(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 105359141 B
(45)授权公告日 2021.10.01

(21)申请号 201480037071.7
(22)申请日 2014.03.26
(65)同一申请的已公布的文献号
申请公布号 CN 105359141 A
(43)申请公布日 2016.02.24
(30)优先权数据
61/824,544 2013.05.17 US
61/843,203 2013.07.05 US
14/044,429 2013.10.02 US
(85)PCT国际申请进入国家阶段日
2015.12.28
(86)PCT国际申请的申请数据
PCT/US2014/031919 2014.03.26
(87)PCT国际申请的公布数据
WO2014/186057 EN 2014.11.20
(73)专利权人 甲骨文国际公司
地址 美国加利福尼亚
(72)发明人 D·阿兰 刘国洪(托马斯)
龚宇(杰夫)
(74)专利代理机构 中国贸促会专利商标事务所有限公司 11038
代理人 鲍进
(51)Int.Cl.
G06F 16/25(2019.01)
(56)对比文件
US 2005228808 A1,2005.10.13
CN 102349081 A,2012.02.08
廖林伟等.面向商业智能的ETL模型研究综述.《信息技术》.2012,(第4期),第25-29页.
张忠平等.结构图ETL概念模型的设计方法.《计算机工程与应用》.2009,第45卷(第6期),第161-164页.
审查员 凌燕翔

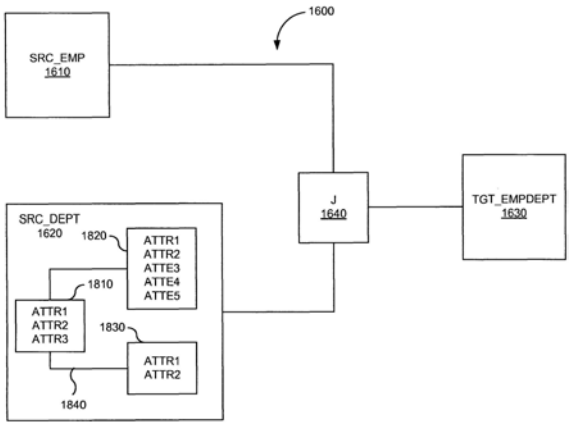
权利要求书3页 说明书27页 附图30页

(54)发明名称

支持基于流的ETL和基于实体关系的ETL的组合

(57)摘要

本公开涉及支持基于流的ETL和基于实体关系的ETL的组合。公开了结合用于使映射的设计和维护容易的一种或多种技术的数据集成系统。当组件被添加到现有设计时,数据集成系统消除了指定所有输入和输出属性的需求。在一方面,实现了允许实体关系在逻辑设计的流视图中被添加和/或编辑的组件类型。因此,可以以对映射设计器的该部分的最小努力将表示数据集的组件的属性添加并传播到下游组件。



CN 105359141 B

1.一种使用数据流提取-变换-加载ETL和实体关系提取-变换-加载ETL的组合来在数据源和数据目标之间生成数据集成场景的方法,所述方法包括:

在一个或多个计算机系统处在数据集成场景的逻辑设计的用户界面视图中,从用户接收经由所述数据集成场景的所述逻辑设计的组件定义的实体关系集合,其中,所述实体关系集合具有在数据集中的第一实体的属性和所述数据集中的第二实体的属性之间定义的关系;

利用与所述一个或多个计算机系统关联的一个或多个处理器,基于所述实体关系集合确定数据流模型,其中所述数据流模型是提供用于所述逻辑设计的语义定义的操作语义模型,其中所述语义定义限定所述数据集中的实体关系,所述数据集中的所述实体关系识别数据目标中的列如何直接从所述数据源的属性以及从所述数据集的所述第一实体的所述属性和所述第二实体的所述属性填充;

利用与所述一个或多个计算机系统关联的一个或多个处理器,在所述逻辑设计的所述用户界面视图中生成指示所述数据流模型的信息,其中,包括所述第一实体的所述属性以及所述第二实体的所述属性的属性集被暴露给所述逻辑设计的下游组件;

在所述一个或多个计算机系统处在所述逻辑设计的所述用户界面视图中通过引入一个或多个组件或数据集来从用户接收所述逻辑设计中的改变;

利用与所述一个或多个计算机系统关联的一个或多个处理器,通过重新路由之前生成的指示所述数据流模型的信息中的组件之间的连接来确定更新的数据流模型,而不必改变数据目标中的列如何直接从所述数据源的属性以及从所述数据集的所述第一实体的所述属性和所述第二实体的所述属性填充;和

在所述用户界面视图中显示更新的数据流模型。

2.如权利要求1所述的方法,还包括:基于声明数据源的属性之间的关系的信息,得出所述数据集中的每个实体的一个或多个属性。

3.如权利要求1或2所述的方法,还包括:接收指定逻辑设计的一个或多个组件的信息,所述信息包括指示改变流经所述逻辑设计的信息的形状的操作的信息。

4.如权利要求1所述的方法,还包括:接收指定所述逻辑设计的一个或多个组件的信息,所述信息包括指示控制流经所述逻辑设计的信息流但是不改变流经所述逻辑设计的信息的形状的操作的信息。

5.如权利要求1所述的方法,还包括:接收指定所述逻辑设计的一个或多个组件的信息,所述信息包括指示具有要存储在目标数据存储中的数据的一个或多个属性的目标组件的信息。

6.如权利要求1所述的方法,其中所述数据流模型包括三路关系,所述三路关系吸收关联实体以使得所述实体被配置为同时参与两个二进制关系。

7.一种存储用于使用数据流提取-变换-加载ETL和实体关系提取-变换-加载ETL的组合来在数据源和数据目标之间生成数据集成场景的计算机可执行代码的非临时性计算机可读介质,所述非临时性计算机可读介质包括:

用于在数据集成场景的逻辑设计的用户界面视图中,从用户接收经由所述数据集成场景的所述逻辑设计的组件定义的实体关系集合的代码,其中,所述实体关系集合具有在数据集中的第一实体的属性和所述数据集中的第二实体的属性之间定义的关系;

用于基于所述实体关系集合确定数据流模型的代码，其中所述数据流模型是提供用于所述逻辑设计的语义定义的操作语义模型，其中所述语义定义限定所述数据集中的实体关系，所述数据集中的所述实体关系识别数据目标中的列如何直接从所述数据源的属性以及从所述数据集的所述第一实体的所述属性和所述第二实体的所述属性填充；

用于在所述逻辑设计的所述用户界面视图中生成指示所述数据流模型的信息的代码，其中，包括所述第一实体的所述属性以及所述第二实体的所述属性的属性集被暴露给所述逻辑设计的下游组件；

用于在所述逻辑设计的所述用户界面视图中通过引入一个或多个组件或数据集来从用户接收所述逻辑设计中的改变的代码；

用于重新路由之前生成的指示所述数据流模型的信息中的组件之间的连接来确定更新的数据流模型而不必改变数据目标中的列如何直接从所述数据源的属性以及从所述数据集的所述第一实体的所述属性和所述第二实体的所述属性填充的代码；和

用于在所述用户界面视图中显示更新的数据流模型的代码。

8.如权利要求7所述的非临时性计算机可读介质，还包括：用于基于声明数据源的属性之间的关系的信息，得出所述数据集中的每个实体的一个或多个属性的代码。

9.如权利要求7至8中任何一项所述的非临时性计算机可读介质，还包括：用于接收指定所述逻辑设计的一个或多个组件的信息的代码，所述信息包括指示改变流经所述逻辑设计的信息的形状的操作的信息。

10.如权利要求7所述的非临时性计算机可读介质，还包括：用于接收指定所述逻辑设计的一个或多个组件的信息的代码，所述信息包括指示控制流经所述逻辑设计的信息流但是不改变流经所述逻辑设计的信息的形状的操作的信息。

11.如权利要求7所述的非临时性计算机可读介质，还包括：用于接收指定所述逻辑设计的一个或多个组件的信息的代码，所述信息包括指示具有要存储在目标数据存储中的数据的一个或多个属性的目标组件的信息。

12.如权利要求7所述的非临时性计算机可读介质，其中所述数据流模型包括三路关系，所述三路关系吸收关联实体以使得所述实体被配置为同时参与两个二进制关系。

13.一种使用数据流提取-变换-加载ETL和实体关系提取-变换-加载ETL的组合来在数据源和数据目标之间生成数据集成场景的系统，所述系统包括：

处理器；及

存储指令的存储器，所述指令在被所述处理器执行时，将所述处理器配置为：

在数据集成场景的逻辑设计的用户界面视图中，从用户接收经由数据集成场景的逻辑设计的组件定义的实体关系集合，其中，所述实体关系集合具有数据集中的第一实体的属性和所述数据集中的第二实体的属性之间定义的关系；

基于所述实体关系集合确定数据流模型，其中所述数据流模型是提供用于所述逻辑设计的语义定义的操作语义模型，其中所述语义定义限定所述数据集中的实体关系，所述数据集中的所述实体关系识别数据目标中的列如何直接从所述数据源的属性以及从所述数据集的所述第一实体的所述属性和所述第二实体的所述属性填充；

在所述逻辑设计的所述用户界面视图中生成指示所述数据流模型的信息，其中，包括所述第一实体的所述属性以及所述第二实体的所述属性的属性集被暴露给所述逻辑设计

的下游组件，

其中所述处理器还被配置为:

在所述逻辑设计的所述用户界面视图中通过引入一个或多个组件或数据集来从用户接收所述逻辑设计中的改变;及

重新路由之前生成的指示所述数据流模型的信息中的组件之间的连接来确定更新的数据流模型，而不必改变数据目标中的列如何直接从所述数据源的属性以及从所述数据集的所述第一实体的所述属性和所述第二实体的所述属性填充;和

在所述用户界面视图中显示更新的数据流模型。

14.如权利要求13所述的系统，其中所述处理器还被配置为基于声明数据源的属性之间的关系的信息，得出所述数据集中的每个实体的一个或多个属性。

15.如权利要求13或14所述的系统，其中所述处理器还被配置为接收指定所述逻辑设计的一个或多个组件的信息，所述信息包括指示改变流经所述逻辑设计的信息的形状的操作的信息。

16.如权利要求13所述的系统，其中所述处理器还被配置为接收指定所述逻辑设计的一个或多个组件的信息，所述信息包括指示控制流经所述逻辑设计的信息流但是不改变流经所述逻辑设计的信息的形状的操作的信息。

17.如权利要求13所述的系统，其中所述处理器还被配置为接收指定所述逻辑设计的一个或多个组件的信息，所述信息包括指示具有要存储在目标数据存储中的数据的一个或多个属性的目标组件的信息。

18.如权利要求13所述的系统，其中所述数据流模型包括三路关系，所述三路关系吸收关联实体以使得所述实体被配置为同时参与两个二进制关系。

支持基于流的ETL和基于实体关系的ETL的组合

背景技术

[0001] 在如今日益快节奏的商业环境中，机构需要使用更专门的软件应用。此外，机构需要确保这些应用在异质硬件平台和系统上的共存并且保证在应用和系统之间共享数据的能力。

[0002] 因此，期望解决与开发数据集成场景相关的问题，其中一些场景可以在本文进行讨论。此外，期望减少与开发数据集成场景相关的缺陷，其中一些场景可以在本文进行讨论。

发明内容

[0003] 至少为了提供对主题的基本理解，本公开内容的以下部分给出对在本公开内容中找到的一个或多个创新、实施例和/或例子的简化综述。该综述不是要提供对任何特定实施例或例子的排它概述。此外，该综述不是要识别实施例或例子的关键/重要元素或者勾勒本公开内容的主题的范围。因此，该综述的一个目的可以是要以简化的形式给出在本公开内容中找到的一些创新、实施例和/或例子，作为对随后给出的更详细描述的序言。

[0004] 在各种实施例中，数据集成系统使用户能够创建独立于平台和技术的逻辑设计。用户可以创建在高级别定义用户想让数据如何在源和目标之间流动的逻辑设计。鉴于用户的基础设施，工具可以分析逻辑设计，并且创建物理设计。逻辑设计可以包括对应于设计中每个源和目标的多个组件，以及诸如连接或过滤的操作，以及访问点。当被转移到物理设计时，每个组件生成对数据执行操作的代码。依赖于底层技术(例如，SQL Server、Oracle、Hadoop，等等)和所使用的语言(SQL、pig，等等)，由每个组件生成的代码可以不同。

[0005] 在一方面，数据集成系统的用户不需要从开始到结束在逻辑设计中的每个组件指定所有数据属性。数据集成系统提供避免需要完全声明流经逻辑设计的信息的多个组件类型，诸如投影器和选择器类型。数据集成系统能够在由预定组件类型表示的操作决定需要什么属性。这简化了设计和维护。在各个方面，提供了充分利用现有RDBMS资源和能力来避免对单独的专用ETL服务器的需求的数据变换和迁移，以实现改进的性能。

[0006] 在一种实施例中，促进数据映射的生成的方法包括接收指定实体关系集合的信息作为逻辑设计的组件。等效数据流模型是基于该实体关系集合确定的。然后，在逻辑流设计中生成指示该等效数据流模型的信息。表示该实体关系集合的数据集的一个或多个属性可以基于声明数据源的属性之间的关系的信息被得出。

[0007] 在更多实施例中，可以接收指定逻辑设计的一个或多个组件的信息，所述信息包括指示改变流经该逻辑设计的信息的形状的操作的信息。可以接收指定逻辑设计的一个或多个组件的信息，所述信息包括指示控制流经该逻辑设计的信息流但是不改变流经该逻辑设计的信息的形状的操作的信息。可以接收指定逻辑设计的一个或多个组件的信息，所述信息包括指示具有要存储在目标数据存储(target datastore)中的数据的一个或多个属性的目标组件的信息。

[0008] 在一方面，在逻辑流设计中生成指示等效数据流模型的信息可以包括将属性列表

导出到下游组件。在另一方面，逻辑设计中的改变可以通过一个或多个关系的引入来接收。然后，可以确定更新的等效数据流模型。

[0009] 在一种实施例中，存储用于促进数据映射的生成的计算机可执行代码的非临时性计算机可读介质包括用于接收指定实体关系集合的信息作为逻辑设计的组件的代码，用于基于该实体关系集合确定等效数据流模型的代码，以及用于在逻辑流设计中生成指示该等效数据流模型的信息的代码。

[0010] 在另一实施例中，促进数据映射的生成的系统包括处理器和存储指令的存储器，当指令被处理器执行时，将处理器配置为接收指定实体关系集合的信息作为逻辑设计的组件、基于该实体关系集合确定等效数据流模型、以及在逻辑流设计中生成指示该等效数据流模型的信息。

[0011] 在一种实施例中，促进数据映射的生成的系统包括被配置为接收指定实体关系集合的信息作为逻辑设计的组件的接收单元；被配置为基于该实体关系集合确定等效数据流模型的确定单元；以及被配置为在逻辑流设计中生成指示该等效数据流模型的信息的生成单元。

[0012] 在一方面，该系统还可以包括被配置为基于声明数据源的属性之间的关系的信息得出代表该实体关系集合的数据集的一个或多个属性的得出单元。在一方面，接收单元还被配置为接收指定逻辑设计的一个或多个组件的信息，所述信息包括指示改变流经该逻辑设计的信息的形状的操作的信息。在一方面，接收单元还被配置为接收指定逻辑设计的一个或多个组件的信息，所述信息包括指示控制流经该逻辑设计的信息流但是不改变流经该逻辑设计的信息的形状的操作的信息。

[0013] 在一方面，接收单元还被配置为接收指定逻辑设计的一个或多个组件的信息，所述信息包括指示具有要存储在目标数据存储中的数据的一个或多个属性的目标组件的信息。在一方面，生成单元包括被配置为将属性列表导出到下游组件的导出单元。在一方面，接收单元还被配置为通过一个或多个关系的引入来接收逻辑设计的改变；并且确定单元还被配置为确定更新的等效数据流模型。

[0014] 在一种实施例中，促进数据映射的生成的系统包括用于接收指定实体关系集合的信息作为逻辑设计的组件的装置；用于基于该实体关系集合确定等效数据流模型的装置；以及用于在逻辑流设计中生成指示该等效数据流模型的信息的装置。在另一方面，该系统还包括用于基于声明数据源的属性之间的关系的信息得出代表该实体关系集合的数据集的一个或多个属性的装置。

[0015] 在另一方面，该系统还包括用于接收指定逻辑设计的一个或多个组件的信息的装置，所述信息包括指示改变流经该逻辑设计的信息的形状的操作的信息。在另一方面，该系统还包括用于接收指定逻辑设计的一个或多个组件的信息的装置，所述信息包括指示控制流经该逻辑设计的信息流但是不改变流经该逻辑设计的信息的形状的操作的信息。

[0016] 在另一方面，该系统还包括用于接收指定逻辑设计的一个或多个组件的信息的装置，所述信息包括指示具有要存储在目标数据存储中的数据的一个或多个属性的目标组件的信息。在另一方面，用于生成指示逻辑流设计中的等效数据流模型的信息的装置包括用于将属性列表导出到下游组件的装置。在另一方面，该系统还包括用于通过一个或多个关系的引入来接收逻辑设计的改变的装置；以及用于确定更新的等效数据流模型的装置。

[0017] 通过参考本公开内容的剩余部分、任何附图以及权利要求，除以上部分之外，还应当认识到对本公开内容主题的本质及其等同物(以及所提供的任何固有的或明确的优点和改进)的进一步理解。

附图说明

[0018] 为了合理地描述和说明在本公开内容中找到的那些创新、实施例和/或例子，可以参考一个或多个附图。被用来描述一个或多个附图的附加细节或例子不应当被认为是对任何要求保护的发明的范围、任何目前描述的实施例和/或例子、或者目前被理解为本公开内容中给出的任何创新的最佳模式的限制。

[0019] 图1是可以结合本发明实施例的系统的简化说明。

[0020] 图2是根据本发明实施例的数据集成系统的框图。

[0021] 图3是可以被用来实现根据本发明实施例的数据集成系统的硬件/软件栈的简化框图。

[0022] 图4是在本发明各种实施例中具有可以为其创建数据集成场景的各种异质数据源的环境的框图。

[0023] 图5A和5B绘出了可以由数据集成系统执行的常规数据集成处理中的简化数据流。

[0024] 图6A和6B绘出了根据本发明实施例、在可以由数据集成系统执行的下一代数据集成处理中的简化数据流。

[0025] 图7是在根据本发明的一种实施例中数据集成系统的ODI工作室与储存库之间交互的简化框图。

[0026] 图8绘出了根据本发明实施例用于创建数据集成场景的方法的流程图。

[0027] 图9是根据本发明实施例用于创建数据集成场景的用户界面的屏幕截图。

[0028] 图10绘出了根据本发明实施例用于创建映射的方法的流程图。

[0029] 图11是根据本发明实施例用于在数据集成场景中提供映射信息的用户界面的屏幕截图。

[0030] 图12是根据本发明实施例用于在数据集成场景中提供流信息的用户界面的屏幕截图。

[0031] 图13绘出了根据本发明实施例用于创建包的方法的流程图。

[0032] 图14是根据本发明实施例用于在数据集成场景中提供包序列信息的用户界面的屏幕截图。

[0033] 图15绘出了根据本发明实施例用于部署数据集成场景的方法的流程图。

[0034] 图16是在根据本发明的一种实施例中组合的基于流且基于实体的映射的简化框图。

[0035] 图17绘出了根据本发明实施例用于生成组合的基于流且基于实体的映射的方法的流程图。

[0036] 图18是在根据本发明的一种实施例中具有数据集视图的图16的映射的简化框图。

[0037] 图19A和19B是在根据本发明的一种实施例中用于组合的基于流且基于实体的映射的逻辑和物理设计的简化框图。

[0038] 图20绘出了根据本发明实施例用于生成组合的基于流且基于实体的映射的物理

设计的方法的流程图。

[0039] 图21是绘出静态E-R和动态ETL模型之间的关系的说明。

[0040] 图22是在一种实施例中提供自动转换系统的顶层设计图的说明。

[0041] 图23A和23B以两种流行的E-R记号示出了三路关系。

[0042] 图24A和24B以两种流行的E-R记号示出了三路关系的等同物。

[0043] 图25利用一系列二进制关系示出了三路关系的等同物。

[0044] 图26利用标准的E-R记号示出了三路关系。

[0045] 图27绘出了为图26中的实体创建的每个表中的行。

[0046] 图28A和28B示出了一种实施例中利用E-R记号的三路关系以及具有源自三个实体的数据的数据流。

[0047] 图29绘出了展开各种数据库建模方法及其语义内容之间的关系的图。

[0048] 图30是可以被用来实践本发明实施例的计算机系统的简化框图。

[0049] 图31是根据本发明实施例用于促进数据映射的生成的系统的简化框图。

具体实施方式

[0050] 介绍

[0051] 在各种实施例中，数据集成系统使用户能够创建独立于平台和技术的逻辑设计。用户可以创建在高级别定义用户想让数据如何在源和目标之间流动的逻辑设计。鉴于用户的基础设施，工具可以分析逻辑设计，并且创建物理设计。逻辑设计可以包括对应于设计中每个源和目标的多个组件，以及诸如连接或过滤的操作，以及访问点。当被转移到物理设计时，每个组件生成对数据执行操作的代码。依赖于底层技术(例如，SQL Server、Oracle、Hadoop，等等)和所使用的语言(SQL、pig，等等)，由每个组件生成的代码可以不同。

[0052] 在一方面，数据集成系统的用户不需要从开始到结束在逻辑设计中的每个组件指定所有数据属性。数据集成系统提供避免需要完全声明流经逻辑设计的信息的多个组件类型，诸如投影器和选择器类型。数据集成系统能够在由预定组件类型表示的操作决定需要什么属性。这简化了设计和维护。

[0053] 图1是可以结合在本公开内容中发现的任何创新、实施例和/或例子的实施例或者结合到其中的系统100的简化说明。图1仅仅是说明结合本发明的实施例并且不限制如权利要求中所述的本发明的范围。本领域普通技术人员将认识到其它变体、修改和备选方案。

[0054] 在一种实施例中，系统100包括一个或多个用户计算机110(例如，计算机110A、110B和110C)。用户计算机110可以是通用个人计算机(仅仅作为例子，包括运行任何适当特色(flavor)的微软公司的Windows™和/或Apple公司的Macintosh™操作系统的个人计算机和/或膝上型计算机)和/或运行各种市售UNIX™或类UNIX操作系统当中任何一种的工作站计算机。这些用户计算机110还可以具有各种应用当中任何一种，包括被配置为执行本发明方法的一个或多个应用，以及一个或多个办公室应用、数据库客户端和/或服务器应用，及web浏览器应用。

[0055] 作为替代，用户计算机110可以是能够经由网络(例如，以下描述的通信网络120)通信和/或显示和导航网页或者其它类型的电子文档的任何其它电子设备，诸如瘦客户端计算机、启用互联网的移动电话，和/或个人数字助理。虽然示例性系统100被示为具有三个

用户计算机，但是任何数量的用户计算机或设备都可以被支持。

[0056] 本发明的某些实施例在联网环境中操作，这可以包括通信网络120。通信网络120可以是本领域技术人员熟悉的任何类型的网络，这些网络可以利用任何各种商用协议，包括但不限于TCP/IP、SNA、IPX、AppleTalk等，支持数据通信。仅仅作为例子，通信网络120可以是局域网(“LAN”)，包括但不限于以太网、令牌环网等；广域网；虚拟网络，包括但不限于虚拟专用网(“VPN”)；互联网；内联网；外联网；公共交换电话网(“PSTN”)；红外线网络；无线网络，包括但不限于依据IEEE 802.11协议套件、本领域中已知的Bluetooth™协议和/或任何其它无线协议中任一种操作的网络；和/或这些和/或其它网络的任意组合。

[0057] 本发明的实施例可以包括一个或多个服务器计算机130(例如，计算机130A和130B)。每个服务器计算机130可以被配置为具有包括但不限于以上讨论的任何一种操作系统，以及任何市售服务器操作系统。每个服务器计算机130还可以运行一个或多个应用，这些应用可以被配置为向一个或多个客户端(例如，用户计算机110)和/或其它服务器(例如，服务器计算机130)提供服务。

[0058] 仅仅作为例子，其中一个服务器计算机130可以是web服务器，仅仅作为例子，web服务器可以被用来处理来自用户计算机110的对网页或其它电子文档的请求。Web服务器还可以运行各种服务器应用，包括HTTP服务器、FTP服务器、CGI服务器、数据库服务器、Java服务器，等等。在本发明的一些实施例中，web服务器可以被配置为提供可以在用户计算机110当中一个或多个上的web浏览器中操作的网页，以执行本发明的方法。

[0059] 在一些实施例中，服务器计算机130可以包括一个或多个文件和/或应用服务器，这可以包括一个或多个可被在用户计算机110和/或其它服务器计算机130当中一个或多个上运行的客户端访问的一个或多个应用。仅仅作为例子，服务器计算机130当中一个或多个可以是能够响应于用户计算机110和/或其它服务器计算机130而执行程序或脚本的一个或多个通用计算机，包括但不限于web应用(在一些情况下，这可以被配置为执行本发明的方法)。

[0060] 仅仅作为例子，web应用可以被实现为以任何编程语言，诸如Java、C或C++，和/或任何脚本语言，诸如Perl、Python或TCL，以及任何编程/脚本语言的组合所写的一个或多个脚本或程序。(一个或多个)应用服务器还可以包括数据库服务器，包括但不限于可从Oracle，Microsoft、IBM等商购的那些，数据库服务器可以处理来自在用户计算机110和/或另一服务器计算机130之一上运行的数据库客户端的请求。

[0061] 在一些实施例中，根据本发明的实施例，应用服务器可以动态地创建网页，用于显示信息。由应用服务器提供的数据可以被格式化为网页(例如，包括HTML、XML、Javascript、AJAX等)和/或可以经由web服务器被转发到其中一个用户计算机110(例如，如上所述)。类似地，web服务器可以从其中一个用户计算机110接收网页请求和/或输入数据和/或将网页请求和/或输入数据转发到应用服务器。

[0062] 根据更多的实施例，服务器计算机130当中的一个或多个可以充当文件服务器和/或可以包括实现由其中一个用户计算机110和/或另一服务器计算机130上运行的应用所结合的本发明方法所必需的文件当中的一个或多个。作为替代，如本领域技术人员将认识到的，文件服务器可以包括所有必需的文件，从而允许这种应用被用户计算机110和/或服务器计算机130当中的一个或多个远程调用。应当注意的是，依赖于特定于实现的需求和参

数，本文关于各种服务器(例如，应用服务器、数据库服务器、web服务器、文件服务器，等等)所描述的功能可以由单个服务器和/或多个专用服务器执行。

[0063] 在某些实施例中，系统100可以包括一个或多个数据库140(例如，数据库140A和140B)。(一个或多个)数据库140的位置是任意的：仅仅作为例子，数据库140A可以驻留在服务器计算机130A(和/或用户计算机110当中的一个或多个)本地的存储介质上(和/或驻留在其中)。作为替代，数据库140B可以远离用户计算机110和服务器计算机130当中任何一个或全部，只要它可以与这些计算机当中一个或多个通信就可以(例如，经由通信网络120)。在特定的一组实施例中，数据库140可以驻留在本领域技术人员熟悉的存储区域网络(“SAN”)中。(同样，用于执行用户计算机110和服务器计算机130所具有的功能的任何必要的文件都可以适当地本地存储在相应的计算机上和/或远程存储。)在一组实施例中，数据库140当中一个或多个可以是适于响应于SQL格式的命令而存储、更新和检索数据的关系数据库。例如，数据库140可以由数据库服务器控制和/或维护，如上所述。

[0064] 数据集成概述

[0065] 图2是根据本发明实施例的数据集成系统200的简化框图。图2是可以结合在本公开内容中给出的一个或多个发明的各种实施例或实现的数据集成系统200的简化说明。图2仅仅是说明本文所公开的发明的实施例或实现不应当限制如权利要求中所述的任何发明的范围。本领域普通技术人员可以通过本公开内容以及本文给出的示教认识到对附图中示出的那些实施例或实现的其它变化、修改和/或备选方案。

[0066] 在这种实施例中，数据集成系统200包括信息源202、信息集成204和信息目的地206。一般而言，信息从信息源202流到信息集成204，由此信息可以被信息目的地206消费、获得或者以别的方式使用。数据流可以是单向的或双向的。在一些实施例中，一个或多个数据流可以在数据集成系统200中存在。

[0067] 信息源202表示被配置为提供数据来源的一个或多个硬件和/或软件元件。信息源202可以提供对数据的直接或间接访问。在这种实施例中，信息源202包括一个或多个应用208和一个或多个储存库210。

[0068] 应用208表示传统应用，诸如桌面、被托管的、基于web的或者基于云的应用。应用208可以被配置为为了一个或多个预定目的而接收、处理和维护数据。应用208的一些例子包括客户关系管理(CRM)应用、金融服务应用、政府和风险合规应用、人力资本管理(HCM)、采购应用、供应链管理应用、项目或投资组合管理应用，等等。应用208可以包括被配置为用于以各种人类可读和机器可读的格式操纵和导出应用数据的功能，如本领域中已知的。应用208还可以访问存储库210中的数据并在其中存储数据。

[0069] 储存库210表示被配置为提供对数据的访问的硬件和/或软件元件。储存库210可以提供数据的逻辑和/或物理分区。储存库210还可以提供报告和数据分析。储存库210的一些例子包括数据库、数据仓库、云存储，等等。储存库可以包括通过集成来自一个或多个应用208的数据所创建的中央储存库。存储在储存库210中的数据可以从操作系统上载。在源中可用之前，数据可以通过附加的操作传递。

[0070] 信息集成204表示被配置为提供数据集成服务的一个或多个硬件和/或软件元件。直接或间接的数据集成服务可以在信息集成204中提供。在这种实施例中，信息集成204包括数据迁移212、数据入库214、主数据管理216、数据同步218、联合220和实时消息传送222。

可以理解，信息集成204可以包括与这里所示的那些不同的、提供数据集成功能的一个或多个模块、服务或其它附加元素。

[0071] 数据迁移212表示被配置为提供数据迁移的一个或多个硬件和/或软件元件。一般而言，数据迁移212提供用于在存储类型、格式或系统之间传送数据的一个或多个过程。数据迁移212通常提供手动或编程选项来实现迁移。在数据迁移过程中，一个系统上的或者由一个系统提供的数据被映射到另一个系统，从而提供用于数据提取和数据加载的设计。数据迁移可以涉及一个或多个阶段，诸如创建将第一系统的数据格式关联到第二系统的格式和需求的一个或多个设计的设计阶段，其中数据从第一系统被读取的数据提取阶段，数据清除阶段，以及其中数据被写到第二系统的数据加载阶段。在一些实施例中，数据迁移可以包括确定数据在上述任何阶段中是否被准确处理的数据验证阶段。

[0072] 数据入库214表示被配置为提供用于报告和数据分析的数据库的一个或多个硬件和/或软件元件。数据仓库通常被视为数据的中央储存库，它是通过集成来自一个或多个不同源的数据创建的。数据入库214可以包括数据的当前存储以及历史数据的存储。数据入库214可以包括典型的基于提取、变换、加载(ETL)的数据仓库，由此中间(staging)层、数据集成层和访问层包含关键功能。在一个例子中，中间层或中间数据库存储从一个或多个不同的源数据系统中每一个当中提取的原始数据。集成层通过变换来自中间层的数据来集成不同的数据集，从而常常将这种变换后的数据存储在操作数据存储(ODS)数据库中。然后，集成的数据被移动到常常被称为数据仓库数据库的另一个数据库。数据可以被布置成分层的组(常常被称为维度)并且被布置成事实和汇总事实。可以提供访问层来帮助用户或其它系统检索数据。数据仓库可以被细分为数据集市，由此每个数据集市存储来自仓库的数据的子集。在一些实施例中，数据入库214可以包括商业智能工具，提取、变换并将数据加载到储存库中的工具，以及管理和检索元数据的工具。

[0073] 主数据管理216表示被配置为管理数据的主拷贝的一个或多个硬件和/或软件元件。主数据管理216可以包括一致地定义和管理主数据的一组过程、管理、策略、标准和工具。主数据管理216可以包括用于除去重复数据、标准化数据并且结合规则以便消除不正确的数据进入系统以便创建主数据的权威来源的功能。主数据管理216可以提供用于收集、汇总、匹配、整合、质量保证、持久化和在整个组织中分发数据以确保正在进行的信息的维护和应用使用中的一致性和控制的过程。

[0074] 数据同步218表示被配置为同步数据的一个或多个硬件和/或软件元件。数据同步218可以提供在从源到目标的数据中建立一致性并且反之亦然。数据同步218还可以提供数据随时间推移的连续协调。

[0075] 联合220表示被配置为整合来自组成源的数据视图的一个或多个硬件和/或软件元件。联合220可以透明地将多个自治的数据库系统映射到单个联合数据库。组成数据库可以经由计算机网络互连并且可以在地理上分散。联合220提供了合并几个完全不同的数据库的备选方案。例如，联合数据库，或虚拟数据库，可以提供所有组成数据库的合成。联合220可能无法提供完全不同的组成数据库中的真正数据集成，而仅仅是在视图中提供。

[0076] 联合220可以包括提供统一用户界面的功能，从而使用户和客户端能够利用单个查询存储和检索多个不连续的数据库中的数据-即使组成数据库是异质的。联合220可以包括将查询分解为子查询以提交到相关的组成数据源并且合成子查询的结果集的功能。联合

220可以包括对子查询的一个或多个封装，以便将其转化为适当的查询语言。在一些实施例中，联合220是通过导出模式的公布和访问操作使它们的数据能够让联合的其它成员获得的自治组件的集合。

[0077] 实时消息传送222表示被配置为提供受实时约束(例如，从事件到系统响应的操作期限)的消息传送服务的一个或多个硬件和/或软件元件。实时消息传送222可以包括保证在严格时间限制内的动作或响应的功能。在一个例子中，实时消息传送222可以负责从一个数据库取一些订单和消费者数据，将其与保存在文件中的一些雇员数据结合，然后将集成的数据加载到Microsoft SQL Server 2000数据库中。因为订单需要在它们到达的时候被分析，所以实时消息传送222可以尽可能接近实时地将订单传递至目标数据库并且只提取新的和改变的数据，以保持工作量尽可能小。

[0078] 信息目的地206表示被配置为存储或消费数据的一个或多个硬件和/或软件元件。在这种实施例中，信息目的地206可以提供对数据的直接或间接访问。在这种实施例中，信息目的地206包括一个或多个应用224和一个或多个储存库226。

[0079] 应用224表示传统应用，诸如桌面、被托管的、基于web的或基于云的应用。应用224可以被配置为为了一个或多个预定目的而接收、处理和维护数据。应用224的一些例子包括客户关系管理(CRM)应用、金融服务应用、政府和风险合规应用、人力资本管理(HCM)、采购应用、供应链管理应用、项目或投资组合管理应用，等等。应用224可以包括被配置为以各种人类可读和机器可读的格式操纵和导入应用数据的功能，如本领域已知的。应用224还可以访问储存库226中的数据并在其中存储数据。

[0080] 储存库226表示提供对数据的访问的硬件和/或软件元件。储存库226可以提供数据的逻辑和/或物理分区。储存库226还可以提供报告和数据分析。储存库226的一些例子包括数据库、数据仓库、云存储，等等。储存库可以包括通过集成来自一个或多个应用226的数据创建的中央储存库。存储在储存库226中的数据可以通过信息集成204被上载或导入。在使其在目的地可用之前，数据可以通过附加的操作传递。

[0081] 数据集成系统

[0082] 图3是可以被用来实现根据本发明实施例的数据集成系统200的硬件/软件栈的简化框图。图3仅仅是说明本文所公开的发明的实施例或实现不应当限制如权利要求中所述的任何发明的范围。本领域普通技术人员可以通过本公开内容以及本文给出的示教认识到对附图中示出的那些实施例或实现的其它变化、修改和/或备选方案。在根据这种实施例的数据集成系统200中找到的组件的一个例子可以包括，ORACLE DATA INTEGRATOR，由位于加州Redwood Shores的Oracle提供的ORACLE FUSION中间件产品系列的成员。ORACLE DATA INTEGRATOR是基于Java的应用，它使用一个或多个数据库执行基于集合的数据集成任务。此外，ORACLE DATA INTEGRATOR可以提取数据，通过Web服务和消息提供变换后的数据，并且创建响应面向服务的体系架构并在其中创建事件的集成过程。ORACLE DATA INTEGRATOR是基于ELT[提取-加载和变换]体系架构而不是常规的ETL[提取-变换-加载]体系架构。用于ORACLE DATA INTEGRATOR的用户手册的拷贝附到本公开内容并且通过引用被结合于此，用于所有目的。

[0083] 在各种实施例中，数据集成系统200提供新的声明设计方法，以定义数据变换和集成过程，从而产生更快更简单的开发和维护。因此，数据集成系统200分离声明规则与实现

细节。数据集成系统200还为数据变换和验证过程的执行提供独特的E-LT体系架构(提取-加载变换)。实施例中的这种体系架构消除了对单独ETL服务器和专用引擎的需求。在一些实施例中,数据集成系统200代替地充分利用RDBMS引擎的固有能力。

[0084] 在一些实施例中,数据集成系统200集成在一个或多个中间件软件包中,诸如ORACLE FUSION MIDDLEWARE平台,并且变成中间件栈的组件。如图3中所绘出的,数据集成系统200可以提供运行时组件作为Java EE应用。

[0085] 在这个例子中,数据集成系统200的一个组件是储存库302。储存库302表示被配置为存储关于IT基础设施、所有应用的元数据、项目、场景和执行日志的配置信息的硬件和/或软件元件。在一些方面,储存库302的多个实例可以在IT基础设施中共存,例如,开发、QA、用户、验收和生产。储存库302被配置为允许交换元数据和场景的几个单独的环境(例如:开发、测试、维护和生产环境)。储存库302还被配置为充当版本控制系统,其中对象被归档并指定版本号。

[0086] 在这个例子中,储存库302由至少一个主储存库304和一个或多个工作储存库306组成。为在数据集成系统200中使用而开发或配置的对象可以存储在这些储存库类型之一当中。一般而言,主储存库304存储以下信息:包括用户、简档和权限的安全信息,包括技术、服务器定义、模式、语境、语言等的拓扑信息,以及版本化和归档的对象。一个或多个工作储存库306可以包含真正的开发对象。

[0087] 几个工作储存器可以在数据集成系统200中共存(例如,具有单独的环境或者匹配特定的版本生命周期)。这一个或多个工作储存库306存储用于模型的信息,包括模式定义、数据存储结构和元数据、字段和列定义、数据质量约束、交叉引用、数据沿袭,等等。这一个或多个工作储存库306还可以存储项目,包括商业规则、包、过程、文件夹、知识模块、变量等,以及场景执行,包括场景、调度信息和日志。在一些方面中,这一个或多个工作储存库306可以只包含执行信息(通常用于生产目的),并且被指定为执行储存库。

[0088] 在各种实施例中,储存库302存储一个或多个ETL项目。ETL项目定义或以别的方式指定建模源或目标中的数据的数据属性的一个或多个数据模型。ETL项目还提供数据质量控制以及定义移动和变换数据的映射。数据完整性控制确保数据的整体一致性。对于由特定源或目标强加的约束和声明规则,应用数据不是总有效。例如,订单可能被发现没有消费者,或者订单行没有产品,等等。数据集成系统200提供检测这些约束违背并且为了回收或报告目的而存储它们的工作环境。

[0089] 在数据集成系统200的一些实施例中,存在两种不同类型的控制:静态控制和流控制。静态控制暗示被用来验证应用数据的完整性的规则的存在。这些规则当中的一些(被称为约束)已经在数据服务器中实现(利用主键、参考约束,等等)。数据集成系统200允许附加约束的定义和检查,而无需在源中直接声明它们。流控制涉及实现其自己的声明规则的变换和集成过程的目标。在将数据加载到目标之前,流控制根据这些约束验证应用的进入的数据。流控制过程一般被称为映射。

[0090] ETL项目可以被自动化到可以为了在运行时环境中执行而部署的包中。因此,数据集成流的自动化通过序列化包中的不同步骤(映射、过程等)的执行并且通过产生包含用于这些步骤当中每一个的就绪代码的生产场景来实现。包通常由组织成执行图的一系列步骤组成。包是被用来生成生产场景的主要对象。它们表示数据集成工作流并且可以执行作业,

诸如像:开始对数据存储或模型的逆向工程设计过程、向管理员发送电子邮件、下载文件并且解压其、定义映射必须以其执行的次序,以及定义利用变化的参数对执行命令迭代的循环。

[0091] 场景被设计为将源组件(映射、包、过程、变量)投产。场景来自对这个组件的代码的生成(SQL、外壳等)。一旦被生成,源组件的代码就被冻结并且该场景被存储在储存库302中,诸如工作储存库306当中的一个或多个。场景可以被导出,然后导入不同的生产环境。

[0092] 在各种实施例中,数据集成系统200以被Java图形模块和调度代理访问的模块化方式围绕储存库302组织。图形模块可以被用来设计和建立存储在储存库302中的一个或多个集成过程。管理员、开发人员和操作人员可以使用开发工作室访问储存库302。代理可以被用来调度和协调与存储在储存库302中的集成过程关联的一组集成任务。例如,在运行时,部署在桌面、web服务上或者以别的方式与源通信的代理协调一个或多个集成过程的执行。代理可以检索存储在主储存库304中的代码、连接到各个源和目标系统,并且编排整体数据集成过程或场景。

[0093] 在这种实施例中,数据集成系统200包括可以包括以上讨论的图形模块和/或代理当中一个或多个的桌面308。桌面308表示一个或多个桌面或工作站计算设备,诸如个人计算机、笔记本电脑、上网本电脑、平板电脑,等等。桌面308包括Java虚拟机(JVM)310和Oracle数据集成器(ODI)工作室312。Java虚拟机(JVM)310是可以执行Java字节码的虚拟机。JVM 310最经常被实现为在现有操作系统上运行,但是也可以被实现为直接在硬件上运行。JVM310提供Java字节码可以在其中被执行的运行时环境,从而启用诸如运行时web服务(WS)314和代理316的特征。JVM 310可以包括Java类库,实现Java应用编程接口(API)的一组标准的类库(在Java字节码中),以及构成Java运行时环境(JRE)的其它元素。

[0094] 代理316被配置为调度和协调与存储在储存库302中的一个或多个集成过程关联的一组集成任务。例如,在运行时,代理协调集成过程的执行。代理可以检索存储在主储存库304中的代码,连接到各个源和目标系统,并且编排整个数据集成过程或场景。

[0095] 再次参考图3,ODI工作室312包括被配置为设计数据集成项目的硬件和/或软件元件。在这个例子中,ODI工作室312包括四个被用来创建和管理数据集成项目的图形模块或导航器,即,设计器模块318、操作器模块320、拓扑模块322和安全模块324。设计器模块318是被配置为定义数据存储(表、文件、Web服务等)、数据映射和包(集成步骤的集合,包括映射)的模块。在各种实施例中,设计器模块318定义用于数据变换和数据完整性的声明规则。因此,项目开发在设计器模块318中发生。此外,数据库和应用元数据在设计器模块318中被导入和定义。在一种实施例中,设计器模块318使用元数据和规则生成数据集成场景或加载生产计划。一般而言,设计器模块318被用来设计数据完整性检查并且建立变换,诸如像:现有应用或数据库的自动逆向工程设计、变换和集成映射的图形开发和维护、映射中数据流的可视化、自动文档生成以及所生成的代码的定制。

[0096] 操作器模块320是被配置为查看和管理生产集成作业的模块。因此,操作器模块320管理和监视生产中的数据集成过程并且可以显示带错误计数、处理的行数、执行统计、被执行的实际代码等等的执行日志。在设计时,开发人员还可以使用操作器模块320联系设计器模块318用于调试目的。

[0097] 拓扑模块322是被配置为创建和管理到数据源和代理的连接的模块。拓扑模块322

定义基础设施的物理和逻辑体系架构。基础设施或项目管理员可以通过拓扑模块322在主储存库中注册服务器、数据库模式和目录以及代理。安全模块324是被配置为管理用户和他们的储存库特权的模块。

[0098] 一般而言，用户或过程与设计器模块318交互，以便为源和目标326创建具有一个或多个数据集成过程的数据集成项目。每个数据集成过程包括至少一个数据集成任务。在一些实施例中，数据集成任务由指示什么数据位要被变换并与其它位组合以及数据如何被真正提取、加载等的技术细节的一组商业规则定义。在优选实施例中，数据集成任务是利用构建数据映射的声明方法指定的。映射是填充被称为目标的一个数据存储的对象，其数据来自被称为源的一个或多个其它数据存储。一般而言，源数据存储中的列通过映射被链接到目标数据存储中的列。作为包步骤，映射可以被添加到包中。如以上所讨论的，包定义数据集成作业。包是在项目之下创建的并且由有组织的步骤序列组成，每个步骤可以是映射或过程。包可以具有入口点和多个出口点。

[0099] 在一些实施例中，当创建新映射时，开发人员或技术业务人员与设计器模块318交互，以便首先定义哪些数据要被集成并且哪些业务规则应当被使用。例如，开发人员可以指定哪些表要被连接，哪些过滤器要被应用，以及哪些SQL表达式要被用来变换数据。被使用的SQL的特定方言(dialect)由代码要在其上执行的数据库平台确定。然后，在单独的步骤中，技术人员可以与设计器模块318交互，以选择最高效的方式来提取、组合，然后集成这种数据。例如，技术人员可以使用特定于数据库的工具和设计技术，诸如增量加载、批量加载实用工具、渐变维度和更改数据的捕获。

[0100] 在这种实施例中，映射可以为源和目标326创建。源和目标326可以包括一个或多个传统应用328、一个或多个文件/XML文档330、一个或多个应用332、一个或多个数据仓库(DW)、商业智能(BI)工具和应用、企业过程管理(EPM)工具和应用334，以及一个或多个JVM 336(包括运行时web服务340和代理342)。

[0101] 图4是在本发明各种实施例中具有可以为其创建数据集成场景的各种异质数据源的环境400的框图。在这个例子中，环境400包括ODI工作室312和储存库302。储存库302包含生成集成场景400所需的所有元数据。用户或过程与ODI工作室312交互，以利用数据完整性控制402和声明规则404创建集成场景400。

[0102] 订单应用406表示用于跟踪消费者订单的应用。创建“订单应用”数据模型来表示存储在订单应用406中的数据以及任何数据完整性控制或条件。例如，“订单应用”数据模型可以基于超结构化查询语言(HSQL)接口并且包括五个数据存储，SRC_CITY、SRC_CUSTOMER、SRC_ORDERS、SRC_ORDER_LINES、SRC_PRODUCT和SRC_REGION。

[0103] 参数文件408表示从生产系统发出的平面文件(例如，ASCII)，该文件包含销售代表的名单以及分成年龄范围的年龄段。在这个例子中，创建“参数”数据模型来表示该平面文件中的数据。例如，“参数”数据模型可以基于文件接口并且包括两个数据存储，SRC_SALES_PERSON和SRC_AGE_GROUP。

[0104] 销售管理应用410表示用于跟踪销售的应用。销售管理应用410可以是用来自订单应用406和参数文件408的数据的变换填充的数据仓库。创建“销售管理”数据模型来表示存储在销售管理应用410中的数据以及任何数据完整性控制或条件或变换。例如，“销售管理”数据模型可以基于超结构化查询语言(HSQL)接口并且包括六个数据存储，TRG_CITY、TRG_

COUNTRY、TRG_CUSTOMER、TRG_PRODUCT、TRG_PROD_FAMILY、TRG_REGION和TRG_SALE。

[0105] 图5A和5B绘出了可以由数据集成系统200执行的常规数据集成处理中的简化数据流。在这个例子中，来自订单应用406、参数文件408和一个或多个其它可选的或附加源的数据流经针对销售管理应用410的传统ETL过程。数据变换发生在单独的ETL服务器500中。该场景需要专用或专有资源，导致较差的性能，并造成高成本。

[0106] 图6A和6B绘出了根据本发明实施例、在可以由数据集成系统200执行的下一代数据集成处理中的简化数据流。在这个例子中，来自订单应用406、参数文件408和一个或多个其它可选的或附加的数据源的数据流经针对销售管理应用410的E-LT过程。数据变换充分利用现有资源，从而导致更高的性能和效率。如上所述，现有的ETL系统需要专用和/或专有的基础设施来执行数据变换。这样做部分地是为了适应未知的用户基础设施。例如，在不知道正在使用什么类型的数据库的情况下，现有的ETL系统不能预料什么变换操作将在给定系统中可用。但是，这导致利用不足的资源，诸如能够在没有任何专用和/或专有的基础设施的情况下执行适当的数据变换的用户的现有数据库和服务器。

[0107] 根据实施例，本发明通过使用户能够根据用户的特定需求定制数据集成过程来充分利用用户的现有基础设施。例如，当设计数据集成计划时，它可以被分成可由单个系统执行的离散部分，称为执行单元。一旦数据集成计划已经被分成多个执行单元，就可以基于用户的基础设施和系统资源向用户给出物理计划。这个计划可以由用户进一步定制，以改变哪些用户系统执行哪些执行单元。例如，可以向用户给出其中连接操作对第一数据库执行的计划，并且用户可以通过将连接操作移动到第二数据库来定制计划。

[0108] 如图6B中所示，这导致不依赖于作为现有ETL系统的特征的独立变换服务器的提取-加载-变换(E-LT)体系架构。相反，如上所述，数据变换可以在用户的现有基础设施上被执行。E-LT体系架构为用户提供更大的灵活性，同时降低与获取并维护专用变换服务器相关联的成本。

[0109] 再次参考图3，代理可以被用来调度和协调与集成过程相关联的一组集成任务。例如，在运行时，代理协调集成过程的执行。代理可以检索存储在主储存库304中的代码，连接到各个源和目标系统，并且编排整体的数据集成过程或场景。在各种实施例中，有两种类型的代理。在一个例子中，独立的代理被安装在桌面308上，诸如代理316。在另一个例子中，应用服务器代理可以被部署在应用服务器326上(诸如部署在Oracle WebLogic服务器上的Java EE代理)并且可以受益于应用服务器层特征，诸如用于高可用性需求的群集。在还有另一个例子中，代理可以被部署在源和目标326上，诸如代理342。

[0110] 在这种实施例中，数据集成系统200包括可包括以上讨论的一个或多个代理的应用服务器344。应用服务器344表示一个或多个应用服务器、web服务器，或被托管的应用。在这个例子中，应用服务器344包括FMW控制台346、servlet(小服务程序)容器348、web服务容器350，以及数据源连接池352。

[0111] FMW控制台346表示被配置为管理应用服务器344的各方面的一个或多个硬件和/或软件元件，其中所述各方面诸如与Servlet容器348、web服务容器350和数据源连接池334相关联的信息。例如，FMW控制台346可以是被用来管理Oracle WebLogic服务器域的基于浏览器的图形用户界面。FMW控制台346可以包括功能，以配置、起动和停止WebLogic服务器实例，配置WebLogic服务器群集，配置诸如数据库连接(JDBC)和消息传送(JMS)的WebLogic服

务器服务，配置安全参数，包括创建和管理用户、组和角色，配置和部署Java EE应用，监视服务器和应用的性能，查看服务器和域日志文件，查看应用部署描述符，以及编辑选定的运行时应用部署描述符元素。在一些实施例中，FMW控制台346包括在生产中向FMW控制台346提供对数据集成过程的访问的ODI插件354，并且可以显示带错误计数、处理的行数、执行统计、所执行的实际代码等等的执行日志。

[0112] Servlet容器348表示被配置为扩充应用服务器344的能力的一个或多个硬件和/或软件元件。Servlet最常被用来处理或存储从HTML表单提交的数据，提供诸如数据库查询的结果的动态内容，并且管理在无状态HTTP协议中不存在的状态信息，诸如将物品填充到适当消费者的购物车中。servlet通常是Java EE中符合Java Servlet API的Java类，其中Java Servlet API是Java类可以通过其对请求作出响应的协议。为了部署和运行servlet，servlet容器348被用作与servlet交互的web服务器的组件。因此，servlet容器348可以扩充由web服务容器350的公共web服务356和数据服务358提供的功能，以及访问由数据源连接池352提供的数据池。Servlet容器348还负责管理servlet的生命周期，将URL映射到特定的servlet并且确保URL请求者具有正确的访问权限。

[0113] 在这个例子中，servlet容器348包括与ODI SDK 362关联的Java EE应用360、ODI控制台364，以及与Java EE代理368关联的运行时web服务366。ODI SDK 362提供用于数据集成和ETL设计的软件开发工具包(SDK)。ODI SDK 362使工作中常见并且很重复的工作自动化，从而允许用户把重复的任务写成脚本。

[0114] ODI控制台364是提供对储存库302的web访问的Java企业版(Java EE)应用。ODI控制台364被配置为允许用户浏览设计时对象，包括项目、模型和执行日志。ODI控制台364可以允许用户查看流映射、追踪所有数据的源并且甚至深入到字段级，以理解被用来建立数据的变换。此外，最终用户可以通过ODI控制台364启动和监视场景执行。在一方面，ODI控制台364为管理员提供查看和编辑诸如数据服务器、物理和逻辑模式的拓扑对象以及管理储存库302的能力。

[0115] 数据场景设计和开发

[0116] 如以上所讨论的，设计场景来将源组件(映射、包、过程、变量)投产。场景来自对这个组件的代码(SQL、外壳等)的生成。场景可以被导出，然后导入不同的生产环境。

[0117] 图7是在根据本发明的一种实施例中数据集成系统的ODI工作室与储存库之间交互的简化框图。在图7所示的实施例中，图3的ODI工作室312使用元数据和规则来生成用于生产的数据集成场景700。一般而言，设计器模块318被用来设计数据完整性检查并建立变换，诸如像：现有应用或数据库的自动逆向工程设计、变换和集成接口的图形开发和维护、接口中数据流的可视化、自动文档生成，以及所生成的代码的定制。

[0118] 图8绘出了根据本发明实施例用于创建数据集成场景的方法800的流程图。图8中所绘出的方法800中处理的实现可以在被诸如计算机系统或信息处理设备的逻辑机器的中央处理单元(CPU或处理器)执行时由软件(例如，指令或代码模块)执行、由电子设备或专用集成电路的硬件组件执行，或者由软件和硬件元件的组合执行。图8中所绘出的方法800开始于步骤810。

[0119] 在各种实施例中，用户可以启动与ODI工作室312的设计器模块318的会话并且连接到储存库302。用户可以与一个或多个用户接口特征交互，以创建新的数据集成项目或者

从存储在例如主储存库304中的现有数据集成项目选择。一般而言，设计器模块318被用来管理元数据、设计数据完整性检查，并建立变换。在各种实施例中，通过设计器模块318被处理的主要对象是模型和项目。数据模型包含数据源或目标(例如，表、列、约束、描述、交叉引用等)中的所有元数据。项目包含用于源或目标的所有加载和变换规则(例如，映射、过程、变量等)。

[0120] 在步骤820中，创建一个或多个数据模型。在步骤830中，创建一个或多个项目。图9是根据本发明实施例用于创建数据集成场景的用户界面的屏幕截图。在这个例子中，导航面板910显示信息并且包括用于与数据模型交互的功能。导航面板920显示信息并且包括用于与项目交互的功能。如以上所讨论的，用户不仅可以创建数据模型，而且可以为数据模型中的数据开发任何数据完整性检查。此外，用户可以为项目指定提供数据完整性的接口、过程、变量并且为从源到目标加载数据的流中的数据提供变换。在步骤840中，生成一个或多个数据集成场景。图8在步骤850结束。

[0121] 图10绘出了根据本发明实施例用于创建映射的方法1000的流程图。图10中所绘的方法1000中处理的实现可以在被诸如计算机系统或信息处理设备的逻辑机器的中央处理单元(CPU或处理器)执行时由软件(例如，指令或代码模块)执行、由电子设备或专用集成电路的硬件组件执行，或者由软件和硬件元件的组合执行。图10中所绘出的方法1000开始于步骤1010。

[0122] 在步骤1020中，接收目标数据存储信息。例如，用户可以与设计器模块318的一个或多个用户接口特征交互，以提供目标数据存储信息。在一种实施例中，用户可以将包括来自导航面板910的一个或多个数据模型的目标数据存储信息拖放到映射或流面板上，其中映射面板可视地表示选定的数据模型的各方面以及任何关联的变换或数据完整性检查。

[0123] 在步骤1030中，接收源数据存储信息。例如，用户可以与设计器模块318的一个或多个用户接口特征交互，以提供源数据存储信息。在一种实施例中，用户可以将包括来自导航面板910的一个或多个数据模型的源数据存储信息拖放到与目标数据存储信息相同的映射或流面板上，其中映射面板可视地表示选定的数据模型的各方面以及任何关联的变换或数据完整性检查。

[0124] 在各种实施例中，源数据存储信息和目标数据存储信息可以由一个或多个数据模型以及可选地操作组成。操作的一些例子可以包括一个或多个数据集操作(例如，联合、连接、交集等)、数据变换、数据过滤操作、约束、描述、交叉引用、完整性检查，等等。在更多实施例中，这些操作当中的一些可以在设计器模块318中被预先配置并可视表示。在其它实施例中，定制的操作可以被提供，从而允许用户指定实现操作的逻辑、映射等。

[0125] 在步骤1040中，接收映射信息。例如，用户可以与设计器模块318的一个或多个用户接口特征交互，以便将源数据存储信息映射到目标数据存储信息。在一种实施例中，用户可以可视地将源数据存储信息中数据元素的属性与目标数据存储信息中数据元素的属性连接。这可以通过匹配源数据存储信息和目标数据存储信息中表的列名来完成。在更多实施例中，一种或多种自动映射技术可以被用来提供映射信息。

[0126] 图11是根据本发明实施例用于在数据集成场景中提供映射信息的用户界面的屏幕截图。在这个例子中，面板1110中源数据存储信息的属性被映射到面板1120中目标数据存储信息的属性。

[0127] 再次参考图10，在步骤1050中，接收数据加载策略。数据加载策略包括关于来自源数据存储信息的实际数据如何在提取阶段被加载的信息。数据加载策略可以在设计器模块318的流选项卡中定义。在一些实施例中，数据加载策略可以依赖于映射的配置为流自动计算。

[0128] 例如，可以为流建议一个或多个知识模块。知识模块(KM)是跨不同的技术实现可重用变换和ELT(提取、加载和变换)策略的组件。在一方面，知识模块(KM)是代码模板。每个KM可以专用于整个数据集合过程中的单独任务。KM中的代码看起来接近它将利用替代方法被执行的形式，从而使其能够一般性地被许多不同的集成作业使用。被生成和执行的代码是从在设计器模块318中定义的声明规则和元数据得出的。其一个例子是通过从Oracle数据库10g捕获的变化数据提取数据并且将变换后的数据加载到Oracle数据库11g中的分区事实表11g中，或者从Microsoft SQL Server数据库创建基于时间戳的提取物并且将这种数据加载到Teradata企业级数据仓库中。

[0129] KM的能力在于它们的可重用性和灵活性-例如，加载策略可以为一个事实表开发，然后该加载策略可以应用到所有其它事实表。在一方面，使用给定KM的所有映射继承对该KM所作的任何改变。在一些实施例中，提供了五种不同类型的KM，它们当中每一个覆盖从源到目标的变换过程中的一个阶段，诸如集成知识模块(IKM)、加载知识模块(LKM)，以及检查知识模块CKM。

[0130] 参考图4，用户可以定义在环境400中从SRC_AGE_GROUP、SRC_SALES_PERSON文件以及从SRC_CUSTOMER表中检索数据的方式。为了定义加载策略，用户可以选择对应于SRC_AGE_GROUP文件的加载的源集合并且选择对应于SQL的LKM文件，以实现从文件到SQL的流。在一方面，LKM负责从远程服务器向临时区域(staging area)加载源数据。

[0131] 在步骤1060中，接收数据集成策略。在定义加载阶段之后，用户定义策略，以采用所加载数据到目标的集成。为了定义集成策略，用户可以选择目标对象并且选择IKM SQL增量更新。IKM负责将最终变换后的数据写到目标。当IKM被起动时，它假设用于远程服务器的所有加载阶段都已经执行了它们的任务，诸如所有远程源数据集都已经被LKM加载到临时区域中，或者源数据存储在与临时区域相同的数据服务器上。

[0132] 在步骤1070中，接收数据控制策略。一般而言，CKM负责检查数据集的记录与既定的约束一致。CKM可以被用来维护数据完整性并参与整体数据质量主动性(initiative)。CKM可以以两种方式被使用。首先，检查现有数据的一致性。这可以在任何数据存储上或者在映射内进行。在这种情况下，被检查的数据是当前在数据存储中的数据。在第二种情况下，目标数据存储中的数据在其被加载之后被检查。在这种情况下，CKM在写到目标之前模拟目标数据存储对结果流的约束。

[0133] 图12是根据本发明实施例用于在数据集成场景中提供流信息的用户界面的屏幕截图。

[0134] 在步骤1080中，接口被生成。图10在步骤1090结束。

[0135] 数据集成场景包和部署

[0136] 如以上所讨论的，数据集成流的自动化可以在数据集成系统200中通过序列化包中的不同步骤(映射、过程等)的执行并且通过产生包含用于这些步骤当中每一个的就绪代码的生产场景来实现。包由组织成执行图的一系列步骤组成。包是被用来生成生产场景的

主要对象。设计场景以将源组件(映射、包、过程、变量)投入生产。场景来自对这种组件的代码(SQL、外壳等)的生成。场景可以被导出,然后导入不同的生产环境。

[0137] 图13绘出了根据本发明实施例用于创建包的方法的流程图。图13中所绘的方法1300中处理的实现可以在被诸如计算机系统或信息处理设备的逻辑机器的中央处理单元(CPU或处理器)执行时由软件(例如,指令或代码模块)执行、由电子设备或专用集成电路的硬件组件执行,或者由软件和硬件元件的组合执行。图13中所绘出的方法1300开始于步骤1310。

[0138] 在步骤1320中,接收包步骤信息。包步骤信息包括识别步骤、元素、属性、组件等的信息。在一个例子中,用户可以与设计器模块318的一个或多个用户接口特征交互,以创建、识别或以其它方式指定用于包的一个或多个步骤。在一种实施例中,一个或多个组件被选择并放在图上。这些组件在包中被示为步骤。

[0139] 在步骤1330中,接收包步骤序列信息。包步骤序列信息包括识别步骤的次序、依赖关系等的信息。一旦步骤被创建,步骤就被排序或重新排序为数据处理链。在一个例子中,用户可以与设计器模块318的一个或多个用户接口特征交互,以提供包的一个或多个步骤的顺序或排序。数据处理链可以包括被定义为第一步骤的独特步骤。一般而言,每个步骤具有一个或多个终止状态,诸如成功或失败。处于一些状态,诸如失败或成功,的步骤后面可以跟者另一个步骤或者包的结束。在一方面,在诸如失败的一些状态的情况下,序列信息可以定义重试次数。在另一方面,包可以具有几个可能的终止步骤。

[0140] 图14是根据本发明实施例用于在数据集成场景中提供包序列信息的用户界面的屏幕截图。

[0141] 在步骤1340中,包被生成。图13在步骤1350结束。

[0142] 如以上所讨论的,数据集成流的自动化可以通过序列化包中不同步骤(映射、过程等)的执行来实现。然后,可以为生产场景产生包含用于包的这些步骤当中每一个的就绪代码的包。在各种实施例中,包被部署成在生产环境中自动运行。

[0143] 图15绘出了根据本发明实施例用于部署数据集成场景的方法1500的流程图。图15中所绘的方法1500中处理的实现可以在被诸如计算机系统或信息处理设备的逻辑机器的中央处理单元(CPU或处理器)执行时由软件(例如,指令或代码模块)执行、由电子设备或专用集成电路的硬件组件执行,或者由软件和硬件元件的组合执行。图15中所绘出的方法1500开始于步骤1510。

[0144] 在步骤1520中,检索集成场景。在一种实施例中,包是从储存库302中检索的。在步骤1530中,集成场景被部署到一个或多个代理。在步骤1540中,集成场景由这一个或多个代理执行。在一方面,集成场景可以以几种途径被执行,诸如从ODI工作室312、从命令行或者从web服务。场景执行可以例如经由操作器模块320等被查看和监视,如以上所讨论的。图15在步骤1550结束。

[0145] 组合的基于流的ETL和基于实体关系的ETL

[0146] 在大多数数据集成系统中,映射需要构成映射的部分的所有输入和输出属性的明确定义。在典型的基于流的ETL工具中,连接器在属性级进行。这导致非常简明的映射模型。但是,这也产生了大量对象并且由于属性级连接器的数目而使构造和维护映射变得繁琐。

[0147] 在各种实施例中,数据集成系统200结合用于使映射的设计和维护容易的一种或

多种技术。组件可以被简单地添加到现有设计，而不需要指定所有输入和输出属性，并且允许组件级的连接器被重新路由。在一方面，提供面向数据集和流的设计的组合，以处理复杂性连同改变。实体关系可以在设计的逻辑视图中指定，因此允许数据存储、连接、过滤和查找被添加或除去，而无需对映射的一般性改变。

[0148] 如在本文所使用的，数据集一般表示来自一组数据存储的数据流。几个数据集可以利用操作，诸如像Union和Intersect之类基于集合的操作符，被合并成接口目标数据存储。在各种实施例中，数据集可以在设计的逻辑视图中被添加、除去和排序。因此，数据集成系统200使用户能够在单个视图中组合基于流的ETL和基于实体关系的ETL。因此，数据集成系统200大大容易了映射的设计和维护。数据集成系统200还使得将组件添加到现有设计变得简单，通常仅需要组件级连接器被重新路由。

[0149] 图16是在根据本发明的一种实施例中组合的基于流且基于实体的映射1600的简化框图。在这个例子中，映射1600包括表示数据源SRC_EMP的组件1610、表示数据集DATASET的数据集1620和表示数据目标TGT_EMPDEPT的组件1630。为了更新数据目标TGT_EMPDEPT，需要数据源SRC_EMP和DATASET的连接。表示JOIN的组件1640被添加到映射1600，其作为输入连接到组件1610和数据集1620并且作为输出连接到组件1630。组件1640被配置为提供连接表达式，诸如(SRC_EMP.DEPTNO＝DATASET.DEPTNO)。

[0150] 在传统的数据集成系统中，映射1600需要构成表示JOIN的组件1640的一部分的所有输入和输出属性的明确定义。与此相反，在各种实施例中，映射开发人员可以在数据集1620中定义实体关系，以提供数据目标TGT_EMPDEPT的列如何直接从流经组件1640并且因此对组件1630可见的、由组件1610表示的数据源SRC_EMP的属性以及由数据集1620表示的DATASET的属性填充。

[0151] 图17绘出了根据本发明实施例用于生成组合的基于流且基于实体的映射的方法1700的流程图。图17中所绘的方法1700中处理的实现可以在被诸如计算机系统或信息处理设备的逻辑机器的中央处理单元(CPU或处理器)执行时由软件(例如，指令或代码模块)执行、由电子设备或专用集成电路的硬件组件执行，或者由软件和硬件元件的组合执行。图17中所绘出的方法1700开始于步骤1710。

[0152] 在步骤1720中，接收一个或多个组件。如以上所讨论的，一些类型的组件影响流经映射的数据的形状，而其它类型的组件控制数据流但是不从根本上改变流的形状。在步骤1730中，接收一个或多个数据集。例如，映射设计器可以添加、编辑或从设计除去数据集。映射设计器可以与关系编辑器交互，以指定数据集中各个属性之间的实体关系。在一方面，数据集成系统200被配置为提取既定的实体关系，以确定将被暴露给设计的下游组件的属性。在步骤1740中，基于组件和数据集，生成映射。在各种实施例中，设计的逻辑视图和物理视图可以被更新，以反映对组件和数据集的改变。在各个方面，数据集成系统200基于流的数据集视图中的得出关系自动生成物理设计。图17在步骤1750结束。

[0153] 数据集成系统200还使得在组件和其它数据集中能够简单地添加到现有的设计，通常只需要组件级连接器被重新路由。例如，如果过滤器组件被添加到设计中，则改变组件级连接器将不需要改变某些下游组件的属性赋值。在另一个例子中，添加另一个数据集允许映射设计器直接从映射的设计视图中指定或声明实体关系。

[0154] 图18是在根据本发明的一种实施例中具有数据集视图的映射1600的简化框图。在

这个例子中，组件1620包括一个或多个实体1810、1820和1830。为了将实体关系添加到映射1600，用户只需要添加或定义实体属性之间的关系，诸如关系1840。在各种实施例中，这种改变将不需要对映射1600中任何下游赋值的改变，因为从一个或多个实体关系产生的输出属性可以直接从设计视图中提供的信息得出。在传统的流工具中，处于列级的一切都将需要通过新数据集的引入被重新链接。

[0155] 图19A和19B是在根据本发明的一种实施例中用于组合的基于流且基于实体的映射的逻辑和物理设计的简化框图。在这个例子中，图19A的视图1910包括表示逻辑设计的流视图中数据源的组件A、B和C以及表示数据目标的组件T。组件A、B、C被表示为描述逻辑设计的数据集视图中实体关系的数据集。因此，数据集具有声明的属性集，如从下游组件——诸如从数据集视图中由映射创建者定义的实体关系描述的组件T——看到的。组件J1和J2表示数据集视图中组件的属性之间的逻辑操作。

[0156] 在这个例子中，图19B的视图1920包括在物理设计的流视图中表示数据源的组件A、B和C以及表示数据目标的组件T。属性集是从数据集视图中定义的实体关系得出的并且被用来创建物理设计。

[0157] 图20绘出了根据本发明实施例用于生成组合的基于流且基于实体的映射的物理设计的方法2000的流程图。图20中所绘的方法2000中处理的实现可以在被诸如计算机系统或信息处理设备的逻辑机器的中央处理单元(CPU或处理器)执行时由软件(例如，指令或代码模块)执行、由电子设备或专用集成电路的硬件组件执行，或者由软件和硬件元件的组合执行。图20中所绘出的方法2000开始于步骤2010。

[0158] 在步骤2020中，接收组件定义。例如，组件定义可以包括规则、操作、过程、变量、序列，等等。在步骤2030中，接收数据集定义。例如，映射设计器可以在逻辑设计的流视图中添加或编辑实体关系。在步骤2040中，基于来自流设计的得出关系信息生成物理设计。图20在步骤2050结束。

[0159] 因此，数据集成系统200使用户能够创建独立于平台和技术的逻辑设计。用户可以创建在高级别定义用户想让数据如何在源和目标之间流动的逻辑设计。鉴于用户的基础设施，工具可以分析逻辑设计，并且创建物理设计。逻辑设计可以包括对应于设计中每个源和目标的多个组件，以及诸如连接或过滤的操作，以及访问点。当被转移到物理设计时，每个组件生成对数据执行操作的代码。依赖于底层技术(例如，SQL Server、Oracle、Hadoop，等等)和所使用的语言(SQL、pig，等等)，由每个组件生成的代码可以不同。

[0160] 因此，数据集成系统200的用户不需要在逻辑设计中预定义数据集组件。数据集成系统200提供允许映射设计器在逻辑设计的数据集视图中声明实体关系的工具。数据集成系统200能够在由预定组件类型表示的操作决定需要什么属性。这简化了设计和维护。

[0161] 实体关系建模

[0162] 关系数据库设计已基于实体关系建模，或E-R建模。按照惯例，E-R设计已被用于描述问题域的静态配置。更动态的方面，诸如从数据存储中提取数据并且将它们“消息传送”到形状中，一般被认为是不同的问题。从90年代中期开始，已经有朝这些所谓“ETL工具”的稳定努力。ETL工具可以帮助人类设计人员创建关于动态数据流的规范，通常被称为ETL模型。

[0163] 图21是绘出静态E-R和动态ETL模型之间的关系的说明。一个有趣的问题是是否有

可能在ETL设计过程中消除所绘出的人为因素。作为替代，以另一种方式提出这个问题，E-R模型包含足够的操作信息使得动态数据流模型可以自动形成而无需人的干预吗？

[0164] 通过利用E-R模型自动化ETL设计过程，有许多好处。一个好处是ETL设计器的生产率。E-R模型可以比ETL过程更容易使其正确。E-R模型还具有数据库工程师理解的标准记号系统。但是这些不能对任何ETL工具说。就算不是全部也有大部分对设计器的一部分需要陡峭的学习曲线。另一个好处是对改变的更好适应性。当完成E-R模型时不需要“中间人”，ETL过程需要也是如此。

[0165] 在各种实施例中，公开了提供从E-R模型到ETL模型的自动转换的技术。这是基于当数据库工程师读E-R图时数据流模型通常已在其脑海里建立的观察。利用这种无声的数据流模型，工程师可以理解E-R模型并且能够与其他人通信。工程师甚至基于这种模型来创建软件。当E-R模型变得复杂时，这个现象更明显。因此，发明人认识到，在每个E-R模型中可以有一个或多个隐藏的数据流模型。在一方面，为E-R模型提供等效数据流模型，这对于引导自动转换系统的创建已经被证明是准确的。

[0166] 图22是在一种实施例中提供自动转换系统2200的顶层设计图的说明。图22可以仅仅是说明本文公开的发明的实施例或实现不应当限制如权利要求中所述的任何发明的范围。本领域普通技术人员将通过本公开内容和本文给出的示教认识到附图中示出的那些实施例或实现的其它变体、修改和备选方案。

[0167] 如图22中所示，E-R模型作为输入连同“用户指令”集合被提供给自动转换系统2200。然后，自动转换系统2200为ETL目的创建等效数据流模型。如本文所使用的，“用户指令”是用户预期数据流模型的计算被考虑的需求集合。例如，用户可以请求关于一系列二进制关系的特定次序，由于逻辑、性能或安全考虑等而请求在指定的机器/位置上处理关系。

[0168] 如本文所使用的，“等效数据流”模型表示用于E-R模型的语义模型。语义模型被用来明确地定义逻辑模型指什么。语义模型可以以多种不同途径来表示，诸如自然语言、集合理论记号、代数方程、数理逻辑或算法记号(一般被称为操作语义)。在各种实施例中，用于E-R模型的语义模型是操作语义模型，被称为“CFO模式”。在一方面，以操作语义格式定义含义提供了双重好处，其一是操作语义模型已经处于与数据流模型一致的逐步形式。其次，与其它正规语义模型相比，操作语义模型容易让人类理解，并且它比自然语言解释更严格。

[0169] E-R模型(或图)中的二进制关系可以一般地映射到ETL模型中的连接。但是，多路关系需要一些工作，因为关于其存在普遍的误解。图23A和23B以两种流行的E-R记号示出了三路关系。参考图23A，模型2310是利用标准E-R记号绘制或另外表示的。在这个例子中，模型2310包括三个实体，PET、PET_TYPE和PET_OWNER，它们在被称为“Pet-of-Type-and-Owner”的三路关系中关联。图23A的直观理解是三实体可以同时交互。

[0170] 但是，在实践当中，不使用标准的E-R记号。相反，更常见的是看所谓的“鱼尾纹(Crow’s Feet)”记号。两种记号系统之间的差异仅仅是表面的。参考图23B，模型2320是利用鱼尾纹记号绘制或另外表示的。在这个例子中，模型2320再次包括三个实体，PET、PET_TYPE和PET_OWNER，它们在被称为“Pet-of-Type-and-Owner”的三路关系中关联，该三路关系被示为在中间具有边角线(corner line)的方框(也被称为关联实体)。创建关联实体，以便同时将三个其它实体绑定到一起。

[0171] 一个常见的误解是错误地将多路关系等同为一系列二进制关系。图24A和24B以两

种流行的E-R记号示出了三路关系的等同物。参考图24A，模型2410是以标准E-R记号利用两个二进制关系被绘制为或另外表示为图23A的模型2310的等同物。参考图24B，模型2420是以鱼尾纹记号利用等效模型被绘制为或另外表示为模型2320的等同物。

[0172] 这两个模型共有相同的问题，因为它们不要求两个二进制关系必须总是同时成立。例如，PET中的实例，称其为“PET A”，可以关联到PET_TYPE中的实例，称其为“PT A”，但是不需要“PET A”必须也关联到来自PET_OWNER的实例。

[0173] 但是，有可能建模宠物必须同时参与两个二进制关系的事实。图25利用一系列二进制关系示出了三路关系的等同物。在这个例子中，模型2500表示与图24B不同的PET实体，作为关联实体。关联实体中的每个实例毫无例外地关联到所有其它连接的实体。图25可以看起来像一系列二进制关系，但它实际上是被伪装的图23B中的三路关系-PET实体吸收图23B中所示的关联实体。

[0174] 在一方面，存在两种其中PET实体有可能吸收关联实体的特殊情况。首先，一种可能性是每个PET实例参与不多于一个关系实例。其次，另一种可能性是如果PET是弱实体。(弱实体的正式定义是不具有其自己的主键的实体。)假定PET是强实体，那么其自己的主键将必须被用来只识别宠物。它不能被也用来识别关系实例。例如，如果宠物实例参与多于一个关系实例，则在强PET实体中将存在主键冲突。在另一方面，如果PET是弱实体，则其部分键(不唯一)可以与三元关系的键(或者部分或者唯一)组合。在这种情况下，PET吸收该三元关系。

[0175] 因此，不作附加的假设，图23B不能被演变(morphed)成像一系列二进制关系。因此，焦点转向图23B中所绘出的通用三元关系。

[0176] 图26利用标准的E-R记号示出了三路关系。用于该例的模式包括PET、PET_TYPE、PET_OWNER实体加上关于它们的一个或多个三元关系。还提供了一些附加信息。在这个例子中，模型2600将PET表示为三元关系的可选参与者，如由E-R图中的基数范围0…m指示的。其它两个实体都是关系的完全参与者。

[0177] 假定p、t、o分别是PET、PET_TYPE和PET_OWNER的实例。以下是“Pet-of-Type-and-Owner”关系中的可能实例：

[0178] • (p,t,o)

[0179] • (<missing>,t,o)

[0180] 在这里，<missing>表示来自实体的值的不存在。这些候选元组是否是有效的关系实例是由如下定义的三路连接条件确定的：

[0181] PET.type_id＝PET_TYPE.id and PET.owner_id＝PET_OWNER.id

[0182] 注意：值<missing>能够匹配任何其它值。因此，在这个例子中，元组(<missimg>,t,o)是关系的有效实例，因为以下条件被评估为真。

[0183] <missing>＝PET_TYPE.id and<missing>＝PET_OWNER.id

[0184] 用于实体的三个示例表是通过以下语句创建的。

[0185] create table PET(

[0186] id number,

[0187] name varchar2(30),

[0188] tid number,

[0189] oid number);

[0190] create table PET_TYPE(

[0191] id number,

[0192] name varchar2(30));

[0193] create table PET_OWNER(

[0194] id number,

[0195] name varchar2(30));

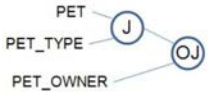
[0196] 图27绘出了每个表中的行。如图所示,每个实体只有一个实例。在一方面,用户可以输入三路连接条件:

[0197] PET.type_id=PET_TYPE.id and PET.owner_id=PET_OWNER.id

[0198] 用户还可以将实体PET标记为可选。这等同于输入图26中所示的E-R模型。一个挑战是如何生成最佳捕捉图26的意义的SQL语句。在各种实施例中,确定将提供良好的多路连接实现的语法。在以下的例子中,使用ANSI连接语法。

[0199] 由于每个ANSI连接是按对的,因此,为了连接三个表,需要两个连接。而且,由于PET是可选实体,因此两个连接当中至少一个必须是外连接。此外,哪两个表先连接也是要考虑的因素。将所有这些考虑放到一起,遇到九种排列,对应于利用ANSI语法的多路连接的九种可能实现。这些情况利用数据流图绘制,并且在下表1中附带其SQl语句和结果示出。

[0200]

	流图	SQL	结果
1		select p.name "pet", t.name "type", o.name "owner" from PET p join PET_TYPE t on (p.tid = t.id) right outer join PET_OWNER o on (p.oid = o.id)	pet type owner -------- -------- ---------- Jeff (注:Type 是无效的。这个不是好结果。)
2		select p.name "pet", t.name "type", o.name "owner" from PET p right outer join PET_TYPE t on (p.tid = t.id) join PET_OWNER o	未选择行。

[0201]

		on (p.oid = o.id)	
3		select p.name "pet", t.name "type", o.name "owner" from PET p right outer join PET_TYPE t on (p.tid = t.id) right outer join PET_OWNER o on (p.oid = o.id)	pet type owner -------- -------- ---------- Jeff (注：Type 是无效的。这个不是好结果。)
4		select p.name "pet", t.name "type", o.name "owner" from PET p join PET_OWNER o on (p.oid = o.id) right outer join PET_TYPE t on (p.tid = t.id)	pet type owner -------- -------- ---------- Cat (注：Owner 是无效的。这个不是好结果。)
5		select p.name "pet", t.name "type", o.name "owner" from PET p right outer join PET_OWNER o on (p.oid = o.id) join PET_TYPE t on (p.tid = t.id)	未选择行。
6		select p.name "pet", t.name "type", o.name "owner" from PET p right outer join PET_OWNER o on (p.oid = o.id) right outer join PET_TYPE t on (p.tid = t.id)	pet type owner -------- -------- ---------- Cat (注：Owner 是无效的。这个不是好结果。)
7		select p.name "pet", t.name "type", o.name "owner" from PET_TYPE t join PET_OWNER o on (1 = 1) left outer join PET p on (p.tid = t.id and p.oid = o.id)	pet type owner -------- -------- ---------- Cat Jeff (注：这是我们期望的结果。值得注意的是：在 PET_TYPE 和 PET_OWNER 之间没有之间连接条件。其缺省值为真。)
8		select p.name "pet", t.name "type", o.name "owner"	未选择行。

[0202]

		from PET_TYPE t full outer join PET_OWNER o on (1 = 1) join PET p on (p.tid = t.id and p.oid = o.id)	
9		select p.name "pet", t.name "type", o.name "owner" from PET_TYPE t full outer join PET_OWNER o on (1 = 1) left outer join PET p on (p.tid = t.id and p.oid = o.id)	pet type owner -------- -------- ---------- Cat Jeff (注：所生成的 SQL 等同于实现#7 中的 SQL。)

[0203] 根据对所有可能实现的检查，实现#7看起来匹配对三路关系的预期。因此，一般而言，多路关系不等同于一系列二进制关系。但是，在各种实施例中，多路关系可以利用二进制连接来实现。因此，在一方面，创建(临时)人类用户不可理解但是对于在生成正确的数据流实现中的使用来说严格的模型。

[0204] 如以上所讨论的，等效数据流模型适合“操作语义模型”类别。已经创建了明确描述系统的意义/意图的操作语义模型。但是，用于等效地表示E-R模型的操作语义模型提供了如本文所讨论的新机会。

[0205] 图28A和28B示出了一种实施例中利用E-R记号的三路关系以及具有源自三个实体的数据的数据流。利用图28A的PET例子，图28B描述了具有源自三个实体的数据的数据流。每个实体提供元组集合。每个元组由列/属性的列表组成。所有元组都通过三个阶段：以下定义的连接、过滤和输出阶段。

[0206] 连接阶段：执行所有输入实体的笛卡尔乘积。如果实体是可选实体(短暂定义的)，则在执行笛卡尔乘积之前所有列都具有值<missing>的特殊元组作为该实体的额外成员首先被添加。

[0207] 过滤阶段：在过滤阶段，出自连接阶段的所有元组被归为三组：

[0208] • 组F包括不满足关系条件的元组。

[0209] • 组S1包括在不比较任何<missing>值的情况下满足关系条件，例如PET.tid＝PET_TYPE.id and PET.oid＝PET_OWNER.id，的元组。

[0210] • 组S2包括满足关系条件但是在比较中使用补充值<missing>的所有其它元组。

[0211] 直观地，组S1包括直接得分成功(scored straight success)的行。并且，由于来自可选实体的可忽略的缺失值，组S2通过了连接条件。

[0212] 输出阶段：利用以下规则输出最终的结果，该最终结果是元组的集合：

[0213] • 来自组F的所有元组都被丢弃。

[0214] • 来自组S1的所有元组都包括在最终结果集中。

[0215] • 来自组S2的元组只有在其对最终结果具有实质贡献时才被包括在最终结果集中。

[0216] 如果元组匹配结果集中的一个元组，则该元组被认为对最终结果没有实质贡献。

在检查两个元组是否匹配时，我们假设<missing>值匹配任何其它值。例如，以下两个元组匹配。

[0217] ('ABC',123)对(<missing>,123)

[0218] 直观地，最终的输出阶段对组S1和S2中的元组执行重复删除。

[0219] 利用图27中的示例数据，连接阶段的结果包含以下两个元组：

[0220] (pet_100,pet_type_1,pet_owner_10)

[0221] (<missing>,pet_type_1,pet_owner_10)

[0222] 在这里，我们使用pet_100表示PET表中具有id＝100的行。应当注意，值<missing>被看作“有效的”宠物，因为PET是可选实体。

[0223] 在第二个阶段，评估多路连接条件PET.tid＝PET_TYPE.id and PET.oid＝PET_OWNER.id。并且只有元组(<missing>,pet_type_1,pet_owner_10)满足该条件。

[0224] 最后一个阶段对这个例子是无关紧要的，因为不需要进行任何重复删除。

[0225] 表1中的实现#7能够返回正确的结果的原因是它在开始评估连接条件之前进行了所有表的笛卡尔乘积。它是与既定的操作语义模型一致的唯一实现。通过确保笛卡尔乘积操作在行在数据流中被过滤之前很好地完成，保护多路关系中固有的同时性属性免于有可能为破坏性的二进制连接。

[0226] 在一些实施例中，用户可能能够通过在指示需要连接阶段操作的实体之间画线来可视地创建模型。例如，假定用户只在PET和PET_OWNER之间画了连接，但是他输入了如下关系条件：

[0227] PET.tid＝PET_TYPE.id and PET.oid＝PET_OWNER.id

[0228] 在看到上面的3-路关系条件后，连接可以在PET和PET_TYPE之间被自动确定并创建。这是因为连接阶段操作需要所有涉及的实体的笛卡尔乘积。仅仅为了实现笛卡尔乘积，关于这个得出的连接的连接条件是1＝1。

[0229] 还假定用户从PET_OWNER到PET_TYPE画了附加的线，从而在三个实体之间形成圆圈。在一方面，产生该圆圈的新线可以被忽略，因为该关系中的所有实体都已经被充分连接。对于人类用户，他可能认为线意味着“二进制关系”，但是通过一直都保持符合图28B的操作语义模型:线仅仅意味着利用笛卡尔乘积将实体连接到一起。

[0230] 在实体被连接之后，图可以被变换成二进制连接的树，其中用于PET和PET_TYPE的连接节点支持1＝1条件。并且多路连接条件被延迟到最后一个连接节点。在整个过程中，连接条件未被掩饰，而是最大限度地被延迟，以确保来自所有表的所有行都有机会交互。

[0231] 相反，如果连接条件被分成两部分(在语法上被允许)，并且将两个子条件分配给两个连接节点，则操作语义模型将被违背，因为过滤阶段将在连接阶段完成之前开始。

[0232] 因此，由于操作语义模型以具体的逐步方式被指定，因此它可以容易地利用任何现有的编程语言被变换成可编程的实现。不需要只使用SQL来实现其。

[0233] 图29绘出了展开各种数据库建模方法及其语义内容之间的关系的图。在这个例子中，E-R模型仍然需要来自语义模型的帮助。CFO模型是这样一种用于消除E-R中的歧义的语义模型，尤其是对于多路关系。如图29中所示，面向对象的模型可以被用于相同的目的。对于正确地将E-R模型转换成OO模型，存在许多专利。但是OO模型缺乏与数据流模型集成的能力。数据流模型明确地写出从源到目标的数据的逐步操作。对那个目的，OO模型仍然太过描

述性。CFO模型就像自动机器，它固有地适于与数据流模型集成。

[0234] 结论

[0235] 图30是可以被用来实践本发明的实施例的计算机系统3000的简化框图。如图30中所示，计算机系统3000包括经由总线子系统3020与多个外围设备通信的处理器3010。这些外围设备可以包括包含存储器子系统3040和文件存储子系统3050的存储子系统3030、输入设备3060、输出设备3070，以及网络接口子系统3080。

[0236] 总线子系统3020提供让计算机系统3000的各种组件和子系统彼此如预期地通信的机制。虽然总线子系统3020示意性地示为单条总线，但总线子系统的备选实施例可以利用多条总线。

[0237] 存储子系统3030可以被配置为存储提供本发明的功能的基本编程和数据结构。提供本发明的功能的软件(代码模块或指令)可以存储在存储子系统3030中。这些软件模块或指令可以由(一个或多个)处理器3010执行。存储子系统中3030还可以提供用于存储根据本发明被使用的数据的储存库。存储子系统3030可以包括存储器子系统3040和文件/盘存储子系统3050。

[0238] 存储器子系统3040可以包括多个存储器，包括用于在程序执行期间存储指令和数据的主随机存取存储器(RAM)3042，以及其中存储固定指令的只读存储器(ROM)3044。文件存储子系统3050为程序和数据文件提供持久性(非易失性)存储，并且可以包括硬盘驱动器，连同关联的可移动介质的软盘驱动器、光盘只读存储器(CD-ROM)驱动器、DVD、光盘驱动器、可移动介质盒，以及其它类似的存储介质。

[0239] 输入设备3060可以包括键盘、诸如鼠标、轨迹球、触摸板或图形输入板的定点设备、扫描仪、条形码扫描仪、结合到显示器中的触摸屏、诸如语音识别系统的音频输入设备、麦克风，以及其它类型的输入设备。一般而言，术语“输入设备”的使用意在包括用于将信息输入计算机系统3000的所有可能类型的设备和机构。

[0240] 输出设备3070可以包括显示器子系统、打印机、传真机，或者诸如音频输出设备的非可视显示器，等等。显示子系统可以是阴极射线管(CRT)、诸如液晶显示器(LCD)的平板设备，或投影设备。一般而言，术语“输出设备”的使用意在包括从计算机系统3000输出信息的所有可能类型的设备和机构。

[0241] 网络接口子系统3080提供到其它计算机系统、设备和诸如通信网络3090的网络的接口。网络接口子系统3080充当从计算机系统3000接收数据和向其它系统发送数据的接口。通信网络3090的一些例子是专用网络、公共网络、租用线路、互联网、以太网、令牌环网、光纤网络等。

[0242] 计算机系统3000可以是各种类型，包括个人计算机、便携式计算机、工作站、网络计算机、大型机、信息站或任何其它数据处理系统。由于计算机和网络的不断变化的本质，在图30中所绘出的计算机系统3000的描述仅仅是作为用于说明计算机系统的优选实施例的具体例子。具有比图30中所绘的系统更多或更少组件的许多其它配置是可能的。

[0243] 图31是根据本发明实施例用于促进数据映射的生成的数据集成系统3100的简化框图。数据集成系统3100的方框可以由硬件、软件或者硬件和软件的组合来实现，以执行本发明的原理。本领域技术人员应当理解，图31中所描述的方框可以组合或分离成子方框，以实现如上所述本发明的原理。因此，本文的描述可以支持本文所述功能框的任何可能的组

合或分离或者进一步定义。

[0244] 如图31中所示，数据集成系统3100被示为包含接收单元3110、确定单元3120和生成单元3130。可选地，数据集成系统3100还可以包括得出单元3140和导出单元3150。

[0245] 在一种实施例中，接收单元3110被配置为接收指定实体关系集合的信息作为逻辑设计的组件。确定单元3120被配置为基于该实体关系集合确定等效数据流模型。生成单元3130被配置为在逻辑流设计中生成指示该等效数据流模型的信息。

[0246] 在实施例的一方面，得出单元3140被配置为基于声明数据源的属性之间的关系的信息得出代表该实体关系集合的数据集的一个或多个属性。在实施例的一方面，接收单元3110还被配置为接收指定逻辑设计的一个或多个组件的信息，所述信息包括指示改变流经该逻辑设计的信息的形状的操作的信息。

[0247] 在实施例的一方面，接收单元3110还被配置为接收指定逻辑设计的一个或多个组件的信息，所述信息包括指示控制流经该逻辑设计的信息流但是不改变流经该逻辑设计的信息的形状的操作的信息。在实施例的一方面，接收单元3110还被配置为接收指定逻辑设计的一个或多个组件的信息，所述信息包括指示具有要存储在目标数据存储中的数据的一个或多个属性的目标组件的信息。

[0248] 在实施例的一方面，生成单元3130包括被配置为将属性列表导出到下游组件的导出单元3150。在实施例的一方面，接收单元3110还被配置为通过一个或多个关系的引入来接收逻辑设计中的改变；并且确定单元3120还被配置为确定更新的等效数据流模型。

[0249] 虽然本发明的具体实施例已经进行了描述，但是各种修改、变更、替换构造和等同物也包括在本发明的范围之内。所描述的发明并不限于某些特殊数据处理环境中的操作，而是可以自由地在多种数据处理环境中操作。此外，虽然本发明已经利用特定的一系列事务和步骤进行了描述，但是对本领域技术人员应当显然，本发明的范围不限于所描述的事务和步骤序列。

[0250] 另外，虽然本发明已经利用硬件和软件的特定组合进行了描述，但是应当认识到，硬件和软件的其它组合也在本发明的范围之内。本发明可以仅以硬件，或仅以软件，或使用它们的组合来实现。

[0251] 因此，本说明书和附图应当被认为是说明性而不是限制性的。但是，显而易见的是，在不背离如权利要求中阐述的本发明的更宽的精神和范围的情况下，可以对其进行添加、减少、删除以及其它修改和变化。

[0252] 其示教可以在本公开内容中给出的一个或多个发明当中任何一个的各种实施例可以以软件、固件、硬件或者其组合以逻辑的形式实现。逻辑可以作为适于指示逻辑机器的中央处理单元(CPU或处理器)执行一组步骤的一组指令存储在机器可存取存储器、机器可读制品、有形的计算机可读介质、计算机可读存储介质或其它计算机/机器可读介质当中或之上，其中这组步骤可以在本公开内容中给出的发明的各种实施例中公开。当代码模块利用计算机系统或信息处理设备的处理器变得可操作时，逻辑可以构成软件程序或计算机程序产品的一部分，当逻辑被执行时，执行在本公开内容中给出的发明的各种实施例中的方法或过程。基于本公开内容和本文提供的示教，本领域普通技术人员将认识到用于以软件、固件、硬件或者其组合实现所给出的一个或多个发明的各种实施例的所公开操作或功能当中任何一个的其它方式、变化、修改、备选方案和/或方法。

[0253] 其示教可以在本公开内容中给出的那些发明当中任何一个的所公开的例子、实现和各种实施例仅仅是说明性的，以便以合理的清晰度向本领域技术人员传达本公开内容的示教。由于这些实现和实施例可以参照示例性说明和具体附图描述，因此所描述的方法和/或具体结构的各种修改和适配会对本领域技术人员变得显然。依赖于本公开内容和在本文发现的这些示教并且所述示教通过其可以推动本领域的所有这种修改、适配或变化都应当被认为在其示教可以在本公开内容中存在的一个或多个发明的范围之内。由此，本描述和附图不应当在限制性的意义上考虑，因为应当理解，公开内容中给出的发明不是要以任何方式限定到具体说明的那些实施例。

[0254] 因此，以上描述和任何附图、说明和图示意在说明而不是限制。因此，本公开内容中给出的任何发明的范围不应当简单地参考以上描述和图中所示的那些实施例来确定，而是应当参考未决的权利要求连同它们的完全范围或等同物来确定。

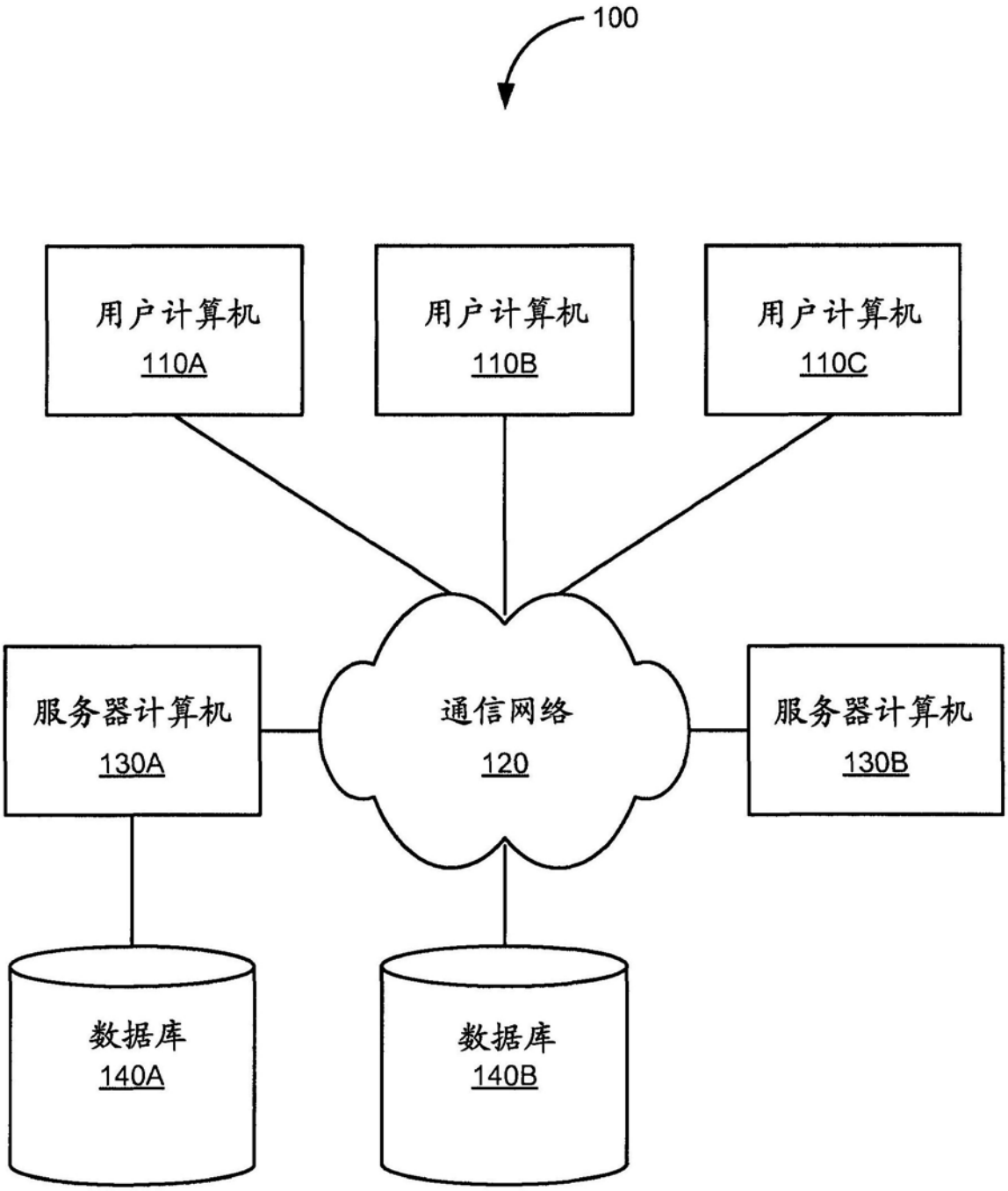

100
用户计算机
110A
用户计算机
110B
用户计算机
110C
服务器计算机
130A
通信网络
120
服务器计算机
130B
数据库
140A
数据库
140B

图1

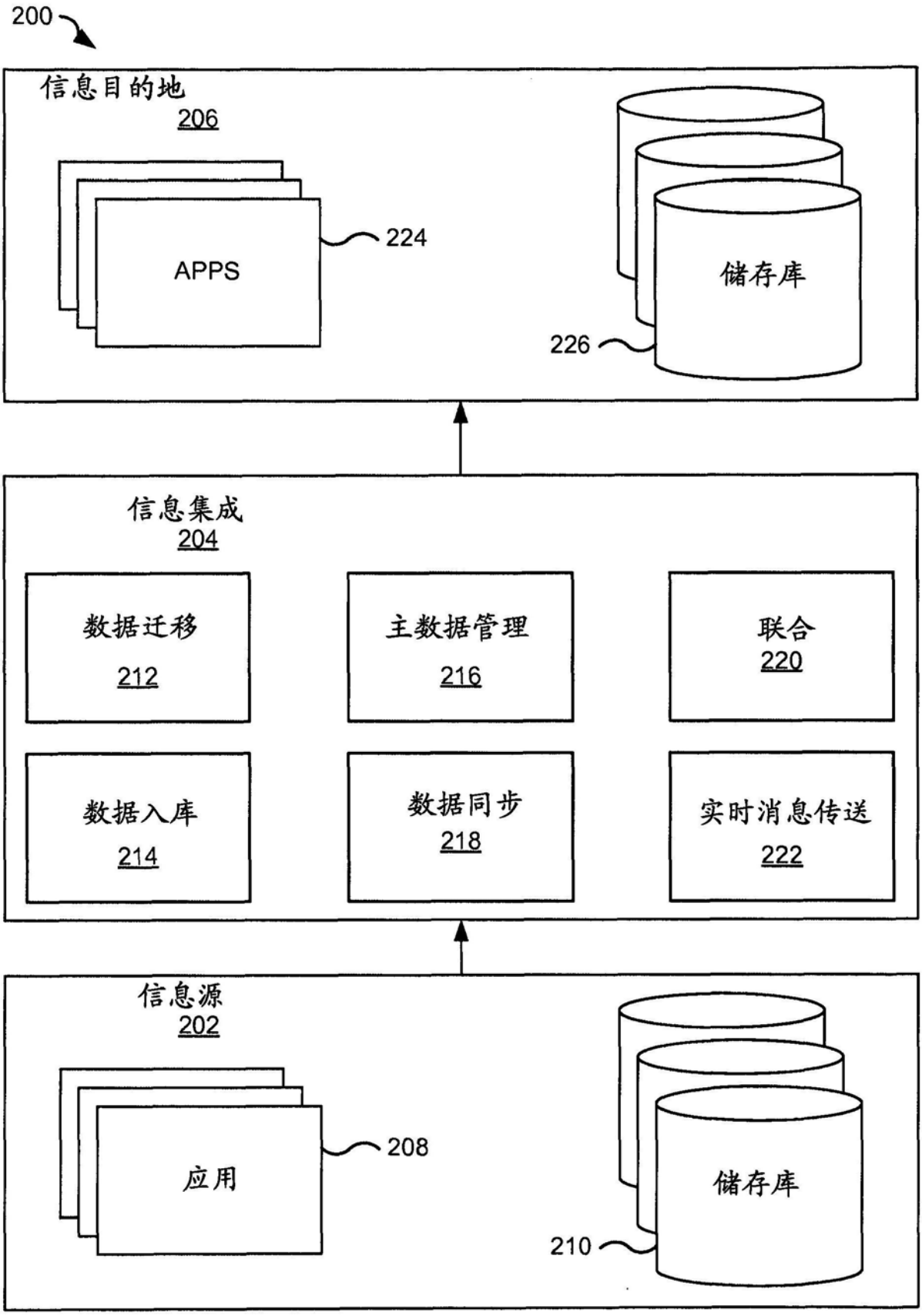

200
信息目的地
206
APPS
224
储存库
226
信息集成
204
数据迁移
212
主数据管理
216
联合
220
数据入库
214
数据同步
218
实时消息传送
222
信息源
202
应用
208
储存库
210

图2

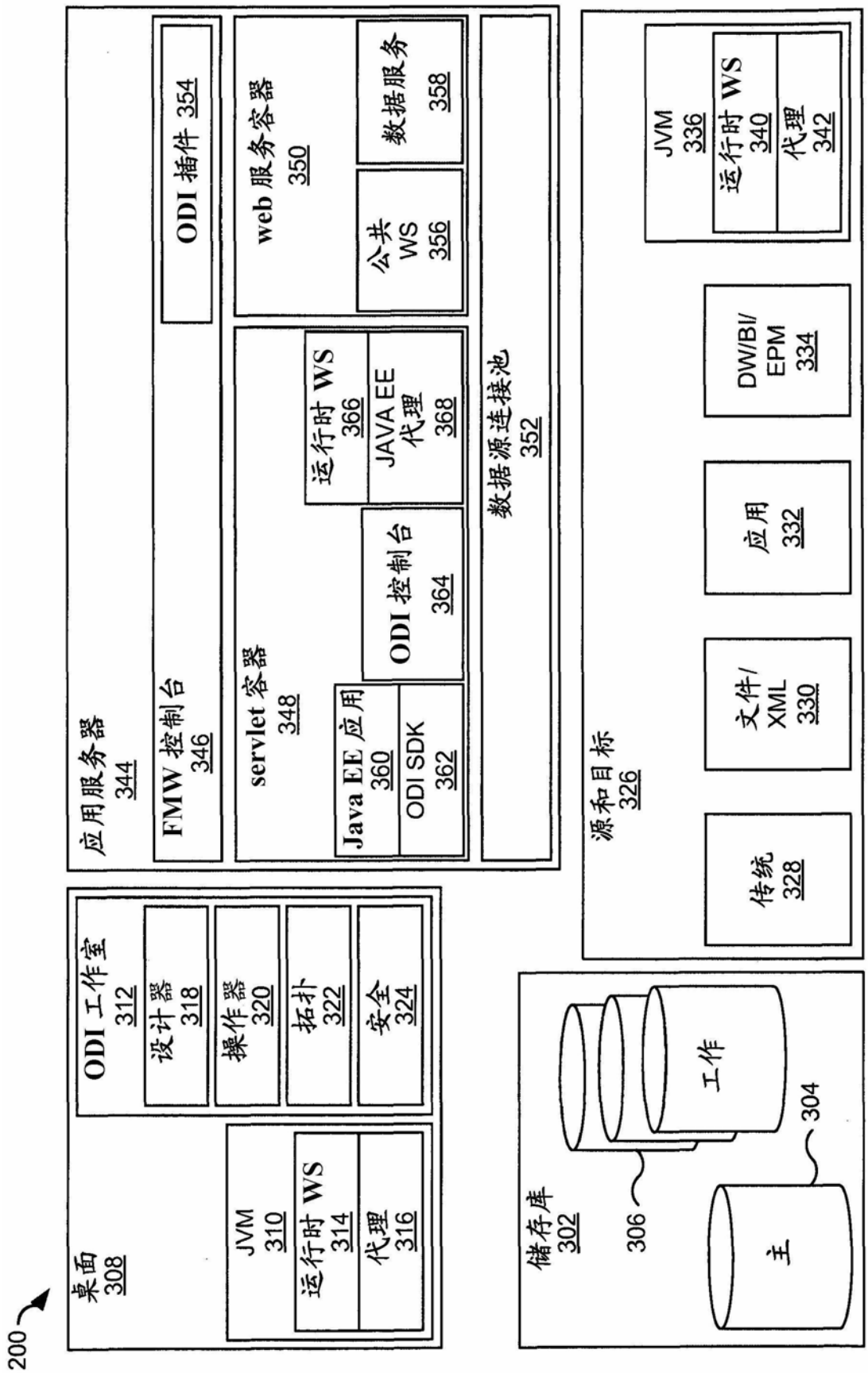

200
桌面
308
JVM
310
运行时 WS
314
代理
316
ODI 工作室
312
设计器
318
操作器
320
拓扑
322
安全
324
应用服务器
344
FMW 控制台
346
ODI 插件 354
servlet 容器
348
Java EE 应用
360
ODI SDK
362
ODI 控制台
364
运行时 WS
366
JAVA EE
代理
368
web 服务容器
350
公共
WS
356
数据服务
358
数据源连接池
352
储存库
302
306
主
工作
304
源和目标
326
传统
328
文件/
XML
330
应用
332
DW/BI/
EPM
334
JVM
336
运行时 WS
340
代理
342

图3

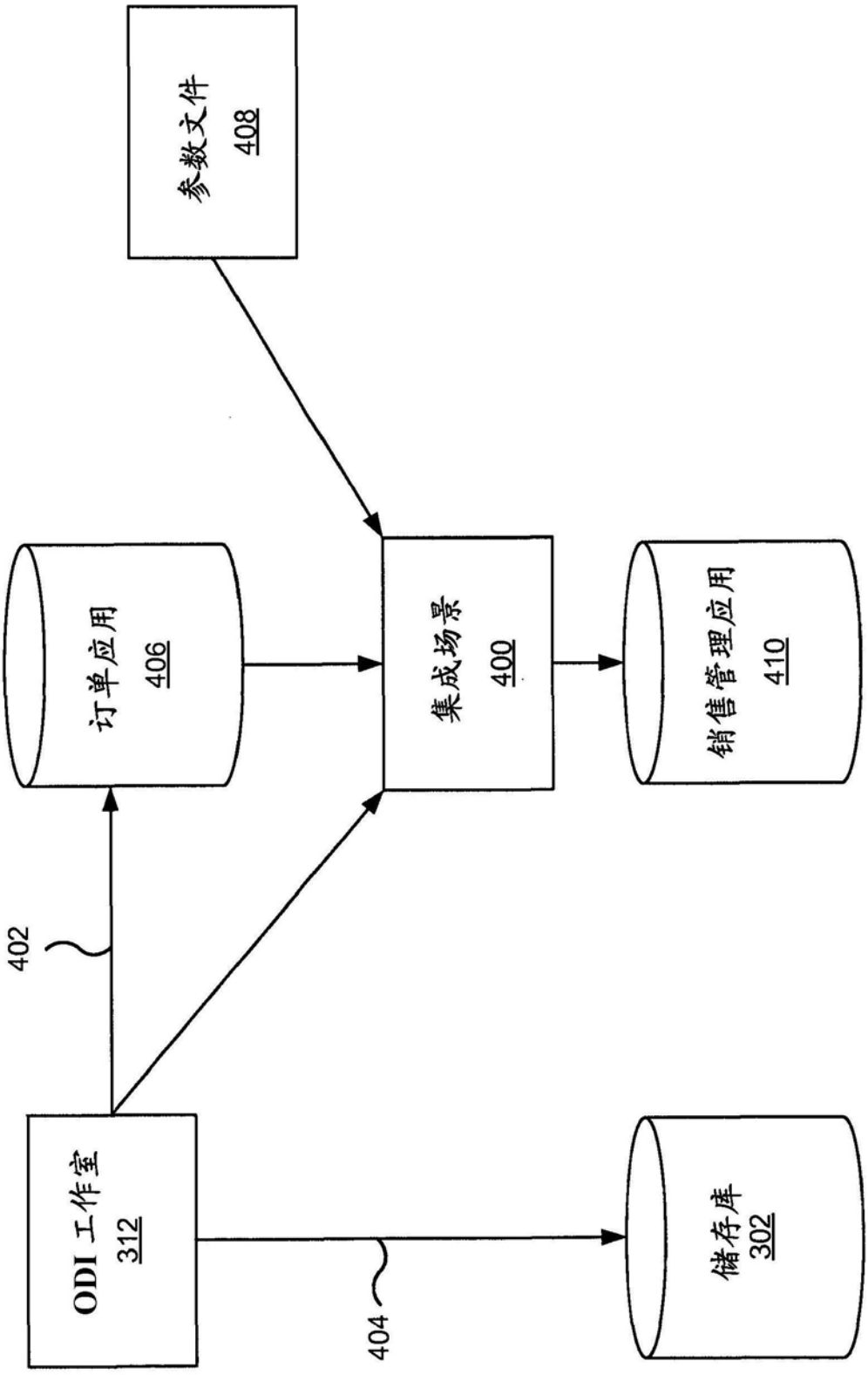

参数文件
408
订单应用
406
集成场景
400
销售管理应用
410
402
ODI 工作室
312
404
储存库
302

图4

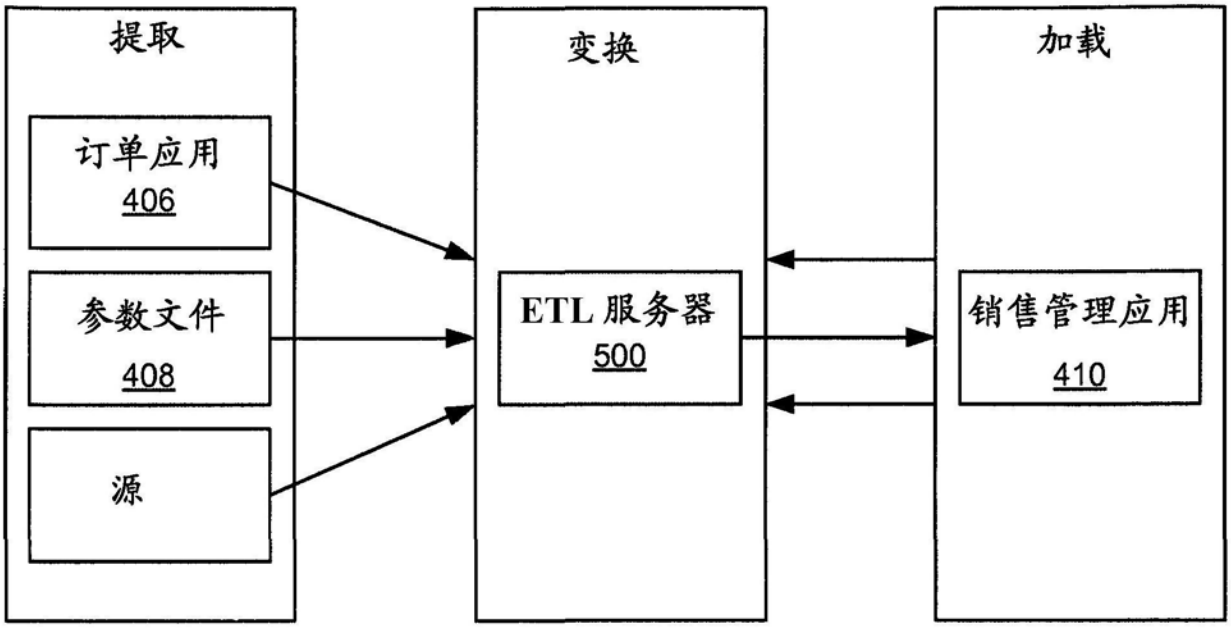

提取
订单应用
406
参数文件
408
源
变换
ETL 服务器
500
加载
销售管理应用
410

图5A

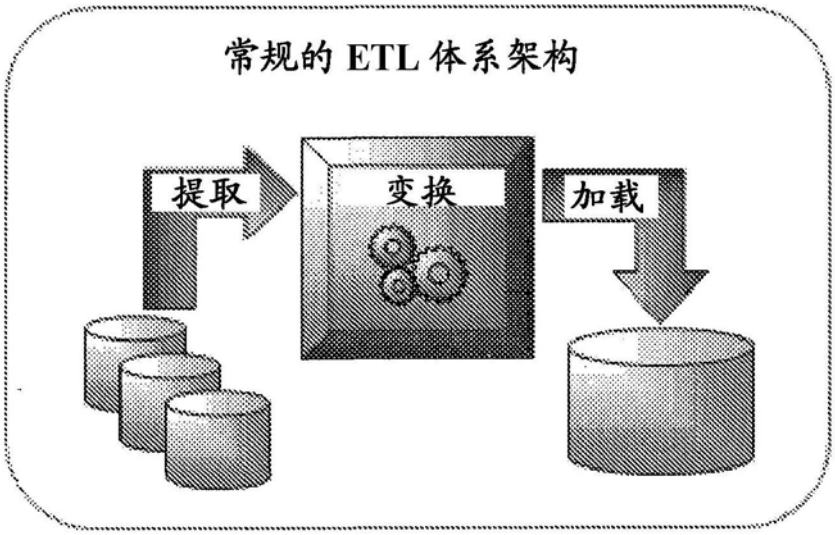

常规的 ETL 体系架构
提取
变换
加载

图5B

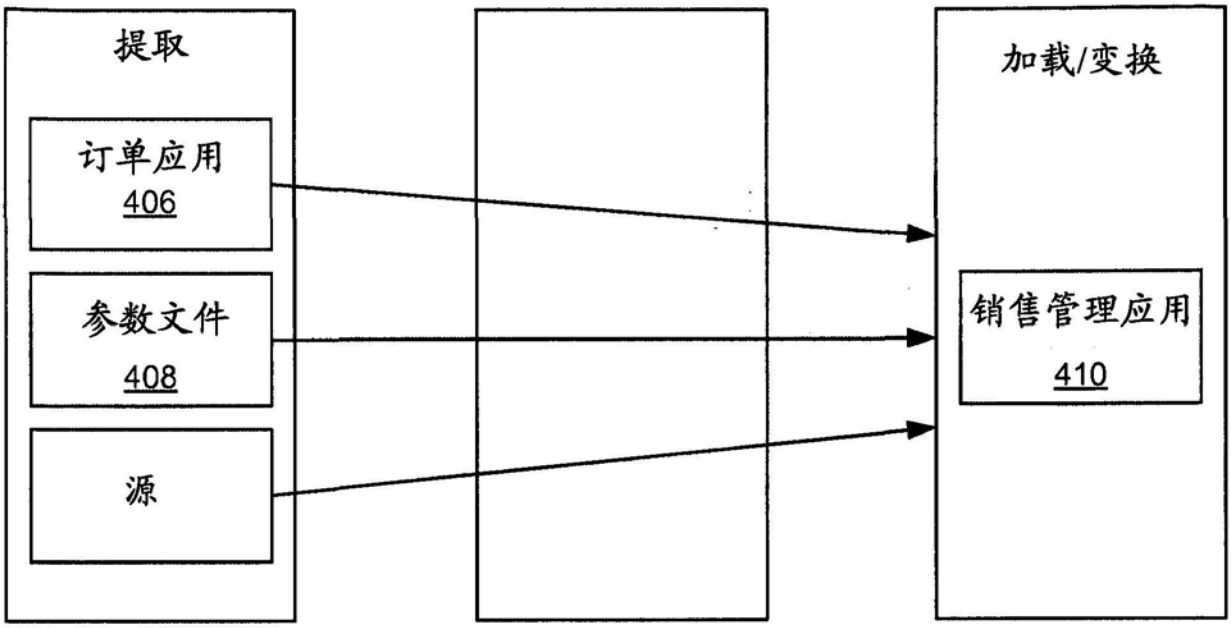


图6A

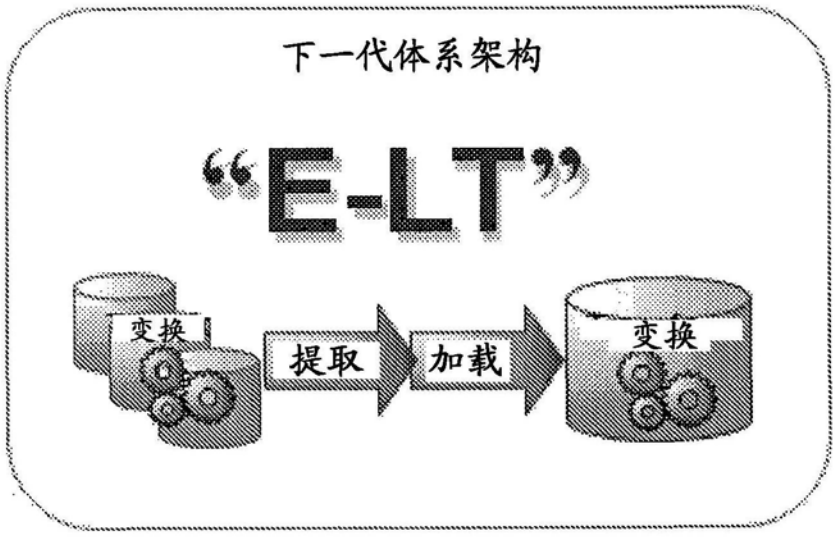


图6B

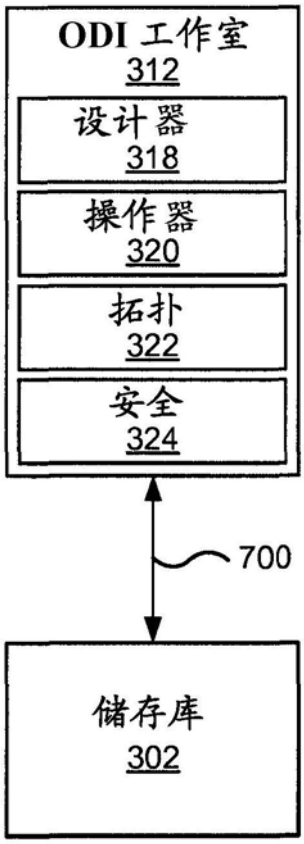


图7

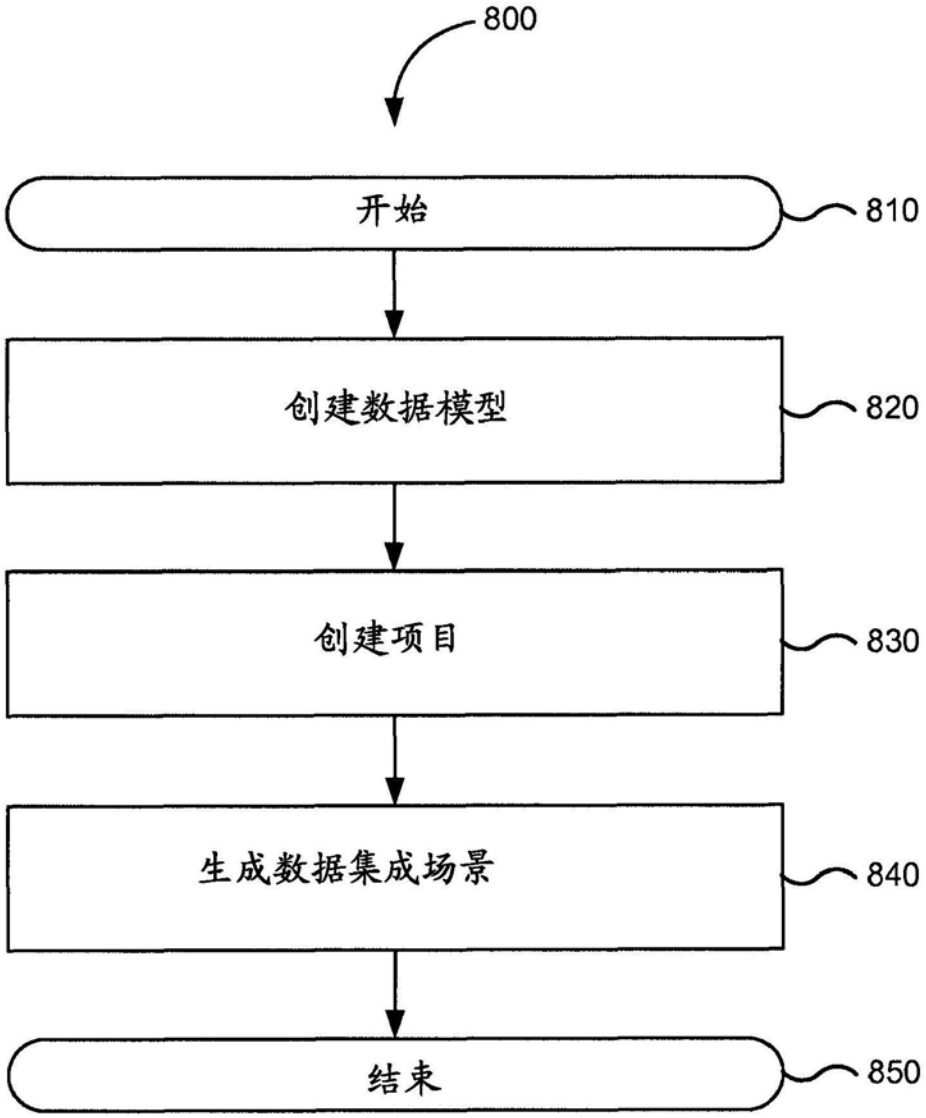


图8

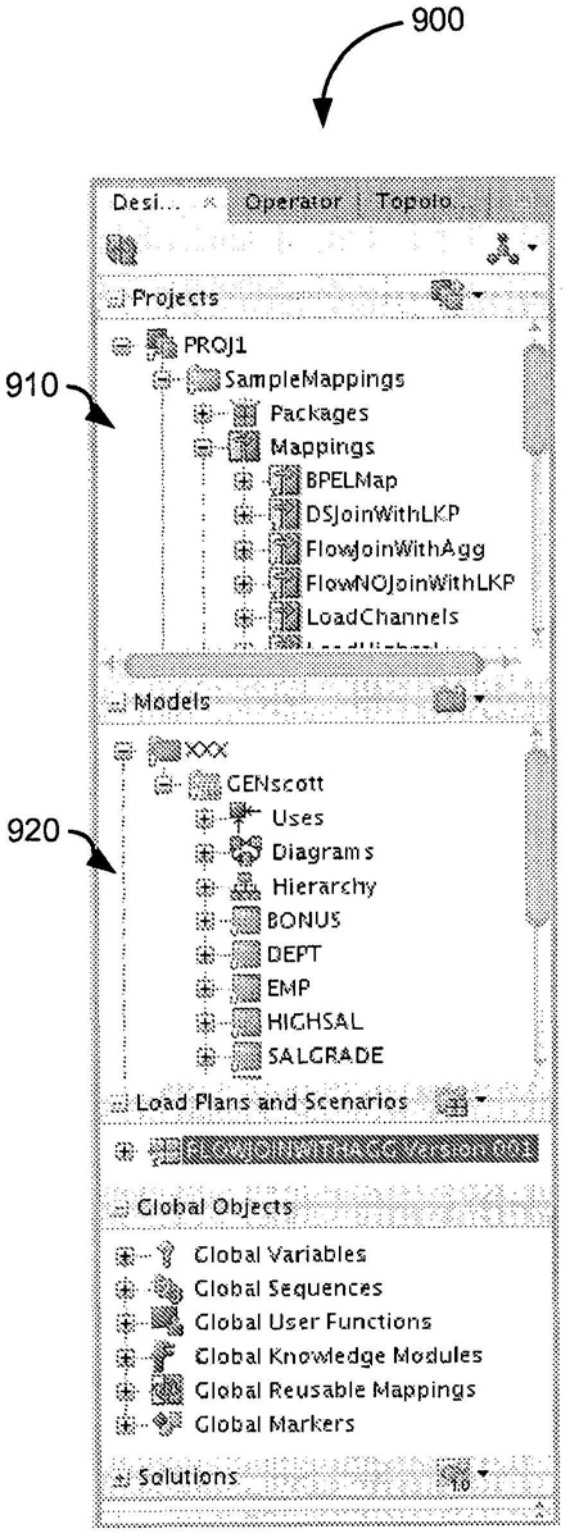

900
910
920
Desi...
Operator
Topolo...
Projects
PROJ1
SampleMappings
Packages
Mappings
BPELMap
DSJoinWithLKP
FlowJoinWithAgg
FlowNOJoinWithLKP
LoadChannels
Models
XXX
GENscott
Uses
Diagrams
Hierarchy
BONUS
DEPT
EMP
HIGHSAL
SALGRADE
Load Plans and Scenarios
Global Objects
Global Variables
Global Sequences
Global User Functions
Global Knowledge Modules
Global Reusable Mappings
Global Markers
Solutions

图9

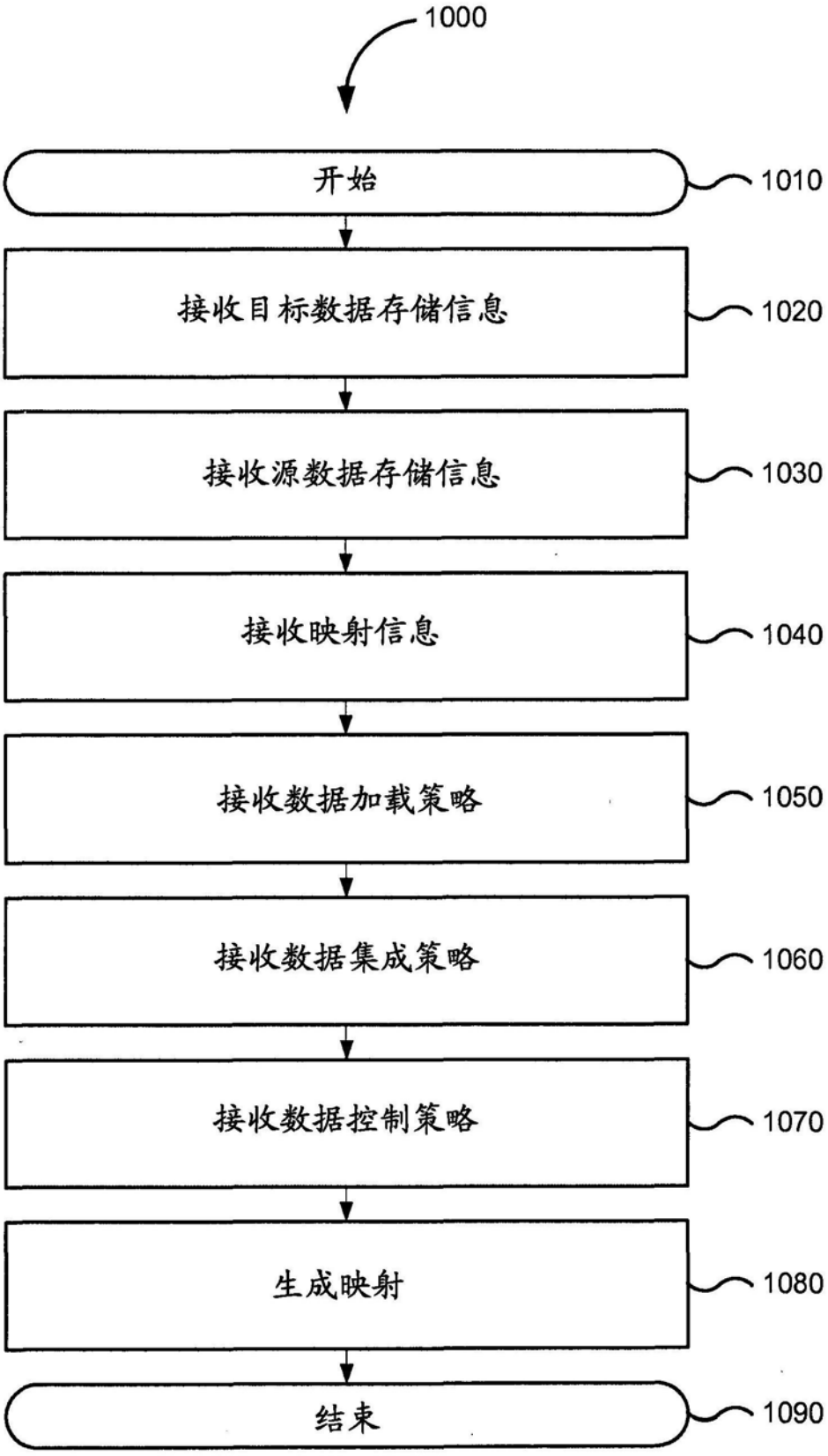

1000
开始
1010
接收目标数据存储信息
1020
接收源数据存储信息
1030
接收映射信息
1040
接收数据加载策略
1050
接收数据集成策略
1060
接收数据控制策略
1070
生成映射
1080
结束
1090

图10

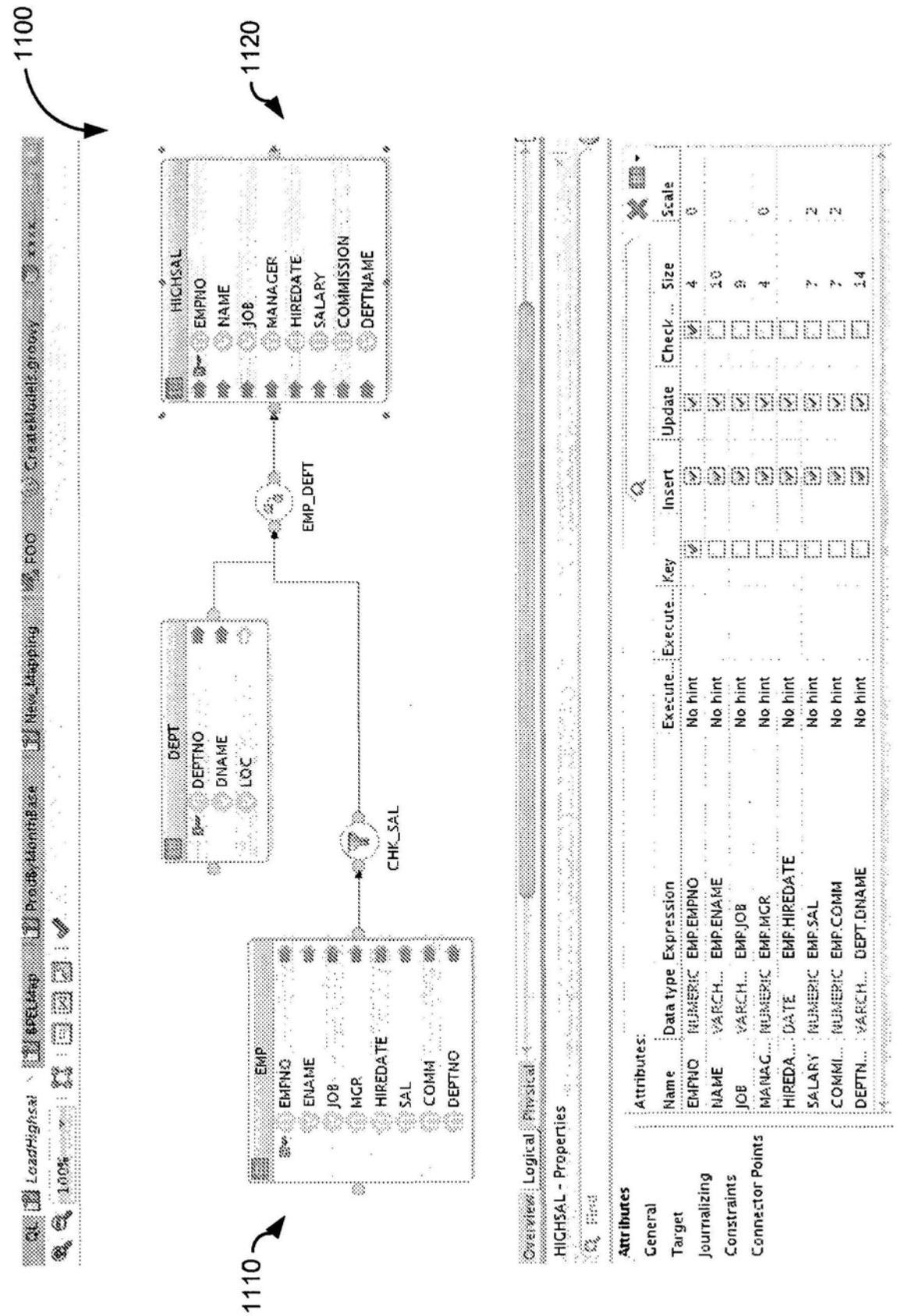

1100
1120
1110
LoadHighsal
EP51Map
ProfitByMonthBase
New_Mapping
FOO
CreateModelGroup
HIGHSAL
EMPNO
NAME
JOB
MANAGER
HIREDATE
SALARY
COMMISSION
DEPTNAME
EMP_DEPT
DEPT
DEPTNO
DNAME
LOC
CHK_SAL
EMP
EMPNO
ENAME
JOB
MGR
HIREDATE
SAL
COMM
DEPTNO
Overview
Logical
Physical
HIGHSAL - Properties
Attributes
General
Target
Journalizing
Constraints
Connector Points
Attributes:
Name
Data type
Expression
Execute...
Execute...
Key
Insert
Update
Check...
Size
Scale
EMPNO
NUMERIC
EMP.EMPNO
No hint
No hint
4
0
NAME
VARCH...
EMP.ENAME
No hint
No hint
10
JOB
VARCH...
EMP.JOB
No hint
No hint
9
MANAG...
NUMERIC
EMP.MGR
No hint
No hint
4
0
HIREDA...
DATE
EMP.HIREDATE
No hint
No hint
SALARY
NUMERIC
EMP.SAL
No hint
No hint
7
2
COMMI...
NUMERIC
EMP.COMM
No hint
No hint
7
2
DEPTN...
VARCH...
DEPT.DNAME
No hint
No hint
14

图11

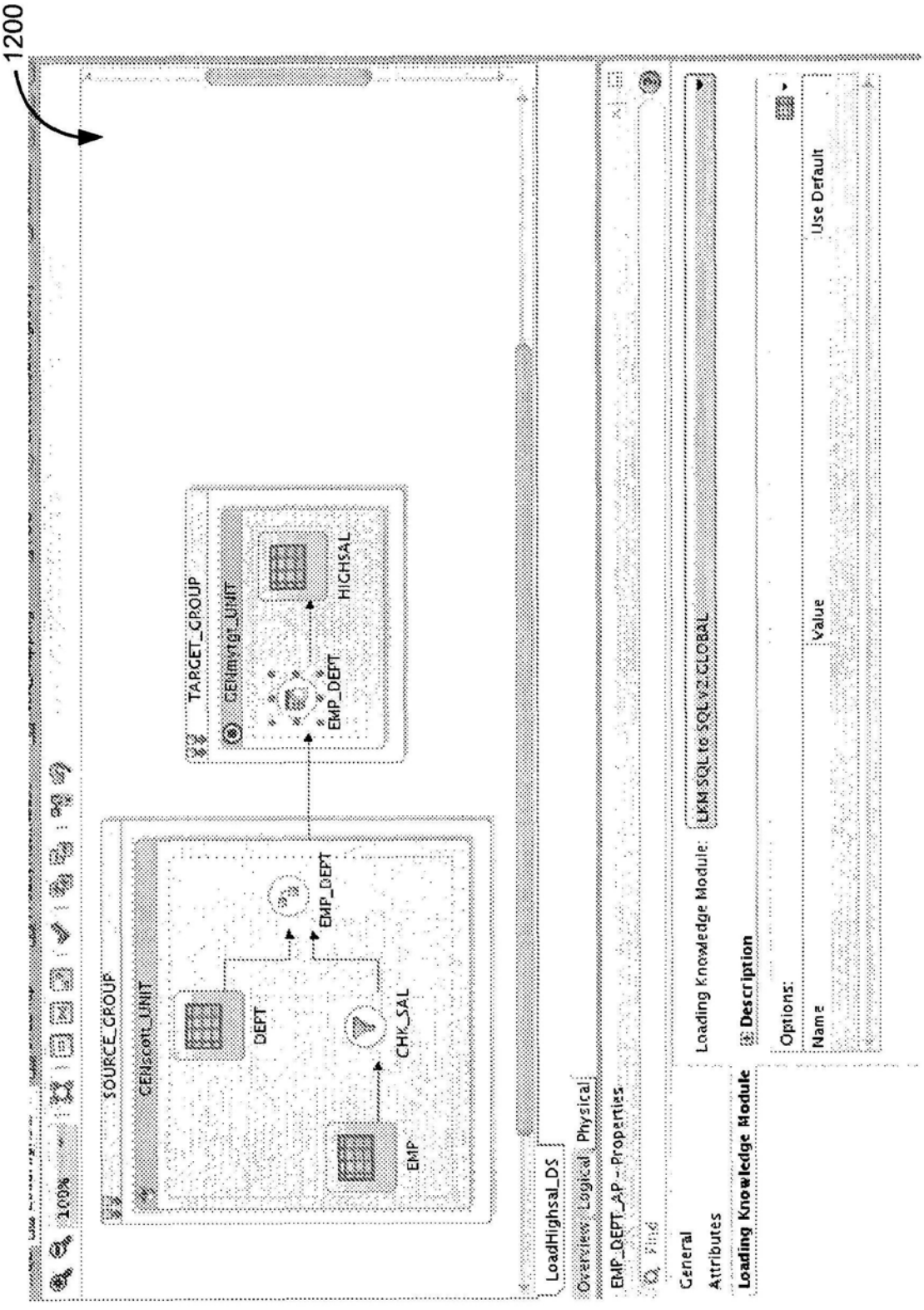

1200
SOURCE_GROUP
CEMscott_UNIT
DEPT
EMP_DEPT
CHK_SAL
EMP
TARGET_GROUP
CEMtrg1_UNIT
EMP_DEPT
HIGHSAL
LoadHighsal_DS
Overview
Logical
Physical
EMP_DEPT_AP - Properties
General
Attributes
Loading Knowledge Module
Loading Knowledge Module:
LKM SQL to SQL v2 GLOBAL
Description
Options:
Name
Value
Use Default

图12

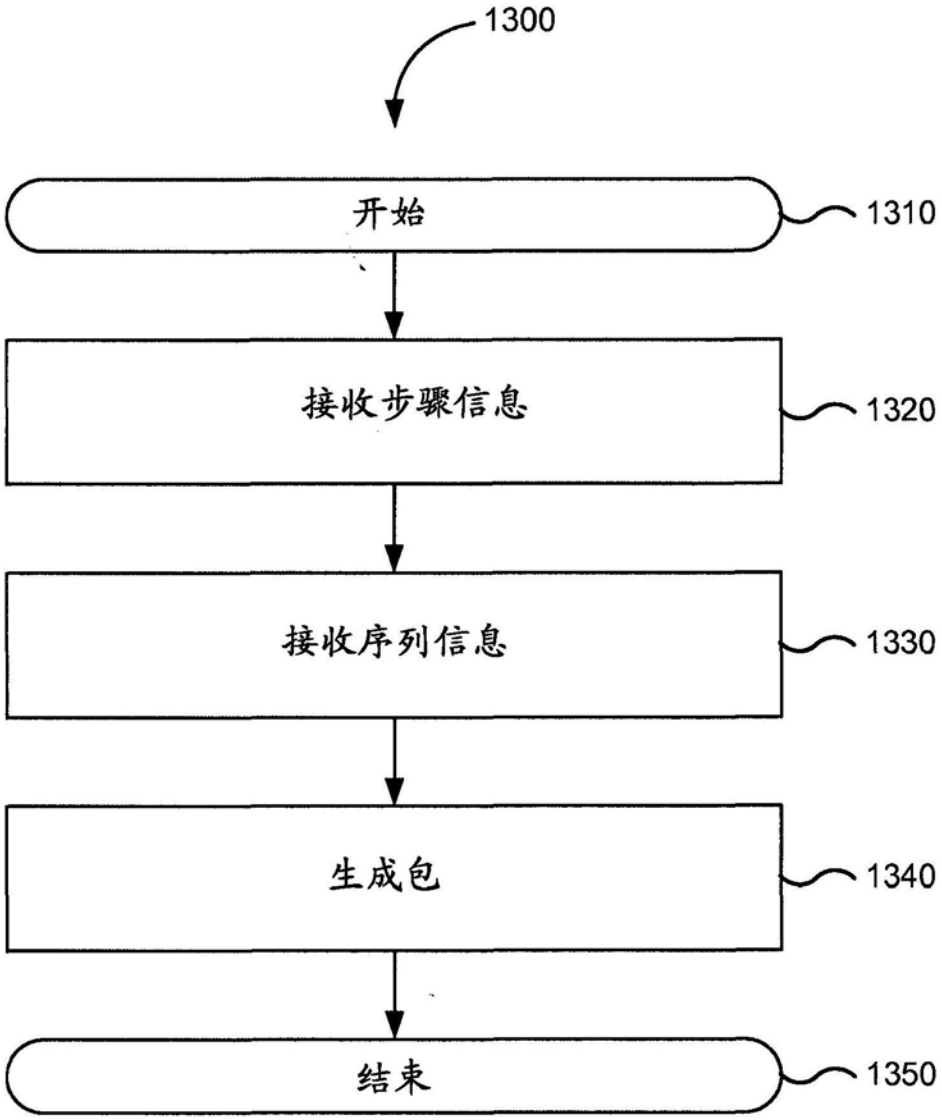


图13

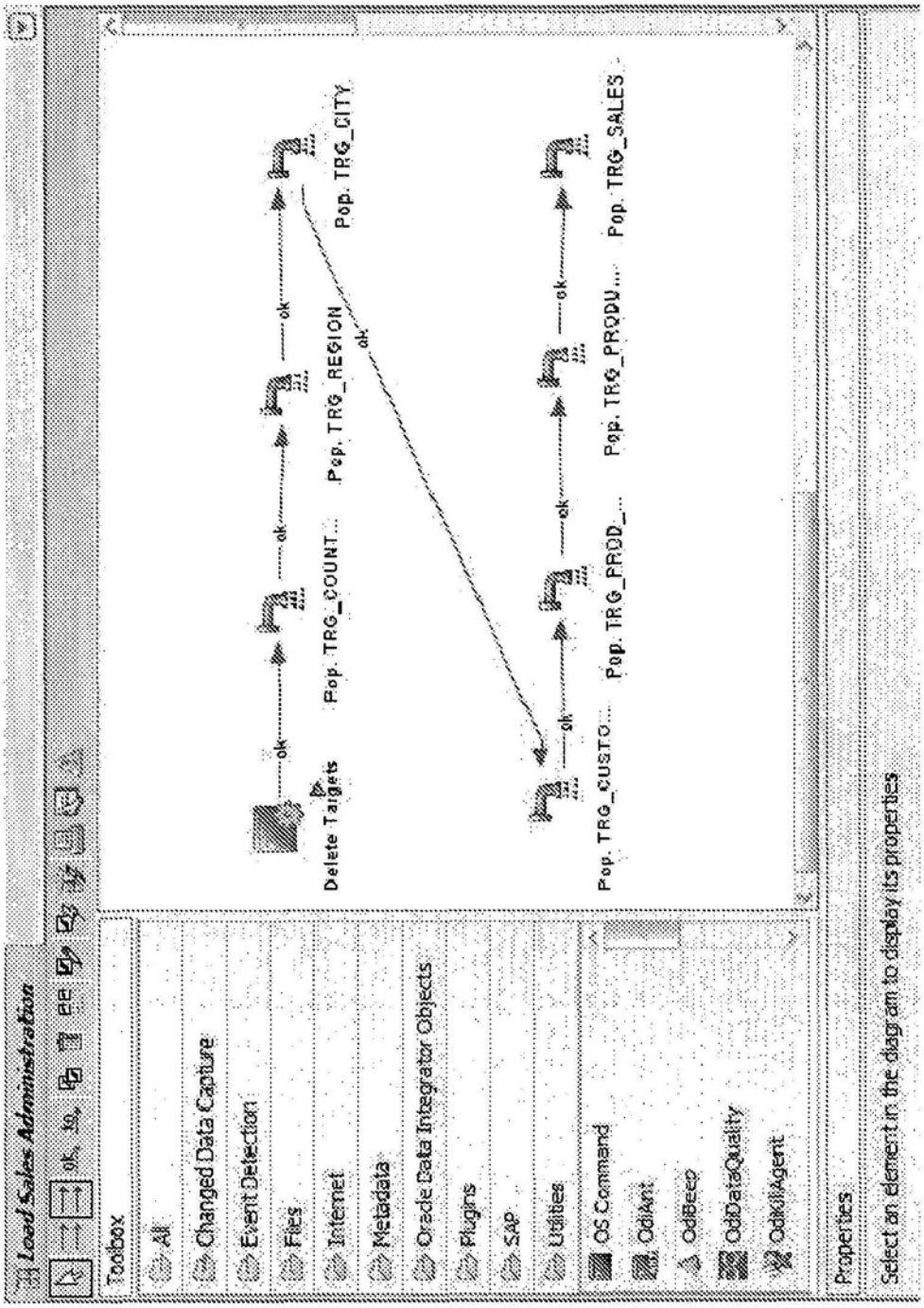

Load Sales Administration
Toolbox
All
Changed Data Capture
Event Detection
Files
Internet
Metadata
Oracle Data Integrator Objects
Plugins
SAP
Utilities
OS Command
OdiAnt
OdiBeep
OdiDataQuality
OdiOlAgent
Properties
Select an element in the diagram to display its properties
Delete Targets
Pop. TRG_COUNT...
Pop. TRG_REGION
Pop. TRG_CITY
Pop. TRG_CUSTO...
Pop. TRG_PROD_...
Pop. TRG_PRODU...
Pop. TRG_SALES
ok

图14

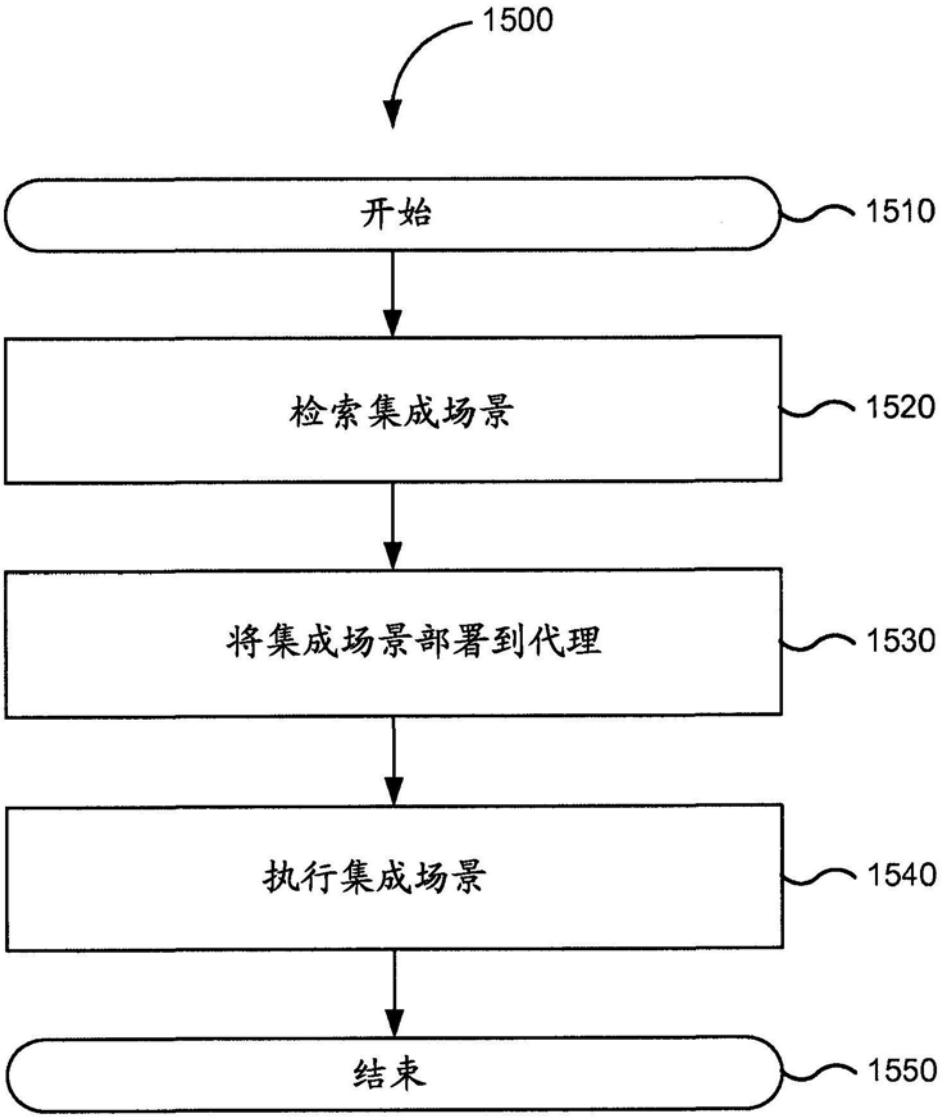

1500
开始
1510
检索集成场景
1520
将集成场景部署到代理
1530
执行集成场景
1540
结束
1550

图15

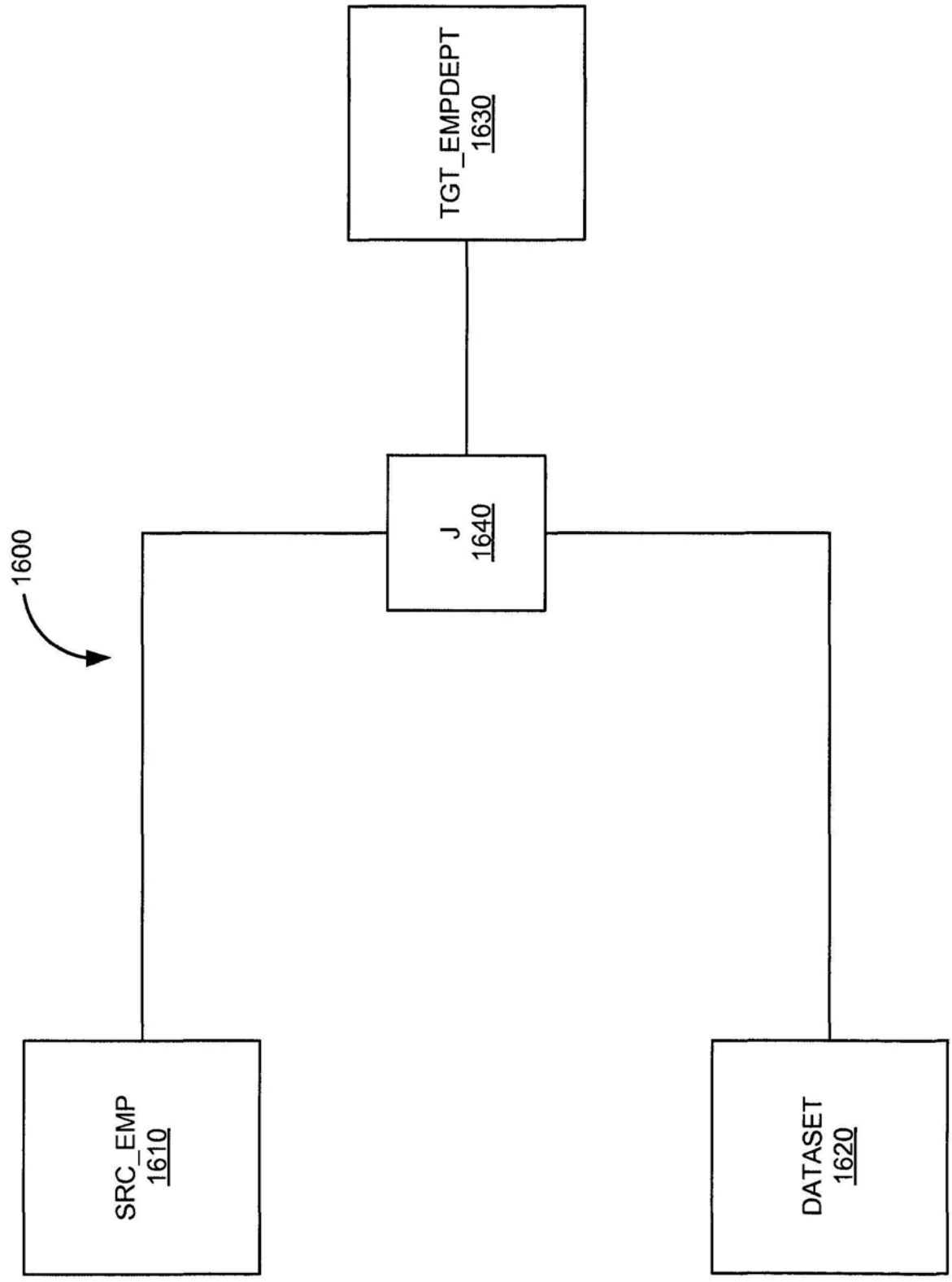


图16

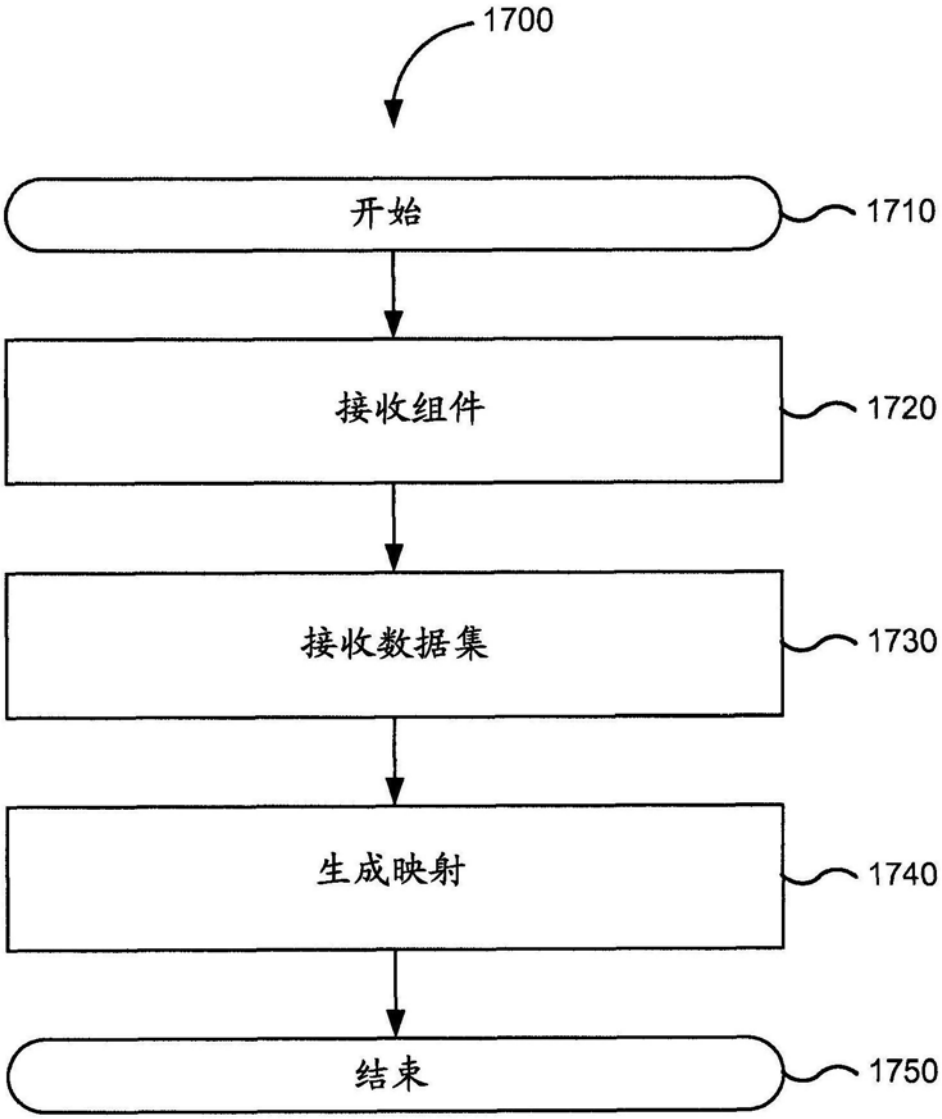


图17

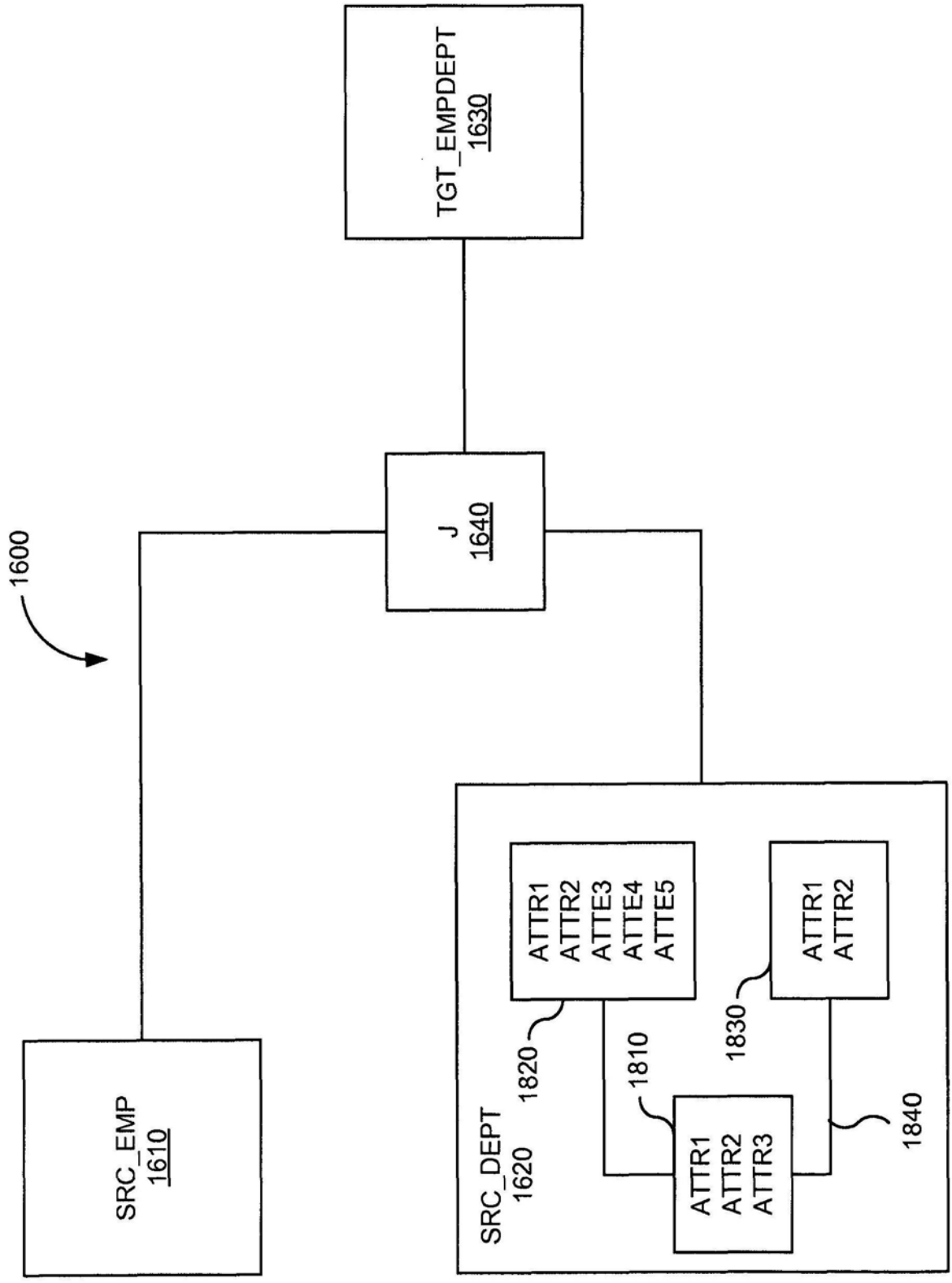

TGT_EMPDEPT
1630
J
1640
1600
SRC_EMP
1610
SRC_DEPT
1620
ATTR1
ATTR2
ATTE3
ATTE4
ATTE5
1820
1810
ATTR1
ATTR2
ATTR3
1830
ATTR1
ATTR2
1840

图18

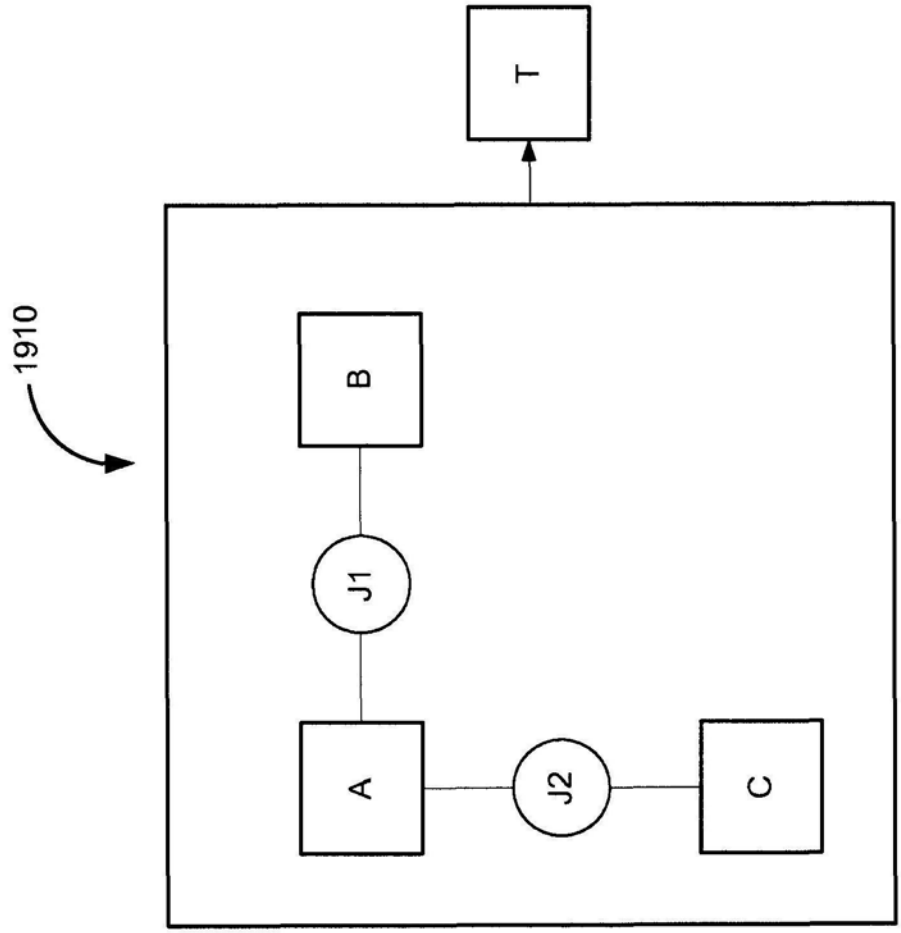


图19A

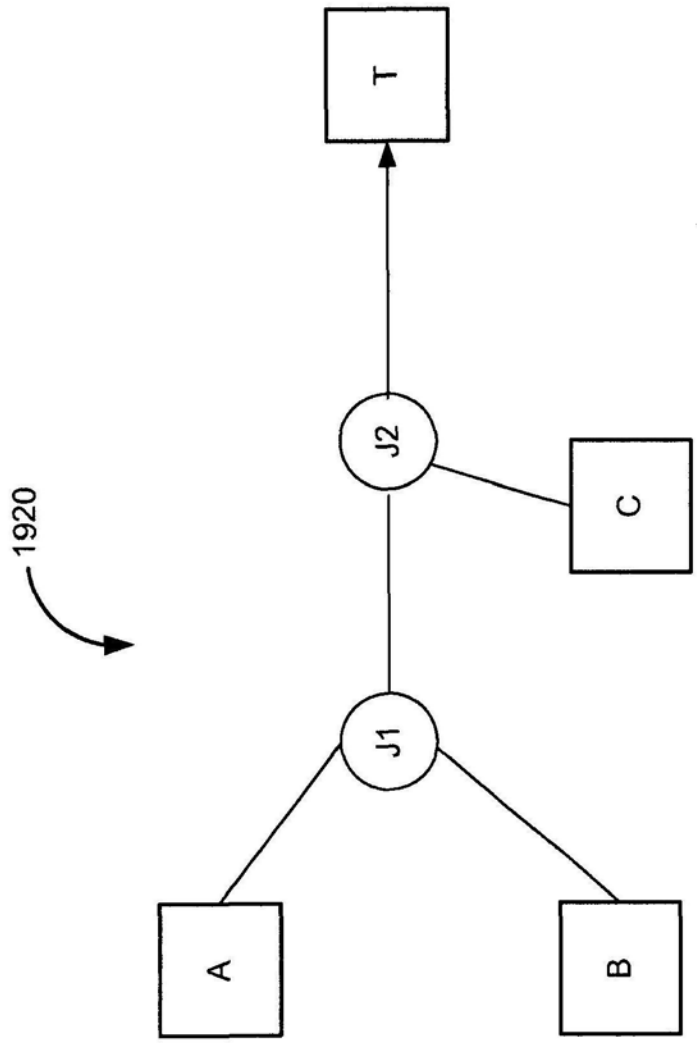


图19B

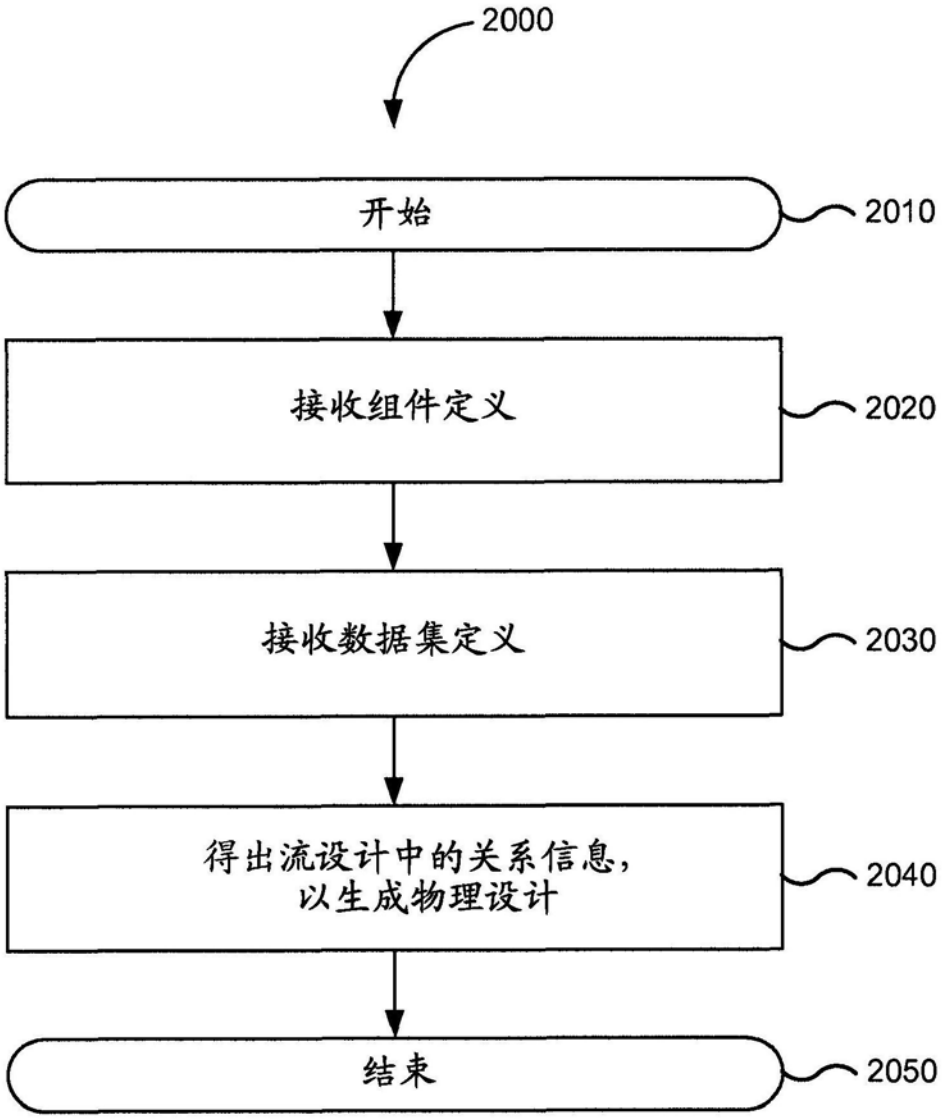

2000
开始
2010
接收组件定义
2020
接收数据集定义
2030
得出流设计中的关系信息，
以生成物理设计
2040
结束
2050

图20

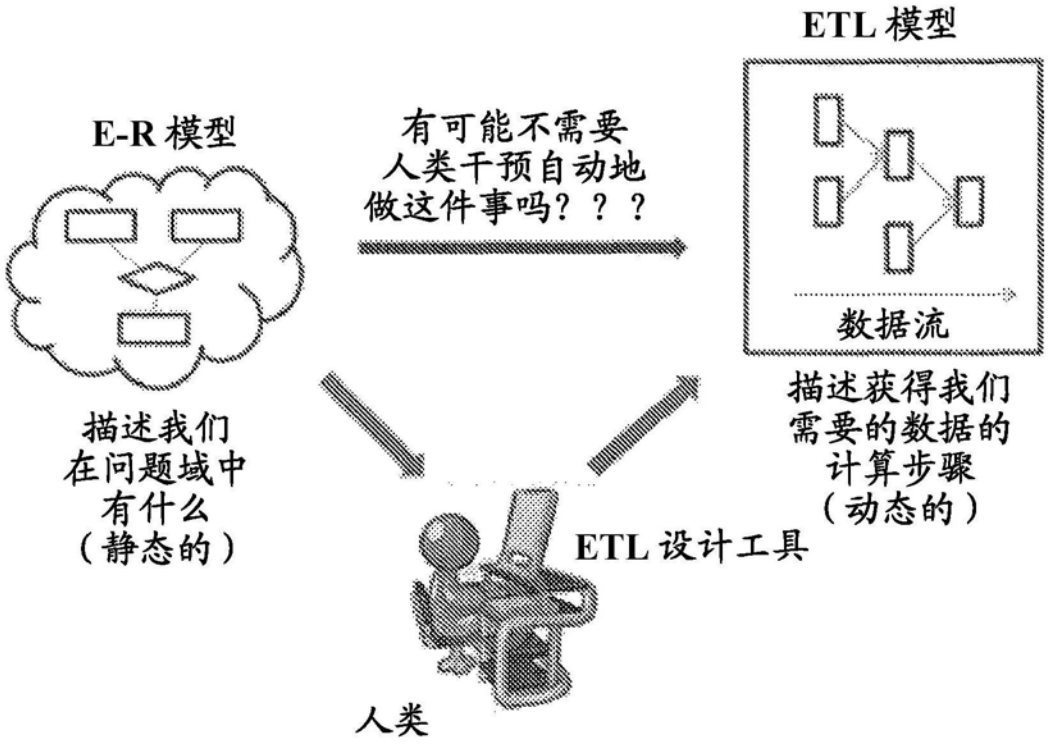

ETL 模型
E-R 模型
有可能不需要
人类干预自动地
做这件事吗？？？
数据流
描述我们
在问题域中
有什么
（静态的）
描述获得我们
需要的数据的
计算步骤
（动态的）
ETL 设计工具
人类

图21

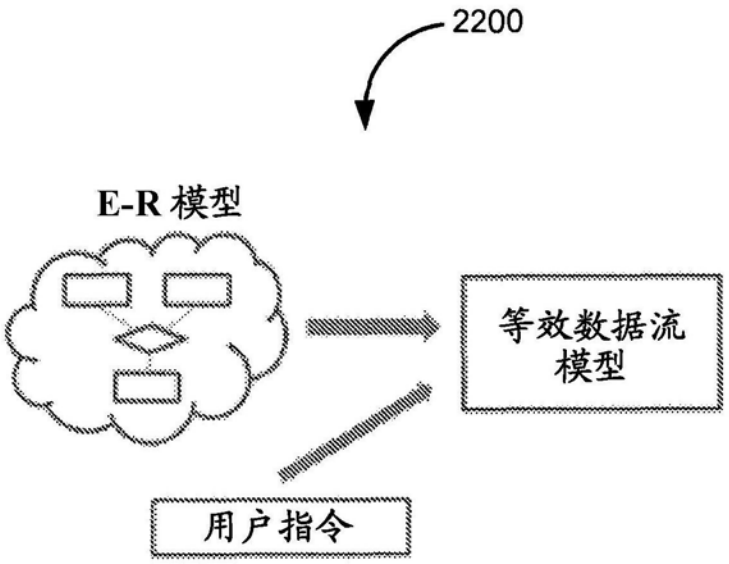

2200
E-R 模型
等效数据流
模型
用户指令

图22

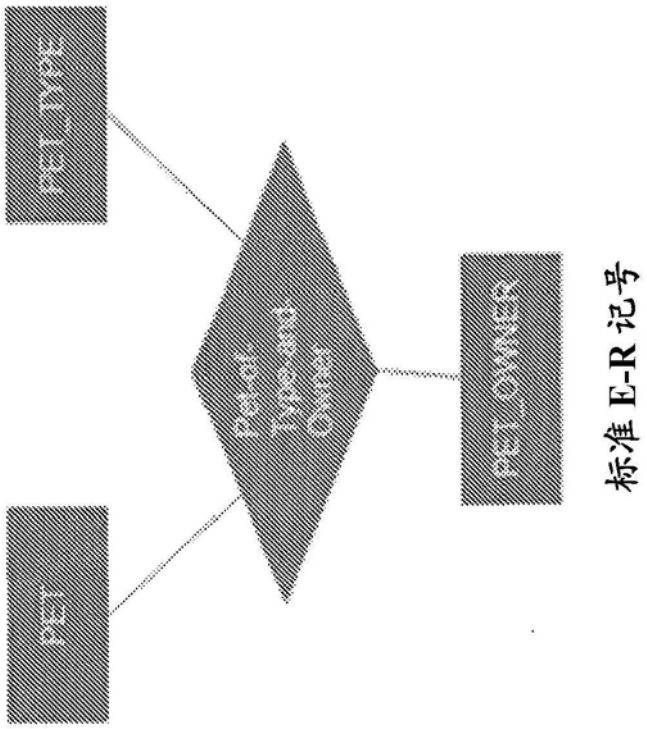

PET_TYPE
Pet-of-Type-and-Owner
PET_OWNER
PET
标准 E-R 记号

图23A

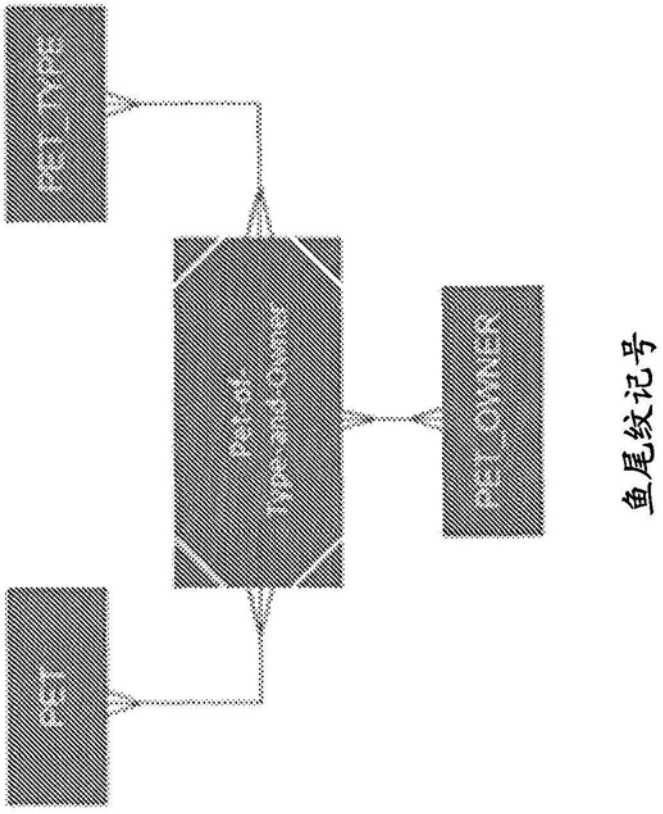

PET_TYPE
Pet-of-Type-and-Owner
PET_OWNER
PET
鱼尾纹记号

图23B

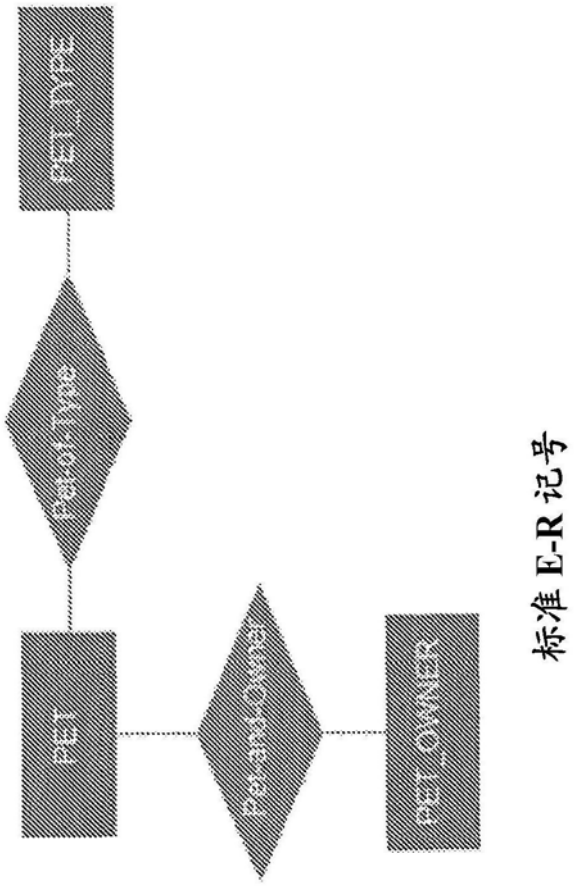

PET_TYPE
Pet-of-Type
PET
Pet-and-Owner
PET_OWNER
标准 E-R 记号

图24A

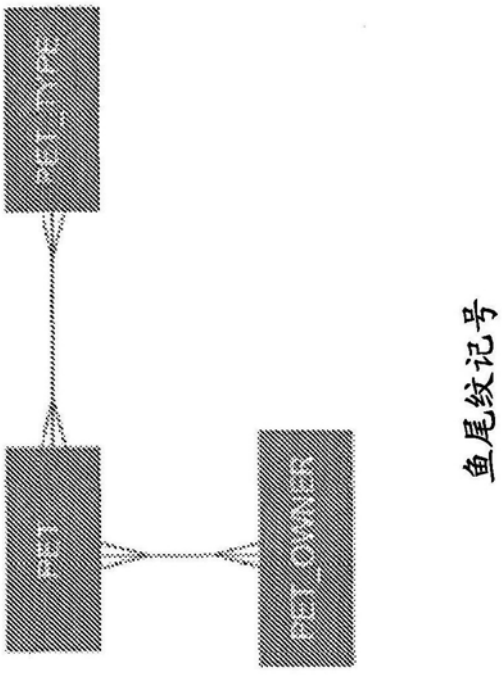

PET_TYPE
PET
PET_OWNER
鱼尾纹记号

图24B

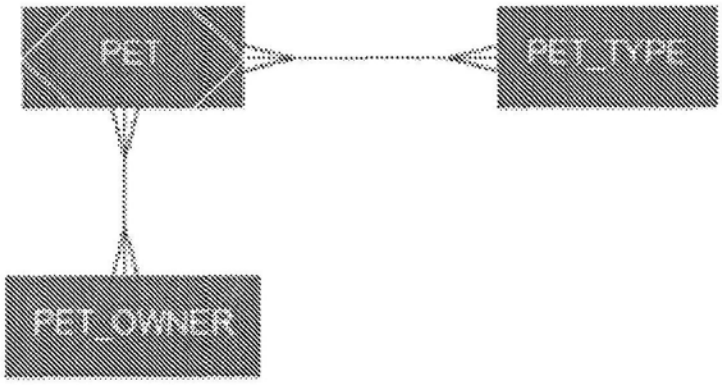

PET
PET_TYPE
PET_OWNER

图25

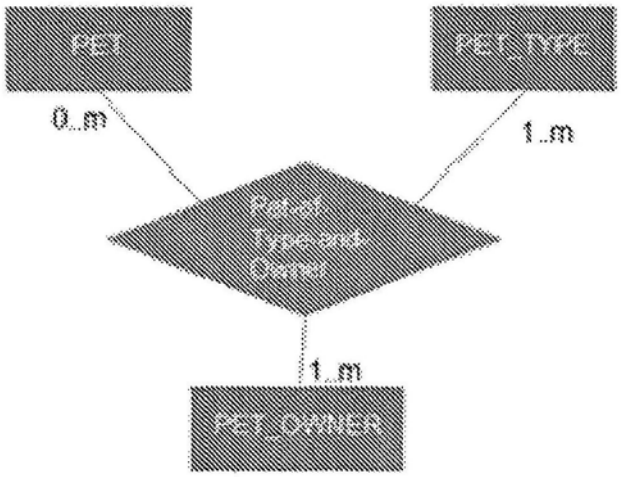

PET
PET_TYPE
0..m
1..m
Pet_of_Type_and_Owner
1..m
PET_OWNER

图26

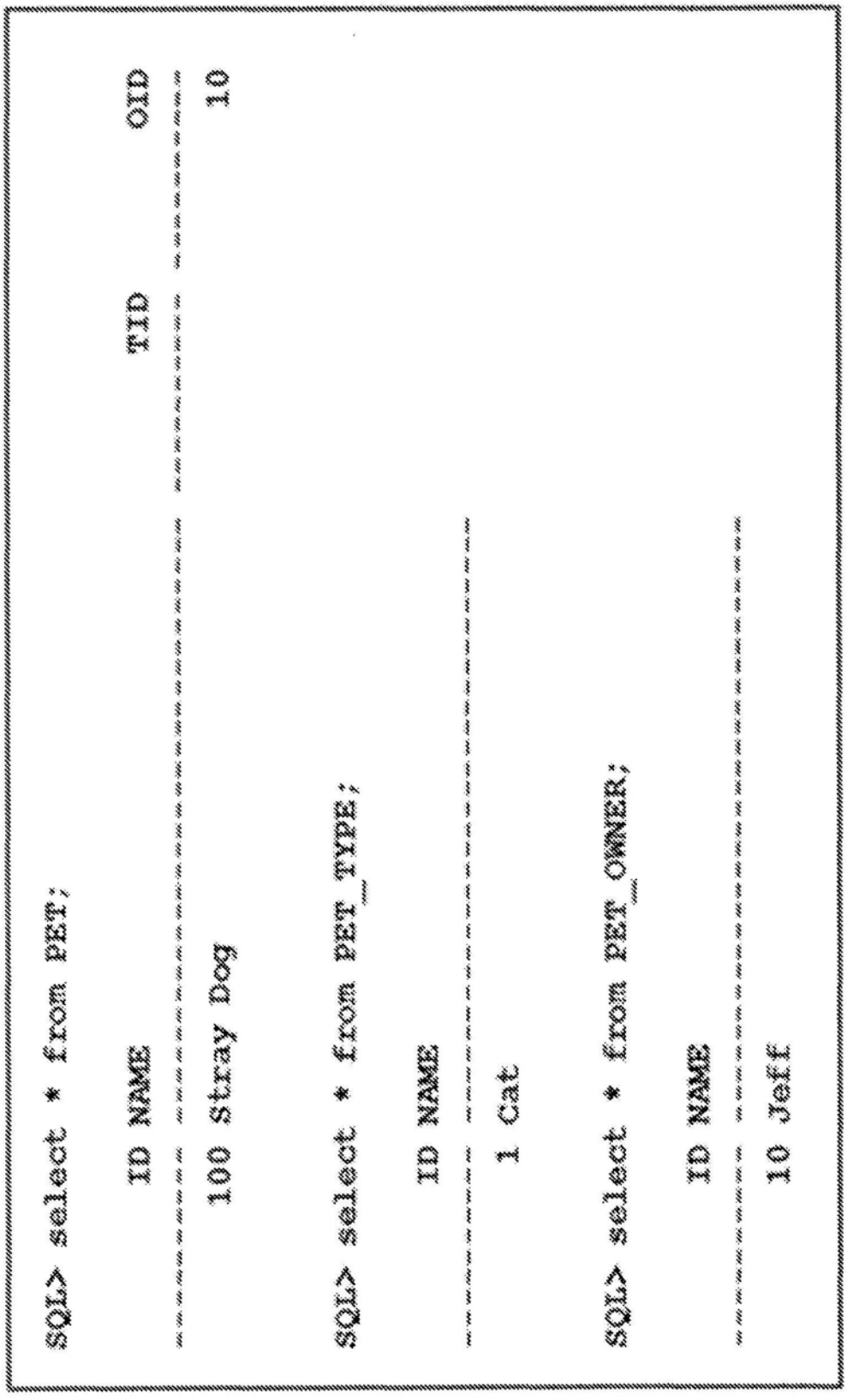

SQL> select * from PET;
ID NAME TID OID
100 Stray Dog 10
SQL> select * from PET_TYPE;
ID NAME
1 Cat
SQL> select * from PET_OWNER;
ID NAME
10 Jeff

图27

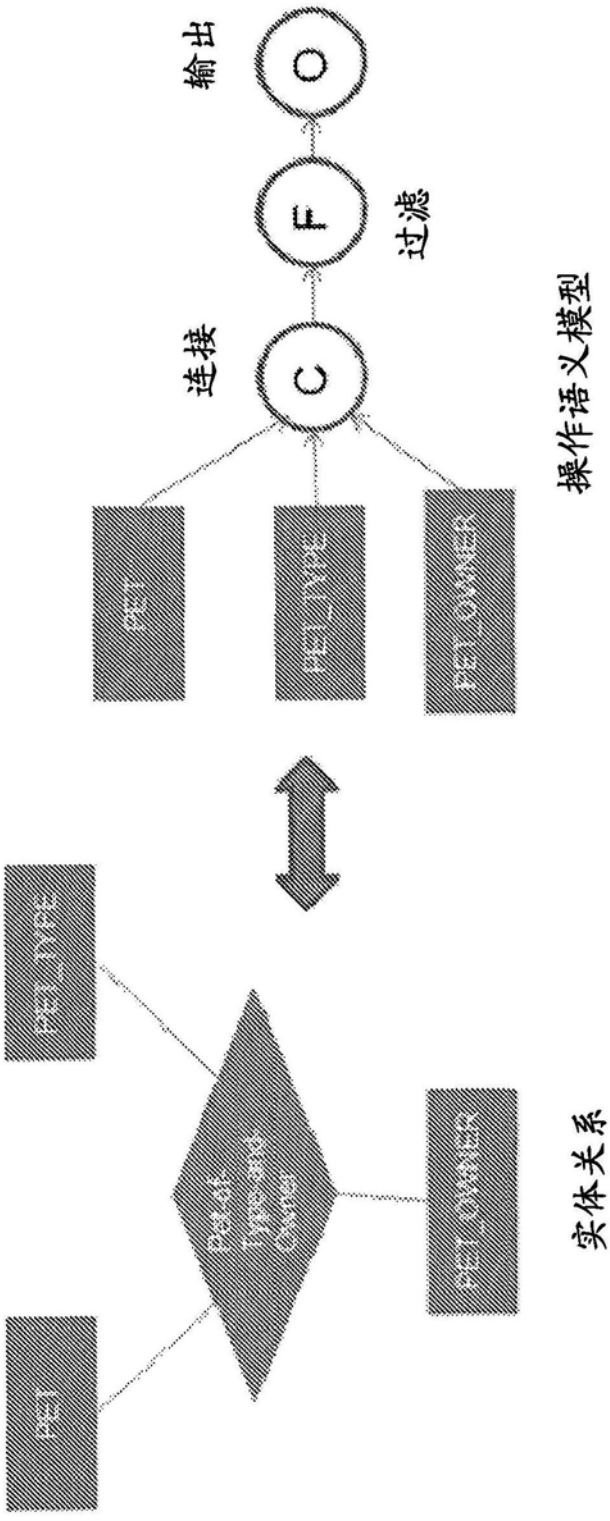

PET
PET_TYPE
Pet of Type and Owner
PET_OWNER
实体关系
连接
C
过滤
F
输出
O
PET
PET_TYPE
PET_OWNER
操作语义模型

图28A

图28B

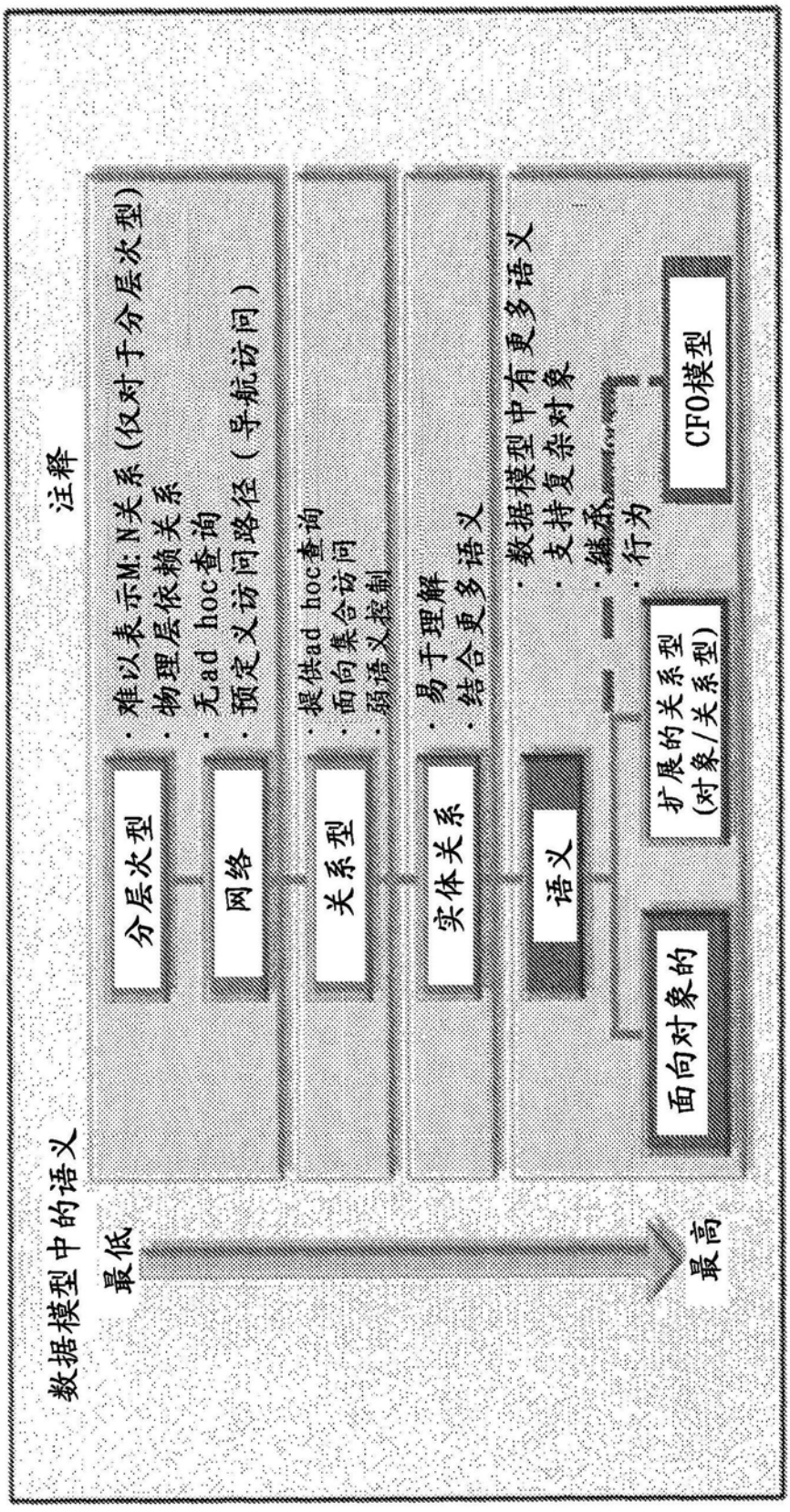

数据模型中的语义
最低
最高
注释
分层次型
网络
关系型
实体关系
语义
面向对象的
扩展的关系型
(对象/关系型)
CFO模型
·难以表示M:N关系(仅对于分层次型)
·物理层依赖关系
·无ad hoc查询
·预定义访问路径(导航访问)
·提供ad hoc查询
·面向集合访问
·弱语义控制
·易于理解
·结合更多语义
·数据模型中有更多语义
·支持复杂对象
·继承
·行为

图29

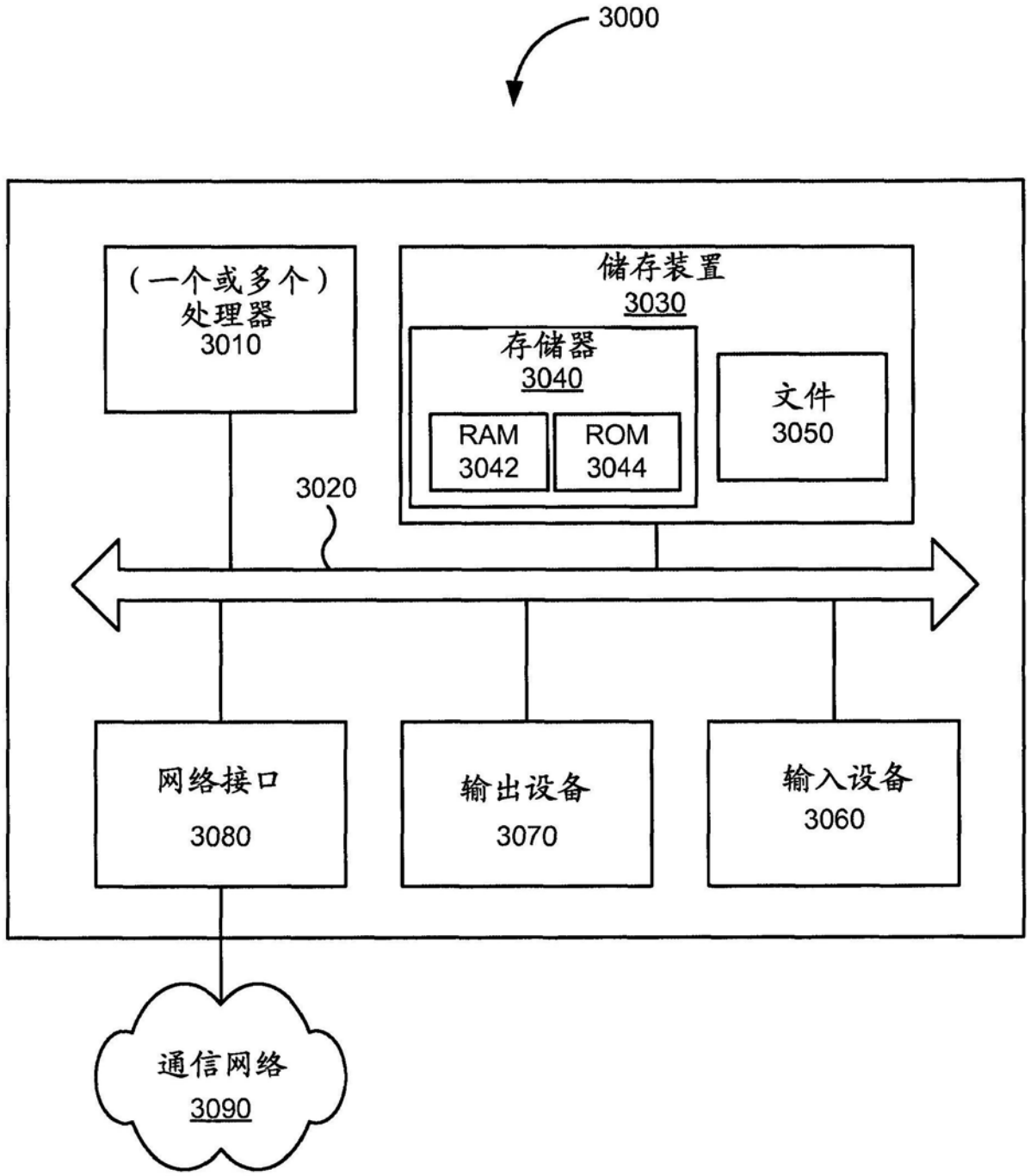

3000
（一个或多个）
处理器
3010
储存装置
3030
存储器
3040
RAM
3042
ROM
3044
文件
3050
3020
网络接口
3080
输出设备
3070
输入设备
3060
通信网络
3090

图30

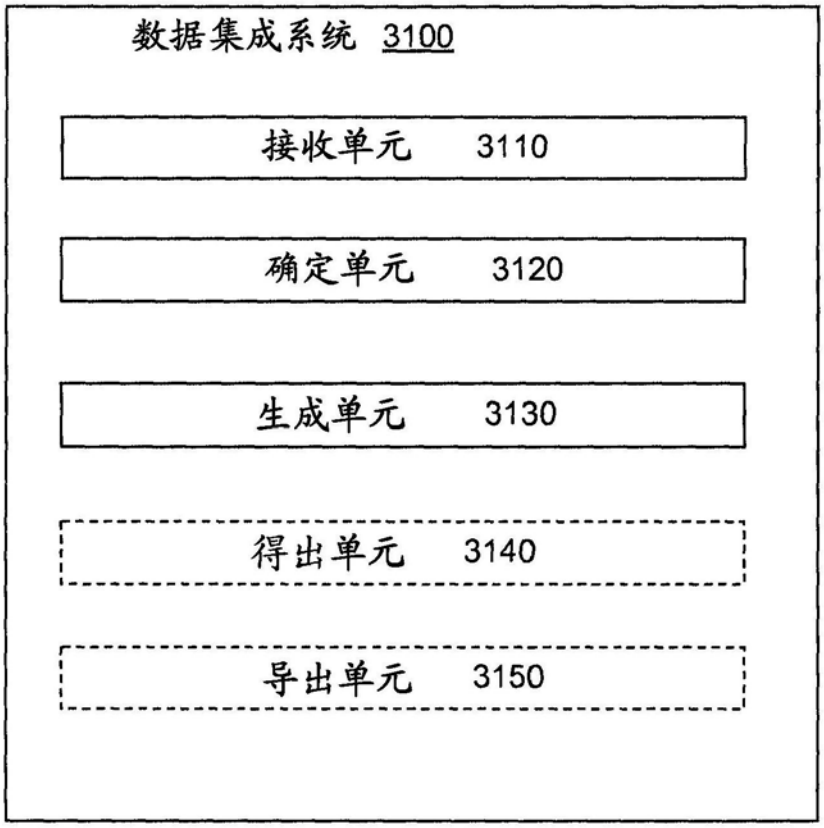

数据集成系统 3100
接收单元 3110
确定单元 3120
生成单元 3130
得出单元 3140
导出单元 3150

图31