数据可能已经以变换后状态或以其原始状态存储在数据库 112 中。例如，该数据可能已经以变换后状态存储，以使得来自许多数据源 102 的数据能够在另一个变换过程中联合。例如，来自 PDA 的查询可以传输到数据集成系统 104，数据集成系统 104 可以从数据库 112 中提取信息。提取后，数据集成系统 104 可以先将数据转换为 PDA 兼容的联合格式，再传输到 PDA。

图 3 是示意图，显示的架构用于为商业企业的多个数据源 102 提供数据集成。数据集成系统 104 的实施例可以包括发现数据阶段 302，有可能在其他过程之间执行从数据源提取数据，以及分析源数据的列值和表结构。发现数据阶段 302 也可产生有关目标数据的表结构、关系和关键字的建议。更完善的概述和审查功能可以包括数据范围确认、计算准确性、如果-则评估的准确性等。发现数据阶段 302 可以规格化数据，比如消除源数据中的冗余依存关系和其他异常。发现数据阶段 302 可以提供附加的功能，比如在数据源 102 之内挖掘例外以便进一步的分析，或者启动主机数据的直接概述。在 IBM 的 Websphere

ProfileStge 产品中可以找到发现数据阶段 302 的商业实施例的非限制性实例。

数据集成系统 104 也可以包括数据准备阶段 304，在此阶段对数据进行准备、标准化、匹配或其他处理，以产生后来将变换的优质数据。数据准备阶段 304 可以执行一般的数据质量函数，比如在数据内协调非一致性或检查正确的匹配（包括一对一匹配，一对多匹配和消除重复）。数据准备阶段 304 还可以提供特定的数据增强功能。例如，数据准备阶段 304 可以确保地址符合万国邮政标准以改进际间通信。数据准备阶段 304 可以使位置数据符合多国的地理编码标准，以便进行空间信息管理。数据准备阶段可以修改地址或添加内容以确保地址信息适合在政府批准的美国地址校正下的美国邮政服务邮费率折扣。类似的分析和数据修改可以提供给加拿大和澳大利亚的邮政系统，它为地址恰当的邮件提供折扣率。在 IBM 的 Websphere QualityStge 产品中可以找到数据准备阶段 304 的商业实施例的非限制性实例。

数据集成系统也可以包括数据变换阶段 308，以变换、丰富和传递变换后的数据。数据变换阶段 308 可以执行过渡服务，比如数据的重新组织和重新格式化，以及执行基于系统用户的商业规则和算法的计算。数据变换阶段 308 也可以把目标数据组织为子集，即数据市场或立方，在一定的分析环境中更好地调协数据处理。数据变换阶段 308 可以采用若干桥接器、译码器或其他接口（如以下一般讨论的），以复盖数据集成系统 104 所使用的多种数据源和数据目标的多种软件和硬件架构。数据变换阶段 308 可以包括图形用户界面、命令行界面或这些的某种组合，以跨越平台 100 设计数据集成作业。在 IBM 的 Websphere DataStge 产品中可以找到数据变换阶段 308 的商业实施例的非限制性实例。

使用并行执行系统 310 或者以串行或组合的方式，可以执行数据集成系统 104 的阶段 302、304、308，以最优化系统 104 的性能。

数据集成系统 104 也可以包括元数据管理系统 312，以管理与数据源 102 相关联的元数据。一般来说，元数据管理系统 312 可以跨越

数据集成环境中的全部工具提供元数据的互换、集成、管理和分析。例如，元数据管理系统 312 可以提供全异的数据源中数据的公共的、普遍可访问的视图，比如 IBM 的 Websphere ODBC MetaBroker、CA ERwin、IBM Websphere ProfileStage、IBM Websphere DataStage、IBM Websphere QualityStage、IBM DB2 Cube 视图和 Cognos Impromptu。元数据管理系统 312 也可以为数据谱系提供分析工具以及对数据结构变化的影响分析。元数据管理系统 312 可以进一步用于为数据集成系统 104 内的数据准备数据定义、算法和商业环境的商业数据词汇，其词汇可以发表以在整个企业中使用。在 IBM 的 Websphere MetaStge 产品中可以找到元数据管理系统 312 的商业实施例的非限制性实例。

除非上下文另有指定或需求，否则术语“映射”是指在视图、模型或模型事例之间使元数据和元-元数据相关的设计时间的活动，而“变换”是指对应的运行时间的活动。也应当注意，以下说明涉及元数据管理系统，其中原子数据项实际上是被模型化数据源的元数据。同样，元数据管理系统之内的元数据实际上是描述这种元数据的元数据，也称为元-元数据。进一步把元-元数据抽象为元-元-元数据也是可能的和适当的。为了避免混淆，以下命名法一般遵循数据、元数据、元-元数据层次，其中数据表示一个或多个数据源/目标的基础数据。不过应当承认，偶而元数据可以简单地指的是数据（由元数据管理系统所管理的数据），元-元数据往往对应地指的只是元数据，即从元数据管理系统内模型的观点来看的元数据。更一般地说，从上下文来看，用途应当清楚。不过，在用途模糊之处，应当以最广泛的可能意义进行解释。

图 4 显示了元数据管理系统 5202 的架构，例如它可以是以上介绍的任何元数据管理系统或元数据设施 312。元数据管理系统 5202 可以包括多个外部用户 5204，比如工具或客户，通过多个视图 5208、知识库 5210 与中心 5206 通讯，知识库 5210 包括至少一个模型 5212，它包括涉及模型 5212 的操作元数据的至少一个操作类 5214，和/或涉

及模型5216的设计元数据的至少一个设计类5216。为了与知识库5210中的模型5212的交互，可以提供元数据服务5218。

用户5204可以是以上介绍的接口系统202的任何一种，或者任何其他客户设备、工具或软件接口的其他程序，用户可以通过它们运行查询或以其他方式探查在数据库中的数据。用户5204可以使用视图5208运行查询，视图5208适应由用户5204所采用的数据模型与由中心5260所采用的数据模型之间的通讯。例如，视图5208可以包括字段、数据类型、数据分级结构、数据关系、时间信息、数据源信息或用户5202显示数据或使用数据的方式相关的任何其他信息，以及向外部用户5204提供的视图5208中的数据模型和由中心5260内部采用的数据模型之间任何适宜的映射。尽管图4仅仅展示了两幅视图5208，应当承认可以使用任何数量的视图5208，在相同类型的外部用户5204不止一个时，所述视图5208可以是相同的视图5208，而在外部用户5204不同时，视图5208可以是不同的视图5208，或者可以是符合元数据管理系统处理能力的这些视图5208的任何数量与组合。

也应当承认，外部用户5202可以使用用户5202独特的数据或元数据，在中心5260内没有对应部分。例如，Erwin设计工具使用Erwin独特的对象“坐标”，描述对象在图形“画布”中出现之处。中心5260可以设计为通过以对用户5202透明的方式，支持对中心模型的扩展而应付特殊的情况。作为选择，视图5208除了连接中心5260之外，也可以提供到适宜外部数据的直接映射，或者转而提供该直接映射。

中心5206通常可以使用由数据主题或其商业环境所定义的数据模型5212。因而通常期望数据的中心模型在单一应用程序内不会频繁变化。在对中心模型进行改变之处，可能需要对一幅或多幅视图5208进行对应的更新。中心5260可以使用在知识库5210内存储的一个或多个模型5212与基础数据（如针对企业数据的元数据）交互。虽然针对知识库5210的设计类5216使用中心是用途广泛的一种有用架构，但是应当承认，操作类5214典型情况下不会需要这样的中心5260。更一般地说，本文介绍的元数据管理系统5202可以设计为没有任何中

心 5260。这种架构可用于例如在多种视图的设计模型之间具有很少的或根本没有公共性之处。在这种情况下，针对元数据管理系统 5202 中多种视图之间的通讯可以采用其他技术，比如动态地产生非持久的逻辑中心作为中央连接器。无论在元数据管理系统 5202 中是否采用了中央中心 5260，本文介绍的系统的其他原理都是可适用的。

使用面向对象的技术可以在比如 Eclipse 和 Eclipse Modeling Framework（“EMF”）的平台中存储和处理模型 5212。模型 5212 可以包括元数据和到数据源和/或目标中相关结构的映射，以及任何其他有用的更抽象的元数据模型。模型的这些方面可以包含在持久存储在知识库 5210 之内的知识库对象中。

知识库 5210 可以存储一个或多个模型 5212，它们含有操作类 5214 和设计类 5216。模型可以包括元数据、元-元数据、或者数据的任何其他有用的描述或功能特征。例如，模型 5212 可以包含用于重量的数值，以比如盎司为单位。如果系统用户希望以英镑指定重量实现新的数据源或集成以英镑指定重量的现有数据源，可以把这种信息包括在模型 5212 中，以使得针对这些完全不同数据源的对应元数据能够在中心 5206 之内一致对待，并且通过可以对数据提供不同透视（或相同的透视）的一个或多个视图 5208 呈现给外部用户 5204。更一般地说，模型 5212 可以包含元数据管理系统期望的集成和任何其他用途有益的基础数据和元数据有关的任何信息。模型 5210 能够有效地捕捉有关数据以及它如何变化的信息，以在全企业或在企业之间实现数据用途的一致对待和可扩充。

在知识库 5210 中创建模型 5212 时，它可以被自动地划分到设计和操作组件中，虽然它们是一并和/或统一查询，但是可以独立管理。使用面向对象的技术，可以为模型 5212 存储操作类 5214，并且在若干类之间继承任何适合的属性、方法等。确切地说，操作类 5214 可以包含外部过程的模型操作方面，即提供运行结果的持久存储。操作类 5214 可以被标注时间戳，或以其他方式标注以便唯一引用。应当承认，虽然 Eclipse 平台是建立和维护本文介绍模型的一种有用工具，但是

任何面向对象工具或技术都可以同样地采用。在以下说明中，术语“属性”一般将是指面向对象描述或其他类似描述的多种特征，比如通用置标语言（“UML”）类模型的元素，包括类、子类、包、包结构、属性、属性、方法、关系、继承等。因此操作类、包结构等可以是操作属性，作为本文所用的术语。

设计类 5216 也可以从模型 5212 列示并继承所有的属性、方法等。这些设计类 5216 内的信息也可以包括版本信息，以使得可以或者顺序地或者以分支或者以其组合方式保留多个对象事例。用户根据企业计算系统的需求和设计目标可以操作、编辑、更新或以其他方式控制和管理这些版本化的元数据对象。使用版本控制或类似的技术，这些设计类 5216 的元数据对象可以共享或者向各个用户或团队输出。通常当尝试不同设计，或者当基础数据有变化时可以采用不同的版本。应当承认，在运行时可执行的文件创建之前，多种设计可以协调，分支合并。也应当承认，尽管如以上介绍 EMF 对知识库 5210 内的模型类可以是有用的平台，但是也可以采用任何类似的模型框架，比如对象管理集团公司的 Meta-Object Facility。

企业计算系统可以包括数据集成系统 104。企业计算系统也可以包括计算机、大型机、便携式设备、数据源和其他设备的任何组合，通过一个或多个局域网本地连接，以及/或者通过一个或多个广域网或公共网远程连接，例如使用因特网之上的虚拟私有网。企业计算系统内的设备可以互连为单一企业计算系统，以共享数据、资源、通讯和信息技术管理。典型情况下，企业计算系统内的资源由通用实体使用，比如商家、协会或者政府实体或大学。不过，在一定的商业模式下，企业计算系统的资源可以被许多不同的实体所拥有（或租赁）和使用，比如在应用程序服务提供商对远程执行应用程序提供请求式访问的情况下。企业计算系统也可以包括多种工具，它们通过各自的转换引擎（在基于桥接器的系统中可以是桥接器）访问公共的数据结构，本文称知识库信息管理器（“RIM”）（以下也指“中心”）。RIM 可以包括以上介绍的任何数据源 102。工具通常包括例如各种类型的数据

库管理系统和访问 RIM 中存储的共享数据的其他应用程序。这些工具、RIM 和转换引擎可以在单一计算机系统上处理和保留，也可以在例如由网络互连的许多计算机系统上处理和保留，该网络在不同的组成部分之间传递数据访问请求、转换的数据访问请求以及响应。

在这些工具执行的时候，可以产生数据访问请求以启动数据访问操作，即从 RIM 中检索数据或在 RIM 中存储数据。可以采用以下将介绍的原子数据模型和格式在 RIM 中存储数据。典型情况下，这些工具将观看以各种不同特征数据模型和格式在 RIM 中所存储的数据，如以下的介绍，而且每个转换引擎在收到数据访问请求后，将根据需要在各自工具的特征模型和格式与 RIM 的原子模型格式之间转换数据。例如，在从 RIM 中检索数据项的检索类型的访问操作期间，转换引擎将识别 RIM 中的一个或多个原子数据项，它们共同地包括响应该访问请求将被检索的数据项，并且转换引擎将使得 RIM 能够向转换引擎之一提供这些原子数据项。转换引擎又将把它从 RIM 收到的原子数据项汇总为该工具的特征模型和格式所需要的一个或多个数据项，即数据的“视图”，并且把汇总后的数据项提供给发布该访问请求的工具。在更新 RIM 中数据的数据存储期间，转换引擎可以以工具之一的特征模型和格式接收要存储的数据。转换引擎可以把该数据转换为用于 RIM 的原子模型和格式，并且把转换后的数据提供给 RIM 进行存储。如果数据存储访问请求使数据能够被更新，RIM 就可以以来自转换引擎的新供给的数据取代当前的数据。相反，如果数据存储访问请求提供了新数据，RIM 就可以以转换引擎所提供的原子格式向在 RIM 中的当前数据添加该数据。

元数据服务 5218 可以用于创建、编辑、删除或以其他方式操作知识库 5210 中的对象、类 5214、5216 和模型 5212，或者查询和探查其中包含的模型 5212 和任何其他数据。服务 5218 可以通过用户界面、命令行界面、程序界面或其他界面呈现给用户。服务 5218 可以提供若干功能，比如版本划分、分支排列、合并及知识库 5210 内支持的任何其他操作。以下更详细地介绍了这些操作的某一些。元数据服务 5218

也可以包括例如数据分析服务，比如影响分析（针对一种模型类型事例的改变如何影响模型中的其他类型事例）、操作分析（通过事件元数据反映的可执行对象的历史）、数据谱系（在数据仓库或跨越企业计算系统中数据移动的历史）、版本挖掘（探查元数据对象的版本历史）、对象差异（探查元数据对象之间的差异）以及对象合并（根据指定的规则组合相同类的两个对象）。元数据服务 5218 也可以包括为变换元数据输入和输出若干服务，例如当元数据进入知识库 5210 和/或出来时。元数据服务 5218 可以使用例如 J2EE 平台实现，并且通过面向服务的架构比如 SOA 提供给用户。同样，知识库 5210 内的事务处理可以使用例如 J2EE 应用服务器内的粒媒容器管理。应当承认，服务 5218 也可以作为用户界面中的一个或多个工具提供给终端用户。

应当注意，尽管以上介绍的功能（例如版本划分、分支排列、版本挖掘等）主要针对元数据对象即元数据不同事例之内和之间的细节，但是这种对元数据管理的一般方式可以容易地被抽象化，以针对元模型的管理，即元-元数据管理或管理元数据模型的模型。

因此，本文介绍了元模型化工具，它们为定义元模型之间的映射而提供、为元模型产生界面以及便利元数据模型的实现和变换。提供元模型化工具时可以通过提供图形用户界面，以提供对许多有关功能的访问。例如，界面提供的工具可以定义、确认、测试和分析元模型和映射，以及输出元数据模型。界面提供的工具也可以用于对元模型、元模型映射以及任何通过元模型化工具产生的元数据模型的事例进行文档编辑。元模型化工具有可能有效地用于例如部署新版本的企业模型。图表，比如 IBM Rational XDE 的服务可以支持图表化、模型化和映射。

元模型化工具可以部署为例如面向服务中的若干服务。元模型化工具可以为元数据模型提供中心化管理的映射规范，具有与以上讨论的元数据工具一致的同步、版本化、历史跟踪和其他适合的能力。因此，尽管映射模型可以表示中心与视图（或其他模型）之间的对象变换，但是来自这种元模型化透视的映射模型也可以或转而表示不同元

数据模型之间的映射，最终可以模型本身之间的变换中，比如升级到更新版本的元数据模型时。例如，元模型化工具可以提供与模型定义分离并松散耦合的独立的规范语言，以允许开发控制和实现的灵活性。元模型化工具可以有益地提供对开发环境内的映射规范的动态浏览，并且可以提供以多种详细级别自动产生文档的工具。利用集成套装的元模型化工具，可以开发对应的测试框架，以产生测试元数据并动态地执行映射，以使得能够获得即刻的结果并且加入到现行开发中。

为了维持模型的操作属性和设计属性之间概念上的分离，知识库 5210 可以逻辑地和/或物理地分成两个或多个知识库，比如用于持久存储设计类 5216 及属性的通用知识库（未显示）和用于持久存储操作类 5214 及属性的操作知识库（未显示）。因此，当知识库 5210 登记模型 5212 时，操作类 5214 可以持续在操作知识库中，而设计类 5216 可以持续在通用知识库中。使用类内的注解定义它们的关联，可以区分一个物理或逻辑知识库内的操作类和设计类。应当承认，其他技术也已公知，并且可以用于将模型的类分为操作方面和设计方面，或者提供逻辑上或物理上分开的操作和设计知识库。例如，公共/操作的分离可以隐式地设计在模型的类结构内，也可以使清单或其他列表或者编程设备伴随该模型，并且列出每个类或属性与其适宜知识库的关联。无论如何实现，这种方案都可以有益地允许针对模型 5212 的设计和操作要素的持续进行不同的处理。例如，设计类 5216 可以由若干团队的程序员开发和修改，因而需要稳健的版本区分能力和协调性。相反，操作类 5214 可能针对不同作业需要唯一标识，比如通过使用时间戳或其他唯一标识符。从而为每组类都可以定义适宜的服务，同时维持用户能够查询、变换或以其他方式操作的单一模型。

图 5 显示了通过一个或多个视图或模型与（元数据的）数据库的通讯。服务 5302、用户界面 5303 或任何其他界面可以与数据库 5312 通讯，它可以是以上介绍的任何数据源 102，以便向数据库 5312 提交查询。通讯可以通过元数据模型进行，比如知识库 5304 提供的视图 5308 和中心 5310，知识库 5304 是比如以上介绍的知识库 5210。这些

元数据模型可以包括关于数据的任何信息，比如字段、字段名、字段属性、数据类型、数据层次、数据关系、临时信息、数据源信息或者与结构、位置或数据使用有关的任何其他信息，或者关于这种数据的元数据（即元-元数据）。

服务 5302 可以产生的查询使用服务 5302 固有数据的视图，即具有服务 5302 所定义的结构和格式。这种查询可以由服务 5302 建立，无须关于数据库 5312 中数据结构的任何信息。知识库 5304 向发出请求的服务 5302 提供的视图 5308 可以被映射到中心 5310，它向多个不同的视图提供模型，包括接收该查询的视图 5308，用于元数据的一致表达。中心 5310 又可以被映射到数据库 5312 内部所使用的结构。通过利用视图 5308、中心 5310 和数据库 5312 之间的映射信息，该查询可以有益地转换为使用数据库 5312 固有的数据模型或语法的查询。这可以导致显著的性能优势，因为该查询能够受益于对数据库 5312 的任何优化或调整。作为进一步的优点，可以独立地查询映射信息，以研究对具体查询 5302 的可能最优化。

相反，当创建可执行文件时其他现有技术“压平”了元数据模型，使得该查询必须面对整个数据库执行，查询结果再使用呈现给服务 5302 的视图 5308 分析。实际上，来自数据库 5312 的全部潜在相关对象必然在中心 5310 中例示，并且被变换到视图 5308 内，这样它们才能够在内存中处理，以执行查询。这就使内存承担了显著的负担，并且失去了设计到数据库 5312 内的所有性能优势。通过使用模型之间的映射信息，把查询自身转换为数据库 5312 的固有语法，只有查询结果需要被例示和变换，以便向外部服务 5302 呈现。

同样，用户界面 5303 可以通过知识库 5304 所提供的许多模型与数据库 5312 通讯。用户可以使用在用户界面 5303 中数据表达所对应的结构和格式的字段，在用户界面 5303 中创建查询。视图 5308 可以接收该查询并使用任何可得到的映射信息将其转换成对中心 5310 的查询，并且再使用任何可得到的映射信息将其转换成对数据库 5312 的查询，以便以数据库 5312 固有的语法显示整个查询。

应当承认，尽管单一视图 5308 显示为既用于用户界面 5303 也用于服务 5302，但是每个都可以有其自己的观察数据的外部模型，这些模型可以由知识库 5304 保存和提供。查询可以面对数据库 5312 执行，产生的结果可以通过中心 5310 和视图 5308 以用户界面 5303 容易可用的形式返回。更一般地说，尽管图 5 描绘了两层结构，与数据集成系统的星型架构一致，但是彼此为任何相对关系的任何数量的元数据模型都可以受益于本文介绍的访问数据库的技术，只要涉及多种模型中元数据之间关系的映射信息可得到。

图 6 显示了知识库服务 5304，包括转换引擎，它提供视图 5308 和中心 5310 之间的元数据转换服务。转换引擎可以提供若干查询在由不同模型和数据库 5312 所用的多种固有的元数据结构之间的转换，比如以上介绍的那些，以及模型之间若干对象的变换。所述转换引擎或多个转换引擎可以配备为知识库 5304 内的服务，如图 6 中的一般描述，其中可以登记和/或存储转换引擎。知识库服务 5304 可以访问转换引擎，将该查询转换为中心 5310 的格式。虽然未显示，但是在中心 5310 和数据库 5312 之间也可以提供类似的转换。更一般地说，转换引擎可以从外部模型以多种查询语言或程序语言接收查询，并且使用针对各自模型和数据库 5312 的可用映射信息，将查询转换成针对数据库 5312 最优化的结构中的查询。因此查询通常可以以视图 5308（或其他模型）固有的术语表达，并且以数据库 5312 固有的术语呈现给数据库 5312。

虽然转换引擎是一种概念上的转换查询方法，如以上介绍，但是应当承认，也可以设计出其他方法并用于本文介绍的系统。一般来说，这些方法将受益于分开存储系统使用的元数据模型之间的映射信息，以及到数据库 5312 的任何映射。通过以转换引擎或者某种其他工具或服务在运行时可访问的形式保留映射信息，元数据管理系统可以实现显著的性能成就。

图 7 显示的知识库服务为多个外部服务 5302 提供了转换引擎。服务 5302 可以是例如数据变换阶段 308、数据准备阶段 304、RTI 服务 2704、用户界面或者可能对数据库 5312 中的元数据执行查询的任

何其他服务或外部客户。服务 5302 可以以视图 5308 固有的语法向视图 5308 提出查询。转换引擎可以把查询转换为中心 5310 固有的语法，它又可以转换为使用数据库 5312 固有语法的查询。通过访问转换引擎把查询结果转换回服务 5302 固有的语法，查询结果可以返回到服务 5302。以这种方式服务 5302 与数据库 5312 可以使用它们自己的固有语法高效地通讯。应当承认，本文为描述若干查询而使用的术语“语法”是指任何语法、结构、格式、程序语言和/或界面，只要它们有可能用于表示若干查询，可以是外部的比如对若干服务或数据库，也可以是内部的比如在若干元数据模型之间。

图 8 至图 10 描绘了元数据模型可以如何被映射到关系数据库中的模式以进行持久存储。一般来说，可以使用面向对象的关系管理工具描述元数据模型。当这样的元数据模型登记到知识库时，比如通用知识库和操作知识库，使用以下讨论的多种技术可以将内存中的模型映射到关系数据库中的模式。这种策略尤其应服从使用比如 Apache Object/Relational Bridge(“OJB”)工具的管理，以保持面对关系数据库的 Java 工具。作为显著的优点，这种方法允许真正的设计灵活性，同时利用已面世的商业关系数据库的高性能。参考以下图 8 至图 10 将介绍许多特定映射，它们可以有益地用于存储元数据模型。

图 8 描绘了元数据模型和关系数据库之间的对应。元数据模型 5602 可以包括多个面向对象的类 5604，它们定义了模型 5602 的多种属性，比如关于元数据的信息，包括字段、字段名、字段属性、数据类型、数据层次、数据关系、临时信息、数据源信息或者与结构、位置或数据使用有关的任何其他信息。关系数据库 5608 可以包括多个表 5610，表示了用于物理上存储模型 5602 的关系模式。模型 5602 和数据库 5608 之间的映射，如一般地由图中的垂直箭头所描述，可以是通过模型 5602 中若干类 5604 到数据库 5608 中若干表 5610 的一对一的映射。以这种方式，类 5602 的各个方面在若干表 5608 之一中都具有对应的方面，使得模型 5602 的结构在数据库 5608 中被完全复制。从而能够保留模型 5602 和数据库 5608 之间概念上线性的转换。这样的

表达通常提供更高的性能，并且可以在执行时被直接编译，或易于预编译；不过，对模型 5602 的改变可能需要整个数据库 5608 的重建以及针对编译版本的对应改变。

图 9 显示了元数据模型到关系数据库的另一种映射。元数据模型 5702 可以是例如以上介绍的元数据模型 5602。模型 5702 和数据库 5704 之间的映射，如一般地由图中的垂直箭头所描述，可以是从模型 5702 内若干类的属性到数据库 5704 内若干表 5706 中若干项中。表 5706 可以组织为使一定的用途最优化，比如通过在分开的物理表中组织版本数据或运行时的人为现象，无论由模型 5702 所使用的面向对象的结构如何。这种方法有利地允许在普通的表结构内完全表达任意模型的特征。这种方法可以提高可扩展性，因为对模型 5702 的任何改变将仅需要对数据库 5702 中受到影响的任何项进行更新，比如一两行的更新，而没有另外影响在表 5706 中所存储的描述。一般说来，这表示了在为持久性而使用数据库 5704 的相对高性能与模型 5702 与其持久形式之间映射的相对可扩展性之间的设计折衷。

图 10 显示了以上图 8 和图 9 中介绍的模型映射的组合。元数据模型 5802 可以是例如以上介绍的元数据模型 5602。模型 5802 和数据库 5808 之间的映射，如一般地由图中的垂直箭头所描述，可以部分地是从模型 5802 内的若干类 5804 直接到数据库 5808 内的表 5810，后者具有对应的结构，如以上参考图 8 的介绍。模型 5802 可以由用户修改，比如向类 5804 增加属性 5806。对数据库 5808 中存储的模型可以进行对应的改变，比如通过在普通表 5814 中记录描述项 5812，如以上参考图 9 的介绍。因此模型的静态部分可以被映射到更持久、固定的模式，而模型的非静态即用户可配置部分可以被映射到可扩展性的描述模式。以这种方式，存储模型 5802 的关系模式可以是混合式的，它有利地组合了模型的相对固定部分的性能与模型的用户可配置部分的可扩展性。

每个登记的模型都可以是持久的。当登记第一模型比如视图时，该模型可以被传递到连同第二模型比如中心的登记过程以及第一模型

到第二模型的映射。在第一模型的属性能够被映射到第二模型时，除了映射自身以外，不需要附加的持久性机构。不过，在第一模型的属性不能够被映射到第二模型时，为了保持未映射的属性可以提供某种机构。应当承认，任何具体模型都可以未映射、部分映射或全部映射到另一个模型。在属性需要持久性的那些事例中，即它们未被映射到现有模型时，就可以采用以上参考图 8 至图 10 所介绍的用于可扩展模型的一切技术，以提供模型持久性的存储机制。确切地说，最普通的表形式通过许多设计周期可以提供所期望的持久机构，同时通过为模型未映射的部分复制类结构，可以有利地部署运行时的模型。

应当进一步承认，以上介绍的普通结构可以为若干可扩展模型提供反身存储机制。该存储机制可以“理解”其环境，并可以寻找模型描述，为一切对象确定有关的类、属性、映射等。这些反身能力可以用于提供更高级别的设计环境，其中模式比如以上介绍的普通表格式能够以容纳扩展的方式保持模型属性。

图 11 描绘了向外部元数据揭示多个内部服务的架构。在一定的事例中，元数据可以驻留在由本文介绍的元数据管理系统所管理的元数据模型之外，比如在若干分开的企业或若干企业应用程序之间共享数据的情况下。访问这样的外部元数据的架构可以包括具有第一视图 5904 的外部元数据 5902、中心 5906 以及通向多个内部服务 5910 的第二视图 5908。

元数据管理系统可以提供外部元数据 5902 的第一视图 5904，它又可以连接到中心 5906，以便为外部元数据 5902 提供公共的内部模型。若干内部服务 5910 可以类似地通过它们自身的元数据视图——第二视图 5908——被映射到中心 5906。通过执行互连的模型 5904、5906、5908，内部服务 5910 以内部服务 5910 固有的形式可以访问外部元数据 5902。内部服务 5910 又可以被部署在面向服务的架构中，以提供对外部元数据 5902 的访问，作为元数据管理系统之内的、或者更广义地遍及全企业的服务。

图 12 描述映射模型驱动的元数据变换，使用解释的映射在若干

模型比如视图和中心之间进行转换。元数据管理系统 6000 可以包括中心 6002、一个或多个转换引擎 6004 以及一个或多个视图 6006、6008。转换引擎 6004 可以包括映射模型 6010，特征为中心 6002 和视图 6006、6008 之间的一个或多个映射。当收到请求时，可以解释这些模型，使用映射模型 6010 确定应当如何把对象的事例表示给请求者。模型 6010 可以以多种形式表示，包括作为模型（如数据结构，比如 Java 类或 EMF 对象或事例），它可以提供更高的设计灵活性，或者作为编译后的代码，它可以向转换引擎 6004 提供更高的执行效率，或者作为解释的代码。更一般地说，单一模型至模型的映射即映射模型 6010 可以例示在任何数量的不同转换引擎 6004 中。同时，不同的转换引擎 6004 可以例示任何数量的不同映射模型 6010，以任何数量的形式，范围从抽象模型到编译后的代码。映射模型 6010 可以登记在转换引擎 6004 的转换登记处（未显示），以提供公共访问和一致性。

在现有系统中，视图至中心的映射典型情况下产生为静态映射，一旦被部署就不会改变。通过把视图 6006、6008 和从视图至中心的映射 6010 视为模型，当元数据事例从视图移动至中心时可以直接地解释映射，反之亦然。视图可以在内部被表示为例如 Java 类、Java 代码或者基础模型的某种解释。同样，映射也能够以多种形式解释，比如 Java 代码、Jython（基于 Java 的脚本）等。当收到请求时，该请求可以被视图模型、映射模型和中心模型所参数化。模型驱动的转换引擎能够接收以模型之一表示的对象，并返回以另一种模型表示的对象。

例如，中心可以是使用解释的 Java 代码所访问的面向对象的构造。同样，可以利用 Java 或某种其他的解释程序语言解释视图 6006、6008。转换引擎 6004 可以使用中心 6002 和视图 6006、6008 之间的元数据模型映射，在中心和视图 6006、6008 之间移动若干请求和对象事例。可以通过用户手工操作，或者响应在一个或多个元数据模型或对象中的变化而自动地（或手工地）动态地修改转换引擎 6004。

应当承认，无论是解释的、编译的或以其他方式执行的，解释/执行模型的软件或软件引擎都可以是同步的或异步的。在异步环境中，

对模型的访问是通过消息服务或其他异步技术。在同步环境中，通过对引擎的应用程序编程接口或其他同步接口可以对引擎进行直接地调用。

图 13 显示了与元数据环境的交互。模型 6102 可以被表示为无版本的类 6104（存储在操作知识库中）和版本化的类 6106（存储在通用知识库中）。为了与模型交互，可以为用户 6110 提供用户元数据环境 6108。在以下说明中使用的“环境”是指基础模型数据和用于模型或元数据的其他上下文信息，一位或多位用户 6110 观察和变换它们时可以通过观察、查询和操作模型和模型数据的一切适当的图形用户界面、命令行界面或其他编程界面，包括存储的模型和元数据事例，无论存储在易失性或非易失性存储器中还是在两者中都有，并且包括其操作属性和设计属性，以上一切数据的一切版本概不例外。尽管普通术语“环境”（或“用户环境”）试图泛指一切模型上下文，一位或多位用户通过它有可能与元数据交互，但是如以下介绍，有几种环境是特别期望的。以下若干实例并不限制本文介绍的系统可能有效地采用的用户环境的数量和多样性。

例如，模型 6102 可以是以上介绍的任何视图或中心，或者任何其他元数据模型。该模型可以包括若干操作类和属性，以及若干设计属性和类。如以上指出，根据各种模型类的目的，模型 6102 可以存储在两个不同的知识库中。因此操作知识库可以被配置为存储使用模型所执行的作业的元数据结果，而通用知识库可以被配置为支持协同和迭代的设计过程。应当承认，操作知识库和通用知识库可以被物理地和/或逻辑地分离，每个都部分地由其中存储的模型类子集所定义，部分地由访问每个所用的若干服务和方法所定义。

用户 6110 可以以多种不同的模式与元数据环境 6108 交互，比如工作空间或组空间。工作空间也称为沙盒，可以在无版本环境中向模型提供动态的编辑，其中例如元数据对若干设计属性的变化或者保存为新模型，或者盖写到现有模型。工作空间可以在用户计算机上本地地存在，或者在用户可以与元数据交互的服务器上远程地存在。典型

情况下，把模型放在工作空间会对其他潜在用户锁住该模型。不过，工作空间也可以提供共享使用，以使得不止一个用户可以编辑该工作空间并保存变化。组空间可以提供版本划分，以使得可以对多个版本进行检出（check out）、检入（check in）、分支排列等。

更一般地说，组空间可以为以上讨论的所有元数据版本划分能力提供元数据环境。例如，版本化的元数据环境可以支持元数据的版本划分，由各个用户创建或编辑。因此，版本化的元数据环境的用户可以检出模型并将该模型检入回去作为新的版本。因此，虽然工作空间可以允许协同编辑，然而组空间却可以实现在版本控制下的元数据协同和/或顺序编辑。

用户界面也可以提供对事件空间的访问，它是与以上介绍的操作属性和/或操作知识库相关联的元数据环境 6108。

用户环境 6108 也可以是或包括联合用户环境，为跨越全企业的许多知识库提供中心化的全局环境。联合用户环境可以提供不同知识库的公共视图，或者可以分开地表示每个知识库。

用户 6110 可以是例如通过图形或命令行界面与元数据环境 6108 交互的用户人员，或者访问知识库中元数据模型的程序或服务，比如以上介绍的发现数据阶段 302、数据准备阶段 304 或数据变换阶段 308。

图 14 描绘了通用知识库 6202，存储着元数据 6204 的多个版本。元数据 6204 可以是例如以上讨论的用于视图和中心的元数据。元数据数据库 6206 可以是以上介绍的任何数据源 102。每个版本的元数据 6204 都可以提供不同的、但是相关的元数据版本，存储在元数据数据库 6206 中。元数据 6204 的若干版本可以由例如从事数据集成项目的开发团队创建，并使用数据库 6206 中存储的若干事例和其他内容进行对比。

图 15 描绘了通用知识库，包含多个对象版本 6304，它表现了元数据数据库 6306 中存储的元数据特征，全部如以上的一般介绍。客户 6308 可以与对象版本 6304 交互，或者直接进行，或者在以上介绍的用户环境之一中，并且可以执行以上一般介绍的一切设计操作。这可

以包括比如元数据模型的动态比较、挖掘、编辑、测试以及任何其他适合的功能。客户也可以使用通用知识库 6302 和对象版本 6304 探查在元数据数据库 6306 中的基础数据。

图 16A 描绘了使版本化的元数据对象的协调。通用知识库 6402 和版本化的对象 6404 可以是以上介绍的通用知识库 6302 和版本化的对象 6304。版本的协调在设计周期的各个位置都可能是所期望的，并且在典型情况下是释放可执行模型是所要求的。将若干版本化的对象 6404 协调为单一事例 6408 可以通过协调过程 6406 控制。有许多公知技术可以用于自动的、半自动的和手工的协调。一般来说，以本文介绍的系统可以使用任何这样的技术。协调过程 6406 可以有利地保留为了协调单一事例 6408 的完全版本历史和协调谱系，以允许对协调过程 6406 的修改，返回到任何先前未协调的状态，或者探查源元数据和协调谱系。在协调期间比如合并中解决了元数据中的直接冲突时，可以恢复先前的属性值，以用于分支和多种版本的替代协调。

图 16B 描绘了跨越若干协调区的分段协调过程。为了为元数据的若干事例管理复杂的协调过程和保持准确的协调谱系，可以提供若干协调区。在讨论图 16B 的协调区之前，先指出元数据事例的某些有用属性。

在企业中的每个元数据事例都可以具有相关联的协调区属性，它定义了该事例与协调区的关联。协调区可以由协调过程的设计者选择，以反映例如数据的组成机构分离，比如人力资源、会计、财务、存货、制造、工资单、工程等。协调区可以是关于地理的，以适合数据和企业的任何粒度程度，比如国家、区域、州、乡镇、建筑、设施等。协调区可以是历史上的或建筑上的以便于分开，例如传统系统与新系统、雇员台式电脑与主机、顾问与雇员。协调区可以反映将商家组织为若干部门或其他小组，比如消费产品、原始设备制造、产品、零售业务、电子商务业务等，或者更一般地分为制造和零售。同样，可以为公司取得的或者从公司分离的新商业单元提供若干协调区。

对每个协调区，关于优先权、例外、联合等都可以定义任何数量

的协调规则。协调的技术众所周知，并且根据协调规则，为了协调在协调区中的元数据事例，可以有效地采用所有这些技术。协调区可以进一步定义匹配类型，它定义了协调结果是如何在引用事例的模型中传播，比如没有匹配（删除了副本）、视图匹配（在视图级别保持版本）、以及/或者额外视图匹配（在中心级别保持版本）。

对象的每个事例也可以具有标识符，唯一地标识协调区内的对象。每一项都能够根据各种上下文或层次描述，比如捕捉这些项的语义上下文。该项可以是对象、类、属性、数据项、数据模型、元数据模型、模型、定义、标识、结构、语言、映射、关系、事例或者其他项或观念，包括另一个语义标识符。语义标识符识别该项时可以根据该项的属性、该项的物理位置、该项与一个或多个其他项的关系比如所在的层次等。在某些情况下，关系可以定义为没有某种具体关系。关系可以基于语义。关系可以包含该项在关系层次中的位置。例如，该项可以基于该项与其他关联项的关系而识别，并且可以直接地与另一项关联、间接地与另一项关联，以及/或者通过一个或多个其他项间接地与另一项关联。关系可以被链接或递推地定义，除静态标识符之外，还允许动态标识符。例如，如果两项之间的关系改变了，收编这两项之一的另一项的语义标识符也会收编两项之间的改变后的关系。

作为更具体的实例，项 **Jim** 可以被标识为 **Jim**，居住在美国某州某城某路 **111** 号，电话号码 **555-555-5555**，社会保险号 **012-34-5678**。作为替代，**Jim** 可以根据他与其他人的关系标识。**Jim** 可以被标识为 **Betty** 的儿子、**Larry** 和 **Jeff** 的哥哥、**Jessica** 的父亲和 **Frank** 的外甥。

语义标识符可以是项的唯一标识符。在以上实例中，如果世界上仅有一个这样的 **Jim**，他是 **Betty** 的儿子、**Larry** 和 **Jeff** 的哥哥、**Jessica** 的父亲和 **Frank** 的外甥，这个语义标识符可能是用于 **Jim** 的唯一标识符。针对项的唯一语义标识符所考虑的关系有可能少于该项与其他项的全部关系。要是世界上仅有一个这样的 **Jim**，他是 **Betty** 的儿子、**Larry** 的哥哥和 **Jessica** 的父亲，仅仅这些关系的存在将足以创建唯一语义标识符。**Jim** 与 **Jeff** 和 **Frank** 的关系将不必考虑。创建基于保证

唯一性的最少数量关系的语义标识符可以是有利的。例如，如果语义标识符存储在数据库 112 中或者由数据集成系统 104 处理，不太复杂的语义标识符将需要的空间更少而且允许处理得更快。

为某项创建唯一语义标识符所必须的关系数量可以随上下文而变。例如，在上下文——上下文 A 之内通过项 1 与两个附加项——项 3 和项 4 的关系，可以区别第一项——项 1 和第二项——项 2。也就是说，在上下文 A 中，项 1 的唯一语义标识符可以是它直接地与项 3 和项 4 关联，并且通过项 3 和项 4 与任何数量的其他项间接地关联。在不同的上下文——上下文 B 中，项 1 可以通过它与项 3（但是也许不是项 4）的关系，以及它与另一项——项 5 的关系及与项 6 没有关系被唯一地识别。因此，在本文介绍的数据集成方法和系统的实施例中，项的语义标识符，比如涉及数据集成作业或数据集成平台的项，可以用该项的依赖于上下文的标识符所提供。在实施例中，这种依赖于上下文的标识符可以以原子格式存储，比如在数据知识库中。

上下文 A 和上下文 B 可以是两个不同的输入、映射、运行版本、模型、元代理模型、事例、工具、视图、对象、类、项、关系、属性或者上述所有的任何联合。协调或比较设施可以比较在不同输入、运行版本、模型、元代理模型、事例、工具和/或若干项中项标识的值和/或语义，并且根据该比较结果，判断或帮助确定采取什么动作，或者制止什么动作。例如，协调引擎可以比较输入事例 A 所使用的模型和元代理 B 所使用的模型。根据这种比较结果，可以决定元代理 B 能够访问输入事例 A 的数据和元数据，无须变换或修改，并且该比较设施可以指挥元代理 B 继续运行。在另一个实例中，可以将工具 A 与工具 B 进行比较，而且可以确定执行跨工具对象的合并，其中每种工具都能够访问其他工具的对象。在若干实施例中协调设施可以触发转换设施从而帮助跨工具对象的合并，比如创建桥、元代理、中心等，以转换需要转换的任何对象，比如基于不同语法而处理在每个各自工具中具体项标识的转换，或者基于工具之间的其他差异的转换，由比较结果所确定。

在若干实施例中存储、保留、记录、处理和/或解释语义标识符所依照的语法，可以以字符串的结构或格式存储、保留、记录、处理和/或解释。例如语法可以是：列名::表名::数据库名。这种语法可以关系到例如标识数据库中表的列的语义标识符。以这种语法组成的字符串可以是：年龄::雇员::雇员数据库。这个字符串可以关系到例如标识在具体雇员数据库中雇员年龄的语义标识符。在（以上实例）上下文 A 中，项 1 的语义标识符所对应的字符串可以是：与项 3 的直接关系::与项 4 的直接关系。语义标识符和对应的字符串还可以包括项 1 和项 5 之间缺乏直接关系，比如出现在上述上下文 B 中的情况。

可以对语法字符串进行语法分析。可以截断、修改语法和/或字符串，以及/或者可以重新排序语法和/或字符串的元素。转换引擎可以执行截断、修改和/或重新排序。当语义标识符的唯一性不需要语法和/或字符串中包括的全部关系时，截断语法和/或字符串可能是有益的。假设在语法字符串的给定上下文中，所有的项都与项 3 直接有关；例如项 3 是数据库，其中存储了所有这些项。就可以截断语法字符串，比如创建省略涉及项 3 关系的字符串，而仍然保持唯一的语义标识符。截断语法和/或字符串可以减少存储需求，提高处理效率。改变语法和/或字符串中的关系顺序也可能是有益的，例如减少了数据集成过程的处理时间。如果首先处理的公共关系较少，为了识别该项系统将很有可能需要访问和处理的与该项相关联的关系也很少。例如，如果涉及项 3 的项非常少，涉及项 4 的项更少，而涉及项 2 的项很多，取决于上下文，一个语法字符串可以允许识别项 9 的时间短于另一个语法字符串。在一个上下文中唯一地标识某项有可能仅仅需要语法字符串的一定元素，而在另一个上下文中却需要语法字符串的全部元素。

协调引擎可以使用元数据事例的标识以及定义协调规则的协调区和一切匹配类型规范，对元数据事例执行协调。协调操作可以采用语义标识符唯一地识别协调区内的事例，并且可以转换或以其他方式修改另一个协调区中被协调事例的语义标识符的格式、语言和/或数据模型。协调操作可以包括往来于一个或多个数据工具、语言、格式和/

或数据模型往来于至少一种其他数据工具、语言、格式和/或数据模型的协调或映射。例如，协调操作可以包括往来于或者若干已知数据集成工具之间的协调或映射，比如 IBM 的 WebSphere DataStage 7、IBM 的 WebSphere QualityStage、Business Object 工具、IBM-DB2 Cube Views、UML 1.1、UML 1.3、ERStudio、IBM 的 WebSphere ProfileStage、PowerDesigner（具有对 Packages 和 Extended Attributes 的附加支持）和/或 MicroStratege 工具。协调引擎和/或协调操作可以选择性地体现在元代理中。可以按照批量、实时和/或连续原理运行、执行和/或实施协调操作。可以作为服务，例如作为面向对象服务架构的一部分提供协调操作，或者使之可用。

对于语义标识符、数据库 112、包括一个或多个语义标识符的数据库 112、信息系统、包括一个或多个语义标识符的信息系统或其他项，协调操作一旦存在，它就能够被协调往来于、映射到、链接到、用于或关联到共享至少一项协调操作的一切其他语义标识符、数据库 112、包括一个或多个语义标识符的数据库 112、信息系统、包括一个或多个语义标识符的信息系统或其他项。在若干实施例中，比如使用原子数据知识库作为转换操作的中心，协调操作的映射能够在若干其他行为之间，根据在原始语义上下文和转换的语义上下文之间向后和向前执行的操作跟踪协调。取决于上下文，数据项的适当标识符可以改变，比如通过改变或截断语法和/或字符串以使存储能够更高效或处理能够更快，或者通过改变语义上下文改变时形成唯一标识符所使用的关系。因此，动态标识符可以联合可退回协调的益处以及在使用数据项的多种上下文中快速处理、高效率数据处理和有效操作的益处。

图 16B 描绘了若干协调区。一般来说，元数据对象或项在其自己的数据星座内被唯一地标识，不过，协调过程也必须通过可以联合来自不同数据源之对象的不同事例的协调过程管理标识。可以为来自许多数据源的元数据定义许多协调区 6450-6458。例如，图 16B 左侧的协调区 6450-6454 可以是来自企业各种单元的源数据，比如公司的部门或分立数据库。使用以上介绍的技术，为 6450-6454 这些数据源协

调区在每一个中的每个元数据事例都可以定义协调区、规则、匹配类型和标识符。根据协调规则，协调引擎可以把来自两个协调区（如协调区 6450 和 6452）的数据协调到新的协调区 6456 内，其中每项都被唯一地标识，并且表示来自源协调区元数据事例的协调后版本。这种新的协调区 6456 又可以与一个或多个其他协调区（如协调 6454）进行协调，以提供另一个协调区 6458，表示企业内元数据事例的完全协调。与此同时，任何协调区在其自身和数据源之间都可以具有一个或多个协调区，以便在将其引入具体协调区之前，更精细地协调来自一个或多个数据源的元数据。因此，图 16B 的模式可以以任何方式重复、变更和/或扩充，以完成一切任意的模式或流程的元数据事例协调。

作为具体实例，第一协调区 6450 可能表示人力资源的元数据，可以包括所有新雇员的起薪。第二协调区 6452 可能表示工资单数据，包括全体雇员周薪信息。用户比如在公司会计部门中的某个人，可以把这些协调区协调到新协调区 6456 内以跟踪工资信息。为了准确和一致，可以分析这个协调区 6456 内的元数据，并且可以进行修改直到获得满意的协调结果。另一个协调区 6454 可能表示公司财务数据库的元数据。财务数据库可以包括公司的全部财务数据，包括公司工资费用的元数据。这种数据可以表现为具有高质量的特点，并且可以被审计或在公司的其他领域内以其他方式使用。设计协调规则时可以遵从关于数据质量的一切信息，比如在的一种数据源表示由已知具有低质量保证标准外部承包商所准备的编制，而另一种数据源却表示来自公司内受过良好培训和监督雇员的数据项。来自这个协调区 6454 的元数据可以与来自另一个协调区 6458 中的另一个协调区 6454 的工资元数据进行协调，协调区 6458 包含的元数据表示公司内员工工资的完全集成的视图。为了进一步扩展本实例，图 16B 的所有协调区 6450-6458 可以特定到具体部门，并且可以与来自其他具体部门或来自具体收获的集成协调区进一步集成。同样，来自不同具体部门、地理位置、子公司、功能商务单位等的数据也可以使用以上介绍的分段协调顺序地集成。

应当承认，从以上介绍的分段协调过程产生了许多显著优点。一个优点是协调谱系的保存。在复杂的数据集成环境中，很有可能存在元数据的多个源，包括分层文件、平面文件、联合数据源等。在这种环境中，跟踪集成和协调的过程以及保持沿着该路径反向协调步骤的能力可能都重要。由以上介绍的技术所提供的协调谱系允许完全地审计、检查和修改协调谱系，并且为完全协调的模型中的每个元数据事例都提供了通向原始数据源的规定通路。

作为另一个优点，分段协调提供了集成模型中数据源的可见性。例如，分析师和管理人可以使用图 **16B** 的完全协调区 **6458** 作为商业分析工具的元数据模型。在基于分析工具形成商业决策以前，检验数据资源及其质量可能是有益的，甚至是必须的。作为另一个实例，商业决策可能需要数据的具体视图。对上门推销活动来说，地址的街道名称可能是关键的，而对邮递活动而言邮政编码可能是重要的。不同的数据源可以以不同的详细程度和不同的准确程度携带相关信息。适当时可以检查和修改协调过程，以便为设计推销活动表示元数据在分析工具中所期望的最佳视图。继续这个实例，一种数据资源可以定义非常精细和很准确的地址，但不需经常更新，例如两年一次或在收到信息时间歇地更新。另一种数据资源可以包含极其新近的信息（比如电话号列表），包括街道地址但是没有邮政编码。对集成的企业数据模型通过检验协调谱系，管理人可以认识到只有邮政编码事例有可能过时，从而重新设计协调过程（或选择性地，集成数据模型本身），以便合成来自街道地址的最新邮政编码。

作为再一个优点，分段协调提供了从集成视图向数据资源上行传播协调和修改的能力。这可以根本改进来自企业内部原始数据源的元数据和数据的数据结构和质量。

以上的一般方法在高度异类的数据环境中可以具有具体的用途。例如，在复杂的具体环境中，许多分立组比如制造、会计、人力资源和工程，每个都可以保留单独的数据仓库，其中具有专门针对该组的宽阵列的数据库。在这种环境中，可以有利地使用数据集成，以容许

改进商业智能的方式集成单独的数据库。集成可以是组内纵向的，比如将若干数据库集成为该组的综合性元数据模型，集成也可以是横向跨组的，比如将来自每个组的工资单集成为综合性工资单元数据模型。完整的公司范围数据集成可以包括组内集成的交替步骤和跨组集成。

图 17 描绘了版本化元数据对象的协调。通用知识库 6502、版本化对象 6504、协调过程 6506 和协调后的单一对象事例 6508 都可以如同参考以上附图所介绍的。此外，每个对象版本 6504 和单一事例 6508 是指元数据数据库 6510 中存储的元数据。应当承认，在元数据数据库 6510 中的元数据可以变化，或者是由于独立于模型的变化（如公司希望跟踪存货的新的附加特征时，或在某种数据集成作业的影响下），或者是由于对元数据的改变（如为商业分析目的向模型加入了某个数字的五日移动平均值）。因此所期望的可能是按照元数据对象的版本划分元数据数据库 6510 或以其他方式保留对元数据改变的历史。利用这种附加信息，用户对元数据的各版本就具有向后和向前移动的完全灵活性。

图 18 显示了在元数据过程中使用并发的实例。在这个实例中，在协调过程 6604 中协调了多个元数据事例 6602。在协调期间，大量的元数据可能需要被合并或盖写，以创建元数据模型的协调后版本。改进该过程可以通过将协调过程 6604 构建为若干独立过程对象，它们可以流入各个处理器 6606 进行独立的或管道式执行。独立过程对象可以流入包含多个处理器 6606 的单一硬件设备 6608，也可以流入不同的硬件设备 6610、6608，也可以流入通过网络可用的任何其他处理器或处理器组。

并发和并行处理的相关概念在本领域中众所周知，所以这里不必详细介绍。一般来说，并发和并行机制适应的情况是过程能够分成主要是自引用对象组件的“程序块”，也称为子图（指对象依赖关系的有向图），能够独立地或在管道中处理。协调过程可以容易地被模型化为管道用于并发执行。例如，该过程可以包括向来自新元数据源的对象流分配标识的任务、从先前元数据源取来潜在冲突候选者的任务、

进行协调的任务、将协调结果合并到元数据对象的输出集的任务以及存储合并后元数据对象的任务。其他元数据过程也可以适于并发，比如元数据的引入。

以下附图介绍了与元数据管理相关联的几种方法。应当承认，这些过程可以由硬件、软件或它们的某种组合实现。这些过程可以在一台或多台微处理器、微控制器、嵌入式微控制器、可编程数字信号处理器或其他可编程设备实现，连同存储程序指令、程序数据和程序输出或其他中间或最终结果的内部和/外部存储器。应当进一步承认，这些过程可以作为计算机可执行代码实现，使用结构化编程语言比如 C、面向对象的编程语言比如 C++或 Java 或者其他高级或低级的任何编程语言而创建，编译或解释后运行在硬件和软件平台的任何同类或异类组上，平台包括计算机、网络或者其组合。该过程也可以采用各种各样的工具、平台和架构，以实现可伸缩的企业元数据管理系统。尽管以上提供了软件平台的若干特定实例，但是也存在着其他平台和技术并且可以有利地用于本文介绍的系统。

图 19 是从用户界面 6702 到元数据数据库 6712 的查询过程中涉及实体的图示。查询可以在用户界面 6702 处开始，用户在那里以用户界面固有的语法准备查询。该查询可以被传递到元数据模型 6704，比如视图。转换引擎 6708 或描述第一元数据模型 6704 比如视图与第二元数据模型比如中心之间映射的映射信息的应用程序又可以转换该查询。中心 6710 可以使用附加的转换或映射步骤将转换后的查询传递到数据库 6712，以将基于中心的查询转变为数据库 6712 固有语法中的查询。通过多种实体和任何适合的转换可以将该结果传递给原始发布该查询的用户界面 6702。

图 20 显示了从元数据模型扩充元数据数据库过程中涉及的实体。用户可以使用适当的编辑界面向视图 6802 加入属性等。转换引擎可以更新，以便在视图模型 6802 和中心 6804 之间转换元数据。虽然星型模型的中心 6804 通常以一致的形式维护，但是中心 6804 也可以被更新，取决于对视图 6802 变化的属性和为此的原因。此外，转换引擎也

可以更新，用于该中心和数据库 6808 之间的转换。数据模型 6804 和/或转换引擎也可以使用适合的数据库专用命令向数据库 6808 加入适合的行、列或表，以适应反映数据库 6808 内的新视图 6802。如果对数据库 6808 进行了改变，这些改变可以通过模型链被推回到视图 6802。

图 21 显示了从工具 6902 访问知识库 6910 过程中涉及的实体。工具 6902 可以是依据视图 6904 而通讯的第三方工具。工具 6902 可以通过视图 6904 请求映射后的元数据，由转换引擎可以将其转换为中心 6908 的形式。该中心可以通过另一个转换引擎进一步转换该请求，以对知识库 6910 中的映射后数据进行物理访问。因此该请求可以通过一连串的查询变换到达知识库。结果为一个或多个元数据对象，又可以通过许多转换或变换引擎传递，如从知识库 6910 移动到中心 6908，到视图 6904，最后到提出请求的工具 6902。因此在一方面存在着从外部工具访问知识库的方法，可以包括通过一个或多个模型将查询变换到知识库，并提供一个或多个对象，比如通过从知识库 6910 到工具 6902 的一个或多个对象变换的映射后元数据。优选情况下，这种方法以知识库固有的语法向知识库 6910 呈现查询，同时以工具 6902 固有的语法向外部工具 6902 呈现结果。

图 22 显示了工具访问版本化和非版本化元数据模型过程中涉及的实体。工具 7002 可以与用户环境 7004 通讯，它可以是例如以上所介绍的事件用户环境、团队用户环境或工作用户环境。用户环境 7004 可以实施为 Java 空间或适合使用元数据工具的任何其他框架或平台。根据用户环境 7004 的类型和工具 7002 的属性以及在用户环境 7004 中所执行的操作，用户环境可以与非版本化模型 7008 即操作知识库中的操作类和属性进行通讯，或可以与版本化模型 7010 即通用知识库中的设计类和属性进行通讯。在用户环境中可见的元数据模型可以被编辑并且写回到版本化模型 7010，或者作为对现有版本的替换或者作为元数据模型的新版本。应当承认，如果版本化元数据 7010 已经被另一个工具或用户检出，可以防止工具 7002 检出通用知识库中的版本化元

数据 7010。

图 23 显示了用户界面访问通用知识库中多版本元数据过程中涉及的实体。用户界面 7102 可以向通用知识库 7104 发布访问元数据的一个或多个版本 7108 的请求，并且可以进一步查询通用知识库 7104 关于元数据的其他版本以及多种版本之间变化的属性和按时间记录。

图 24 显示了元数据版本的协调过程中涉及的实体。版本 7202 可以通过协调过程 7212 与另一版本 7204 协调。利用另外的协调过程 7214、7218 可以对两个或多个其他版本 7208、7210 执行类似的协调。在每个协调之后或在所有的协调之后，可以把协调后版本合并为元数据新版本，以反映从先前版本的变化。这种协调可以在若干阶段中执行，或者一次全部完成，可以对协调冲突、协调顺序等选择性地实行用户控制。

图 25 显示了在协调过程中使用并发所涉及的实体。该协调过程可以是以上介绍的协调过程，只不过每个分立的协调都可以独立地传递到多个处理器 7304——它们可以在群集 7302 中也可以彼此物理远离，并且以管道或并行方式执行，取决于每个协调阶段之间依存关系的属性。

尽管已经连同一定的优选实施例介绍了本发明，但是应当理解，本领域的普通技术人员会认可其他实施例，以及它们落在本公开的范围之内。

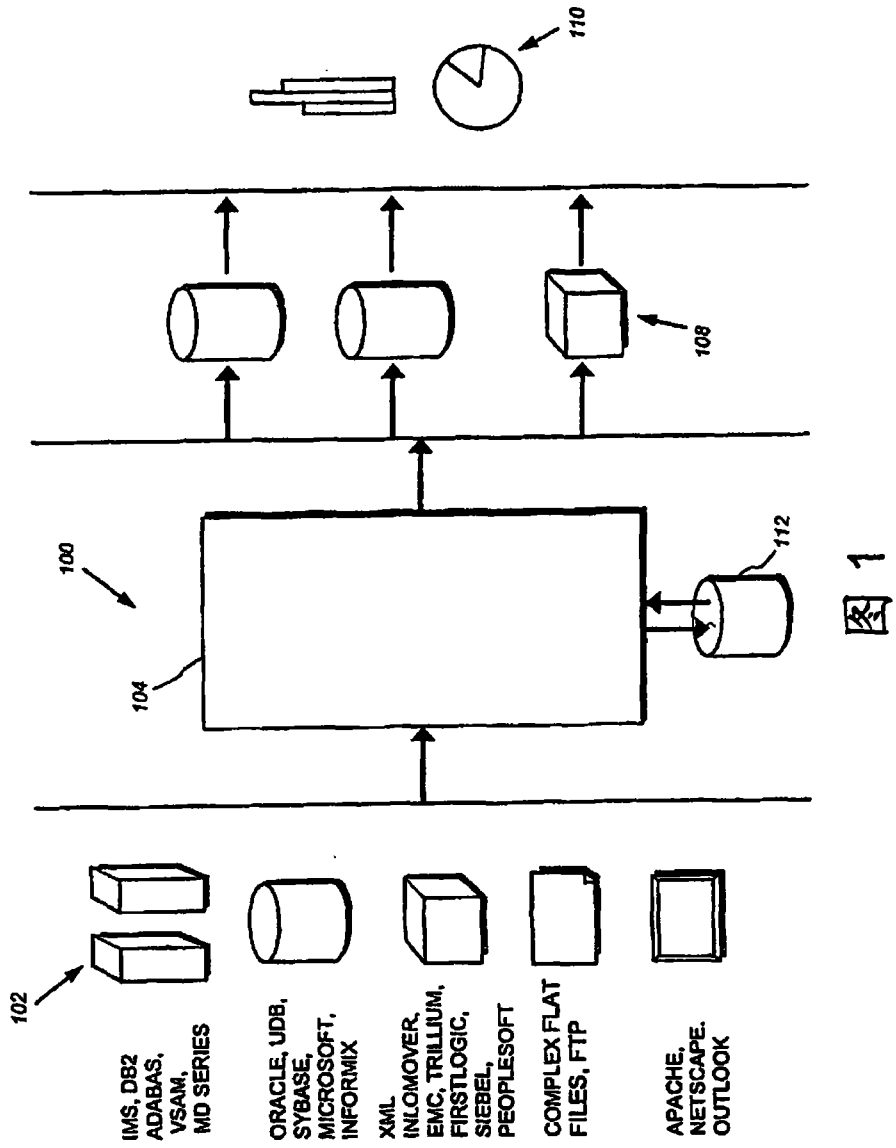

110
108
100
112
104
102
IMS, DB2
ADABAS,
VSAM,
MD SERIES
ORACLE, UDB,
SYBASE,
MICROSOFT,
INFORMIX
XML
INI,OMOVER,
EMC, TRILLIUM,
FIRSTLOGIC,
SIEBEL,
PEOPLESOFT
COMPLEX FLAT
FILES, FTP
APACHE,
NETSCAPE.
OUTLOOK

图 1

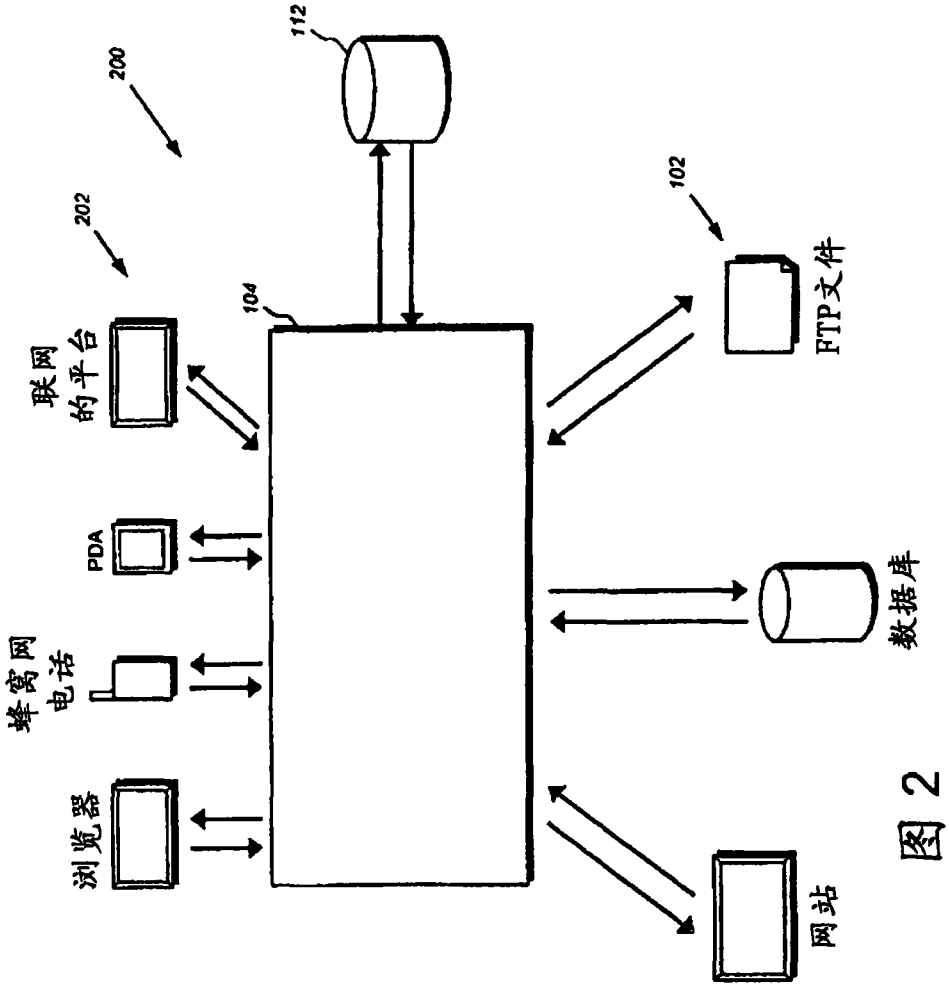


图 2

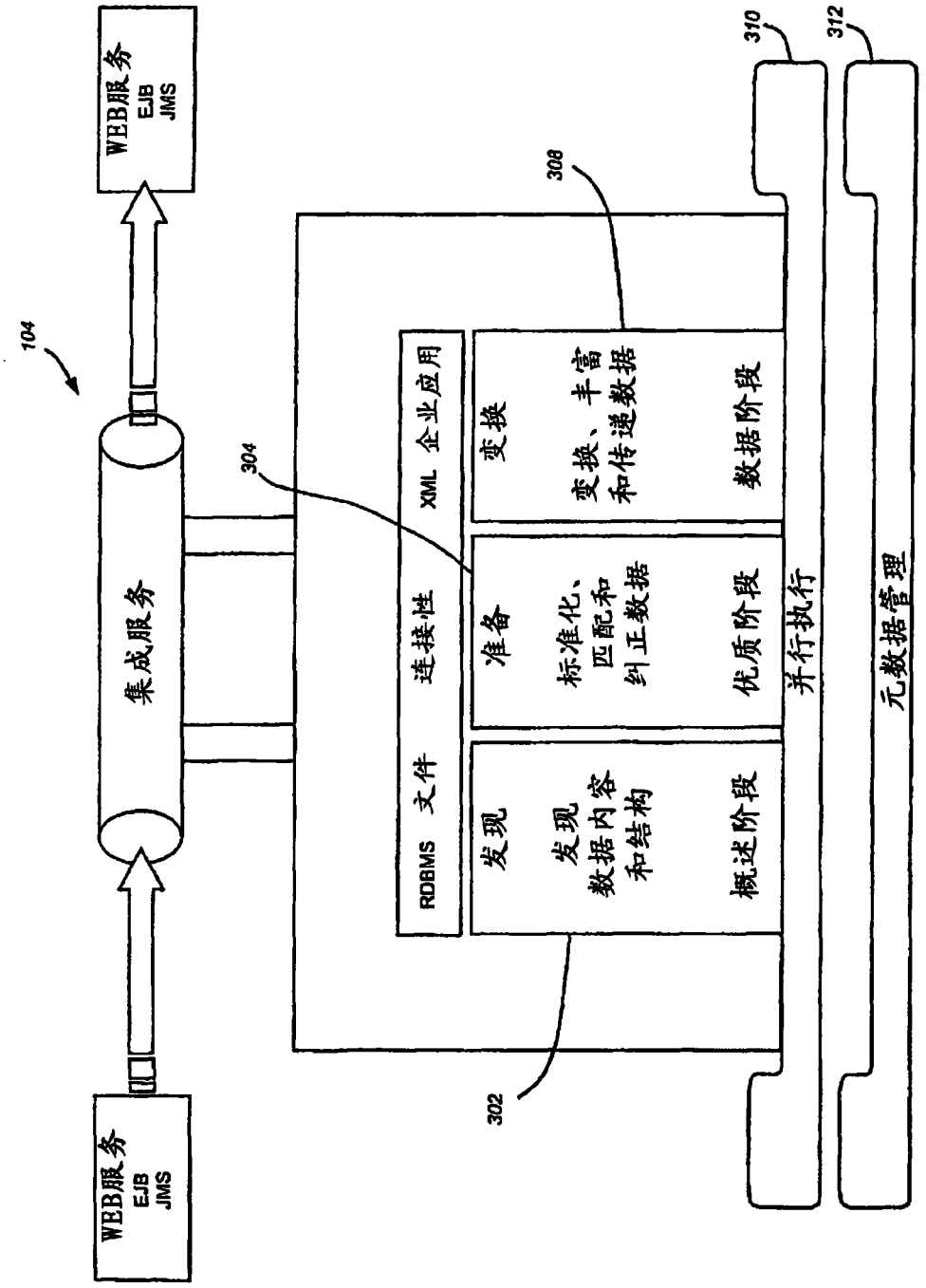

WEB服务
EJB
JMS
104
集成服务
WEB服务
EJB
JMS
304
RDBMS 文件 连接性 XML 企业应用
发现
发现数据内容和结构
概述阶段
302
准备
标准化、匹配和纠正数据
优质阶段
308
变换
变换、丰富和传递数据
数据阶段
并行执行
310
元数据管理
312

图 3

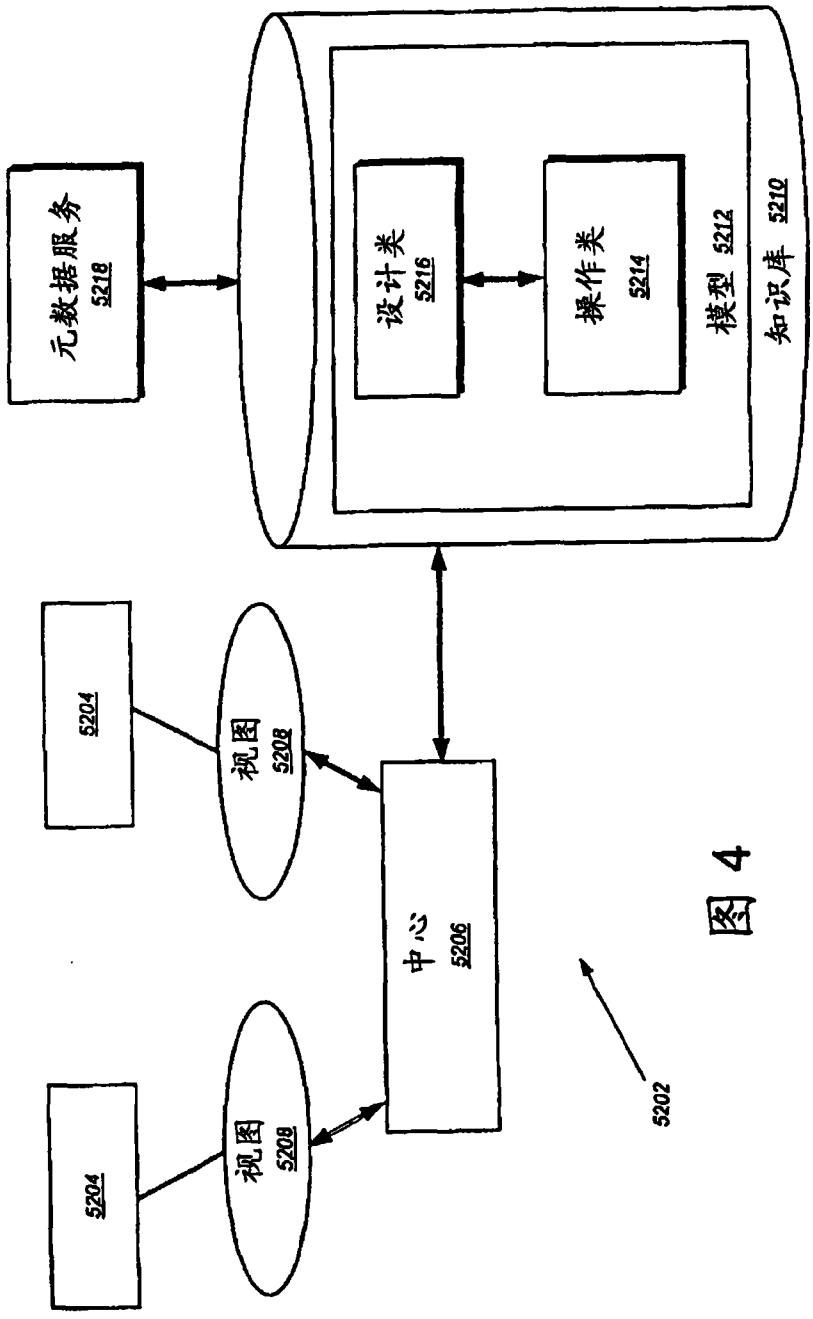

元数据服务
5218
设计类
5216
操作类
5214
模型 5212
知识库 5210
5204
视图
5208
中心
5206
5204
视图
5208
5202

图 4

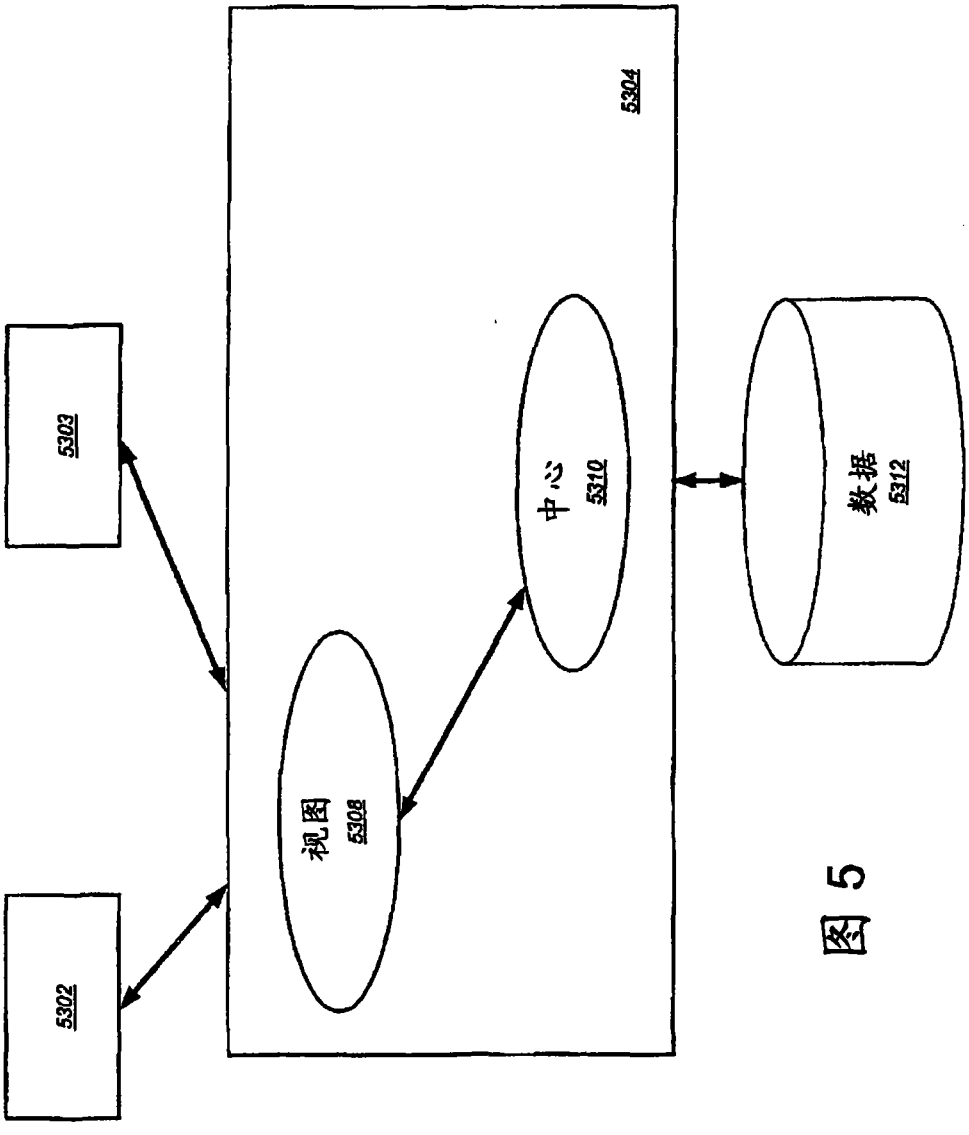


图 5

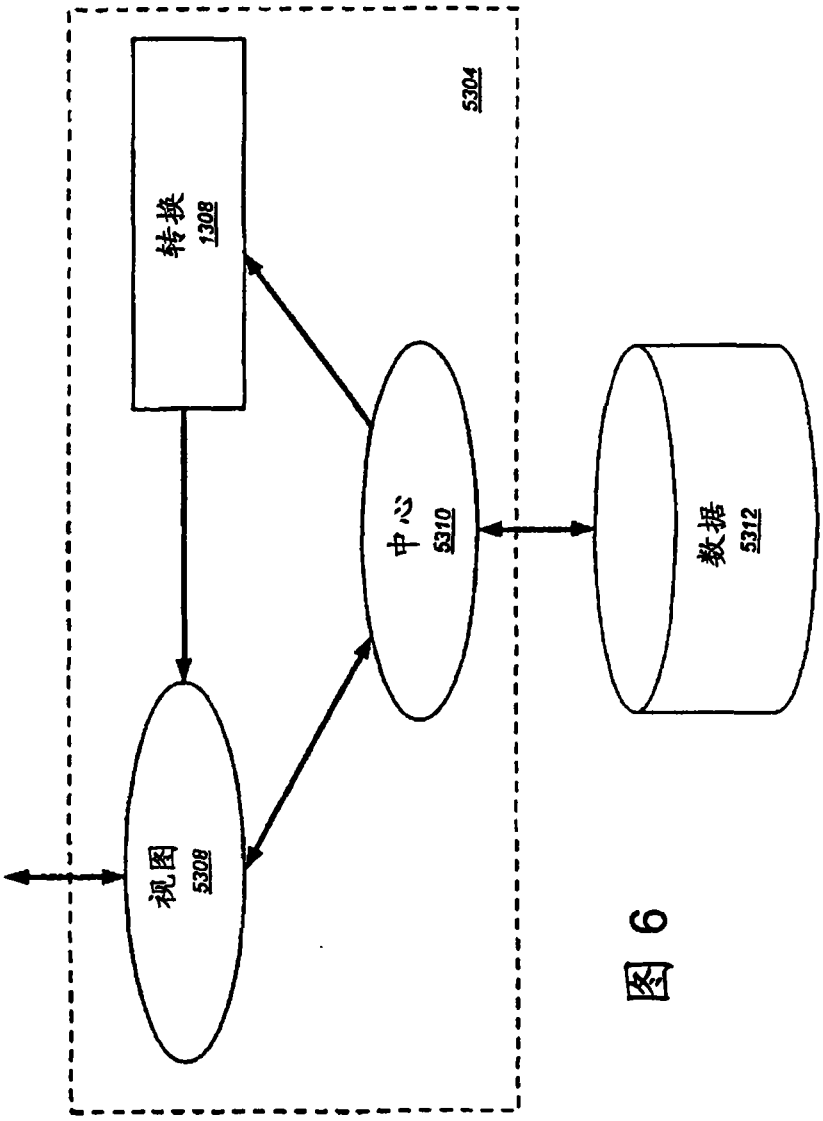


图 6

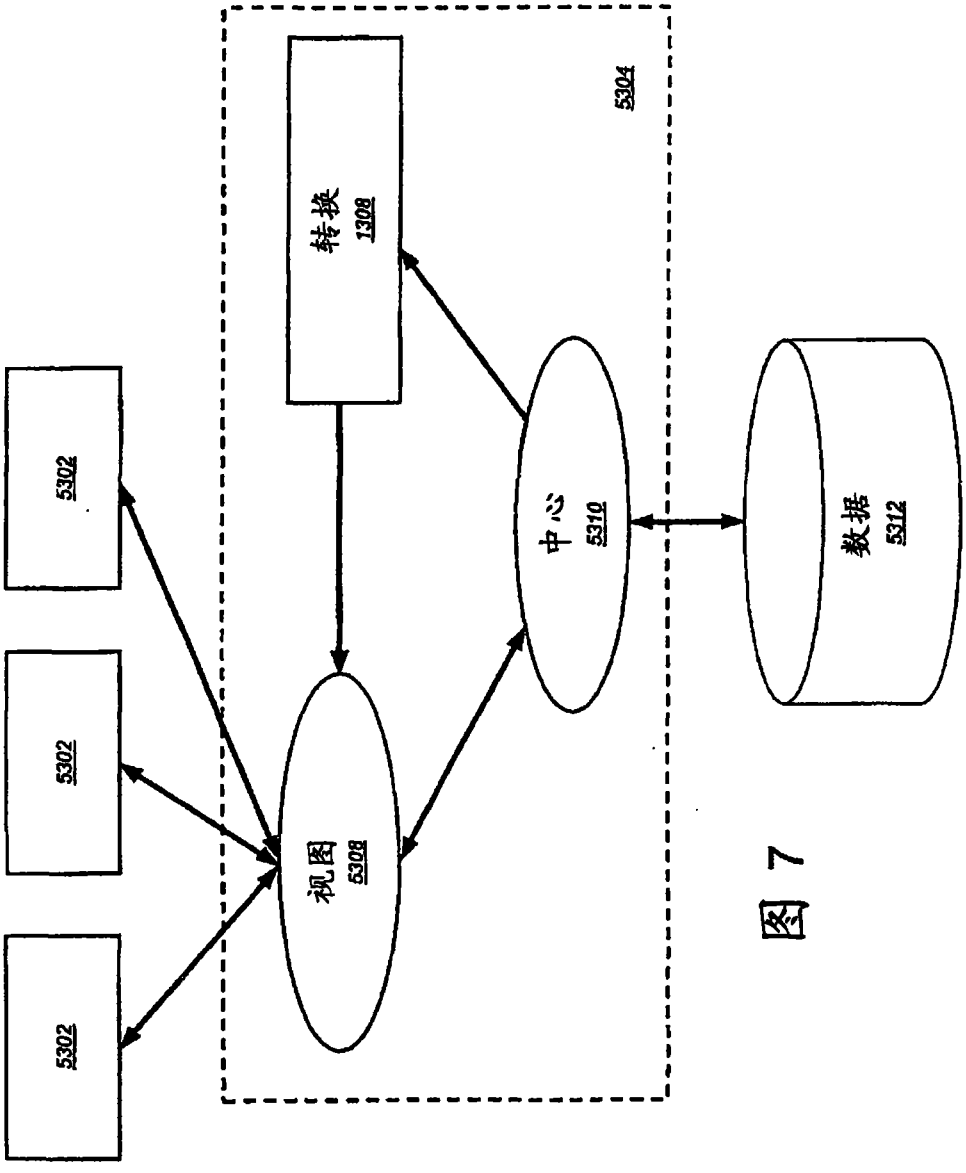


图 7

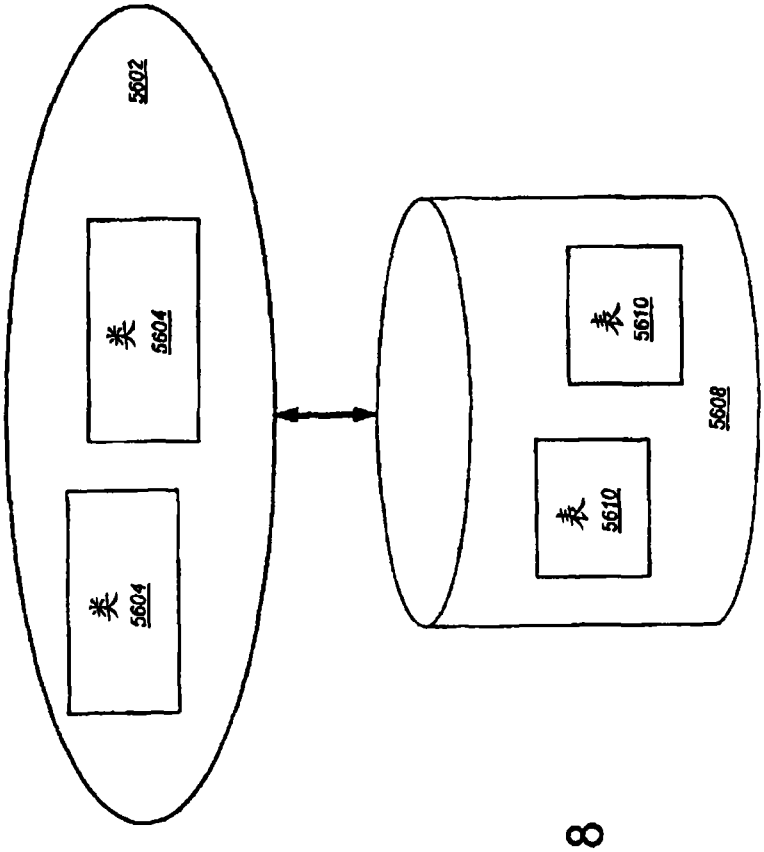


图 8

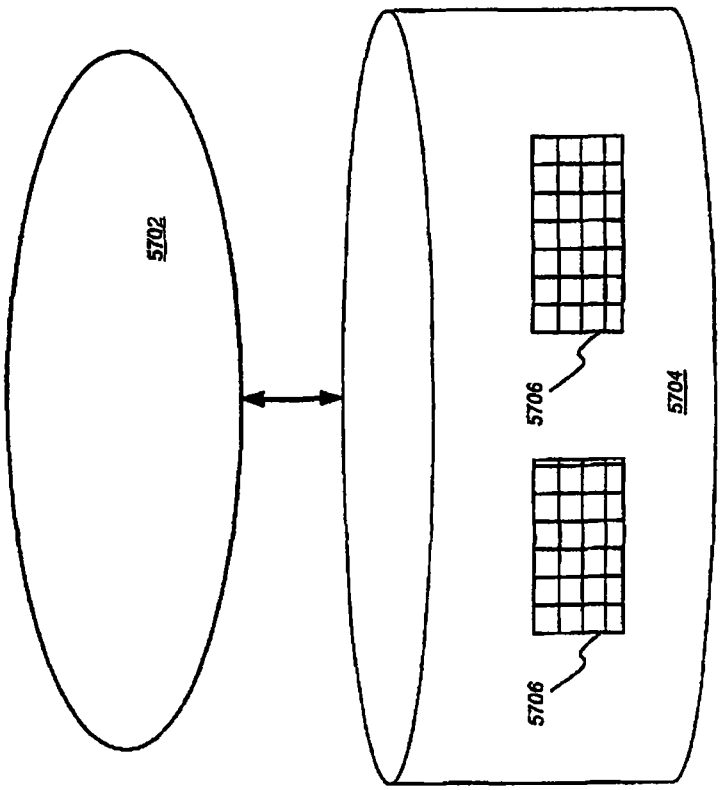


图 9

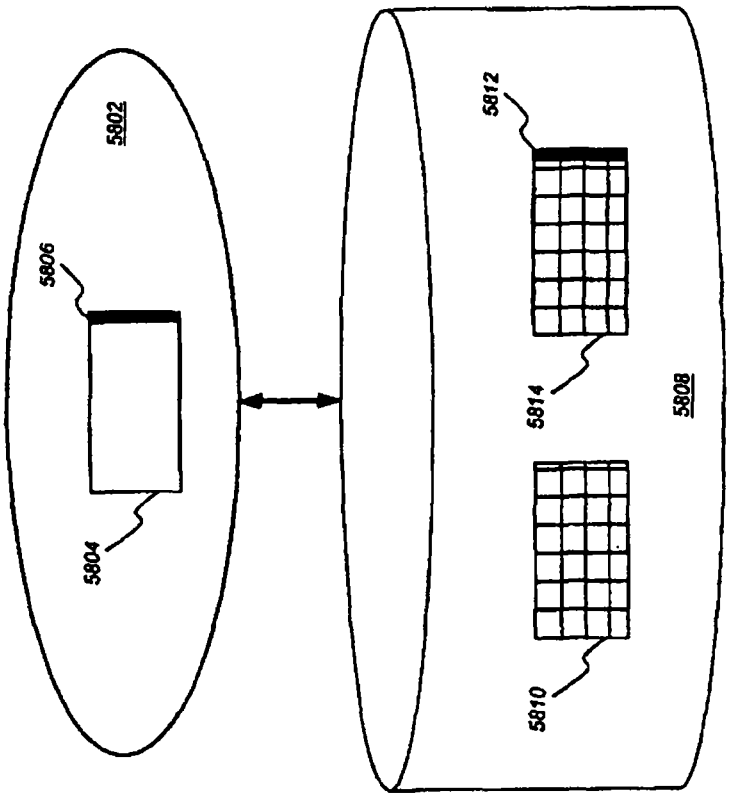


图 10

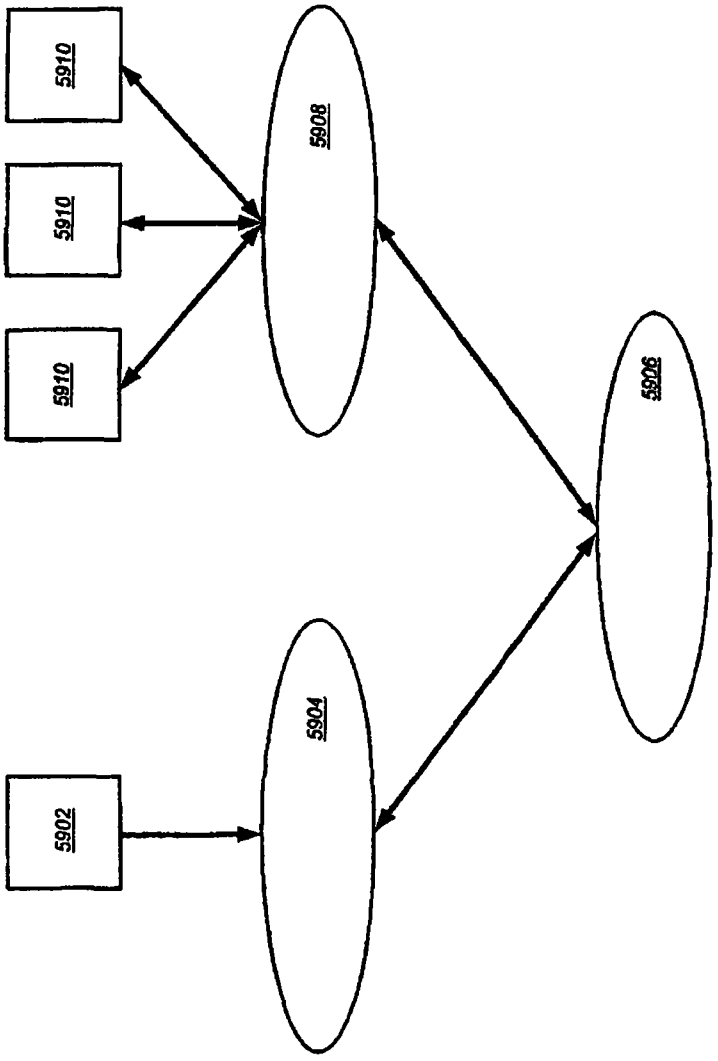


图 11

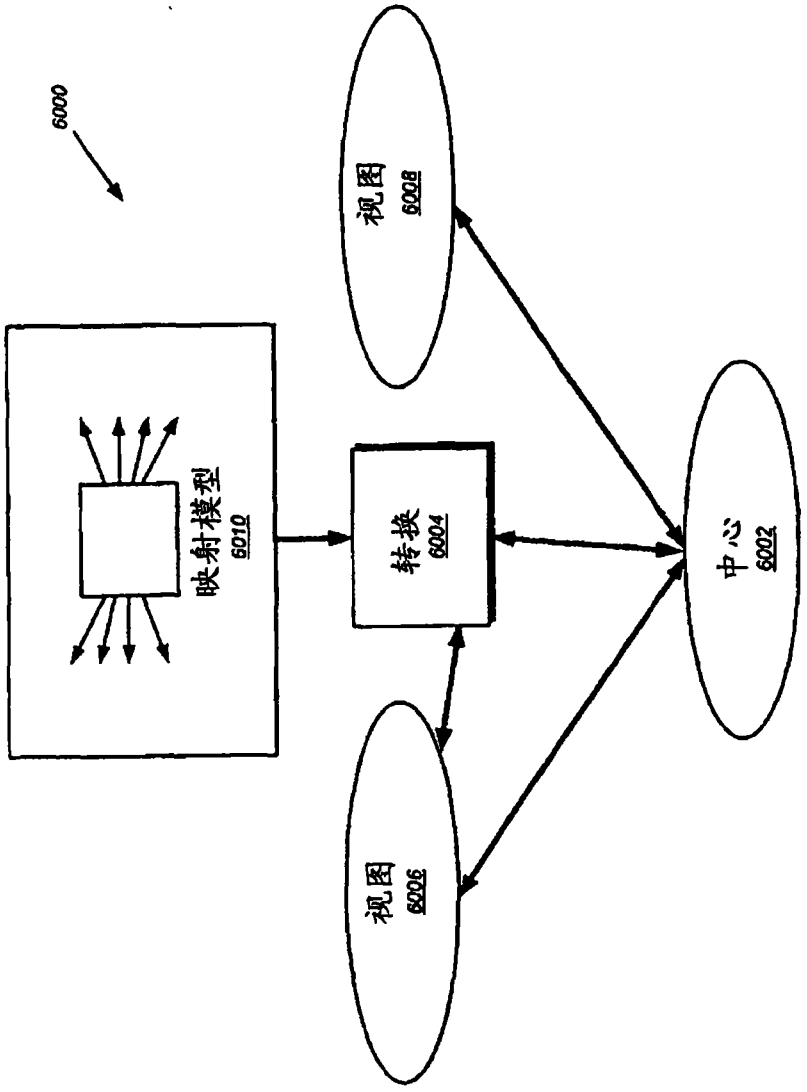


图12

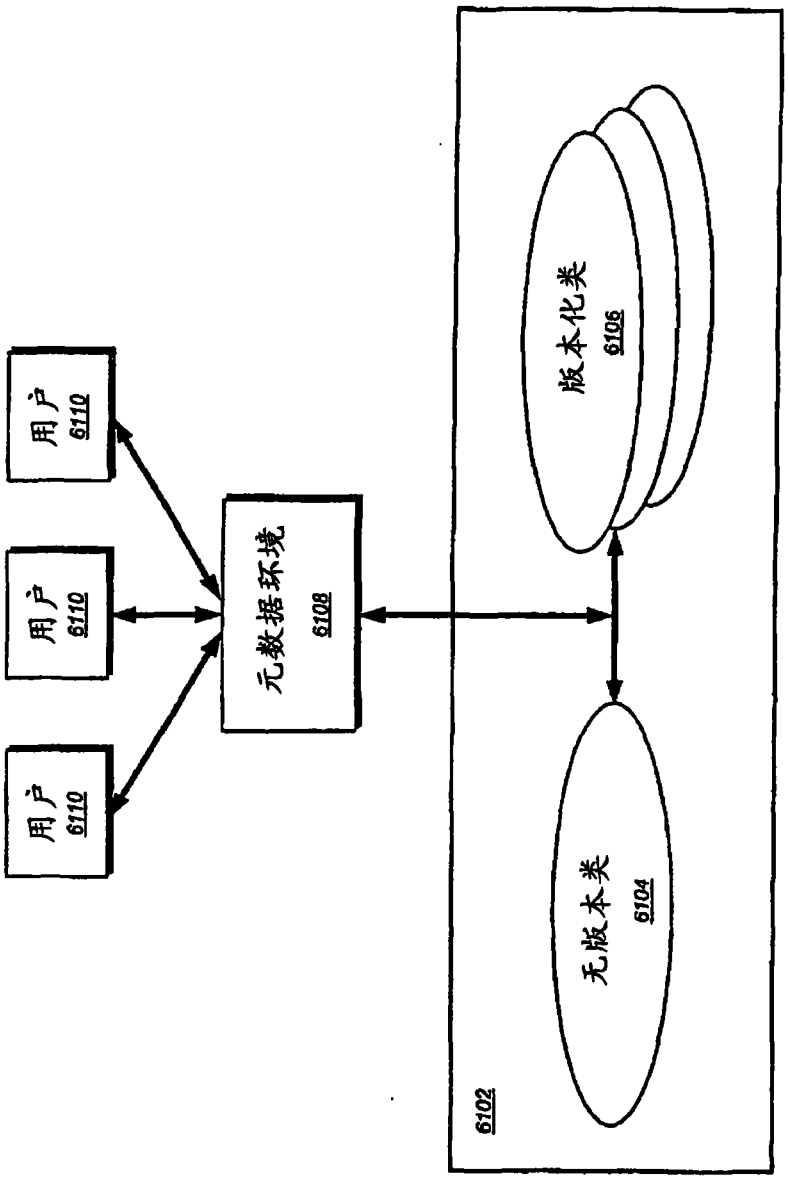


图 13

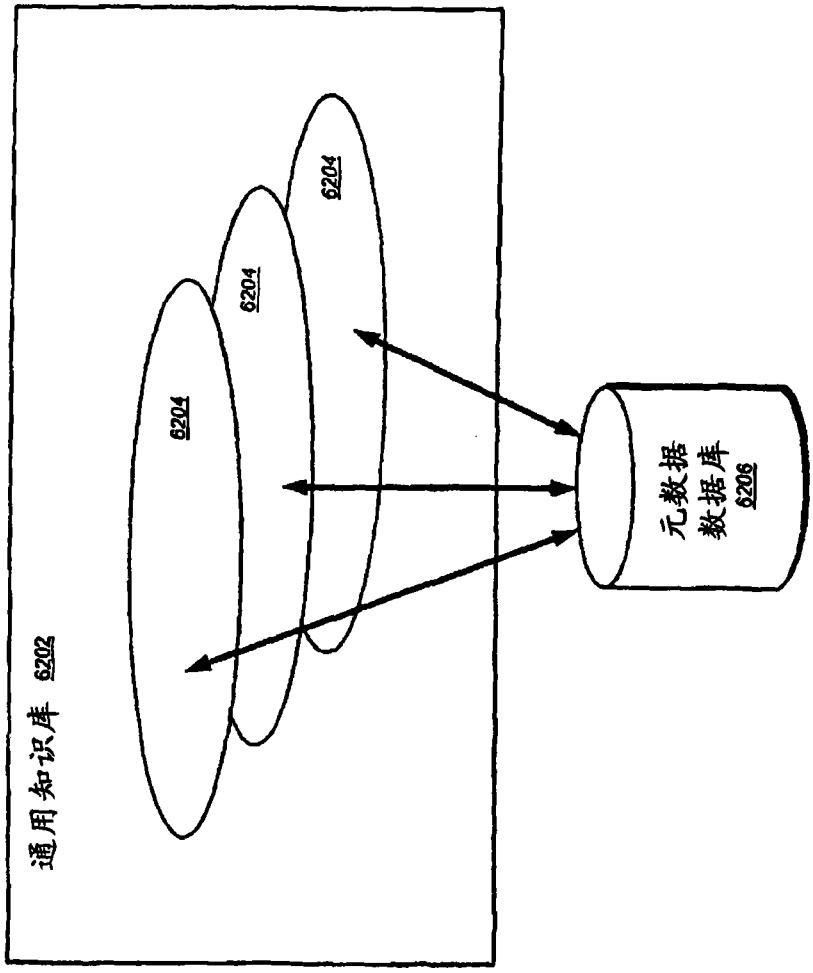


图 14

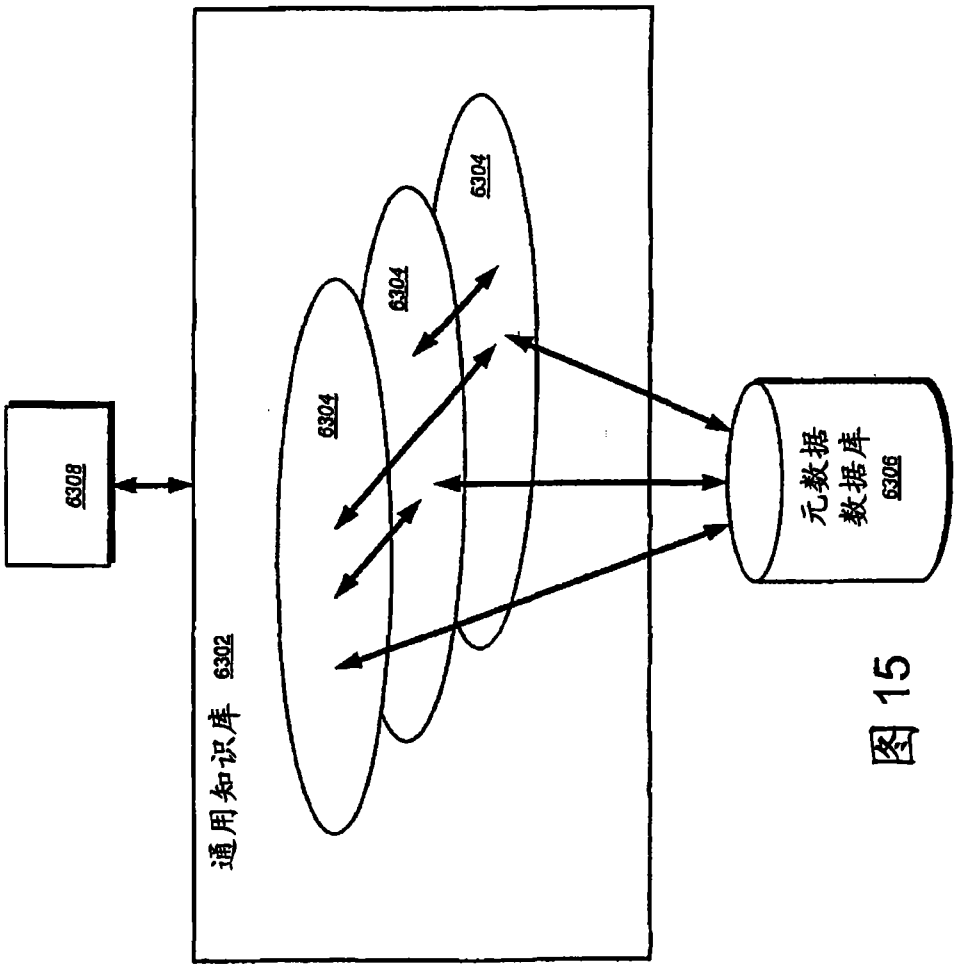


图 15

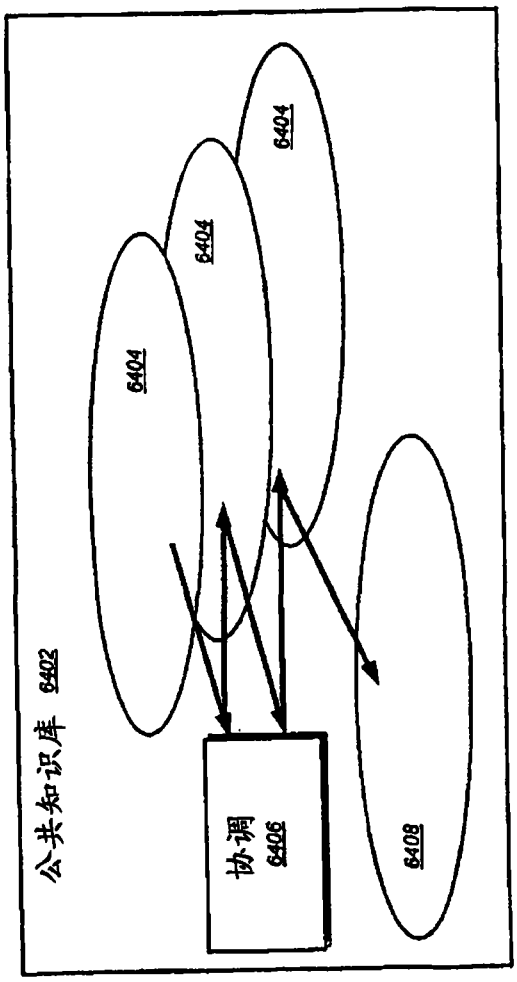


图16A

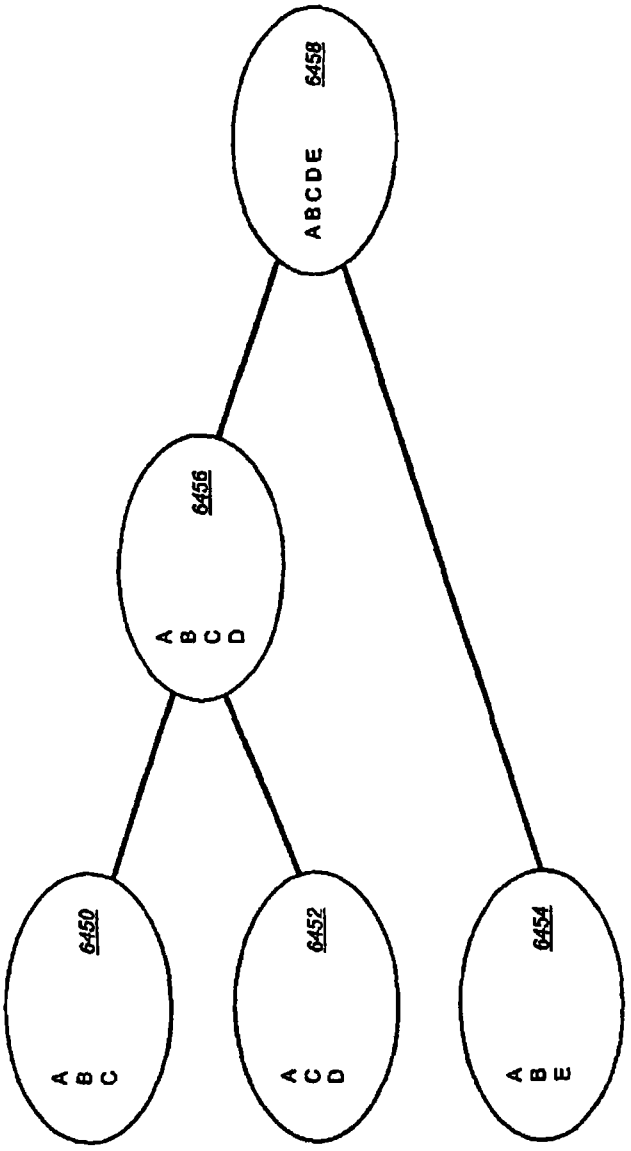


图16B

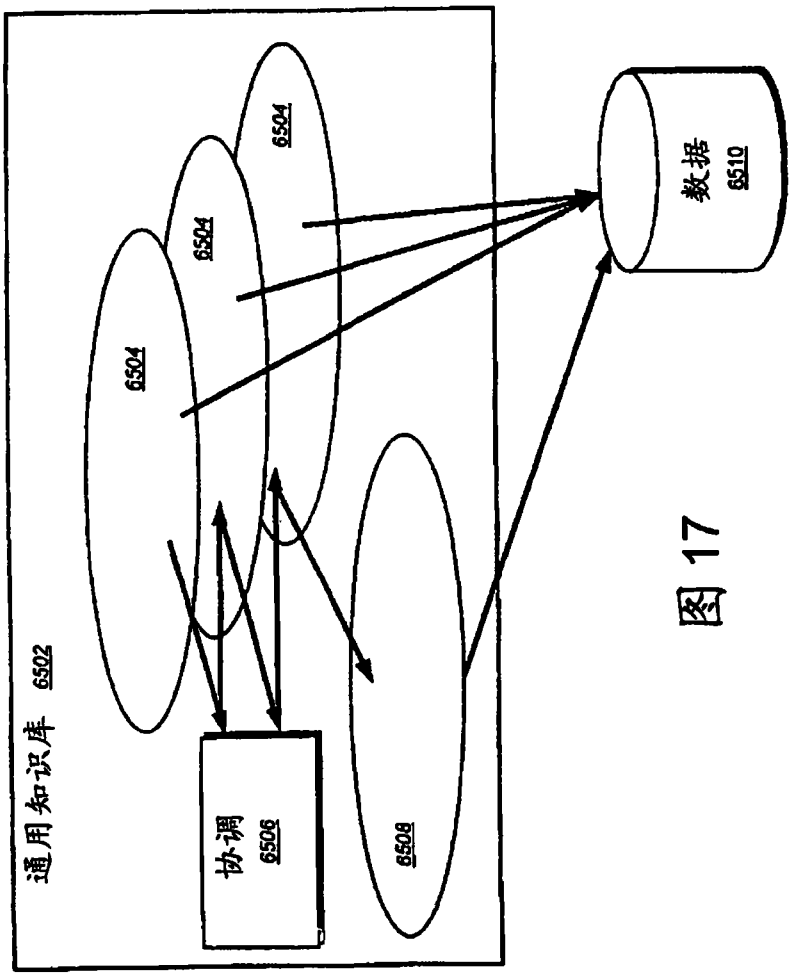


图 17

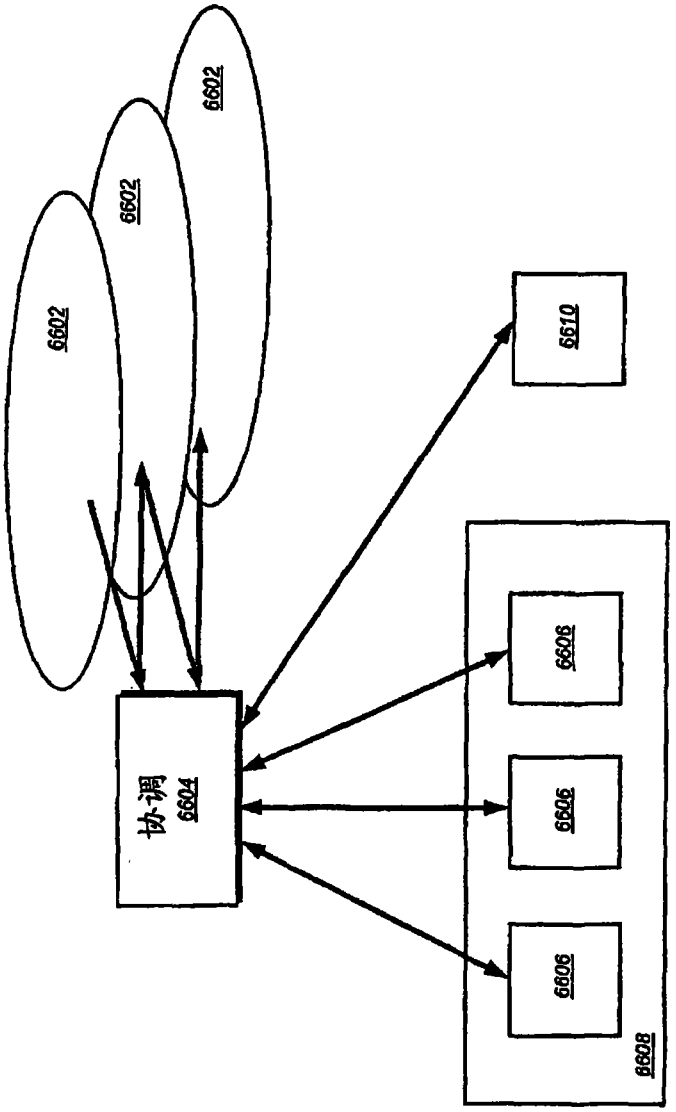


图18

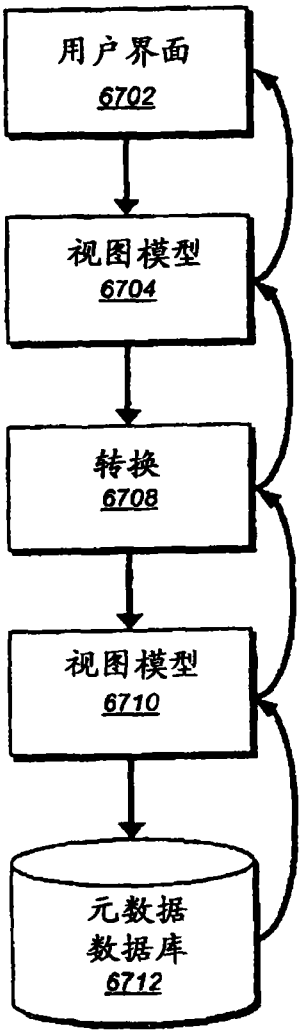


图 19

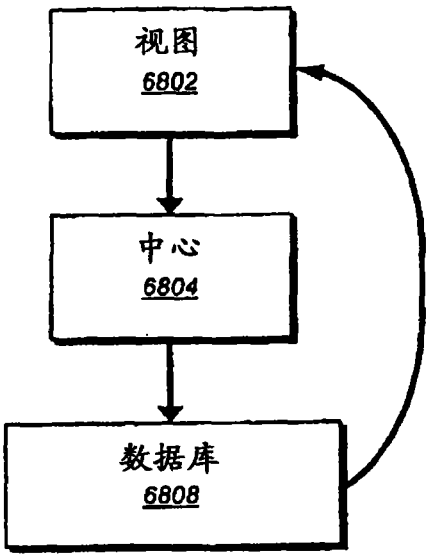


图 20

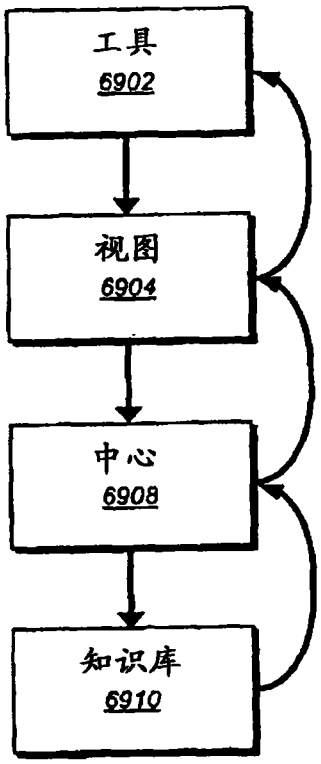


图 21

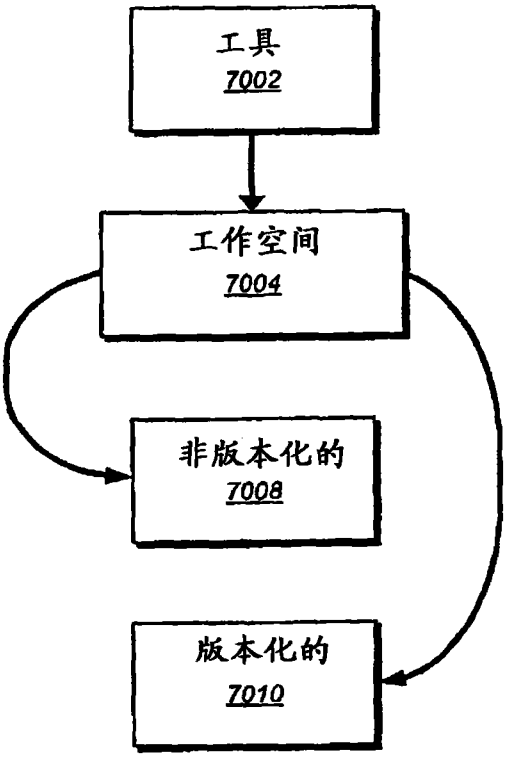


图 22

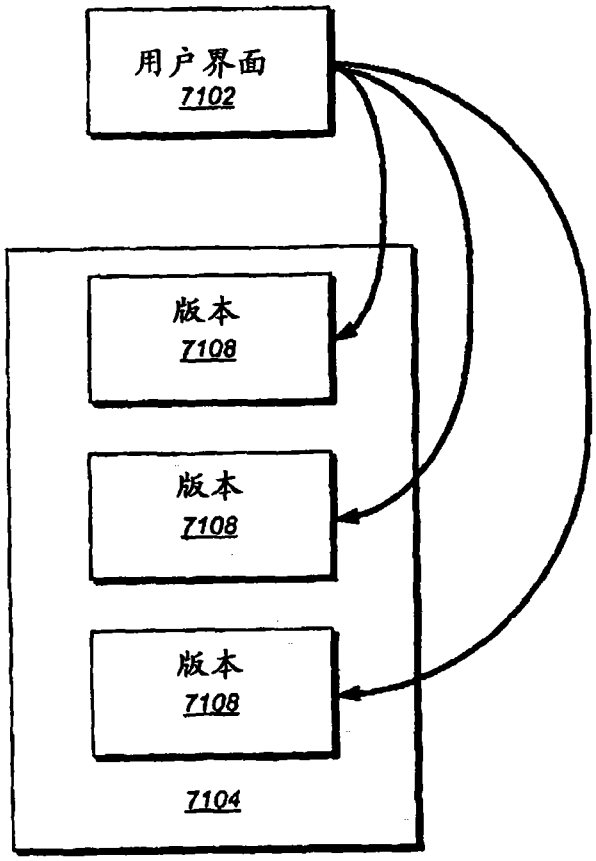


图 23

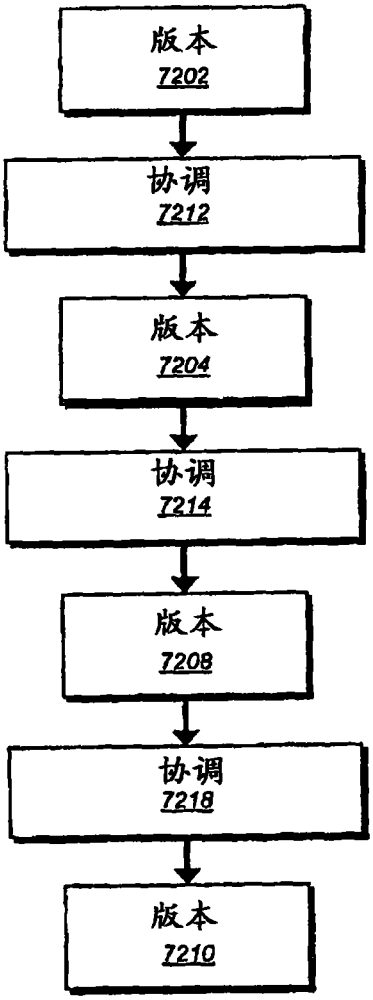


图 24

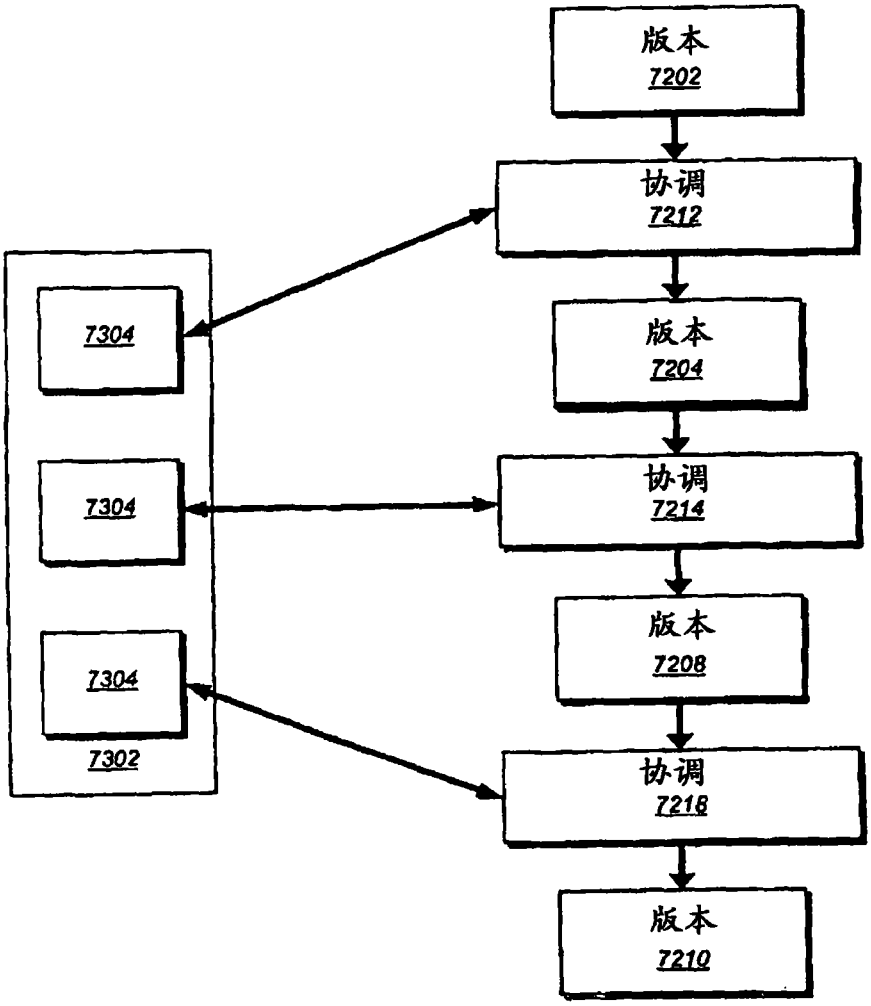

版本
7202
协调
7212
版本
7204
协调
7214
版本
7208
协调
7218
版本
7210
7304
7304
7304
7302

图 25