(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 1821957 B
(45)授权公告日 2014.03.12

(21)申请号 200610005450.5
(22)申请日 2006.01.18
(30)优先权数据
60/654,237 2005.02.18 US
11/230,983 2005.09.20 US
(73)专利权人 微软公司
地址 美国华盛顿州
(72)发明人 B·阿尔巴哈瑞 H·J·M·迈耶 M·E·蒂姆
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 张政权
(51)Int.Cl.
G06F 9/44(2006.01)
G06F 17/30(2006.01)
(56)对比文件
US 5295256 A,1994.03.15, 全文.
CN 1125990 A,1996.07.03, 全文.
WO 9503586 A1,1995.02.02, 全文.
审查员 王亮

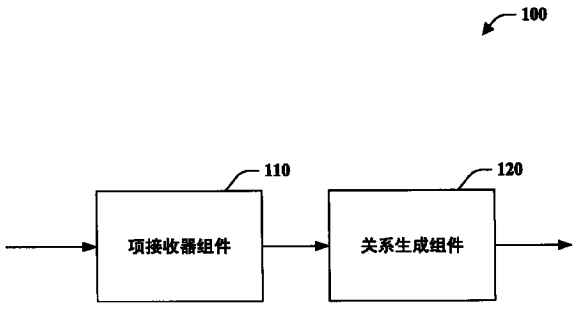
权利要求书1页 说明书14页 附图9页

(54)发明名称

关系建模

(57)摘要

本发明涉及项之间的关系。可以在项本身外部定义项之间的关系，从而提供模块化、灵活且可扩展的系统。例如，可以在包含用于根据指定的关系返回特定元素或值的方法的类中定义项之间的关系。而且，可以扩展编译器或类似的系统来接受简单属性格式的关系表达式，并指引这样的调用至特定的方法。



CN 1821957 B

1. 一种计算机实现的关系系统，所述关系系统包括以下计算机可执行组件：

项接收器组件，用于获取两个或多个程序项；

关系生成组件，用于从所述接收器组件接受项，并生成定义所述项之间的一个或多个关系的构造，其中所述关系生成组件生成所述项之间的关系而不修改项；以及

为关系生成项专用名的影响组件，所述影响组件包括人工智能组件，以根据相关元素的名称来推断关系，并通过接收并提供由外部元数据信息驱动的命名方案来协助所述关系生成组件。

2. 如权利要求 1 所述的系统，其特征在于，所述项是数据类型和 XML 文档之一。

3. 如权利要求 1 所述的系统，其特征在于，所述构造是类。

4. 如权利要求 3 所述的系统，其特征在于，所述类包括根据指定关系检索特定项元素的方法。

5. 如权利要求 4 所述的系统，其特征在于，所述类是静态类。

6. 如权利要求 1 所述的系统，其特征在于，所述一个或多个关系是包括一对一、一对多、多对一和多对多的二元关系。

7. 如权利要求 1 所述的系统，其特征在于，所述一个或多个关系是由对所述项封装导航计算的方法指定的。

8. 如权利要求 1 所述的系统，其特征在于，所述一个或多个关系包括组成、关联和链接之一。

9. 如权利要求 8 所述的系统，其特征在于，所述关联是公值、条件和实体之一。

10. 一种便于数据交互的计算机实现的方法，包括以下计算机可执行动作：

接收两个或多个程序项之间的关系表达式，所述关系用相应编程语言来编写并用静态类来表示，其中静态类提供一个或多个静态方法，所述静态方法允许使用由所述静态类表示的关系来在项之间导航；

定位与所述表达式相关联的项外部的方法；

执行所述方法来使用项外部的外部构造来计算所述项之间的关系使得项之间的关系被定义而不修改项；

使用由所述静态类表示的关系从一组类实例中导航所述项；

为所述关系生成项专用名，其中人工智能组件被用于根据相关元素的名称来推断关系，以及；

接收并提供由外部元数据信息驱动的命名方案。

11. 如权利要求 10 所述的方法，其特征在于，还包括返回结果。

12. 如权利要求 10 所述的方法，其特征在于，接收关系表达式包括接收属性记号中类型之间的关系。

13. 如权利要求 12 所述的方法，其特征在于，定位方法包括将所述表达式从属性记号扩展至方法名。

14. 如权利要求 13 所述的方法，其特征在于，定位方法还包括标识含有一个或多个关系方法的关系类。

15. 如权利要求 10 所述的方法，其特征在于，所述关系包括二元、组成、关联和链接之一。

关系建模

[0001] 相关申请的交叉引用

[0002] 本申请要求于2005年2月18日提交的、名为“OBJECT ORIENTEDRELATIONSHIP MODELING(面向对象关系建模)”的美国专利临时申请第60/654,237号的优先权。该临时申请的整体通过引用包含在此。

背景技术

[0003] 程序设计语言是专门用来为执行任务而将指令传递至计算机或微处理器的形式语言。近年来,面向对象程序设计成为设计员和程序员用来实现计算机系统内的功能的多个惯用且流行的模型之一。面向对象的程序设计是独特的,至少因为其前提是按照对象或事物而不是像其它模型那样按照动作来查看。

[0004] 对象技术的好处源于三个基本原理:封装、多态以及继承。对象隐藏或封装其数据和相关联方法的内部结构。并非展示实现细节,而是对象呈现清晰地表示它们的抽象而无需无关信息的接口。多态比封装更进一步。多态允许为不同的数据类型使用相同的代码,即多种形状一个接口的想法。从而,软件组件可以请求另一组件而不必确切地了解该组件。接收请求的组件解释该请求,并根据其变量和数据来判断如何执行该请求。第三个原理是继承性,它使得开发员能够再次使用之前存在的设计和代码。该能力允许开发员避免从零开始创建所有软件。相反,通过继承,开发员可以导出继承和修改其它类的状态和行为的子类。

[0005] 面向对象的程序设计模型通常由基于类的方法来定义。在该系统中,对象是包括状态和行为两者的实体。对象的状态和行为均由类定义,类标识了特定类型的对象。基于类定义创建的对象被认为是在动态类型中反映的该类的实例。因此,类指定对象可以包含的数据(即,状态)以及该对象可以执行的方法、功能或行为。方法用于通过变更其中包含的数据来修改相关联对象的内部状态。对象中这样的数据和方法的组合通常被称为面向对象程序设计中的封装。封装规定对象的状态仅由与该对象相关联的良好定义的方法改变。当对象的行为限于这样的良好定义的位置和接口时,对象中的改变(例如,代码修改)将对系统中的其它对象和元素产生最小的影响。

发明内容

[0006] 以下提供了本发明的简化概述,以提供对所要求保护的本发明的主题的某些方面的基本理解。该概述不是本发明的广泛概观。它既不旨在标识本发明的关键或重要的元素,也不描绘本发明的范围。该概述的唯一目的是以简化的形式呈现本发明的某些概念,作为之后呈现的更详细描述的序言。

[0007] 简言之,本发明涉及项和/或其元素之间的关系的表达式。更具体地,关系被作为一级概念。根据本发明的一方面,关系可以由项外部的构造,诸如类来表示,该构造提供计算和/或导航关系的机制或方法。根据本发明的另一方面,可以利用数据类型属性记号来调用关系方法。

[0008] 本发明的各方面至少因为它们提供了用于与项之间的关系交互的可扩展且易于使用的系统和方法而是有益的。通过使得关系成为一级编程对象，可以在现有项之间创建新的关系，而不必修改项。这是有价值的，至少因为它允许定义可能不受程序员控制的某些或全部项或元素之间的关系，或者在不可能修改这样的元素来反映新关系的情况下允许定义这样的关系。而且，可以通过属性表示容易地实现类方法的调用，并将其映射至实际方法记号。

[0009] 为了达到前述和相关的目的，此处结合以下描述和附图来描述所要求保护的本发明的某些说明性方面。这些方面指示了可实现本发明的各种方法，所有这些方法旨在落入所要求保护的本发明的范围之内。当结合附图考虑阅读以下详细描述时，其它优点和新颖的特征将变得明显。

附图说明

[0010] 图 1 是关系系统的框图。

[0011] 图 2 是示例性关系系统的框图。

[0012] 图 3 是包含影响组件的关系系统的框图。

[0013] 图 4 是编译系统的框图。

[0014] 图 5 是便于与数据交互的接口系统的框图。

[0015] 图 6 是集成开发系统或环境的框图。

[0016] 图 7 是定义关系的方法的流程图。

[0017] 图 8 是编译方法的流程图。

[0018] 图 9 是数据交互方法的流程图。

[0019] 图 10 是协助程序开发的方法的流程图。

[0020] 图 11 是示例性编译环境的示意性框图。

[0021] 图 12 是示出合适的操作环境的示意性框图。

[0022] 图 13 是示例计算环境的示意性框图。

具体实施方式

[0023] 现在参考附图描述本发明的各方面，在所有附图中，同样的标号指的是相同或对应的元素。然而，应该理解，附图和关于附图的详细描述不旨在将所要求保护的本发明限于所揭示的具体形式。而是相反，本发明旨在覆盖落入所要求保护的本发明的精神和范围内的所有修改、等效方案和替换方案。

[0024] 如在本申请中所用的，术语“组件”和“系统”等指的是计算机相关的实体，它们或者是硬件、硬件和软件的组合、软件或者是执行中的软件。例如，组件可以是，但不限于，运行在处理器上的进程、处理器、对象、可执行代码、执行的线程、程序和 / 或计算机。作为说明，运行在服务器上的应用程序和服务器本身都可以是组件。一个或多个组件可以驻留在执行中的进程和 / 或线程内，且组件可以位于一台计算机上和 / 或分布在两台或多台计算机之间。

[0025] 术语“示例性”在此处用来指的是作为示例、实例或说明。此处被描述为“示例性”的任何方面或设计不必被解释为与其它方面或设计相比是较佳的或更有利的。而且，此处

提供了各种示例性代码片段。应该理解，这些示例是为说明清楚的目的而提供的，并且应理解，这些示例不意味着将所揭示的本发明的范围限于在所要求保护的本发明的各方面描述中所使用的语言、体系结构和 / 或特征。

[0026] 可以将基于人工智能的系统（例如，显式和 / 或隐式训练的分类器、基于知识的系统 ...）用于执行根据如下所述的本发明的一个或多个方面的推断和 / 或概率判断和 / 或基于统计的判断。如此处所用的，术语“推断”一般指的是从通过事件和 / 或数据捕捉到的一组观察值中推出或推断系统、环境和 / 或用户的状态的过程。推断例如可以被用来标识特定的上下文或动作，或者可以生成状态的概率分布。推断可以是概率性的，即，基于对数据和事件的考虑对所关注的状态的概率分布的计算。推断也可以指的是用于从一组事件和 / 或数据中组成更高级事件的技术。这样的推断导致从一组观察到的事件和 / 或存储的事件数据中构造出新的事件或动作，而不论原先的事件是否在时间上紧密相关，也不论原先的事件和数据是来自一个还是若干个事件和数据源。可以将各种分类方案和 / 或系统（例如，支持矢量机、神经网络、专家系统、贝叶斯信任网络、模糊逻辑、数据融合引擎 ...）用于执行关于本发明的自动和 / 或推断出的动作。

[0027] 另外，所揭示的本发明可以被实现为系统、方法、装置或制品，它们使用生产软件、固件、硬件或其任何组合的标准程序设计和 / 或工程技术来控制基于计算机或处理器的设备，以实现此处所详细描述的各方面。如此所用的术语“制品”（或者“计算机程序产品”）旨在包括可从任何计算机可读设备、载波或介质来访问的计算机程序。例如，计算机可读介质可以包括，但不限于，磁性存储设备（例如，硬盘、软盘、磁条 ...）、光盘（例如，高密度盘 (CD)、数字多功能盘 (DVD)...）、智能卡、以及闪存设备（例如，卡、棒、跳转驱动器 (jump drive)...）。另外，可以理解，载波可以用来承载诸如在发送和接收电子邮件或访问诸如因特网或局域网 (LAN) 等网络时使用的计算机可读电子数据。当然，本领域的技术人员可以认识到，可以对该配置进行各种修改，而不背离所要求保护的本发明的范围和精神。

[0028] 最初转向图 1，根据本发明的一方面示出关系系统 100。关系系统 100 可以包括项接收器组件 110 和关系生成系统 120。项接收器组件 110 接收、检索或获取项和 / 或其元素。这些项可以包括，但不限于，数据类型、对象、网页以及 XML 文档。关系生成组件 120 从项接收器组件 110 中接收、检索或以获取多个项。关系生成组件 120 分析项并指定和 / 或定义项之间的关系。例如，关系可以在诸如类或更具体地静态类等程序构造中定义。类可以包括封装用于根据特定关系检索各种项或其元素的集合的功能的类外部的方法或对方法的引用。

[0029] 应该注意，关系生成系统 100 以一种以上方式有益。例如，系统 100 支持诸如项和它们之间的关系或链接等概念的分离，以及开发中的模块性。而且，项和关系的分离提供灵活性及可扩展性，因为项不可能总是可供修改或者修改它们可能是不可行的。例如，考虑这样的情形，其中存在以几十年前的某种传统格式定义的人和关于每个人的数据或属性的集合。在此之后，修改该集合来为每个人添加电话号码可能是不可能或不可行的。现在，可以生成分离的关系构造以将人集合与分离的手机集合相关联。

[0030] 图 2 示出了示例性关系交互系统 200。提供系统 200 以便于描述和讨论本发明的各方面。系统 200 包括一个或多个关系方法 210 和两个项 A220 和 B230。如图所示，项 A220 与项 B230 之间的关系不被定义在项属性内，也不被定义为项属性。相反，该关系被定义在

项外部作为一级概念。关系方法 210 封装该计算来与项和项元素交互。

[0031] 作为示例而非限制，考虑对象至关系的映射情形。具体地，在该情形中期望针对数据库与对象的关系来进行编程。概念上包含类型 T 的对象的数据库表可以在程序设计语言中使用诸如 C# 中的 IENumerable<T>（枚举）等集合类型来表示，其中 T 是带有映射至表字段的属性的类。数据库关系然后可以在该程序设计语言中使用静态类表示。该类提供通过封装连接条件来导航这些关系的静态方法。例如，项 A220 可以是对应于顾客 (Customer) 表的对象、类型或类，而项 B230 可以是对应于定单 (Order) 表的对象、类型或类。例如：

[0032] public class Customer{...}

[0033] public class Order{...}

[0034] 可以如下以诸如静态类等编程构造与项 A 220 和 B 230 分离地定义关系 210：

[0035] public static class OrderRelationship{

[0036] public static Customer GetCustomerGivenOrder(Order order)；

[0037] public static IEnumerable<Order>GetOrdersGivenCustomer(Customer, customer)；

[0038] public static IEnumerable<Customer>GetCustomersGivenOrders(IEnumerable<Order>orders)；

[0039] public static IEnumerable<Order>GetOrdersGivenCustomers(IEnumerable<Customer>customers)；

[0040] }

[0041] 此处，类方法提供一种根据可能存在于顾客与定单之间的多个关系来导航数据存储的机制。此处，由类和类方法封装二元关系。二元关系可以包括一对一、一对多、多对一和多对多关系。第一方法“GetCustomerGivenOrder（给定定单获取顾客）”捕捉给定一特定定单来检索一顾客的一对一关系。第二方法“GetOrdersGivenCustomer（给定一顾客获取多个定单）”是多对一关系。此处，检索与特定顾客相关联的多个定单。第三方法和第四方法“GetCustomersGivenOrders（给定多个定单获取多个顾客）”及“GetOrdersGivenCustomers（给定多个顾客获取多个定单）”是多对多关系。具体地，第一方法检索与指定定单相关联的顾客集合。给定一组定单，第四方法可以检索相关的顾客。

[0042] 前述和后继的示例不旨在限制所附权利要求书的范围。本发明的各方面可适用于其中项之间存在关系或导航的任何情形。例如，考虑链接文档的情形。项 A220 和项 B230 可以是各种格式中的任何一种的电子文档，这些格式包括但不限于超文本和 XML。并非包括诸如从项 A220 至项 B230 的超链接等链接。文档之间的关系 210 可以被定义为在其外部。这使得可以定义关系，而不必修改文档之一。

[0043] 也构想了除二元关系和链接之外的其它关系，且它们也属于权利要求书的范围内，这些其它关系包括但不限于组成和关联。如果一个项被嵌套在另一项内，则该项与另一项具有组成关系。这样，项或实体可以组成任何其它项或实体。下表示出消息 - 参与者组成关系：

[0044]

Id	**主题**	**参与者**	
1	Hey!	**ID**	**电子地址**
		1	Jili
		2	Michael
2	Yay!	**ID**	**电子地址**
		3	Jili
		4	Ben

[0045] 表 1

[0046] 根据关联，存在若干不同的类型，包括引用、公值、条件和实体。引用关联可以对应于主键－外键关系。以下示例示出了顾客－定单引用关联关系，其中表 2 对应于顾客而表 3 对应于定单：

[0047]

Id	名字
1	Fred
2	Wilma

[0048] 表 2

[0049]

Id	顾客	...
1	(1)	...
2	(2)	...
3	(1)	...

[0050] 表 3

[0051] 公值关联是其中跨两个或更多项共享公值的关系。例如，下表示出在人（表 4）和器材（表 5）之间所示的音乐家公值关联：

[0052]

Id	名字	器材
1	Fred	钢琴
2	John	吉他
3	Wilma	钢琴

[0053] 表 4

[0054]

Id	名字
1	吉他
2	长笛
3	钢琴

[0055] 表 5

[0056] 条件关联是由查询准则表达的关系。下表提供了联系人－文档条件关联：

[0057]

Id	电子邮件地址

1	元素 0 :benja@xyz.com 元素 1 :mbtyalor@xyz.com
2	元素 0 :jili@xyz.com

[0058] 表 6

[0059]

Id	作者
1	mbtyalor@xyz.com
2	jili@xyz.com
3	benja@xyz.com

[0060] 表 7

[0061] 实体关联围绕一个项或实体具有 n 个端点，它担当通过其它类型关系至其它实体的中心。链接关系仅是含有一个中心以及两个基于引用的端点的实体关联的特殊情况。以下的示例性表示出了雇佣关系实体关联，其中表 8 对应于雇佣关系，表 9 对应于人，表 10 对应于雇主 ：

[0062]

Id	受雇日期	雇员 Id	雇主 Id
1	01/01/01	2	1
2	01/01/02	2	2
3	01/01/03	1	2

[0063] 表 8

[0064]

Id	名字
1	Fred
2	Wilma
3	Barney

[0065] 表 9

[0066]

Id	名字
1	Zoo
2	Boo

[0067] 表 10

[0068] 图 3 示出了根据本发明一方面的关系系统 300。类似于图 1 的系统 100，系统 300 包括项接收器组件 110 和关系生成组件 120。如上所述，接收器组件 110 可以接收和 / 或检索多个元素，仅举几例，诸如数据对象、网页或 XML 文档。关系生成组件 120 可以从接收器组件 110 接收和 / 或检索元素。关系生成组件 120 可以定义或指定项之间的关系或链接。而且，关系生成组件 120 可以提供用于响应于指定的关系来检索特定项或其元素的方法或其引用。同样类似于系统 100，系统 300 可以定义类中的关系，而类可以包括用于计算关系的方法。然而，系统 300 也可以包括影响组件 310。影响组件 310 是用于影响关系命名的机制。根据本发明的一方面，影响组件 310 可以包括，或者可以被通信地耦合至，试探或人工智能组件、方法或机制，以基于相关元素的名称来推断或推理出关系的名称。例如，如上例所述，两个项“Customer”和“Order”可以含有名为“GetCutomerGivenOrder”的关系方法。除此之外或者作为替代，影响组件 310 可以例如通过接收和 / 或提供由外部元数据信息驱动的命名方案来协助关系生成组件 120。

[0069] 如上所述，关系可以是二元关系或普通的n元关系（例如，中心-辐轴关系）等等。二元关系可以是如在实体关系讨论中所用的一对一、一对多、多对一或多对多关系。类型S和T之间的关系可以使用静态类来建模。考虑捕捉若干二元关系的类的以下示例：

[0070] publicstaticclassRelationshipName_Relationship{

[0071] publicstaticIEnumerable<T>GetTsGivenS(S,s)；

[0072] publicstaticSGetSGivenT(Tt)；

[0073] publicstaticIEnumerable<S>GetSsGivenTs(IEnumerable<T>ts)；

[0074] publicstaticIEnumerable<T>GetTsGivenSs(IEnumerable<S>ss)；

[0075] }

[0076] 在一对一关系的情况下，GetTsGivenS（给定S获取T）方法返回T的单个实例，而在多对多关系的情况下，GetSGivenT（给定T获取S）方法返回诸如IEnumerable（枚举）等集合类型。

[0077] 关系类的名称、静态方法和形参名称可以通过多种方式导出，包括但不限于，使用基于类型名称的试探和使用由外部元数据信息驱动的命名方案。同样如之前所提供的，以下对“Customer”和“Order”之间的关系建模：

[0078] public class Customer{...}

[0079] public class Order{...}

[0080] public static class OrderRelationship{

[0081] public static Customer GetCustomerGivenOrder(Order order)；

[0082] public static IEnumerable<Order>GetOrdersGivenCustomer(Customer, customer)；

[0083] public static IEnumerable<Customer>GetCustomerGivenOrders(IEnumerable<Order>orders)；

[0084] public static IEnumerable<Order>GetOrdersGivenCustomers(IEnumerable<Customer>customers)；

[0085] }

[0086] 给定表示顾客表中一组顾客的变量，如：

[0087] IEnumerable<Customer>customers = ...；

[0088] 可以如下使用以下静态方法来检索与该组顾客有关的一组定单：

[0089] IEnumerable<Order>orders = OrerRelationship.GetOrderGivenCustomer(customers)；

[0090] 该静态方法调用封装了连接（join）条件：

[0091] SELECT（字段）FROM Customer JOIN Order ON（条件）

[0092] n元关系（此处也被称为实体关联）是例如一组特定类型的二元关系与其它类型的二元关系。一个示例是中心和辐轴关系。这样的关系的设计模式如下：

[0093] public static class RelationshipName_Relationship{

[0094] public static S1GetSpokel GivenHub(hub)；

[0095] public static IEnumerable<Hub>GetHubsGivenSpokel(spokel)；

[0096] public static S2GetSpoke2GivenHub(hub)；

[0097] public static IEnumerable<Hub>GetHubsGivenSpoke2(spoke2)；

[0098] public static IEnumerable<S1>GetSpoke1 GivenHub(IEnumerable<Hub>hubs)；

[0099] public static IEnumerable<Hub>GetHubsGivenSpoke1(IEnumerable<S1>spoke1s)；

[0100] public static IEnumerable<S2>GetSpoke2GivenHub(IEnumerable<Hub>hubs)；

[0101] public static IEnumerable<Hub>GetHubsGivenSpoke2(IEnumerable<S2>spoke2s)；

[0102] }

[0103] 技术上，可以区分关系各方上的基数，但是为简明起见，假定中心和辐轴之间的所有个别二元关系都是一对多关系。

[0104] 与简单的二元关系情况一样，关系类和静态方法的名称可以通过多种方式导出，范围从基于类型名称的试探到由外部元数据信息驱动的命名方案。以下示例对“Employment（雇佣关系）”、“Person（个人）”和“Organization（组织）”之间的关系建模。在“Employment”和“Person”以及“Employment”和“Organization”之间存在两个一对多二元关系，“Employment”用作中心。

[0105] public class Person{..}

[0106] public class Organization{..}

[0107] public class Employment{..}

[0108] public static class EmploymentRelationship{

[0109] public static IEnumerable<Employment>GetEmploymentsGivenEmployee(Person

[0110] employee)；

[0111] public static Person GetEmployeeGivenEmployment(Employmentemployment)；

[0112] public static IEnumerable<Employment>GetEmploymentsGivenEmployer(Organization

[0113] employer)；

[0114] public static OrganizationGetEmployerGivenEmployment(Employmentemployment)；

[0115] public static IEnumerable<Employment>GetEmploymentGivenEmployee

[0116] (IEnumerable<Person>employees)；

[0117] public static IEnumerable<Person>GetEmployeeGivenEmployment

[0118] (IEnumerable<Employment>employments)；

[0119] public static IEnumerable<Employment>GetEmploymentGivenEmployer

[0120] (IEnumerable<Organization>employers)；

[0121] public static IEnumerable<Organization>GetEmployerGivenEmployment

[0122] (IEnumerable<Employment>employments)；

[0123] }

[0124] 给定表示个人表中的一组人的变量，如：

[0125] IEnumerable<Person>people = ... ;

[0126] 以下静态方法用于获取关于该组人的一组雇佣关系：

[0127] IEnumerable<Employment>employments =

[0128] EmploymentRelationship.GetEmploymentGivenEmployee(people) ;

[0129] 以下静态方法用于获取关于该组雇佣关系的一组组织：

[0130] IEnumerable<Organization>employers =

[0131] EmploymentRelationship.GetEmployerGivenEmployment(employments) ;

[0132] 后一种静态方法调用封装连接条件：

[0133] SELECT(字段)FROMPersonJOINEmploymentON(条件)JOINOrganizationON(条件)

[0134] 类型或类关系可以被建模为与类型属性的关系。例如，考虑包含类型个人(Person)和类型组织(Organization)的情形，其中组织雇佣个人，而个人被组织雇佣。这在诸如C#等面向对象语言中如下表示：

[0135] publicclassPerson{

[0136] public IEnumerable<Employment>Employments{get ;}

[0137] }

[0138] public class Organization{

[0139] public IEnumerable<Employment>Employments{get ;}

[0140] 然而，使用分离的类建模优于使用属性对关系建模。例如，可以在现有类型之间创建新的关系，而无需修改这些类型。在上述使用属性的情形中，“Person”和“Organization”类型依赖于“Employment”类型。而且，使用属性对关系建模仅允许从实例导航，而从静态类对关系建模允许从实例的集合导航。

[0141] 尽管使用静态方法将关系表示为一级概念的设计模式给予程序员强大的表达能力，但它在句法上是冗长的。而且，这些关系看上去不相同，从而与按照属性建模的关系相比较难发现。

[0142] 图4示出了根据本发明的一方面，在句法上统一访问关系的方式的编译系统400。系统400包括表达式接收器组件410、代码生成组件420和元数据组件430。接收器组件410接收含有项之间的关系，包括但不限于数据类型的程序表达式。该程序表达式可以是如同关系是属性一般指定的简化表达式。例如，给定表示顾客表中的一组顾客的变量：“IEnumerable<Customer>customers = ... ;”，以下的句法可以用于获取与该组顾客相关的一组定单：“IEnumerable<Order>orders = customer.Orders.”。代码生成组件接收该表达式，并从该表达式生成更详细的代码或对方法的调用，诸如：“OrderRelationship.GetOrderGivenCustomers(customers)”。代码生成组件420通过元数据组件430启用该功能。元数据组件430可以检索或接收关于类的元数据，并将其提供给代码生成组件420，以允许从简化的表达式到实际的或更详细的表达式的映射。元数据可以指定“OrderRelationship.GetOrderGivenCustomers(customers)”映射至“customer.Order”。根据本发明的一方面，这样的元数据可以在定义元素之间关系的类中提供。或者，元数据可以由编译系统和/或代码生成组件420利用的某些外部文件或方案来提供。

[0143] 转向图5，示出接口系统500以便于数据交互。接口系统500包括通信上耦合

的导航接口组件 510 和数据接口组件 520。作为示例，数据接口组件 520 可以实现可由导航接口组件 510 调用或执行的方法，反之亦然。导航接口组件 510 可以接收关系表达式。接口组件 510 可以接收经缩写的或全长的表达式，分别诸如“customer.Order”或“OrderRelationship.GetOrderGivenCustomers(customers)”。当提供缩写时，导航接口组件 510 可以将该表达式转换为全长的表达式。然后可以将该表达式从导航接口组件 510 发送至数据接口组件 520。数据接口组件 520 可以提供该表达式，用于在一个或多个项上执行。如果检索到数据，那么数据接口组件 520 可以将结果传回导航接口组件 510。从而，导航接口组件 510 和数据接口组件 520 可以对应于应用程序编程接口 (API)。

[0144] 图 6 示出了根据本发明的一方面的集成开发环境或系统 600。系统 600 可以包括编辑器组件 610 和程序协助组件 620。编辑器组件 610 是专门用于编辑和 / 或开发计算机源代码的文本编辑器。具体地，编辑器组件 610 可以接收关系的指定。文本编辑器在通信上耦合至程序协助组件 620。协助组件 620 可以提供编码协助，包括提示、格式化、着色、工具栏以及出错指示或警告等等。例如，响应于接收到诸如点等项或触发器，程序协助组件 620 可以提供和 / 或使得文本编辑器组件 610 显示用于完成语句的提示。可以就此处提供的关系进行提示。例如，当接收到“customer”，则可以建议“Orders”以提供完整语句“customer.Orders”，表示检索“customer”的所有 order。从而，可以做出表现为项属性，而对应于分离的关系方法的提示或建议。

[0145] 对于若干组件之间的交互描述了前述系统。应该理解，这样的系统和组件可以包括那些组件或其中指定的子组件、指定的组件或子组件中的某一些和 / 或另外的组件。子组件也可以被实现为通信上耦合至其它组件而非包含在父组件中的组件。而且，一个或多个组件和 / 或子组件可以被组合成提供聚合功能的单个组件。组件也可以与为简洁起见此处没有具体描述但本领域技术人员已知的一个或多个其它组件交互。

[0146] 而且，可以理解，以上揭示的系统和以下方法的各个部分可以包括或包含人工智能或基于知识或规则的组件、子组件、进程、装置、方法或机制（例如，支持矢量机、神经网络、专家系统、贝叶斯信任网络、模糊逻辑、数据融合引擎、分类器 ...）。这样的组件及其它组件可以自动化由此执行的某些机制或进程，从而使得该系统和方法的各部分更具有适应性并更高效和智能。作为示例而非限制，影响组件 310 可以使用这样的方法或机制来推断和影响生成的关系名。

[0147] 考虑上述示例性系统，参考图 7-10 的流程图能更好地理解可以根据所揭示的本发明实现的方法。尽管为说明的简单起见，方法被示出和描述为一连串框，但是可以理解和领会，所要求保护的本发明不受框的顺序限制，因为某些框可以按与此处所示和描述不同的顺序和 / 或与其它框同时发生。而且，不是所有示出的框对实现之后描述的方法都是必需的。

[0148] 另外，还应该理解，之后揭示并贯穿本说明书的方法能够被存储在制品上，以便于向计算机传送和传输这样的方法。如此处所用的术语制品旨在包含可从任何计算机可读设备、载波或介质访问的计算机程序。

[0149] 转向图 7，示出了定义关系的方法 700。在 710 处，接收项。这可以包括但不限于，诸如数据类型和文档（例如，XML、文字处理、HTTP...）等程序项。在 720 处，生成诸如类等定义多个项之间的关系的构造。例如，类可以是面向对象的静态类。而且，类可以包括封装

计算和 / 或导航项或元素关系的静态方法。关系可以是二元或 n 元的，以及其它关系。另外，可由与要相关的项相关联的名称或其它元数据影响关系或方法的名称。应该理解，尽管可以自动生成类，但是方法 400 也可以被手动实现，例如通过在 IDE（集成开发环境）或类似系统的帮助下，或在没有其帮助的情况下在面向对象程序设计语言中标识诸如数据类型等各种元素，一级手动指定定义元素之间的关系来实现。

[0150] 图 8 示出了根据本发明的一方面的编译方法 800。在 810 处，接收到关系表达式。关系表达式可以被表示为类的属性，诸如“cutomer.Orders”。在 820 处，编译程序或其它系统可以接收表达式并生成或扩展到与类中定义的例如指定项之间的关系的方法相关联的代码。为了便于这样的代码生成，编译器可以利用例如与关系类相关联的元数据，来确定表达式和更详细的方法调用之间的映射。从而，编译器调用作为在诸如类型等项上直接定义的属性的静态方法。

[0151] 转向图 9，示出用于与数据交互的方法 900。在参考标号 910 处，接收到关系表达式。关系表达式标识两个或多个项或元素以及项之间的关系。根据本发明的一个方面，可以在属性格式中指定表达式，使得关系看上去是项的属性，例如“customer.Orders”、“order.Customer “或”orders.Customers”。在标号 920 处，定位与该关系表达式相关联的方法。在一个实例中，该方法可以驻留在由此与项目本身无关地定义的分离的关系类中。例如，可以确定“customer.Orders”映射至定单关系类，并特别地映射至方法“GetOrdersGivenCustomer(Customer customer)”。在 930 处，可以执行该方法。该方法可便于数据导航和可任选的其它功能，诸如检索、添加、移除或其它数据交互或操纵。在参考标号 940 处，如果合适可以返回结果。

[0152] 图 10 是根据本发明的一方面的程序协助方法 1000。方法 1000 可协助关系的指定。在参考标号 1010 处，特定类型的项被接收、检索或获取。随该项之后，在 1020 处，接收、检索或以获取触发器。触发器可以对应于诸如空格（“”）、点（“.”）、回车等等。在参考标号 1030 处，例如，以基于所定义的关系的提示的形式提供协助。例如，如果用户输入“customer.”，那么可以提供诸如“Orders”等的完成提示。提示可以例如在下拉式菜单中向用户显示。在选取之后，可以完成语句，并被读为“customer.Orders”。这减少了记忆或回想与给定项相关联的所有可能关系的负担，并最小化排字错误等等。

[0153] 除了提示之外，应该注意到，可以提供其它形式的程序协助，诸如关于关系表达式的独特的格式化和 / 或着色。而且，可以提供工具提示，其中例如类型信息在文字框中冒出或在光标滚动或悬停后冒出，和 / 或按下预定的组合键之后冒出。也构想了关于关系表达式的其它程序协助，且这些程序协助也在本发明要求保护的范围之内。

[0154] 图 11 是示出可以用来生成实现代码（例如，可执行代码、中间语言 ...）的编译器环境 1100。然而，环境 1100 的各方面也可以用作例如关于代码编辑器的后台编译器的一部分，以使能够提供智能或上下文敏感的程序设计协助。编译器环境 1100 包括编译器 1120，它包括前端组件 1120、转换器组件 1130、后端组件 1140、出错检验器组件 1150、符号表 1160、分析树 1170、以及状态 1180。编译器 1120 接受源代码作为输入，并产生实现代码作为输出。输入可以包括但不限于，关系表达式、类和 / 或如此处所述的其它构造。编译器环境 1100 的组件和模块之间的关系示出数据的主要流程。为了清楚和简要起见，不示出其它组件和关系。取决于实现，组件可以被添加、省略、分裂成多个模块、与其它模块组合和 /

或模块的其它配置。

[0155] 编译器 1120 可以接受含有与处理元素序列相关联的源代码的文件作为输入。源代码可以包括关系表达式、类、其它表达式、方法和 / 或其它程序构造。编译器 1120 可以结合用于分析构造和生成或注入代码的一个或多个组件来处理源代码。

[0156] 前端组件 1120 读取源代码并对其执行词法分析。本质上，前端组件 1120 读取源代码中的字符（例如，字母数字）序列并将其转换成句法元素或标记，指示常量、标识符、运算符、关键词和标点等等。

[0157] 转换器组件 1130 将标记解析为中间表示。例如，转换器组件 1130 可以检查句法并将标记组合成表达式或其它句法结构，这些句法结构进而又合并成语句树。概念上，这些树形成分析树 1170。而且，如果合适，转换器模块 1130 可以将条目放入列出在源代码中使用的符号名和类型信息及其相关特征的符号表 1130 中。

[0158] 状态 1180 可以用来跟踪编译器 1120 在处理所接收或检索到的源代码以及形成分析树 1170 时的进度。例如，不同的状态值指示编译器 1120 处于类定义或功能的开始处，刚声明了类成员或已经完成表达式。随着编译器行进，它持续更新状态 1180。编译器 1120 可以向之后向编译器 1120 提供输入的外部实体部分或全部地展示状态 1180。

[0159] 基于源代码中的构造或其它信号（或者如果识别了机遇），转换器 1130 或另一组件可以注入代码以便于有效且正确地执行。例如，可以注入代码来扩展理解缩略语或将查询理解转换成序列运算符。编码至转换器 1130 或其它组件内的规则指明为实现所期望的功能必须做什么，并标识将被注入代码或将执行其它操作的位置。注入的代码一般包括一个或多个位置处添加的语句、元数据、或其它元素，但是该术语也可以包括改变、删除或以修改现有的源代码。注入代码可以被存储为一个或多个模板或以某种其它形式储存。另外，应该理解，可以操纵符号表和变换分析树。

[0160] 基于符号表 1160 和分析树 1170，后端组件 1140 可以将中间表示转换成输出代码。后端组件 1140 将中间表示转换为可在目标处理器中执行或可由该目标处理器执行的指令、变量的存储器分配等等。输出代码可以由真实的处理器执行，但是本发明也构想了输出代码由虚拟处理器执行。

[0161] 此外，前端组件 1120 和后端组件 1140 可以执行其它的功能，诸如代码优化，也可以将所述操作作为单个阶段或在多个阶段中执行。编译器 1120 组件的各种其它方面在本质上是常规的，且可以使用执行等效功能的组件代替。另外，在处理源代码的过程中的各个阶段，出错检验器 1150 可以检验诸如词汇结构、句法错误甚至语义错误等错误。在检测到错误后，检验器组件 1150 可以中止编译并生成指示错误的消息。

[0162] 为了提供用于本发明的各方面的环境，图 12 和 13 及以下讨论旨在提供可在其中实现本发明的各方面的合适的计算环境 900 的简要概括描述。尽管以上在可以在一台和 / 或多台计算机上运行的计算机程序的计算机可执行指令的通用语境中描述了本发明，但是本领域的技术人员可以认识到，本发明也可以结合其它程序模块来实现。一般而言，程序模块包括例程、程序、组件、数据结构等，它们执行特定的任务和 / 或实现特定的抽象数据类型。而且，本领域的技术人员可以理解，本发明方法可以使用其它计算机系统配置来实现，包括单处理器或多处理器计算机系统、小型计算设备、大型机、以及个人计算机、手持式计算设备（例如，个人数字助理 (PDA)、电话、手表 . . . ）、基于微处理器的或可编程消费者或工

业电子产品等。本发明所示方面也可以在分布式计算环境中实现，在分布式计算环境中任务是由通过通信网络链接的远程处理设备执行的。然而，本发明的某些（如果不是全部）方面也可以在独立计算机上实现。在分布式计算环境中，程序模块可以位于本地或远程存储器存储设备中。

[0163] 参考图12，用于实现此处所揭示的各方面的示例性环境1210包括计算机1212（例如，台式机、膝上型计算机、服务器、手持式设备、可编程消费者或工业电子产品…）。计算机1212包括处理单元1214、系统存储器1216和系统总线1218。系统总线1218将包括但不限于系统总线1216在内的系统组件耦合至处理单元1214。处理单元1214可以是各种可用的任何处理器。双微处理器和其它多处理器体系结构也可以用作处理单元1214。

[0164] 系统总线1218可以是若干类型的总线结构中的任一种，包括存储器总线或存储器控制器、外围总线或外部总线和/或使用各种可用的总线体系结构中的任一种的局部总线，可用的总线体系结构包括，但不限于，11位总线、工业标准体系结构（ISA）、微通道体系结构（MCA）、扩展的ISA(EISA)、智能驱动器电子接口（IDE）、VESA局部总线（VLB）、外围部件互连（PCI）、通用串行总线（USB）、高级图形接口（AGP）、个人计算机存储卡国际协会总线（PCMCIA）以及小型计算机系统接口（SCSI）。

[0165] 系统存储器1216包括易失性存储器1220和非易失性存储器1222。基本输入/输出系统（BIOS）包含有助于诸如在启动时在计算机1212中的元件之间传递信息的基本例程，它通常存储在非易失性存储器1222中。作为说明，而非限制，非易失性存储器1222可以包括只读存储器（ROM）、可编程ROM(PROM)、电可编程ROM(EPROM)、电可擦除ROM(EEPROM)或闪存。易失性存储器1220可以包括用作外部高速缓存的随机存取存储器（RAM）。作为说明，而非限制，RAM以多种形式可用，诸如同步RAM(SRAM)、动态RAM(DRAM)、同步DRAM(SDRAM)、双数据速率SDRAM(DDR SDRAM)、增强型SDRAM(ESDRAM)、同步链路DRAM(SLDRAM)以及直接Rambus RAM(DRRAM)。

[0166] 计算机1212也包括可移动/不可以移动、易失性/非易失性计算机存储介质。例如，图12示出磁盘存储1224。磁盘存储1224包括，但不限于，如磁盘驱动器、软盘驱动器、磁带驱动器、Jaz驱动器、Zip驱动器、LS-100驱动器、闪存卡或记忆棒等设备。另外，磁盘存储1224可以包括独立或与其它存储介质结合的存储介质，包括但不限于，诸如光盘ROM设备（CD-ROM）、CD可记录驱动器（CD-R驱动器）、CD可重写驱动器（CD-RW驱动器）或数字多功能盘ROM驱动器（DVD-ROM）等的光盘驱动器。为了便于将磁盘存储设备1224连接至系统总线1218，一般使用诸如接口1226等可移动或不可移动接口。

[0167] 可以理解，图12描述了作为用户和在合适的操作环境1210中描述的基本计算机资源之间的中介的软件。这样的软件包括操作系统1228。可被存储在磁盘存储1224上的操作系统1228用来控制和分配计算机系统1212的资源。系统应用程序1230利用了操作系统1228通过存储在系统存储器1216或者磁盘存储1214上的程序模块1232和程序数据1234执行的资源管理。可以理解，本发明可以使用各种操作系统或操作系统的组合来实现。

[0168] 用户通过输入设备1236向计算机1212输入命令或信息。输入设备1236包括，但不限于，诸如鼠标、跟踪球、指示笔、触摸垫等定点设备、键盘、麦克风、操纵杆、游戏垫、圆盘式卫星天线、扫描仪、TV调谐器卡、数码相机、数码摄像机、网络摄像头等。这些和其它输入设备经由接口端口1238通过系统总线1218连接至处理单元1214。接口端口1238包括，例

如串行端口、并行端口、游戏端口和通用串行总线 (USB)。输出设备 1240 使用某些与输入设备 1236 相同类型的端口。从而，例如，USB 端口可以用于向计算机 1212 提供输入，并向输出设备 1240 提供来自计算机 1212 的输出信息。提供输出适配器 1242 来示出存在类似监视器（例如，平板和 CRT）、扬声器和打印机等需要专用适配器的输出设备 1240 的某些输出设备 1240。作为说明而非限制，输出适配器 1242 包括提供输出设备 1240 和系统总线 1218 之间的连接手段的显卡和声卡。应该注意，诸如远程计算机 1244 等其它设备和 / 或设备系统同时提供输入和输出能力两者。

[0169] 计算机 1212 可使用至一台或多台远程计算机，诸如远程计算机 1244 的逻辑连接在网络化环境中操作。远程计算机 1244 可以是个人计算机、服务器、路由器、网络 PC、工作站、基于微处理器的装置、对等设备或其它常见网络节点等，且通常包括上文相对于计算机 1212 描述的许多或所有元件。为简洁起见，对于远程计算机 1244 仅示出存储器存储设备 1246。远程计算机 1244 通过网络接口 1248 被逻辑连接至计算机 1212，并且然后通过通信连接 1250 被物理地连接。网络接口 1248 包括诸如局域网 (LAN) 和广域网 (WAN) 的通信网络。LAN 技术包括光纤分布式数据接口 (FDDI)、铜线分布式数据接口 (CDDI)、以太网 / IEEE802.3、令牌环 /IEEE802.5 等。WAN 技术包括，但不限于，点对点链路、类似综合业务数字网 (ISDN) 及其变体的电路交换网络、分组交换网络和数字用户线 (DSL)。

[0170] 通信连接 1250 指的是用来将网络接口 1248 连接至总线 1218 的硬件 / 软件。尽管为说明清楚，将通信连接 1250 示为位于计算机 1212 内，然而通信连接 1250 也可以在计算机 1212 外部。仅为示例的目的，连接至网络接口 1248 所必需的硬件 / 软件包括内部和外部技术，诸如包括常规电话级调制解调器、线缆调制解调器、电力调制解调器 (power modems)、DSL 调制解调器等的调制解调器、ISDN 适配器以及以太网卡或组件。

[0171] 图 13 是可与本发明交互的示例计算环境 1300 的示意性框图。系统 1300 包括一个或多个客户机 1310。客户机 1310 可以是硬件和 / 或软件（例如，线程、进程、计算设备）。系统 1300 也包括一个或多个服务器 1330。这样，系统 1300 可以对应于两层的客户机服务器模型或多层模型（例如，客户机、中间层服务器、数据服务器）以及其它模型。服务器 1330 也可以是硬件和 / 或软件（例如，线程、进程、计算设备）。例如，服务器 1330 可以容纳通过使用本发明来执行转换的线程。客户机 1310 和服务器 1330 之间的一种可能的通信可以是适于在两个或多个计算机进程之间传输的数据包的形式。

[0172] 系统 1300 包括可以用来促进客户机 1310 和服务器 1330 之间通信的通信架构 1350。客户机 1310 可操作地连接至可用来存储对客户机 1310 本地的信息的一个或多个客户机数据存储 1360。类似地，服务器 1330 可操作地连接至可被用来存储对服务器 1330 本地的信息的一个或多个服务器数据存储 1340。

[0173] 以上描述的包括所要求保护的本发明的各方面的示例。当然，不可能为描述所要求保护的本发明而描述每个可想象的组件或方法的组合，但是本领域的普通技术人员可以认识到，此处所揭示的本发明的众多其它组合和排列是可能的。从而，本发明旨在包括落入所附权利要求书精神和范围内的所有这样的变更、修改和变化。而且，就在详细描述或权利要求书中使用的术语“包括”、“含有”或“具有”及其变体而言，这样的术语旨在类似于解释术语“包含”（当术语“包含”被用作权利要求书中的过渡词时）的方式是包含性的。

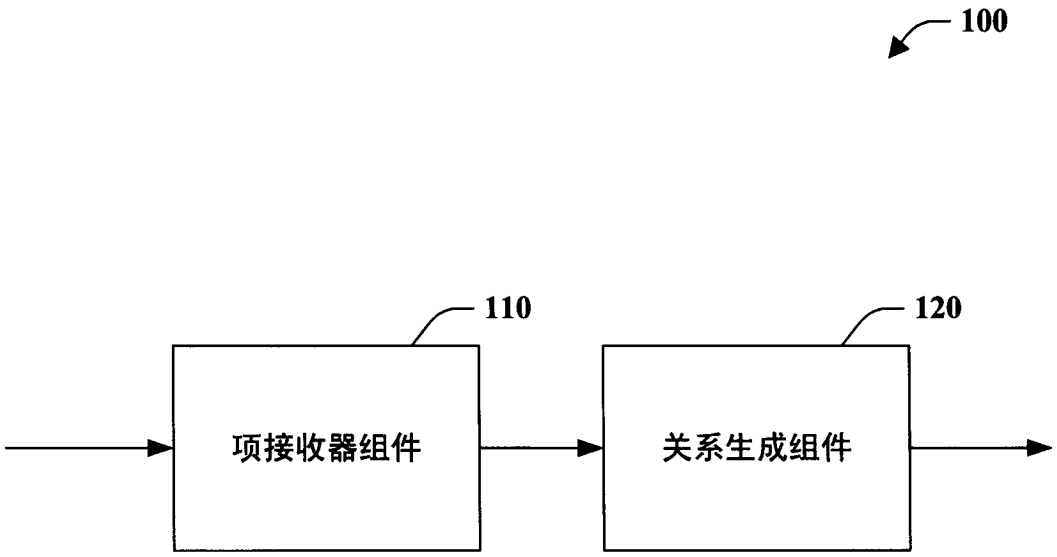


图 1

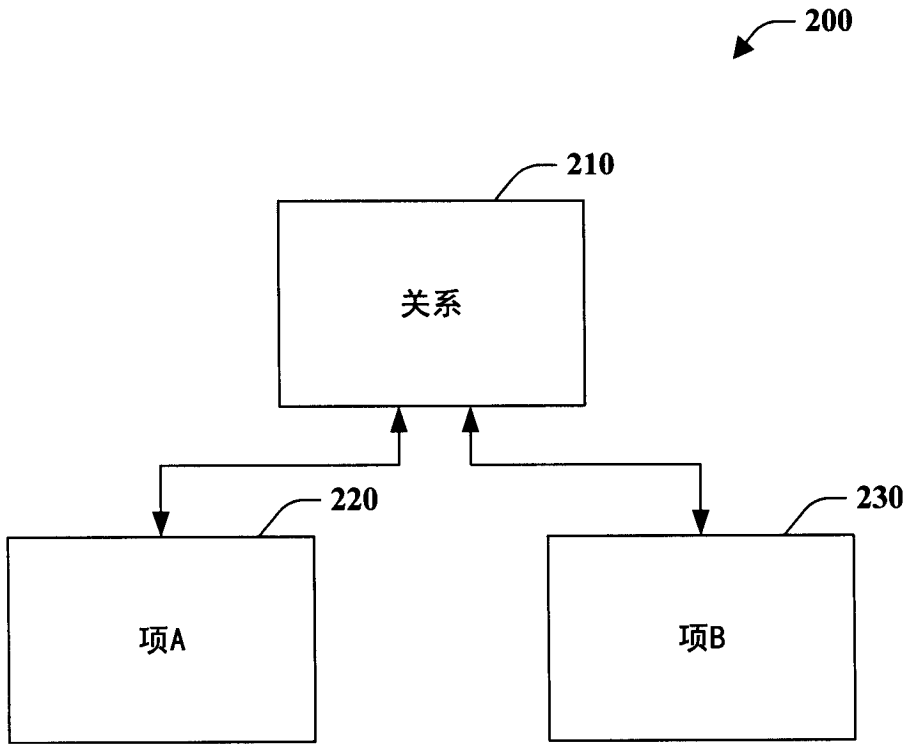


图 2

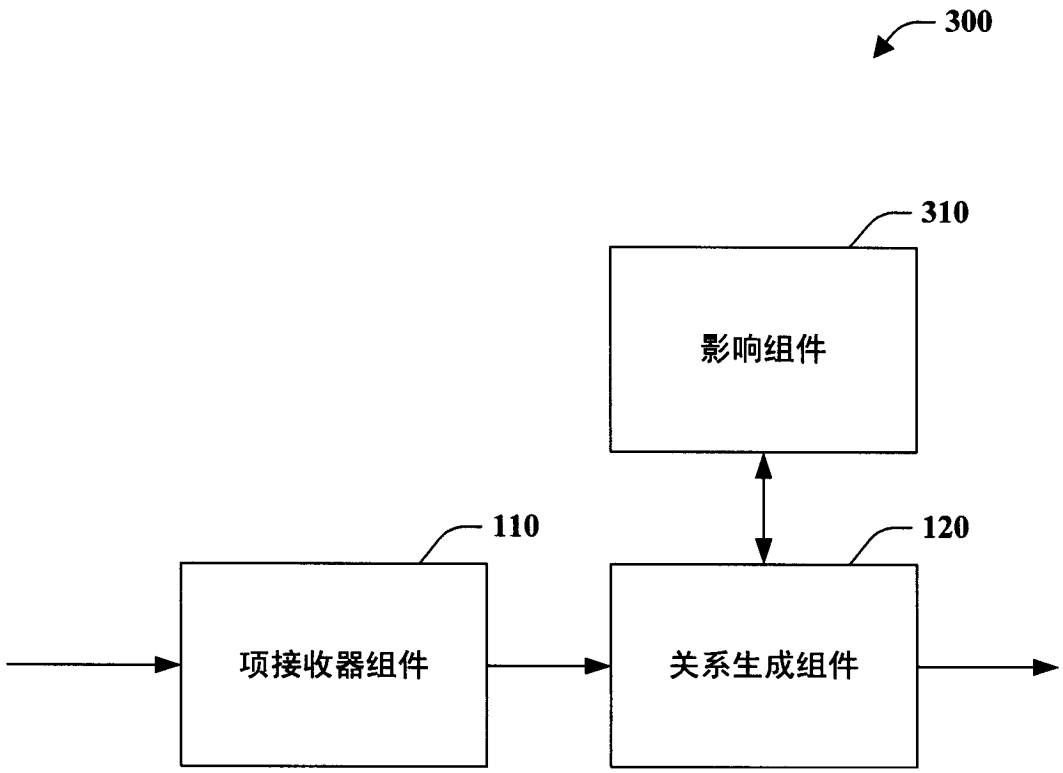

300
310
影响组件
110
120
项接收器组件
关系生成组件

图 3

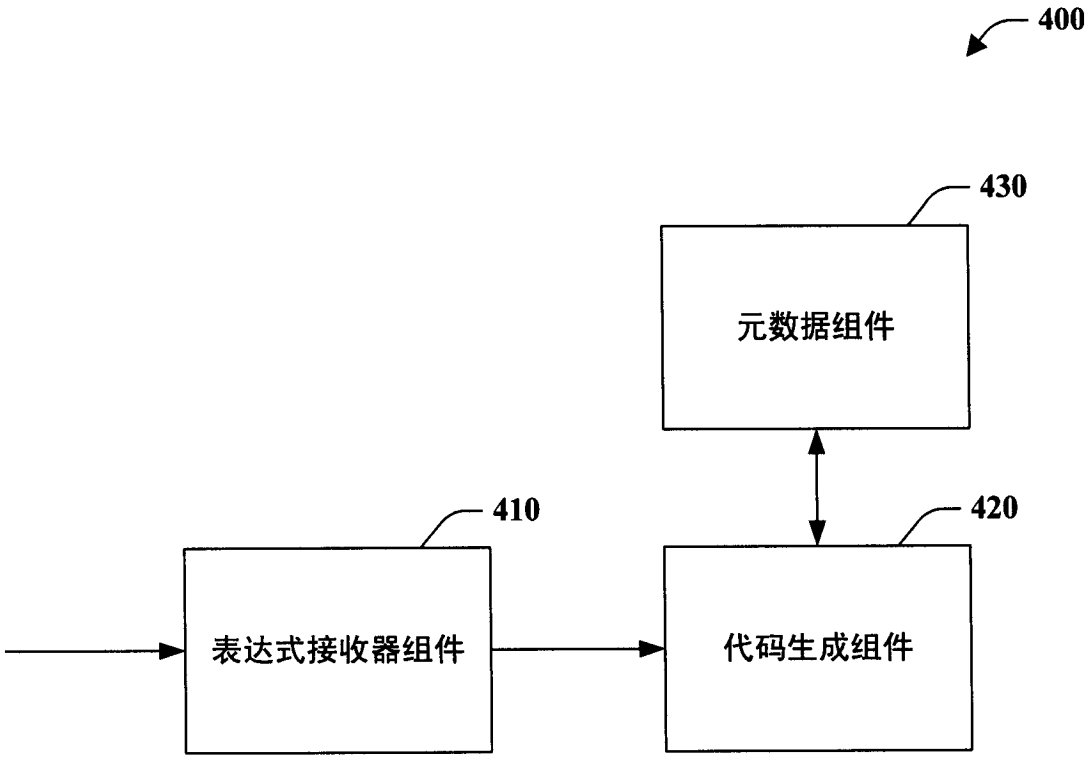

400
430
元数据组件
410
420
表达式接收器组件
代码生成组件

图 4

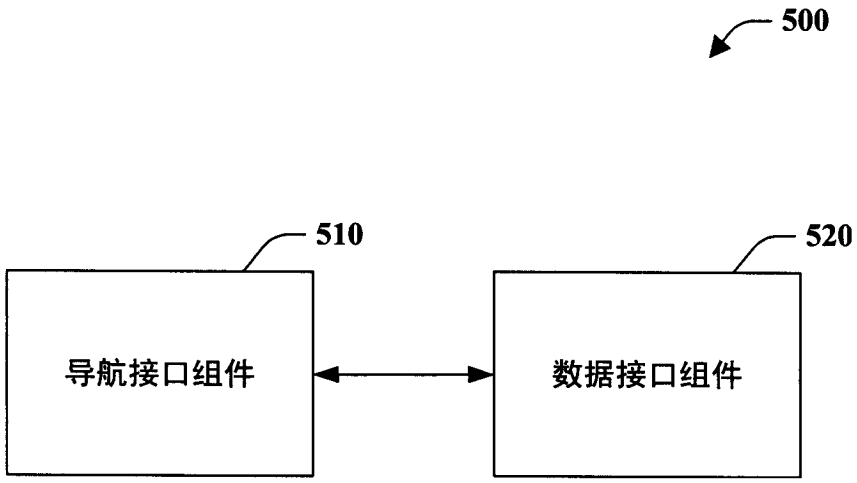


图 5

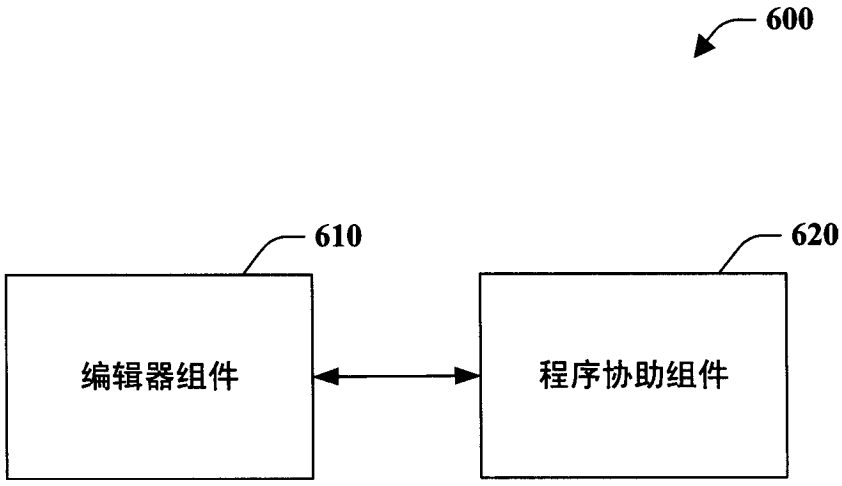


图 6

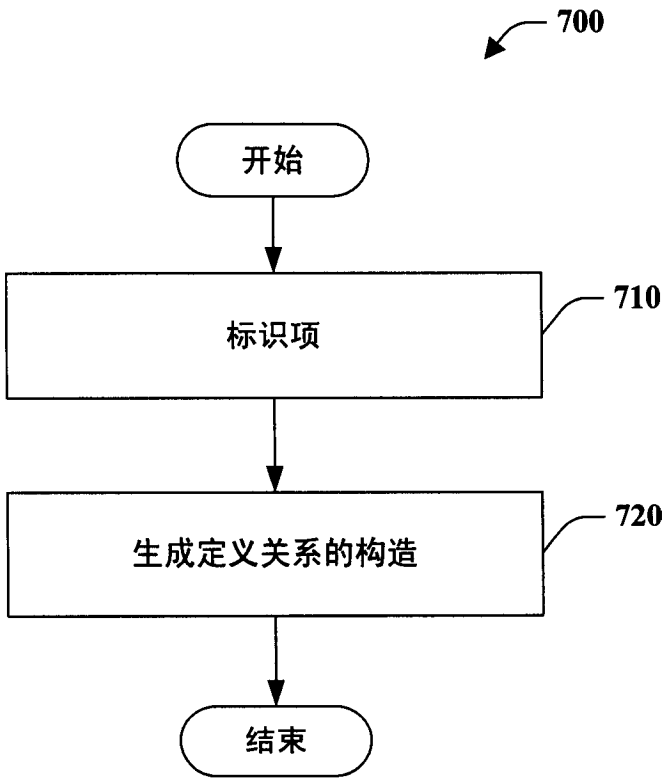

700
开始
标识项
710
生成定义关系的构造
720
结束

图 7

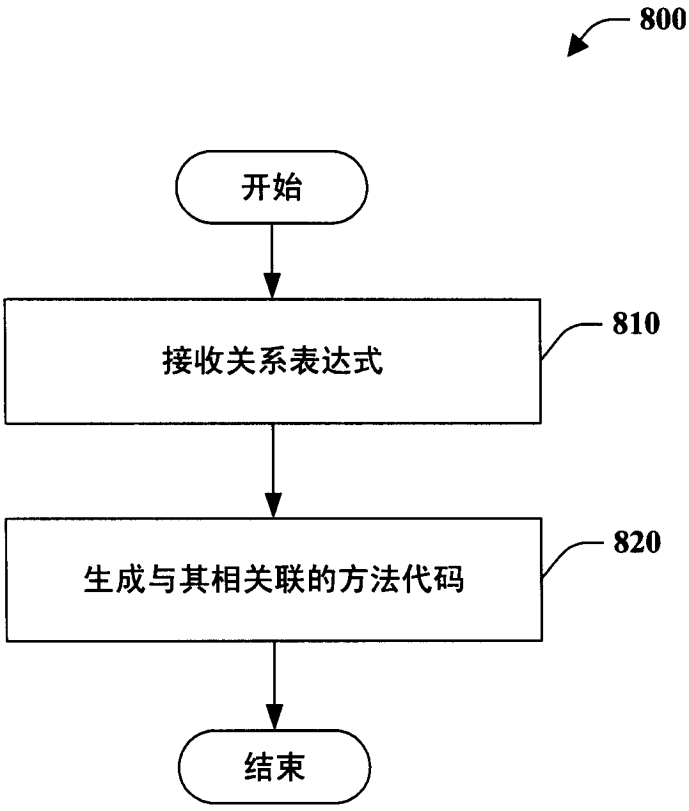

800
开始
接收关系表达式
810
生成与其相关联的方法代码
820
结束

图 8

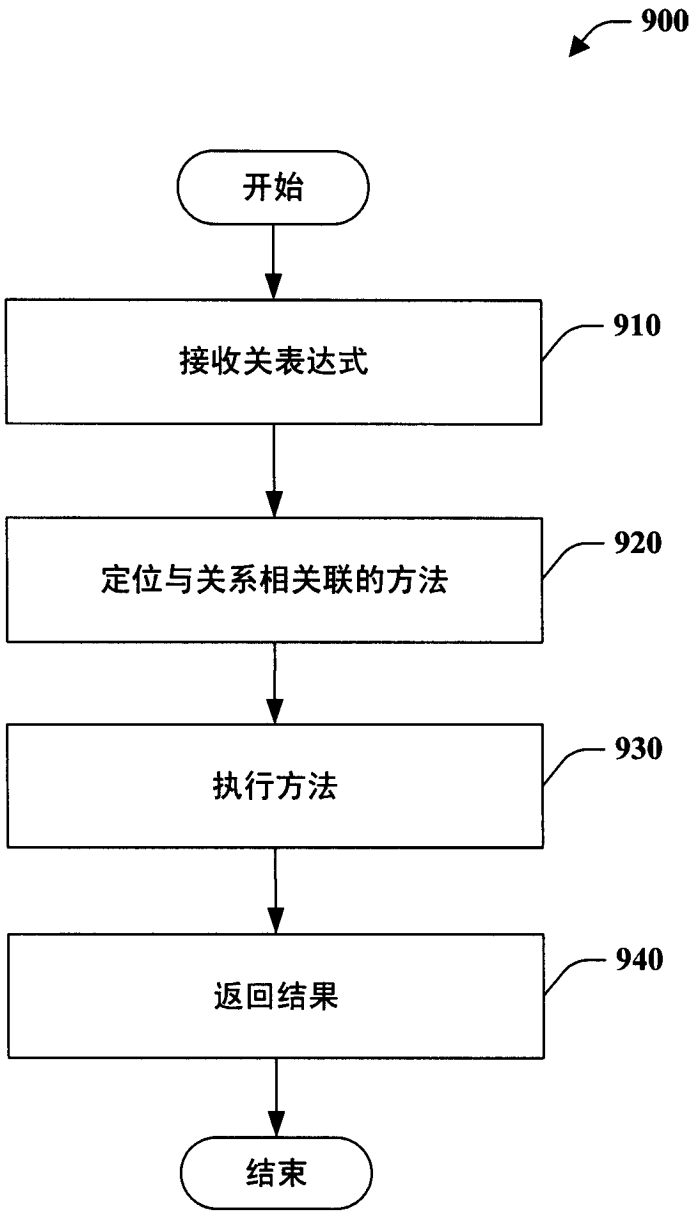


图 9

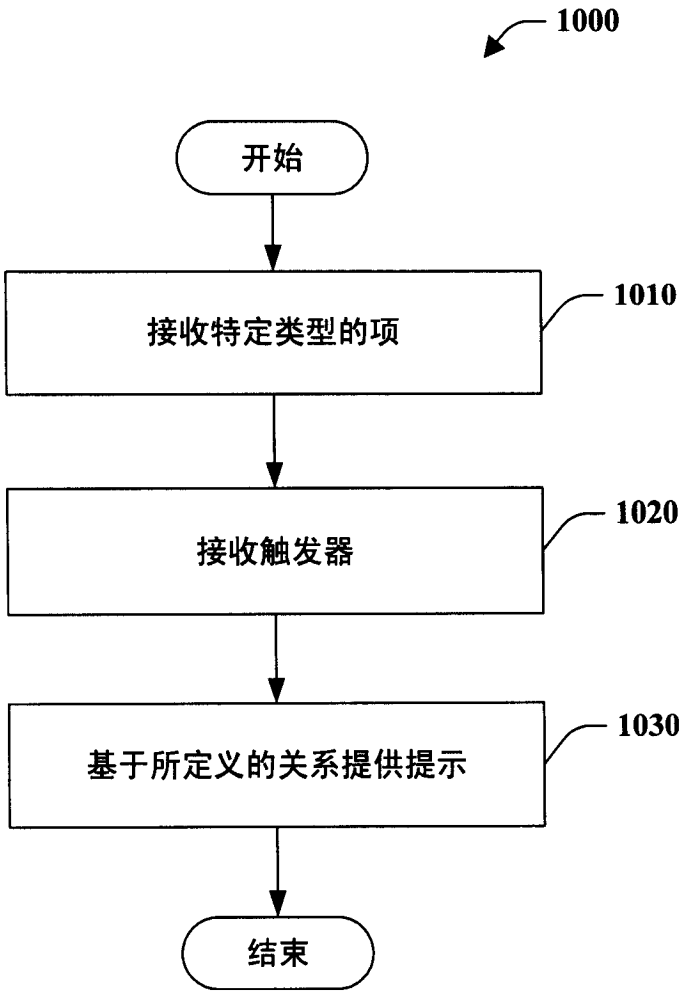


图 10

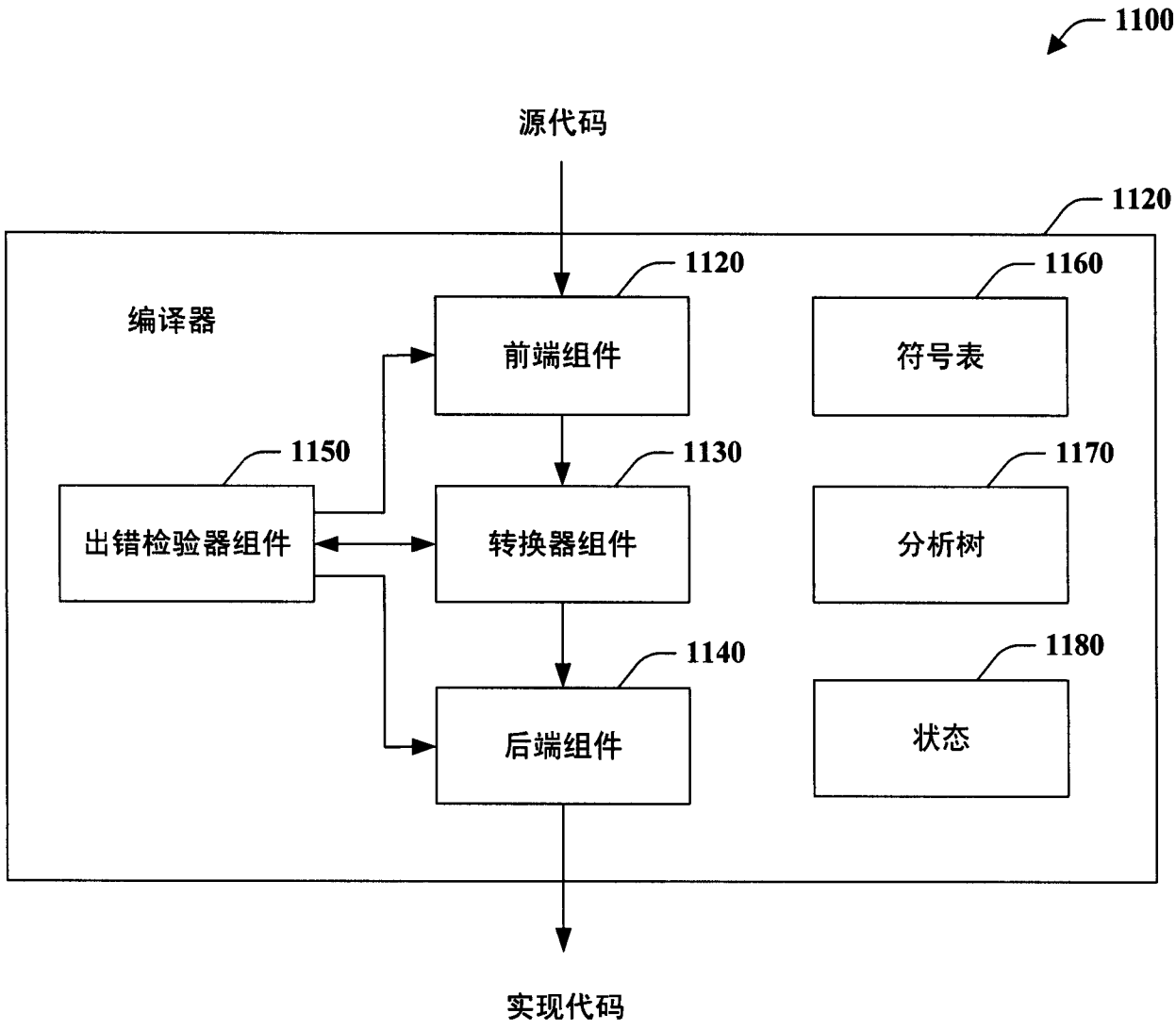

1100
源代码
1120
1120
1160
编译器
前端组件
符号表
1150
1130
1170
出错检验器组件
转换器组件
分析树
1140
1180
后端组件
状态
实现代码

图 11

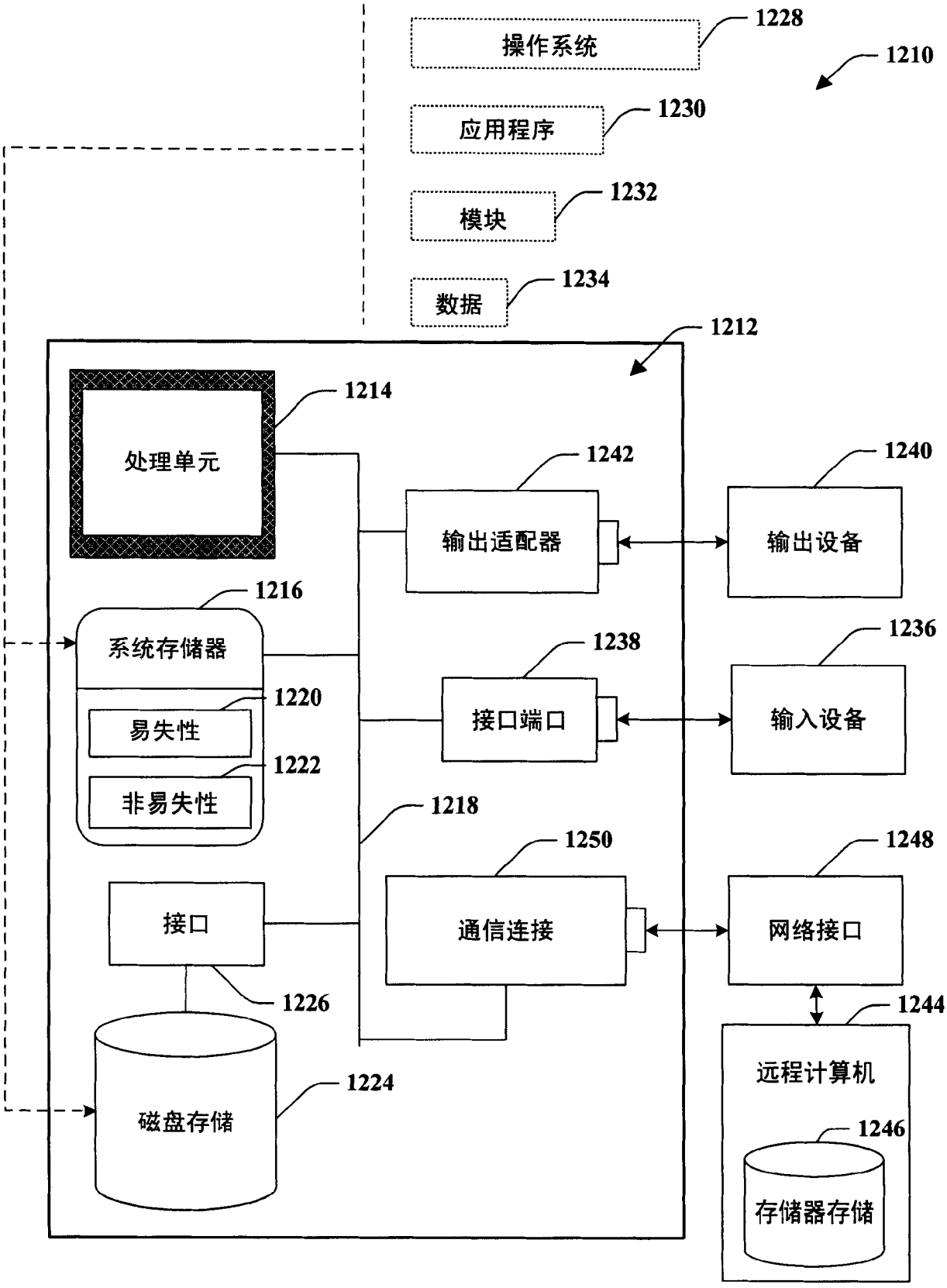

操作系统
1228
1210
应用程序
1230
模块
1232
数据
1234
1212
处理单元
1214
1242
输出适配器
1240
输出设备
1216
系统存储器
1220
易失性
1222
非易失性
1238
接口端口
1236
输入设备
1218
1250
通信连接
1248
网络接口
接口
1226
1244
远程计算机
磁盘存储
1224
1246
存储器存储

图 12

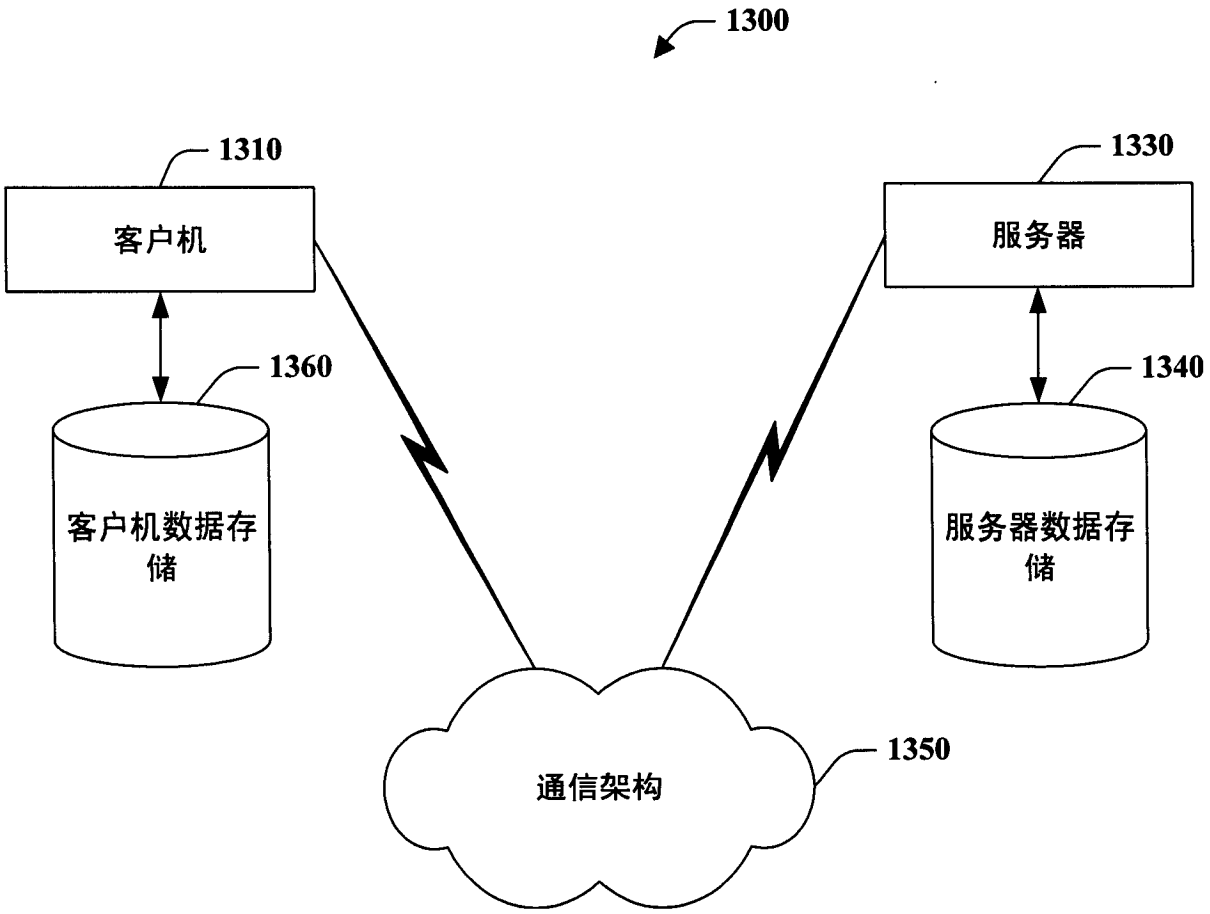


图 13