[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/30 (2006.01)

[12] 发明专利申请公布说明书

[21] 申请号 200680023373.4

[43] 公开日 2008 年 6 月 25 日

[11] 公开号 CN 101208695A

[22] 申请日 2006.6.27
[21] 申请号 200680023373.4
[30] 优先权
[32] 2005. 6.27 [33] US [31] 11/167,902
[86] 国际申请 PCT/US2006/024933 2006.6.27
[87] 国际公布 WO2007/002647 英 2007.1.4
[85] 进入国家阶段日期 2007.12.27
[71] 申请人 起元软件有限公司
地址 美国马萨诸塞州
[72] 发明人 克雷格·W.·斯坦菲尔
J.·斯凯芬顿·沃莱
布龙德·拉森 格伦·约翰·阿林

[74] 专利代理机构 隆天国际知识产权代理有限公司
代理人 郑小军 郑特强

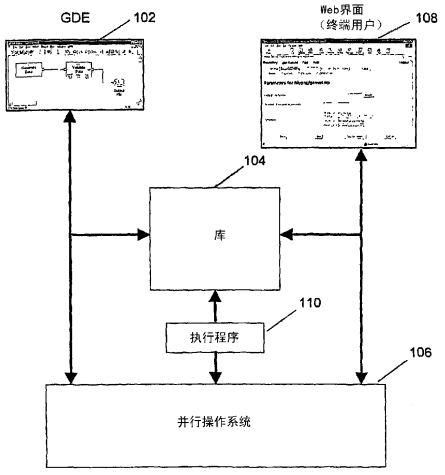
权利要求书 2 页 说明书 37 页 附图 19 页

[54] 发明名称

基于图形计算的元数据管理

[57] 摘要

确定与基于图形计算相关的元数据包括：函数转换与图形的第一部分(2310)相关的元数据，以产生与图形的第二部分(2316)相关的转换后的元数据；确定与所述图形的第二部分相关的图形的第三部分(2317)；以及将转换后的元数据从所述图形的第二部分传送至所述图形的第三部分。



1、一种与基于图形计算相关的元数据的确定方法，包括：

函数转换与图形的第一部分（2310）相关的元数据，以产生与所述图形的第二部分（2316）相关的转换后的元数据；

确定与所述图形的第二部分相关的所述图形的第三部分（2317）；以及

将所述转换后的元数据从所述图形的第二部分传送至所述图形的第三部分。

2、如权利要求 1 所述的方法，其中所述图形的第三部分通过数据流与所述图形的第二部分相关。

3、如权利要求 2 所述的方法，所述数据流包括两个相互连接的图形单元的端口之间的数据流。

4、如权利要求 2 所述的方法，其中所述数据流包括一图形单元的两端口之间的内部数据流。

5、如权利要求 1 所述的方法，其中所述图形的第三部分通过链接与所述图形的第二部分相关，其中所述链接表示与第二部分相关的元数据也应该与所述第三部分相关。

6、如权利要求 1 所述的方法，其中：

所述第一部分包括第一图形单元的第一端口，以及

所述第二部分包括所述第一图形单元的第二端口。

7、如权利要求 6 所述的方法，其中所述函数转换包括元数据定义，其中所述元数据定义包括对与所述第一端口相关的元数据的至少一个引用。

8、如权利要求 7 所述的方法，其中所述元数据定义将用于所述第二端口的元数据定义为所引用的元数据的函数。

9、如权利要求 6 所述的方法，其中所述第一端口是输入端口，所述第二端口是输出端口。

10、如权利要求 1 所述的方法，其中由用户提供被函数转换的元数据。

11、如权利要求 1 所述的方法，其中从所述图形的第四部分传送被函数转换的元数据。

12、如权利要求 1 所述的方法，还包括响应于图形连接方式的改变来传送所述转换后的元数据。

13、如权利要求 1 所述的方法，还包括响应于用户行为传送所述转换后的元数据。

14、如权利要求 1 所述的方法，还包括：

接收来自用户的请求；和

响应于该请求向用户显示与图形单元相关的元数据。

15、如权利要求 14 所述的方法，其中所述来自用户的请求包括来自用户的输入，该输入用于选择将要显示相关元数据的图形单元。

16、如权利要求 15 所述的方法，其中所述来自用户的输入包括在预定量时间内在所选择的图形单元的图形表示附近定位屏幕上的指针。

17、如权利要求 14 所述的方法，其中所显示的元数据包括从另一图形单元传送的元数据。

18、如权利要求 14 所述的方法，其中所显示的元数据在运行所述图形之前被显示。

19、一种在计算机可读介质上存储的软件，用于确定与基于图形计算相关的元数据，该软件包括使计算机执行以下操作的指令：

函数转换与图形的第一部分（2310）相关的元数据，以产生与所述图形的第二部分（2316）相关的转换后的元数据；

确定与所述图形的第二部分相关的所述图形的第三部分（2317）；以及

将转换后的元数据从所述图形的第二部分传送至所述图形的第三部分。

20、一种与基于图形计算相关的元数据的确定系统，该系统包括：

用于函数转换与图形的第一部分（2310）相关的元数据的装置，以产生与所述图形的第二部分（2316）相关的转换后的元数据；

用于确定与所述图形的第二部分相关的所述图形的第三部分（2317）的装置；以及

用于将转换后的元数据从所述图形的第二部分传送至所述图形的第三部分的装置。

基于图形计算的元数据管理

技术领域

本发明涉及对数据处理系统中的计算控制，尤其涉及基于图形计算的元数据管理。

背景技术

复杂的事务系统通常进行多级数据处理，将一级产生的结果馈送入下一级。经过这些系统的总信息流可以以有向（directed）数据流图的形式被描述，其中图中的顶点代表组件（数据文件或进程），图形中的链接（link）或“边”表示各组件之间的数据流。

相同类型的图示可以用于描述并行处理系统。出于此讨论的目的，并行处理系统包括使用多个中央处理器（CPUs）的计算机系统的任何配置，所述中央处理器可以是本地的（例如 SMP 计算多处理器系统）或本地分布式的（例如连接成群组（cluster）的多个处理器或 MPP）或是远程的或远程分布式的(例如经由 LAN 或 WAN 网络连接的多个处理器)或者它们的任何组合。此外，这些图形可以由组件（数据文件或进程）和流（图形的边或链接）组成。通过明确地或隐含地复制图形的各元素（组件和流），可以表示系统的平行性。

图形还可以用于直接调用计算。来自 Ab Initio Software Corporation, Lexington, MA 的具有图形开发环境（GDE）的“CO>OPERATING SYSTEM ®”体现了此种系统。根据此系统产生的图形提供各种方法：用于从图形组件表示的个别进程中输入和输出信息，用于在进程之间移动信息，以及用于规定进程的运行顺序。该系统包括用于选择进程间通信方法的算法和用于调度（schedule）进程的执行的算法，该系统还用于监控图形的执行。

开发者经常构建各种图形，所述图形通过利用环境变量或命令行自变量（command-line argument）而以一种方式或另一种方式被控制，所述环境变量或命令行自变量可以产生能够被图形编译器在“运行时”（即执行该图形

时）编译成可执行指令的指令（例如外壳脚本（shell script）指令）。从而环境变量和命令行自变量成为特定（*ad hoc*）参数，用于指定诸如文件名、数据选择表达式和关键字（例如分类（sort）关键字）的信息，以使应用更灵活。然而，用户可能需要读取所产生的 shell script 指令并搜索该 shell script 指令，用于引用环境变量和命令行参数，以发现控制特定图形的执行的该组参数。

发明内容

在总的方案中，本发明的特征在于提供一种用于确定与基于图形计算相关的元数据的方法和对应的软件及系统。该方法包括以下步骤：函数转换与图形的第一部分相关的元数据以产生与图形的第二部分相关的转换后的元数据；确定与所述图形的第二部分相关的图形的第三部分；以及将转换后的元数据从所述图形的第二部分传送至所述图形的第三部分。

该方案可以包括一个或多个以下特征：

所述图形的第三部分通过数据流与所述图形的第二部分相关。

所述数据流包括在两个相互连接的图形单元的端口之间的数据流。

所述数据流包括在一图形单元的两端口之间的内部数据流。

所述图形的第三部分通过链接与所述图形的第二部分相关，所述链接表示与第二部分相关的元数据也应该与所述第三部分相关。

所述第一部分包括第一图形单元的第一端口，以及所述第二部分包括所述第一图形单元的第二端口。

所述功能转换包括元数据定义，其包括对与所述第一端口相关的元数据的至少一个引用。

所述元数据定义将用于所述第二端口的元数据定义为所引用的元数据的函数。

所述第一端口是输入端口，所述第二端口是输出端口。

由用户提供被函数转换的元数据。

从图形的第四部分传送被函数转换的元数据。

该方法还包括响应于图形连接方式的改变传送转换后的元数据。

该方法还包括响应于用户行为传送转换后的元数据。

该方法还包括接收来自用户的请求；和响应于该请求向用户显示与图形单元相关的元数据。

所述来自用户的请求包括来自用户的输入，该输入用于选择将要显示相关元数据的图形单元。

所述来自用户的输入包括在预定量时间内在所选择的图形单元的图形表示附近定位屏幕上的指针。

所显示的元数据包括从另一图形单元传送的元数据。

在所述图形被运行之前显示已显示的元数据。

本发明的方案可以包括一个或多个以下优点：

以运行时间参数方式形成的图形界面已被格式化。图形界面已被适当定义，足以使得系统知道哪些参数需要提供以及这些参数应该如何提示。

通过运行时间参数可以直接或间接指定或计算出控制组件的元数据。

可以以控制条件组件的运行时间参数值为依据来修改图形结构，从而基于用户的选择来提供或不提供组件。

对图形运行时间进行参数化的优点是应用可以被充分参数化，使得诸如事务分析员和统计建模型员的终端用户可以请求能满足他们需求的数据。现代企业数据环境的复杂性导致这种事务状态，即，其中在数据收集和预先分析转换进程中通常需要涉及大量的人力。本发明为终端用户提供有力的工具，使得该终端用户可以定义及恢复他们想要的数据，而在每一个查询类型的关键途径中不需要专业的数据分析家。

在图形内传送的元数据包括经由函数转换的元数据，例如被定义成其它元数据的函数的元数据。该传送例如发生在图形运行之前的编辑时间。使得所转换的元数据能够传送，就可以增强用户甚至在图形运行之前就可以查看及/或处理元数据的能力。

存在具有运行时间参数的可再用（可互连）的组件的程序库。通过这些组件，利用具有针对图形中所有运行时间参数而自动确定的提示顺序的组件来构建该图形。在一些情形中，参数需要被重新排序以满足某些限制条件。根据预期的排序（例如由开发者指定的排序）对参数重新排序以满足那些限制条件降低了因严重偏离预期排序的参数顺序而提示用户的可能性。

通过以下描述和权利要求，本发明的其它特征及优点变得明显。

附图说明

图 1A 是示出主元素的相互关系的本发明的一个实施例的框图。

图 1B 是数据流图的框图。

图 2 是具有汇总组件和分类组件 204 的典型图形的框图，该汇总组件和分类序组件 204 具有指定的运行时间参数。

图 3 是表示即将与图形相关的运行时间参数网格的图形对话框的一个实施例图。

图 4 是总结利用运行时间参数的进程的流程图。

图 5 是由关键字提示产生的图形对话框的一个实施例图。

图 6 是由过滤提示产生的图形对话框的一个实施例图。

图 7 是由汇总提示产生的图形对话框的一个实施例图。

图 8 是由重新格式化提示产生的图形对话框的一个实施例图。

图 9A 是第一个图形的框图，其中合并（MergeJoin）组件组合文件 A 和 B 的数据并输出结果至输出文件。

图 9B 是第二个图形的框图，其中汇总组件集合文件 A 的数据并输出结果至输出文件。

图 9C 是一个图形的框图，其中合并组件组合文件 A 和 B 的数据，而汇总组件集合所产生的数据并输出最后结果至输出文件。

图 10 是表示具有条件解释控制的条件的图形对话框的一个实施例图。

图 11 是示出出现损坏情形的图形的示意图。

图 12 是总结包括*完全移除*条件组件的图形的运行时间准备进程的流程图。

图 13 是针对本发明的特定实施例而总结包括*流替换*条件组件的图形的运行时间准备进程的流程图。

图 14 是表示没有运行时间参数的汇总应用的图形的示意图。

图 15 是表示图 14 的汇总应用进行运行时间参数化的版本的图形的示意图。

图 16 是表示用于图 15 的示例性应用的运行时间参数网格的图形对

话框的一个实施例图。

图 17A 是表示来自图 16 的参数网格中的信息的 Web 界面所产生的表格的图形对话框的一个实施例图。

图 17B 是由用户用参数值填充图 17A 的表格的示意图。

图 18 是表示运行时间参数化汇总和组合应用的图形的示意图。

图 19 是表示用于图 18 的示例性应用的运行时间参数网格的图形对话框的一个实施例图。

图 20 是表示来自图 19 的参数网格中的信息的 Web 界面所产生的表格的图形对话框的一个实施例图。

图 21 是表示运行时间参数化的汇总-组合-分类应用的图形的示意图。

图 22 是表示用于图 21 所示的示例性应用的运行时间参数网格的图形对话框的一个实施例图。

图 23A 是在图形中传送元数据的示意图。

图 23B 是图 23A 中图形的组件的子图形的示意图。

图 24 是元数据传送进程的流程图。

图 25A 是包括具有组件内和组件间相关性的参数的图形。

图 25B 和图 25C 是表示图 25A 中的图形的参数之间相关性的相关图。

图 26 是改进的拓扑分类进程示意图。

各个附图中同样的附图标记表示同样的元素。

具体实施方式

概述

图 1A 是示出主单元（principal element）间相互关系的本发明的一个实施例的框图。图形开发环境（GDE）102 提供用于创建可执行的图形并定义图形组件的参数的用户界面。该 GDE 可以是诸如可从本发明的申请人处得到的 CO>OPERATING SYSTEM® GDE。GDE 102 与库 104 和并行操作系统 106 相互通信。Web 界面 108 和执行程序 110 也连接到库 104 和并行操作系统 106。

库 104 优选是可升级的面向对象的数据库系统，其设计用于支持基于图形的应用的开发和执行，以及支持基于图形的应用和其它系统（例如其它操作系统）之间的元数据交换。库 104 是用于所有元数据类型的存储系统，所述元数据包括（但不限于）文件、记录格式、转换函数、图形、作业和监控信息。库在本领域中是公知的；例如可参看美国专利 Nos. 5,930,794; 6,032,158; 6,038,558;以及 6,044,374。

并行操作系统 106 接受在 GDE 102 中产生的数据流图的表示法，并产生与经由该图形定义的处理逻辑和资源相对应的计算机指令。然后并行操作系统 106 通常在多个处理器（不必是同类的）上执行这些指令。合适的并行操作系统是可以从本发明的申请人处得到的 CO>OPERATING SYSTEM®。

Web 界面 108 对库 104 的内容以网页浏览方式进行查看。利用 Web 界面 108，用户可以浏览对象、创建新的对象、改变现有对象、指定应用参数、调度作业等。Web 界面 108 基于存储在库 104 中与图形相关的运行时间参数的信息，自动创建参数化图形的基于表格的用户界面。

执行程序 110 是通过 Web 界面 108 访问的任选的基于库的作业调度系统。执行程序 110 将作业和作业队列在库 104 内保存为对象，Web 界面 108 提供查看工具以处理作业和作业队列。

图 1B 示出简单的数据流图 120，具有通过流 124 连接到过滤组件 126 的输入数据集 122。过滤组件 126 通过流 128 连接到输出数据集 130。数据集可以包括诸如文件或数据库表，该文件或数据库表可以为经由数据流图执行的计算提供数据（例如输入数据集）或接收数据（例如输出数据集）。

在数据流图中用“流”表示的数据流可以被组织为离散数据元素。例如，所述元素可以包括来自被组织为记录（或行）和字段（或列）的数据集的记录。描述对应于记录中的值的字段顺序和数据类型的元数据被称作为“记录格式”。

图形中的组件和数据集具有用于连接到流的输入和/输出端口。流 124 和 128 的“源端”分别与输入数据集 122 的输出端口及过滤组件 126 的输出端口连接。流 124 和 128 的“接收（sink）端”分别与过滤组件

126 的输入端口及输出数据集 130 的输入端口连接。数据集或组件的输入或输出端口与元数据，例如流入或流出端口的数据的记录格式相关联。

根据参数范围规则将包括端口的记录格式或与组件相关联的其它元数据的参数绑定为一个值。可以在设计时间或运行时间（即如下所述的“运行时间参数”）将参数绑定为一个值。参数值可以例如由用户在用户界面（例如响应提示）上定义，从文件中被定义或被定义为根据相同或不同上下文中的另一个参数。例如，通过指定参数具有与另一个参数“相同”的关系，参数可以从不同的上下文（例如在不同组件的上下文中被赋值的参数）导出。

用于图形中的组件可以通过与形成“子图形”的流互连的其它组件来实现。在将子图形用作其它图形中的组件之前，定义组件的各种特性，例如该组件的输入和/或输出端口。在一些情形中，与子图形组件之间关系有关的组件特性应该在该组件被用于图形之前被指定。例如，需要选择子图形组件的运行时间参数的提示顺序。下文将详细描述用于选择图形中组件的运行时间参数的提示顺序的方法。

元数据传送

与端口相关的元数据值，例如记录格式参数，可以通过“传送”而获得。元数据传送可以从“外部”或“内部”发生。对于外部元数据传送，通过传送第二个组件的端口的记录格式值，其经由流而连接到第一个组件，使得用于第一个组件的端口的记录格式参数值可以获得一个值。该值可以传送从流的源端到接收端的下游流，也可以传送从流的接收端到源端的上游流。元数据从已定义元数据的端口传送到未定义元数据的端口。

对于内部元数据传送，定义以用于组件的一个端口的元数据基于实现该组件的子图形而传送到该组件的另一个端口。在一些情形中，内部元数据传送发生在“非转换（non-transforming）”内部数据路径上。例如，用户可以为分类组件的输入端口提供元数据，该分类组件指定流入分类组件的记录的数据类型。因为该分类组件重新排序而不转换记录，因此该数据类型没有被该分类组件变换，数据类型未经改变地传送到分类组件的输出端口，该数据类型准确描述流出该分类组件的记录的数据

类型。

一些组件确实转换（或选择性地转换）了流过它们的数据。例如，用户可以为过滤组件的输入端口提供元数据，该元数据指定流入过滤组件的记录的字段。该过滤组件可以从每一个记录中移除给定字段的值。元数据定义可以用于根据该组件的过滤动作，指定该过滤组件的输出端口的元数据与该输入端口的元数据相关。例如，可以从指定记录字段的元数据中移除被过滤的字段。甚至在知道输入端口元数据之前，也可以提供这种元数据定义。因此，通过允许与端口相关的元数据被指定为一个或多个参数的函数一个或多个参数的函数，该参数包括另一个端口的元数据的，元数据甚至可以在转换内部数据的路径上传送，下文将对此详细描述。

当构建图形，并且用户为该图形中的一些组件的一些端口提供元数据时，可以将此内部和外部元数据的传送选择性配置为在设计时间发生。可选择地，元数据传送可以发生在图形被构建之后，包括在运行时间或正好在运行时间之前。

运行时间参数

运行时间参数允许应用的构建人员将参数设定值（例如分类函数的关键字参数、文件名、记录格式、转换函数等）延迟至运行时间（例如在计算机系统上执行或马上执行程序的时间）。运行时间参数的值可以由终端用户提供，或从其它运行时间参数的组合或存储在对象库内的对象中推导出来。

运行时间参数使应用增加一定程度的灵活性。通过利用这些参数来计算所需要的元数据（数据格式或*类型*，以及程序逻辑或*转换*），可以实现额外的灵活性。类型和转换可以从其它类型和转换、用户提供的参数值和存储的对象（例如来自库中的存储对象）综合得到。这样就可以构建“通用”应用，该“通用”应用工作于任何类型的输入数据，或通过一系列转换而产生数据，所述转换的构建由运行时间参数值来直接控制或由数值间接控制。

在一些实现方式中，当创建或编辑运行时间参数时，开发人员可以为每一个参数指定提示及显示该提示的条件。该系统解释提示指示，以

在条件符合的情况下提供用于接收参数值的图形用户界面（GUI）控制。

运行时间参数的指定

运行时间参数为开发人员提供一种机制，以在图形执行时间（即运行时间）基于外部输入修改图形的状态。在优选实施例中，这些外部值由用户直接输入提供数值。然而，这些外部值还可以来自多个不同源，包括环境变量和命令行参数。GDE 102 产生正确的代码以处理所有这些情形，以及在直接从 GDE 执行图形时向开发人员提示测试值。利用运行时间参数，例如，开发人员可以明确表示通过环境变量而为输入文件路径提供特定名称；该环境变量随后成为图形界面的已知部分。因此，对这些参数就存在有定义明确的界面。这里不需要，例如，读取所产生的 shell script 指令并搜索该 shell script 指令，用于引用环境变量和命令行自变量（argument），以发现控制特定图形执行的一组参数。

图 2 是具有汇总组件 202 和分类组件 204 的典型图形 200 的框图，该汇总组件 202 和分类组件 204 具有指定的运行时间参数。所述运行时间参数（分类组件 204 的关键字和汇总组件 202 的规则）将在用于输入的界面 206 中提供给用户。以下段落描述如何指定运行时间参数，以及创建用于提供提示用户输入的运行时间参数的组合用户界面。

运行时间参数可以以多种方式被指定或定义。一种方式是通过利用在 GDE 102 中显示的允许时间参数网格进行指定或定义。图 3 是表示与图形相关联的运行时间参数网格 300 的图形对话框的一个实施例图。通过在适当字段中的简单填写，可以创建新的运行时间参数。与每一个运行时间参数相关联的对象被创建在库 104 中，并连接到相关的使用该参数的所有图形组件。例如，如果将用以图形分类组件的分类关键字定义为运行时间参数，则用以表示该分类关键字参数的对象将被存储在库 104 中，并连接到相关的分类组件。定义运行时间参数的可选方式就是特别标记图形组件的现有参数，并使其对其它组件是“可见的”（将其导出）。还可以使用这些方法的组合。例如，当创建组件时，开发人员可以将该组件的特定参数指定为运行时间参数。然后，该开发人员利用参数网格来设定图形的所有运行时间参数的默认值（default value）和其它特性，并定义新的运行时间参数。

当运行图形时，参数被处理为可以从用户输入或从外部程序源获得每一

个参数（例如命令行参数或环境变量）的值。在示出的实施例中，运行时间参数网格 300 包括以下字段：

名称 302-该字段包含运行时间参数的名称。

“分数_阀值（score_threshold）”是作为名称的实例。

类型 304-该字段包含运行时间参数内允许的值的类型。“整数”是作为类型的实例。在示出的实施例中所支持的类型为：

- 布尔-其值可以是真或假；
- 选择-其值是一列值中的一个；
- 校正码（collator）-关键字参数值；
- 数据集-外部数据文件名和位置；
- 日期-日期值；
- 表达式-算法、逻辑和/或条件的表达式（例如选择表达式）；
- 浮点-浮点数；
- 整数-整数数字；
- 版面-并行或串行版面定义；
- 记录格式-记录描述或包含记录描述的文件；
- 串-任意字符串；
- 转换-转换描述或包含转换描述的文件。

位置（Loc） 306-该字段与记录格式和转换类型一起使用。其指定类型字段 304 是描述文件位置还是包含嵌入式描述。所支持的位置为：

- 嵌入式-参数包含记录或转换描述；
- 主机-参数包含对主机上文件的参考；
- 本地-参数包含对本地机上文件的参考；
- 库-参数包含对库转换或记录格式的参考。

默认值 308-该字段包含：（1）运行时间参数默认值，即如果从外部程序源没有提供其它值，此时将要使用的运行时间参数的默认值，或（2）规则或表达式，用来描述如何从用户输入得到运行时间值，或如何从用户执行图形时交互地获得信息的。在后一种情形中，如果用户没有提供输入值，则第二个默认值字段（未示出）可用来为运行时间参数提供值。对于“布尔”和“选择”类型，该字段限制用户进行有效选择。对于“版

面”类型，该字段是只读的并显示当前定义的版面定义。对于所有其它类型，该字段优选是简单的文本编辑器，用户可以输入有效字符串到该文本编辑器中。

编辑 310-敲击参数行中的编辑空间 310（或图标；例如铅笔图标）将产生更高级的编辑窗，使用户可以遍历各种用于编辑默认值字段 308 的选项。在所示的实施例中，以下编辑器对于相关类型是可用的：

- 单线编辑-用于整数、浮点、日期和字符串类型；
- 选择对话框-用于布尔和选择类型；
- 关键字编辑器-用于校正码类型；
- 文件浏览器-用于数据集类型和没有嵌入式位置的记录格式和转换类型；
- 转换编辑器-用于具有嵌入式位置的转换类型；
- 记录格式编辑器-用于具有嵌入式位置的记录格式类型；
- 表达式编辑器-用于表达式类型；
- 版面编辑器-用于版面类型。

上述编辑器都可以使用，除非类别字段值（如下所示）是“PL”（用于参数语言）。在此情形中，为用户提供编辑器以定义规则，该规则用于在图形执行时间得到或提示参数值。

描述 312-是自由格式字段，开发人员用它来描述运行时间参数的预期值。如果默认值包含用于询问用户输入值的规则，则在运行时间用它来作为提示。

类别 314-在图形执行时间，该字段定义图形在何处获得相关参数的值。该类别字段 314 所支持的值为：

- 环境-运行时间参数的值预期能在相同名称的环境变量中发现。如果环境变量未被定义，则使用默认值字段 308 中的值。如果需要该参数（例如导出参数），而默认值字段 308 为空时，则将产生运行时间错误并且停止图形的执行。
- 位置-运行时间参数的值预期处在用于调用应用的命令行的相对位置上。例如，如果运行时间参数是已定义的第三位置运行时间参数，则预期其参数值是执行程式中的第三位置命令行的自变

量。必须提供任何指定的位置参数，如果错失一个就会产生运行时间错误。

- 关键字-运行时间参数的值预期是关键字命令行参数。在所示出的实施例中，关键字参数是以下形式：-<参数 名称> <参数值>。关键字参数是可选择的，如果未提供关键字参数且默认值字段 308 为空，而且还需要对应的导出参数，仅在此时产生运行时间错误。
- 固定值-参数的运行时间值总是默认值。这对于在两个或多个运行时间参数之间共享不变的值来说是有用的。
- PL-运行时间参数的默认值包含 PL 表达式，在图形执行时，其被解释为或者从其它参数得到该运行时间参数的值，或者提示用户另外输入。在本发明的任何特定实施例中的组件，其组件描述语言可以选择任何合适的程式语言，例如公开可得的面向对象的程式语言“Python”。这些程式可以在程序控制下构建元数据（类型和转换），并执行条件测试、比较、数据转换、算法和逻辑运算、字符串和列表处理、其他用户输入功能、外部程序式提供的输入和其它运行时间参数，以产生任何运行时间参数的最后值。

在所示出的实施例中，已经在运行时间参数网格 300 上直接创建的用于引用运行时间参数的有用的习惯用法是简单地输入前面带有美元标记“$”的参数名。例如，*$关键字*引用的是名称为*关键字*的运行时间间变量。在所示出的实施例中，根据默认运行时间类别（默认运行时间类别是“环境”）的高级选项对话框中的值，新的运行时间间参数默认“字符串”类型和默认的类别。

因为运行时间参数值可以在运行时间被确认，并且 PL 程式可以提供条件测试，因此可以创建“有条件的”运行时间参数。只有参数的所有条件都能够在运行时间确定时，有条件的运行时间参数才对用户输入产生提示。因此，例如，如果用户响应第一提示，该第一提示请求是否将数据集分类为“NO”，则请求分类关键字的有条件的第二提示就不需要被显示。

因此，在设计阶段（“设计期间”），开发人员指定图形组件的特定参数为“运行时间”参数。然后将与该图形组件相关联的对象以相关

的参数数据（例如图 2 中参数网格 300 的信息类型）进行存储。

图 4 是总结使用运行时间参数的程序的流程图。在运行时间期间，恢复对应于将要执行的应用的参数对象（例如从库中恢复），如步骤 400。对每一个对象进行是否指示用户输入的判断，如步骤 402。如果有指示，则进行是否已满足显示提示的任何条件判断，如步骤 403，这可以包括对在先提示的用户输入进行赋值。如果没有指示，则使用默认值，如步骤 408。可选择地，也可以不需要该参数值（例如，如果用户没有选择启动分类函数，则不需要分类关键字），从而可以忽略该参数值。否则，对用户输入产生提示，如步骤 404。

如果用户没有输入特定参数的值，如步骤 406，则可以选择该参数的默认值，如步骤 408。可选择地，可以给出错误条件以指示缺少用户输入。在任何情况下（假设没有因为缺少用户输入而产生错误条件），通过考虑基于其它值的输入转换、相关性和条件，确定参数的最终值，如步骤 410。

如果确定用户输入没有指示特定参数，如步骤 402，则作出关于参数值是否由外部编程提供的判断，例如通过环境变量或命令行参数提供，如步骤 412。如果不是，则选择该参数的默认值，如步骤 414。可选择地，可以给出错误条件以指示缺少指定类型的可用输入。在任何情形下（假设没有因为缺少外部输入而出现错误条件），通过考虑基于其它参数的输入转换、相关性和条件来确定参数的最终值，如步骤 410。

一旦确定最终的参数值，则作为可选步骤，根据上述的特定条件和规则，所有的条件组件（如下所述）可以被完全移除或被流（即图形链接或边）所替换，如步骤 416。一旦可操作的图形结构完成并且最终的参数值被确定，则以常规方式执行图形，如步骤 418。

测试值

在创建和测试具有运行时间参数的图形的期间，为了支持开发人员，GDE 102 的优选实施例也支持运行时间参数的测试值。当开发人员运行具有运行时间参数的图形或想要查看影响图形组件的基础代码时，GDE 102 显示相关的测试参数网格，在该参数网格中用户可以为一个或多个运行时间参数输入新的测试值。优选地，所用的最后一组测试值与图形一

起被记录和保存。

对于每一个运行时间参数，开发人员在测试值列中输入预期的测试值。编辑字段可以与每一个测试值列相关联。除了在参数类型是 PL 时之外，该测试值字段和编辑字段与运行时间参数网格 200 中的默认值字段和编辑字段作用相同。

如果 PL 表达式指示需以特定运行时间参数的值来提示用户，则该测试值字段和编辑行为以相关 PL 表达式的解释。如果该 PL 表达式基于其它输入而简单得出值，则在标准模式下，该运行时间参数在测试值网格中不可见。

描述运行时间参数如何获得它们的值

在将参数指定为运行时间参数之后，在库 104 中创建对应的对象。如果运行时间参数具有类别字段 214 的值“PL”，则参数的默认值字段 308 包括具有以下优选形式的 prompt_for 伪函数（pseudo-function）：

prompt_for “prompt-kind [modifiers]” options…

如上所示，prompt_for 伪函数可以是用以判断是否基于在先输入来显示提示的条件表达式的部分。

对于这些对象，用户界面用于将直接输入的运行时间参数提供给用户。在优选实施例中，是由 Web 界面 108 提供该功能。特别地，在运行时间期间，由 Web 界面 108 对每一个运行时间参数对象的每一个 prompt_for 伪函数进行解析（parse），以生成具有对应的用户提示的网页（例如以 HTML 形式）。（可选择地，这种网页可以在运行时间之前被生成并在运行时间期间被简单呈现。然而，这种网页产生运行时间的方式提供了更大的灵活性。特别是网页内容可以依赖于在先的用户输入时。）Web 界面 108 的使用与显示这种网页并接收用户输入的常规网页浏览器相结合。

prompt_for 伪函数指示 Web 界面 108 如何提示参数值。特别地，该 prompt_for 伪函数，即字符串常数，指示哪种类别的用户界面（UI）元素（文本框、下拉列表等）需要被提供。该字符串的*修改符（modifiers）*部分，即用逗号隔开的关键字列，为各种提示提供一些共同的选项。在所示出的实施例中，在修改符字符串内的空间不是巨大的。对修改符关

键字的解释如下：

- in place（本位）关键字表明该元素应该直接在应用的总汇级（summary level）用户界面上提供，允许在不“进入”较低级界面的情况下提供该值。如果 in place 未被指定，则在该总汇级界面上提供简单的“编辑”按钮，该按钮将用户带入用以提供参数值的另一个页面。
- 关键字 blank ok 表明用户不需要提供值；该应用将以合理的方式处理默认值。如果 blank ok 未被指定，则在没有提供一些值的情况下该用户无法执行该应用。

以下是具有不同种类修改符的 prompt_for 调用的一些实例：

$ {prompt_for “text, inplace”}

$ {prompt_for “filter, in place”, $ input_type}

$ { prompt_for “radio, blankok, in place”, $ {list 1,2,3}}

此段落的剩余部分列出各种 prompt_kind（提示-类别）和它们对应的选项，并解释每一个 prompt_kind 如何出现在通过 Web 界面 108 产生的网页中。

文本*[大小]*-提供常规的单行文本框*大小*的字符宽（如果*大小*未被提供，则其默认为浏览器针对文本框的默认大小）。

radio（单选）*选项-列表[描述-列表]*-以一组单选按钮的形式提供常规的“选择一个”提示，每一个按钮对应*选项-列表*的每一个元素。如果提供的是*描述-列表*，则以对应的描述标记每一个选项；否则，以*选项-列表*中的对应项的字符串形式标记选项。

Radioplus（单选＋）*选项-列表 [描述-列表]*-y 与 radio 方式类似，但在文本框旁边提供另外的按钮，以允许用户选择不在*选项-列表*中的“写入”值。

Checkbox（复选框）*选项-列表[描述-列表]*-以一组复选框的形式提供常规的“选择零个或多个”提示，一个按钮对应于*选项-列表*的每一个元素。如果提供的是*描述-列表*，则以对应的描述标记每一个选项；否则，以*选项-列表*中的对应项的字符串形式标记选项。

Dropdown（下拉）*选项-列表[描述-列表, 大小]*-以*选项-列表*中元素

的下拉列表的形式提供常规的“选择一个”提示。如果提供的是*描述-列表*，则以对应的描述标记每一个选项；否则，以*选项-列表*中对应项的字符串形式标记选项。如果提供有*大小*，则一次可以查看很多选项；否则，一次只能查看一个。

Multidropdown（多个下拉）*选项-列表[描述-列表，大小]*-以*选项-列表*中元素的下拉列表形式提供常规的“选择零个或多个”提示。如果提供的是*描述-列表*，则以对应的描述标记每一个选项；否则，以*选项-列表*中对应项的字符串形式标记选项。如果提供有*大小*，则一次可以查看很多选项；否则，显示浏览器的默认项数。

Key（关键字）*类型-对象[大小]*-提供由给定的*类型-对象*的字段组成的关键字（也是公知的校正码）的提示。该关键字可以具有与*大小*同样多的部分，默认为*类型-对象*中的字段数。图 5 是通过关键字提示产生的图形对话框 500 的一个实施例图。其后是用以 3-输入关键字的提示的程式文本实例，其中文件*/数据集/固定的*用来定义在下拉框 502 中示出的可用关键字的内容：

${prompt_for “key”, ${dataset_type “/datasets/fixed”},3}

在所示出的实施例中，标准的校勘（collation）顺序是递增的，但用户也可以通过检查相关的复选框 504 来选择关键字的递降（descending）校勘顺序。

filter（过滤）*类型-对象*-提供由针对给定的*类型-对象*的每一个字段的条件组成的过滤表达式的提示。blank ok 修改符对过滤没有影响；空白过滤得到“真”表达式。图 6 是通过 filter 提示而产生的图形对话框 600 的一个实施例图。与每一个表达式文本编辑框 604 相关的可用字段名称 602 由*类型-对象*定义。将比较值输入到文本编辑框 604 中，以及从对应的下拉列表控件 606 中选择比较运算符（例如相等、大于、小于或等于）。

flexifilter（灵活性过滤）*类型-对象*-与 filter 提示类似，但提供由针对给定的*类型-对象*中每一个字段的条件组成的过滤表达式的提示，在每一个字段中从下拉列表选择每一行上的字段名称。这允许将相同字段用以多个条件（例如，*字段* STATE=MA 或*字段* STATE=CA）。

rollup（汇总）*类型-对象 关键字[大小]*-根据由给定关键字汇总的给

定*类型-对象*的字段，提供汇总计算的提示。该汇总可以具有与*大小*一样多的规则，该规则个数默认为*类型-对象*中的字段数。blank ok 修改符对汇总没有影响；空白汇总产生仅为每一组提供关键字值的程序包，图 7 是通过 rollup 提示而产生的图形对话框 700 的一个实施例图。在所示出的实施例中，由下拉框 702 的列来定义可用的汇总计算函数（例如求和、最小值和最大值）。与每一个计算相关联的可用字段名称 704 由*类型-对象*定义。每一个汇总规则具有让用户定义预期表达式的相关文本编辑框 706，用于定义（通过布尔表达式）标准的“地点”文本编辑框 708，以及用于指定将要接收计算结果的字段的输出字段文本编辑框 710，在该标准下源值将参与该计算。在可以明确获得的情形下，输出字段的名称不需要被指定。

reformat（重新格式化）*类型-对象[大小]*-根据给定*类型-对象*的字段提供重新格式化计算的提示。重新格式化可以具有与*大小*一样多的规则，该规则个数默认为*类型-对象*中的字段数。图 8 是由该 reforma 提示而产生的图形对话框 800 的一个实施例图。在所示出的实施例中，reforma 提示包括用于简单复制输入字段至同名（like-named）输出字段的部分 802（或者利用复选框控件个别选择/取消选择，或者利用选择所有或全部不选择按钮全部选择/取消选择）。第二个提示部分包括允许定义重新格式化表达式（例如 total=revenue_1-revenue_2）文本编辑框 804 的列。每一个规则具有相关的输出字段文本编辑框 806 用于指定将要接收重新格式化结果的字段。

outputspec（输出说明）-提供输出数据集说明的提示。所显示的控件包括用于提供可用的格式化选项的下拉控件和用于给输出数据集的特定实例输入名称的文本编辑框。blank ok 修改符对输出数据集说明没有任何影响。

fpath（文件路径）*起点*-提供文件路径的提示。该提示实质上是文本框，但在该提示旁有“浏览”按钮，该按钮将导致用于浏览文件路径的弹出窗口的出现。如果该文本框是非空白的，则将该文本框用作浏览操作的起点；如果该文本框是空白的，则使用*起点*参数。

rpath（库路径）*起点*-提供库路径的提示。该提示实质上是文本框，

但在该提示旁有“浏览”按钮，该按钮将导致用于浏览的弹出窗口的出现。如果该文本框是非空白的，则将该文本框用作浏览操作的起点；如果该文本框是空白的，则使用*起点*参数。

radiofpath（单选文件路径）*选项-列表[描述-列表]*-与 radioplus 类似，但在“写入”口（slot）中提供 fpath 类型的框＋浏览按钮。

radiorpath（单选库路径）*选项-列表[描述-列表]*-与单选＋类似，但在“写入”口中提供 rpath 类型的框＋浏览按钮。

条件组件

在一些实施方式中包括条件组件机制，该条件组件机制允许根据参数值和计算出的元数据来改变图形的组件结构和流。图形的每一个组件具有用以控制该组件是否在运行时间期间出现在图形中的条件。该条件可以被直接计算出来，或通过运行时间参数被间接计算出来。条件组件可被用于各种目的，例如图形的优化或专业化。出于优化目的，如果来自特定数据集的值不被使用，则应用可以省略对所述数据集的处理，从而使图形更有效地运行。出于专业化目的，应用可以根据预期的细节等级来调节多个不同输出数据集的产生，或允许执行图形的多个可选部分的一个部分。

图 9A 是第一个图形的框图，其中合并组件（MergeJoin component）900 从文件 A 和 B 中组合数据，并输出结果至输出文件 902。图 9B 是第二个图形的框图，其中 Rollup 组件 904 汇集文件 A 的数据并输出结果至输出文件 902。图 9C 是一个图形的框图，其中 MergeJoin 组件 906 从文件 A 和 B 中组合数据，而 Rollup 组件 908 汇集所产生的数据并输出最终结果至输出文件 902。利用条件组件，这三个图形可以被组合为单一图形，该单一图形最初看起来很像图 9C 的图形，但其精确结构要一直等到运行时间期间才能被确定。通过设定适当条件，Rollup 组件 908 可以被连接（流）替换，导致类似于图 9A 的图形的运行时间图形。类似地，通过设定适当条件，MergeJoin 组件 906 可以被连接到文件 A 的连接（流）替换，从而导致类似于图 9B 的图形的运行时间图形。

在所示出的实施例中，条件组件可以是定义顶点的任何图形组件（即，诸如输入/输出文件的数据集组件、诸如重新格式化或分类组件的

处理组件或其它图形，其作为子图形而被公知）。在优选实施例中，条件组件由两个特定参数控制：*条件*和*条件解释(interpretation)*。条件是布尔表达式，或是被延迟到运行时间期间才进行赋值。在所示出的实施例中，值“假”和“0”指示假条件，所有其它值（包括空白）指示真条件。条件解释参数具有两个允许的互斥值：*完全移除*和*以流替换*。

图 10 是用以提供具有条件解释控件 1004 的条件 1002 的图形对话框 1000 的一个实施例图。条件解释控件 1004 允许选择*完全移除*解释 1006 或*以流替换*解释 1008。

完全移除：根据该解释，如果满足条件，则从图形中移除组件和其所有被连接的流（即图形链接或边）。有效的*完全移除*条件从图形中功能性地移除该组件和其所有被直接连接的流。*完全移除*条件可被用于任何组件。

从图形中移除的条件组件可能“损坏（poison）”依赖于条件组件的存在的其它连接组件，使得这些连接组件被移除。图 11 是示出发生此种损坏（poisoning）情形时的图形 1100 的示意图。如果针对输入文件组件 1102 的条件的指示是移除，并且对应的条件解释是*完全移除*，则输入文件组件 1102 和其连接流两者都从图形 1100 中被移除。这样就会损坏分类组件 1104 而使其被移除，这是因为分类组件 1104 的输入是有要求的输入端口的，而此时不再有任何数据流连接到该输入端口。这样就会损坏汇总组件 1106，使其被移除，因为汇总组件 1106 的输入是有要求的输入端口的，而此时不再有任何数据流连接到该输入端口。阻止此“损坏不出现”的唯一方法是连接到下游组件的可选用端口或支持（counted）端口。因此，当针对输入文件组件 1102 的条件的指示是移除时，整个分类-汇总图形分支 1108 被从图形 1100 中有效地移除。图 11 中是原始图形结构的名义上为 3 次输入的组合组件 1110 在运行时间期间变成 2 次输入组合组件的结果。

在一个实施方式中，损坏的详细语义（也被公知为“隐含条件”）如下所述：

- 如果组件具有有要求的端口并且没有任何实时流连接到该需要的端口，则从图形中移除该组件和所有连接到它的流。

- 如果从图形中完全移除组件，则从该图形中移除连接到该组件端口的所有流。
- 如果以流替换组件，则从图形中移除连接到除了组件的*指定输入端口*和*指定输出端口*之外的所有端口的所有流。
- 如果有要求的索引端口没有连接任何实时流，则对于具有相同索引的每一个对应的可选用索引端口而言，任何连接到该对应端口的流均被从图形中移除。

这些规则具有一些意外的结果。例如，由于损坏的原因，仅具有可选用端口的组件可以总不被移除。因此，如果期望移除该组件的话，必须明确执行。

图 12 是总结包括*完全移除*条件组件的图形的运行时间准备程序的流程图。如果条件解释是*完全移除*而并未满足该条件，如步骤 1200，则不从图形中移除该条件性的组件（COMPONENT），如步骤 1202。如果满足该条件，如步骤 1200，则该条件性组件连同连接到该组件的所有流，如步骤 1204，均从该图形中被移除。然后，根据上述规则，从图形中移除所有“被损坏的”组件和流，如步骤 1206。

以流替换：根据此解释，如果满足条件，则用流（即图形的边）来替换该组件。*以流替换*条件解释需要附加的信息。参考图 10，当从图形中移除组件时，用户指定输入端口 1010（或支持端口群）和输出端口 1012（或支持端口群）以通过它们进行连接。通过默认，如果这里确实有一个有要求的输入端口或支持端口，和需要的输出端口或支持端口，则这些就是指定的流经（flow-through）连接端口（分别归类为*指定输入端口*和*指定输出端口*）。其中，所谓有要求的端口就是至少要求一个被连接的流的端口。

图 13 是针对本发明特定实施例而总结包括*以流替换*条件组件的图形的运行时间准备程序的流程图。由于在所述实施例中一些组件依赖于某些可用的输入和输出（依据的是在 CO>OPERATING SYSTEM®中可用的组件），因此将多个规则应用到该实施方式中并使用*以流替换*条件：

- 如果条件解释是*以流替换*而该条件并未满足，如步骤 1300，则不从图形中移除该条件组件，如步骤 1302。

- 仅在确实存在有连接到组件的指定输入端口的一个实时的直流，和连接到组件的指定输出端口的一个实时的直流时，具有指定输入端口和指定输出端口的组件才可以被流替换（“实时的”流是指在运行时间期间未被移除的流），如步骤 1304。如果这样的话，则从该图形移除该组件自身，而连接到该组件的指定输入端口的实时的直流和连接到该组件的指定输出端口的实时的直流则被互相连接，如步骤 1306。将直接连接到已被移除的组件的其它端口（即除了特别指定的输入端口和输出端口之外的任何端口）的任何其它流从该图形中移除。如上所述步骤 1308，移除连接到已被移除的组件的任何“被损坏的” 流和组件。
- 对于具有*以流替换*条件的组件，如果其具有连接到多于一个的支持输入群中的*指定输入端口*的实时流，如步骤 1310，则不从图形中移除该组件，因为需要该组件来保证图形的有效性，如步骤 1312。

对具有有要求的输入的实时输入流（fan-in flow）的组件需要进行特别处理。“实时输入流”的意思是该组件可以具有连接到有要求的输入端口的实时的输入流或全部对全部（all-to-all）流，或者该组件业可以具有连接到单独的有要求的输入端口的多于一个的实时直流。对于这些组件而言，对*以流替换*条件的解释应该是用集合所有实时输入流的集合组件来替换该条件性组件，如步骤 1314。然后，如上所述，如步骤 1316，移除连接到已被替换的组件的任何“被损坏的”流和组件。

元数据传送方案

可以通过例如图形开发者、图形使用者或通过从图形的其它部分传送来提供图形的元数据。可以传送各种元数据，包括与数据有关的元数据或关于数据的计算，例如：端口的记录格式（例如流入或流出端口的字段顺序和记录的数据类型）、分类、压缩方法、字符集、二进制表示（大端字节（big-endian）、小端字节）、划分、该组件可能使用哪些计算资源（例如处理器、临时磁盘空间）、数据转换和该组件可能使用的存储量。图形结构的各个方案可以影响元数据的传送。以下将描述其中的两个方案。

在组件移除之后传送

在一些实施方式中，当在移除图形组件之后产生流时，关于定义该流中数据的元数据应该在修订的图形中如何传送，必须做出选择。元数据可以从流的任一端得到。在一些实施方式中，来自该流的上游端的元数据是优选的。

如果该流的上游端是被移除的组件（或是已由集合组件替换的组件），则 GDE 102 通过“遍历”该图形的上游直到找到未被移除的组件，来寻找该流的元数据。由上游组件揭示的元数据用于定义所产生的流的数据特性。

转换后的元数据传送

如上所述，通过允许将与端口相关的元数据指定为一个或多个参数（包括另一个端口的元数据）的函数，元数据甚至可以在用于转换内部数据的路径上传送。例如，图 23A 示出计算来自数据集 2302 和数据集 2304 的数据的组合操作的图形 2300。在此实例中，图形开发者在所述数据集的输出端口处提供元数据。然后将该元数据传送给“智能组合（smart join）”组件 2306，其用于对输入数据集的记录计算组合操作。例如， 元数据从输出端口 2308 传送到输入端口 2310。然后通过“智能组合”组件 2306 转换该元数据并将转换后的元数据从“智能组合”组件 2306 的输出端口 2316 传送到过滤组件 2318 的输入端口 2317。

图 23B 显示了实现“智能组合”组件 2306 的子图形。组件 2306 使用*关键字_字段*参数，该*关键字_字段*参数的值表示由组合组件 2350 执行的组合操作的关键字字段。组件 2306 还将该*关键字_字段*参数用作包括条件分类组件 2354 和 2356 的条件。如果流入到输入端口 2310 的记录已经按照*关键字_字段*进行了分类，则分类组件 2354 不被调节。类似地，如果流入到输入端口 2314 的记录已经按照*关键字_字段*进行了分类，则分类组件 2356 不被调节。如果输入记录的任一流仍还没有按照*关键字_字段*进行分类，则分类组件 2354 和 2356 在所述记录流入到组合组件 2350 之前对它们进行分类。

为了使转换后的元数据传送通过此“智能组合”组件，图形开发者将“智能组合”组件 2306 的输出端口 2316 的元数据（例如描述字段的

元数据）定义为第一个输入端口 2310 的元数据*输入 0.元数据*、第二个输入端口 2314 的元数据*输入 1.元数据*和关键字字段参数*关键字_字段*的函数：

输出.元数据=元数据_组合（关键字_字段，输入 0.元数据，输入 1.元数据）

通过绑定函数自变量至多个值（关于适当上下文）并根据绑定结果执行函数*元数据_组合*，来确定输出端口元数据。在此实例中，因为端口 2310 和 2314 的元数据是未定义的，因此将所传送的元数据绑定至元数据参数*输入 0.元数据*和*输入 1.元数据*。用户为输出端口 2308 提供元数据，输出端口 2308 为从端口 2308 流到“智能组合”组件 2306 的输入端口 2310 的记录指定字段“A”和“B”。用户还为输出端口 2312 提供元数据，输出端口 2312 为从端口 2312 流到“智能组合”组件 2306 的输入端口 2314 的记录指定字段“A”和“C”。此用户提供的元数据传送至端口 2310 和 2314。组合操作的关键字字段是字段 A，从而将“形式参数”*关键字_字段*绑定至值“A”。

通过首先确定*关键字_字段*参数的值是否是*输入 0.元数据*和*输入 1.元数据*指定的字段集的项，函数*元数据_组合*确定输出元数据。如果是，则该输出元数据是两个字段集的并集。如果不是，则输出元数据表示空字段集。

在该元数据传送到“智能组合”组件 2306 的输入端口（或另外例如由用户所提供）之后，“智能组合”组件 2306 的输出端口的转换后的元数据包括字段 A、B 和 C。然后可将转换后的元数据传送到其它组件。在该实例中，转换后的元数据传送到过滤组件 2318。

不管是用户提供的或是在端口之间传送的元数据都能被显示给用户。例如，用户可以利用输入装置（例如鼠标）来选择查看元数据值的组件部分。还可以响应这种用户选择来触发元数据传送。

示例性元数据传送进程

图 24 示出了示例性元数据传送进程 2400 的流程图。进程 2400 可以在例如每次图形变化时、响应用户动作时和/或仅在图形运行之前被执行。进程 2400 产生工作列表，如步骤 2402，该工作列表具有根据多个流确定

的部分顺序而进行排序的图形中的每一组件（例如，如果存在从组件 A 到组件 B 的流，则组件 A 在组件 B 之前出现）。在流没有确定两个组件之间的唯一顺序的情况下，可以将组件标签的字母顺序用作仲裁顺序（tie-breaker）。这为该工作列表中的组件提供稳定的顺序（假设组件的标签是唯一的）。如果对图形重复传送进程 2400（例如在添加新的组件之后），则新的工作列表保持先前工作列表中的组件之间的顺序相同。

进程 2400 在工作列表的起点处开始，并且对于该工作列表中的每一组件，进程 2400 基于实现该组件的子图形的规定（例如子图形中的数据流）在该组件内内部地传送元数据（例如，从输入端口到输出端口，或从输出端口到输入端口），如步骤 2404。该内部元数据传送包括在不转换数据的路径的任一端上的端口之间传送未转换的元数据。内部元数据传送还包括获得具有元数据定义的端口的元数据和/或其它端口的元数据，该元数据定义指的是图形参数，如上所述。当进程 2400 遇到这种元数据定义时，进程 2400 对任何这样的参数赋值，所述参数的值要用来获得元数据。

在对工作列表上的组件执行内部元数据传送之后，进程 2400 从具有元数据的组件的每一个端口外部地传送元数据至不具有元数据的相关组件的端口，如步骤 2406。将通过此外部传送获得元数据的任何组件移至工作列表的末端，如步骤 2408。进程 2400 在工作列表上的最后组件被处理之后终止，如步骤 2410。

支持这种外部元数据传送的组件之间的一种关系是两个组件的端口之间的数据流链接（例如从输入端口到输出端口，或从输出端口到输入端口）。

支持这种外部元数据传送的组件之间的另一种关系是一种链路，其指示一个端口的元数据也可以用于另一个端口。此种“元数据链接”不必对应于数据流链接。例如，端口可以具有连接到与任何特定端口无关的图形中元数据的元数据链接。

组件化子图形中的运行时间参数

在子图形被“组件化”以用作另一个图形中的组件之前，定义该组件的各种特性，例如该组件的输入和/或输出端口。对于包括具有运行时

间参数的组件的子图形，应该选择运行时间参数的提示顺序。因为图形中的组件不是必须按序排序，从而可以存在用于提示用户的多种可能的运行时间参数的全局排序。一些全局排序与关于每一个组件的原始排序不一致。这有利于产生用于提示尽可能保持每一个组件中参数排序的全局排序，同时在适当考虑相关性时重新排序。例如，组件可能将询问“what data would you to process?”的提示排在询问“where would you like to store the processed data?”的提示之前。虽然可以以任一顺序提供所述提示，但期望以该顺序提供所述提示。

因为对提示的运行时间参数进行赋值的过程中需要对未提示的运行时间参数赋值，因此根据所有运行时间参数的赋值顺序来获得提示顺序。用于确定图形的运行时间参数（包括与任何组件不相关的图形的参数）的赋值顺序的一种方法，包括基于表示参数间相关性的一个或多个有向无环图执行拓扑分类。然而，一些拓扑分类算法可能对参数进行不必要的重新排序，从而导致不想要的运行时间参数的提示顺序。

排序实例 1

在第一个实例中，参数排序进程为两个图形组件的参数提供初始参数列表，两个组件为组件 I 和连接到组件 I 的组件 II。在此实例中，参数仅具有“组件内”相关性。也就是说，组件的参数仅与同一组件中的其它参数相关。定义所述参数如下：

组件 I 包括以下参数：

x= $ {prompt_for “text”}

y=x+ $ {prompt_for “text”}

z=x+y+ $ {prompt_for “text”}

q= $ {prompt_for “text”}

组件 II 包括以下参数：

a= $ {prompt_for “text”}

b=a+ $ {prompt_for “text”}

c= $ {prompt_for “text”}

上述列出的参数顺序定义了向用户提示各值的理想顺序。初始参数列表为每一个组件保持此“初始排序”。将“序数”分配给每一个参数

以指示参数在初始排序中的位置。下面的表格列出该初始排序中的参数：

参数	序数	相关性
x	0	
y	1	x
z	2	x, y
q	3	
a	4	
b	5	a
c	6	

“相关性”列表示与所列出的参数相关的其它参数。该相关性对参数赋值实施排序限制条件：在参数被另一个参数使用（例如引用）之前需要被定义。

“通用拓扑分类”算法在每次传递时通过该列表将具有零相关性的参数传送至有序输出列表。在每次传递之后，从相关性列中移除任何被传送的参数。重复此进程，直到所有参数已被传送。输出列表中的参数顺序表示“最后排序”，以使得在其它参数已被赋值之后，与这些其它参数相关的参数被赋值。

在该实例中，在第一次传递时，将参数 x、q、a 和 c 传送到输出列表。在第二次传递时，将参数 y 和 b 传送到输出列表。在第三次和最后一次传递时，将参数 z 传送到输出列表。因此，所述参数的最后排序是：x、q、a、c、y、b、z。同时此排序满足由参数相关性提出的排序限制条件，从而不必对所述参数进行重新排序。在此实例中，初始排序也满足由参数相关性提出的排序限制条件。

满足排序限制条件的用于确定图形参数赋值顺序的其它方法遵守初始排序。例如，一些方法对参数进行排序以满足排序限制条件，根据以初始排序为基础的准则来选择排序。该准则可以包括给予保持接近初始排序的顺序（例如，基于初始排序的改变来最小化度量（metric））优先权的任何类型的准则。在一些情况下，由于根据准则，有多种排序可能

同样满足给定的准则，所以可能没有唯一的“最优”排序。

遵守初始排序的方法实例是“改进的拓扑分类”方法。在此方法中，基于初始排序的准则是在与任何未被传送的参数不相关的前述参数被传送之前，最小化从初始列表传送的参数数目。换句话说，在传送具有零相关性的下一个参数之前，该“改进的拓扑分类”从相关性列中移除所传送的参数。对于上述实例而言，该“改进的拓扑分类”方法产生与初始排序相同的最终排序：x、y、z、q、a、b、c。

遵守初始排序的改进的拓扑分类进程

以下对两个示例性“改进的拓扑分类”进程给出伪代码，两个所述“改进的拓扑分类”进程都遵守由每个参数的分配序数所确定的初始排序。第二个进程包括优选法以改善一些情形的时间效率。所述进程处理从参数的输入数据产生的数据结构。

假设有 *N* 个待排序的参数，则输入数据包括由唯一参数名、与所命名的参数相关的一组参数（称作为“相关集”）和组成的 *N* 个三元组列表，其中任选属性数据对象用于存储与所命名的参数相关的信息。

与该输入数据相关的是一个或多个有向无环图，所述有向无环图表示参数之间的相关性，称作为“相关图”。每个唯一参数名对应于相关图中的节点，有关的相关集对应于从其它节点到该节点的一组链接。所以链接从第一参数的第一节点指向与该第一参数相关的第二参数的第二节点。可选择地，链接方向与参数相关性之间的对应关系可以相反。

输出数据结构 *result_list* 包括来自重新排序的输入数据的 *N* 个参数列表（如果必要的话），从而在参数用于赋值另一个参数之前为该参数赋值，同时给出保持接近初始顺序的顺序优先权。为了产生输出数据结构 *result_list*，通过从工作数据结构 *param_list* 到输出数据结构 *result_list* 每次传送一个参数，所述进程“移除”参数。在所有参数被移除之后完成该输出数据结构。

第一个“改进的拓扑分类”进程包括两个阶段。在第一个阶段中，该进程基于输入数据来构建工作数据结构，其中所述输入数据用于产生所分类的输出数据结构。在第二个阶段中，该进程根据由这些工作数据结构表示的相关性限制条件反复排序并移除参数。

该进程在第一个阶段中构建的一些工作数据结构是字典，所述字典是基于散列法的数据结构。字典中的项可以在O(log*N*)时被有效地存取。以下的示例性数据结构在第一个阶段中被构建：

parm_list（参数列表）*[索引(index)]*：未被移除的参数名有序的列表，由多个 *index* 来编索引（其中 *index*=0 对应该列表中的第一项）。此数据结构是“动态的”（即，在进程执行期间发生改变）。该列表由位置来编索引，使得如果从该列表的中间移除一项，则位于所移除的项之后的项的索引相应地移动。

n_dependencies_dict（n_相关性字典）*[名称(name)]*：由参数名称（*name*）作为关键字（keyed）的字典，该字典的条目包含与关键字参数相关的参数个数。此字典是动态的。

depenters_dict（相关字典）*[name]*：由参数名称（*name*）作为关键字的字典，该字典的条目是字典（也由参数名称作为关键字），表示与该关键字参数相关的参数组。此字典是“静态的”（即，在进程执行期间不发生改变）。

order_dict（排序字典）*[name]*：由参数名称（*name*）作为关键字的字典，其存储序数位置，初始排序中的参数的整数范围从 0 至 *N*-1。此字典是静态的。

attribute_dict（属性字典）*[name]*：由参数名称（*name*）作为关键字的字典，其存储关键字参数的任选属性数据对象。此字典是静态的。

result_list（结果列表）*[index]*：表示进程输出的参数名称和属性的有序列表，其由数字 *index*（其中 *index*=0 对应于该列表中的第一项）来编索引。此数据结构初始为空。此数据结构是动态的。

出于分析所述进程的时间效率的目的，相关图的平均“度”（或从节点的链接数）假设是 *z*。构建这些数据结构花费 O(*N*)时间，除了 *n_dependencies_dict* 和 *depenters_dict* 花费 O(*N***z*)时间。

在第二个阶段，该进程根据分类准则 *by_n_deps_and_order* 对 *param_list* 数据结构中的参数进行分类，该分类准则首先通过与所述参数相关的未被移除的参数个数（即，通过所述参数的值 *n_dependencies_dict*）从最低至最高对所述参数进行排序，然后通过所述参数的序数（即，通

过它们的值 *order_dict*）从最低至最高对所述参数进行排序。然后该进程在已分类的 *param_list* 中消除第一参数。此参数的值 *n_dependencies_dict* 应该是零。（如果在已分类的 *param_list* 中的第一参数的值 *n_dependencies_dict* 不是零，则标记错误）。

为了消除参数，进程将该参数添加到 *result_list*（连同任何相应的属性）并将所有相关者（即 *dependers_dict* 中的参数）的相关计数（即，*n_dependencies_dict* 的值）减 1。最后，从 *parm_list* 中删除该参数。重复此分类和消除所产生的第一参数的步骤，直到所有参数被消除。

以下是 *eliminate（消除)*程序的伪代码定义：

```
def eliminate (list, index):
    result_list.append ((list[index], attribute_dict[list[index]]))
     for depender in dependers_dict[list[index]]:
n_dependencies_dict[depender]=n_dependencies_dict[depender]-1
      delete list[index]
```

eliminate 程序的自变量是 *list*（其值是例如 *param_list*）和 *index*。函数 *result_list.append* 将在位置 *index* 处指示的 *list* 项连同其相关属性添加到 *result_list*。然后，该程序递减每个参数 *depender* 的值 *n_dependencies_dict*，该参数 *depender* 是 *dependers_dict* 数据结构的组员，由正被消除的参数作为关键字。然后，该程序从 *list* 中删除该参数。*eliminate* 程序的运行时间是 O(*z*log*N*)。

以下是第一个“改进的拓扑分类”进程的分类/消除循环的伪代码：

```
while parm_list is not empty:
    parm_list.sort (by_n_deps_and_order)
    while parm_list is not empty and
n_dependencies_dict[parm_list[0]]==0;
    eliminate(parm_list, 0)
    parm_list.sort(by_n_deps_and_order)
if parm_list is not empty and n_dependencies_dict[parm_list[0]]>0;
 delete parm_list[0]
 <record a circularity error and continue>
```

该进程首先利用函数 *parm_list.sort(by_n_deps_and_order)* 执行 *param_list* 的初始分类，该函数根据上述的分类准则 *by_n_deps_and_order* 对 *param_list* 中的参数进行排序。然后该进程执行 *eliminate* 程序，随后是对 *param_list 进行*另一分类，直到 *param_list* 为空。该进程进行检查以确保 *param_list* 中的第一个参数（*index*=0）的相关性数字为零。如果不为零，则该进程移除该参数，记录圆度误差并继续。分类需要 O($N\log N$) 时间并且循环范围为 N，从而估测循环的总运行时间是 O($N^2\log N$)。

第二个“改进的拓扑分类”进程利用了相关图稀疏情形的优点，使得 $z<<N$。在一个初始分类之后，该进程可以保持与其它任何参数不相关的参数列表 *candidates()*的分类。这减少了预期的运行时间，如下所述。

以下是第二个“修正的拓扑分类”进程的伪代码：

```
parm_list.sort(by_n_deps_and_order)
while parm_list is not empty:
 #部分 1
 candidates=[]
 for p in parm_list:
 if n_dependencies_dict[p]==0:
    candidates.append(p)
 #部分 2
 while candidates is not empty and
n_dependencies_dict[candidates[0]]==0:
  this_parm=candidates[0]
  eliminate(candidates,0)
  idx=parm_list.index(this_parm)
  delete parm_list[idx]
  tmp=get_new(this_parm)
  candidates=merge(candidates, tmp)
 #部分 3
 if parm_list is not empty:
   parm_list.sort(by_n_deps_and_order)
```

if n_dependencies_dict[parm_list[0]]>0:

delete parm_list[0]

<record a circularity error and continue>

首先，该进程利用函数 *parm_list.sort(by_n_deps_and_order)* 对 *param_list* 执行初始分类，该函数根据上述分类准则 *by_n_deps_and_order* 对 *param_list* 中的参数进行排序。然后该进程执行具有三个部分（标记为“#部分 1”、“#部分 2”和“#部分 3”）的循环。

在部分 1 中，该进程构建 *candidates* 列表，*candidates* 列表仅包含具有零相关性的参数。该进程浏览 *parm_list* 中的所有参数，将所述参数添加到 *candidates* 并保持它们的相对顺序。

在部分 2 中，该进程执行循环，在所述循环中消除 *candidates* 的参数并将新的参数并入到 *candidates* 中。将 *candidates* 中保存为 *this_parm* 的第一参数从 *candidates* 中消除并将其从 *param_list* 中删除。函数 *get_new(this_parm)*返回参数名称列表，所述参数是新近消除的 *this_parm* 的 *depenters_dict* 的组员，并具有所剩下的零相关性。然后根据 *by_n_deps_and_order* 对这些参数分类（确保根据它们各自的序数进行排序）并将这些参数并入到 *candidates* 中，其中这些参数表示它们的最后相关性已被移除的参数。因此，*candidates* 列表保留根据序数分类的零相关性参数的列表。

如果存在例如由于两个参数相互进行定义而引起的“圆度误差”，则仅输入部分 3。在此情形下，进程再次对 *parm_list* 进行分类，并且如果 *parm_list* 中的第一参数具有非零相关性，则将其删除并且以部分 1 重复循环。

假设没有圆度误差，则仅在开始时对 *N*-参数列表 *parm_list* 进行分类，结果排序时间为 O(*N*log*N*)。此后，仅对新近产生的零相关性参数的较小列表进行分类，该列表是由于消除 *candidates* 列表顶端的参数而产生的。该列表的大小小于 *z*（平均来说），结果排序时间为 O(*z*log*z*)，合并时间为 O(*z*)。所以循环的一次重复时间是 O(*z*log*z*)，总时间是 O(*Nz*log*z*+*N*log*N*)。对于 *z* 不随着 *N* 增加而增大的情形，此时间实际上是 O(*N*log*N*)。

分类实例 2

在另一个实例中，参数分类进程（例如第一个或第二个“改进的拓扑分类”进程）确定图形 2500 的运行时间参数的初始列表，其中如图 25A 所示，图形 2500 具有图形组件 2502、2504 和 2506。图形 2500 还具有与输入数据集 2510 的输出端口 2508 和输出数据集 2514 的输入端口 2512 相关的运行时间参数。在此实例中，参数具有“组件内”和“组件间”相关性。即，组件的参数与同一组件中的参数和不同组件中的参数相关。在此实例中，由于组件之间的流使得与一些参数相关的元数据进行传送，从而产生组件间相关性。

在图 25A 中，由从第一参数或端口到第二参数或端口的虚线箭头表示相关性。指向端口的箭头表示链接参数值从该端口传送至下游端口。从端口指出的箭头表示值从上游端口传送至链接参数。从第一参数指向第二参数的箭头表示第二参数的值与第一参数的值相关（例如引用）。

图 25B 示出相关图 2550，其表示基于图形 2500 在参数 p0、p1、p2、p4、p5 和 p6 之间的排序限制条件。图 25C 示出相关图 2552，其表示基于图形 2500 在参数 p3、p7、p8 和 p9 之间的排序限制条件。

根据图形 2500 中元素的放置顺序，该参数分类进程为各图形单元的十个参数 p0、p2，...，p9 中的每一个分配序数。在图 25A 中，添加到图形 2500 的第一个图形单元（例如通过用户使用 GDE 102 添加）是具有参数 p0、p1 和 p2 的组件 2502。所添加的第二个单元是具有参数 p3、p4 和 p5 的组件 2506。所添加的第三个单元是具有参数 p6 的数据集 2510。所添加的第四个单元是具有参数 p7 的数据集 2514。所添加的最后一个单元是不具有运行时间参数的数据集 2516。下面的表格列出由所分配的序数定义的初始排序中的参数。

参数	序数	相关性
p0	0	
p1	1	p0, p6
p2	2	p6
p3	3	p8

p4	4	p1
p5	5	p1
p6	6	
p7	7	p3
p8	8	
p9	9	p8

在各处理阶段的 *param_list* 和 *result_list* 中的参数的下列列表对应于上述的第一个“改进的拓扑分类”进程。示出在每一阶段根据分类准则 *by_n_deps_and_order* 对 *param_list* 进行的分类。

param_list	*result_list*
p0 p6 p8 p2 p3 p4 p5 p7 p9 p1	空
p6 p8 p1 p2 p3 p4 p5 p7 p9	p0
p1 p2 p8 p3 p4 p5 p7 p9	p0 p6
p2 p4 p5 p8 p3 p7 p9	p0 p6 p1
p4 p5 p8 p3 p7 p9	p0 p6 p1 p2
p5 p8 p3 p7 p9	p0 p6 p1 p2 p4
p8 p3 p7 p9	p0 p6 p1 p2 p4 p5
p3 p9 p7	p0 p6 p1 p2 p4 p5 p8
p7 p9	p0 p6 p1 p2 p4 p5 p8 p3
p9	p0 p6 p1 p2 p4 p5 p8 p3 p7
空	p0 p6 p1 p2 p4 p5 p8 p3 p7 p9

在各处理阶段的 *candidates* 和 *result_list* 中的参数的下列列表对应于上述第二个“改进的拓扑分类”进程。由于在每个阶段参数保持相同的顺序，所以在各阶段之间不必对 *candidates* 进行分类。

candidates	*result_list*
p0 p6 p8	空
p6 p8	p0
p1 p2 p8	p0 p6
p2 p4 p5 p8	p0 p6 p1
p4 p5 p8	p0 p6 p1 p2

p5 p8	p0 p6 p1 p2 p4
p8	p0 p6 p1 p2 p4 p5
p3 p9	p0 p6 p1 p2 p4 p5 p8
p7 p9	p0 p6 p1 p2 p4 p5 p8 p3
p9	p0 p6 p1 p2 p4 p5 p8 p3 p7
空	p0 p6 p1 p2 p4 p5 p8 p3 p7 p9

因此，参照图 26，“改进的拓扑分类”进程 2600 视为输入预期的第一排序 2602 和参数的排序限制条件 2604（例如相关图 2550 和 2552），其中在预期的第一排序中提示用户运行时间参数的值。进程 2600 根据预期的第一排序 2602 提供满足排序限制条件的参数组的新排序 2606。

典型用法

通常，用户坐在 Web 界面 108 的前面并在库 104 中寻找该用户想要运行的应用图形。通过浏览与该应用图形相关的所有对象，Web 界面 108 产生网页形式，其允许用户指定应用的运行时间参数值。一旦指定了所有的运行时间参数，则将应用与参数设定的组合集合为作业，该作业被调度为由执行程序 110 执行。当到达运行该作业的时间时，执行程序 110 以公知方式排队在并行操作系统 106 下执行的应用。并行操作系统 106 收集跟踪信息和作业状态，并在库 104 中存储该信息，从而使用户和管理员可以跟踪作业的进展和性能。

实例

图 14 是表示不具有运行时间参数的汇总应用的图形 1400 的示意图。该图形计算每一种帐户（account）的数目，并将结果写入输出文件。由创建图形的开发者确定此应用的每一个方面：输入文件组件 1402 的名称、输入数据格式、HashRollup 组件 1404 中用于汇总数据的关键字和转换规则、输出格式和输出文件组件 1406 的名称。如所定义的，用户只能精确执行此图形。

图 15 是表示图 14 的汇总应用的运行时间参数化版本的图形 1500 的示意图。该应用的数据流图结构与无运行时间参数化版本非常类似，但该应用更为灵活。终端用户通过运行时间参数可以指定所提取的输入数据集 1502 的名称（被保存的对象，从其可以获得输入文件名和格式），

指定 HashRollup 组件 1504 的汇总关键字和汇总规则，以及指定输出文件组件 1506 的名称。

图 16 是表示图 15 的示例性应用的运行时间参数网格 1600 的图形对话框的一个实施例图。该图是图 2 所示的参数网格版本的填充。需要注意，如上所述，利用 prompt_for 伪函数定义了多个默认参数，从而需要用户通过 Web 界面 108 进行输入。虽然此图形的外观与无运行时间参数化应用图形稍有不同，但一个或多个参数网格（或其它适合的控件）使得开发者能够完全跟踪控制图形执行的所有参数。

图 17A 是表示由 Web 界面 108 根据图 16 的参数网格 1600 中的信息产生的表格 1700 的图形对话框的一个实施例图。在此实例中，表格 1700 给出用户输入的四个运行时间参数：输入数据集库路径 1702、汇总关键字 1704、汇总规则 1706 和输出路径 1708。图 17B 是用户用参数值填充图 17A 的表格 1700 的示意图。利用直接条目和/或与运行时间参数 1702-1708 相关的编辑或浏览器控制按钮，用户提供用于执行相关图形的相应参数值 1710-1716。

图 18 是表示运行时间参数化汇总和组合应用的图形 1800 的示意图。图 19 是表示关于图 18 的示例性应用的运行时间参数网格 1900 的图形对话框的一个实施例图。此处，对该应用的一些方面进行了参数化，但包括组合关键字和输入数据集的大部分方面保持固定。图 20 是表示由 Web 界面 108 根据图 19 的参数网格 1900 中的信息产生的表格 2000 的图形对话框的一个实施例图。需要注意，因为在显示顶层表格时输入到汇总组件的输入类型是已知的，因此能够以 in-place 提示汇总规则 2002。

图 21 是表示运行时间参数化汇总-组合-分类应用的图形 2100 的示意图。虽然类似于图 18 的实例，但在图形 2100 中添加了条件分类组件 2102。图 22 是表示关于图 21 所示的示例性应用的运行时间参数网格 2200 的图形对话框的一个实施例图。sort_key 运行时间参数 2202 仅在用户表示想要分类时才进行提示。为了获得此效果，开发者将 prompt_for 伪函数放入到 if 条件测试内，所述 if 条件测试用于 sort_key 运行时间参数 2202 的默认值 2204。该 if 条件测试引用第二个运行时间参数 do_sort 2206。定义 do_sort 参数 2206 的默认值字段 2208 和描述字段 2210，以产生单选

（radio）提示，该提示询问用户对于文本提示“Should the data be sorted?”的回答是真/假或是/否。如果 do_sort 参数 2206 的值是“真”，则将包括分类组件 2102，作为运行时的图形部分。否则，根据分类组件 2102 的特定条件的说明，将从图形中完全移除分类组件 2102 或用流替换。

程式实施方式

虽然 GDE 102 促进了参数化图形的构建，但有时存在有可以提供基于表格界面的无图形程序。利用应用级 PL 和库 104，可以参数化任意的 shell script 指令。例如，可以将应用的描述写入具有类似以下结构的文件中：

application ***AppName***(
description (“***One-line Description***”),
comment (“***Longer description***”),

parameter ***ParmName1***(
string, kind (keyword), required,
description (“***Short prompt for top-level form***”),
comment (“***Longer prompt for out-of-line form***”),
default ($ {prompt_for…})
),
parameter ***ParmName2***(
type, kind (derived),
default (***PL-expression***)
),
…***more parameters***…
script(=”***scriptname.ksh***”)
)

通用计算机实施方式

本发明可以在硬件或软件中实现，或者在二者的组合（例如可编程的逻辑阵列）中实现。除非特别声明，否则作为本发明的部分算法不与任何特定的计算机或其它装置内在相关。特别地，根据本文的教导，可

以使用各种通用机器及写入其中的程序来执行所要求的方法步骤，或为了更方便实施，还可以构建更多专用装置来执行该方法步骤。然而，本发明优选以在一个或多个可编程的计算机系统上执行的一个或多个计算机程序来实现，每一个可编程的计算机系统包括至少一个处理器、至少一个数据存储系统（包括易失性存储器和非易失性存储器和/或存储元件）、至少一个输入设备或端口以及至少一个输出设备或端口。在处理器上执行程序代码以实现本文描述的功能。

所述程序中的每一个均可以用任何预期的计算机语言（包括机器语言、汇编语言或高级程序语言、逻辑语言或面向对象的编程语言）来实现，以便与计算机系统进行通信。在任何情形下，该语言都可以是编译语言或解释语言。

优选地，将所述计算机程序中的每一个存储在可由通用或专用可编程计算机读取的存储介质或设备（例如固态介质、磁介质或光学介质）上，在计算机系统读取存储介质或设备以执行本文描述的程序时，用于配置并操作计算机。本发明的系统还可以实现为可由计算机读取的存储介质，该可读取的存储介质配置有计算机程序，如此配置的存储介质使得计算机系统能够以特定和预定的方式操作，以实现本文描述的功能。

此处描述了本发明的多个实施例。然而，需要理解的是可以对本发明做出各种改进而不会脱离其精神和范围。例如，可以以不同的顺序执行上述多个功能步骤，而不会对总体程序产生实质影响。例如，可以以相反的顺序执行图 4 中的步骤 402 和 412。因此，还有其它实施例也落在权利要求的保护范围内。

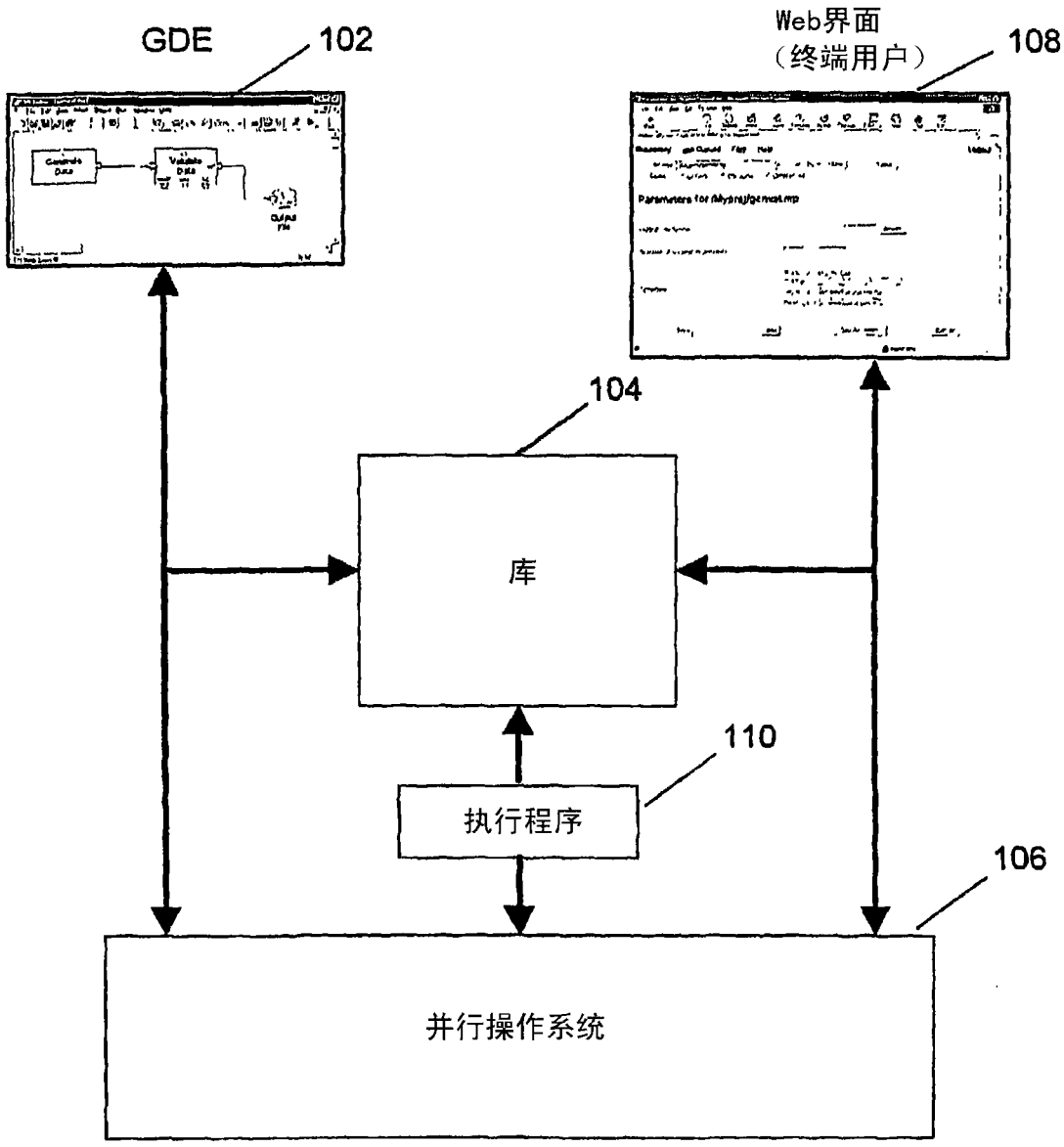


图1A

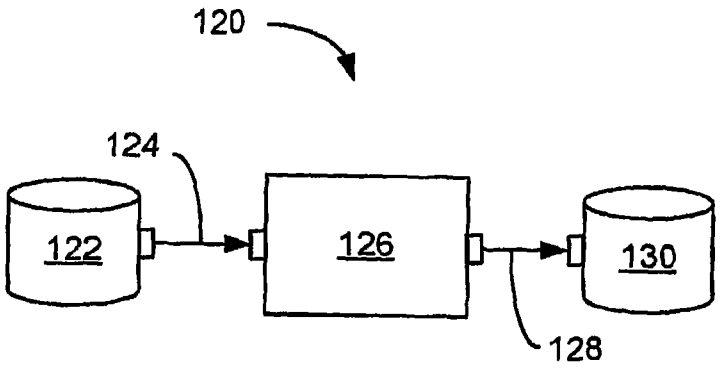


图1B

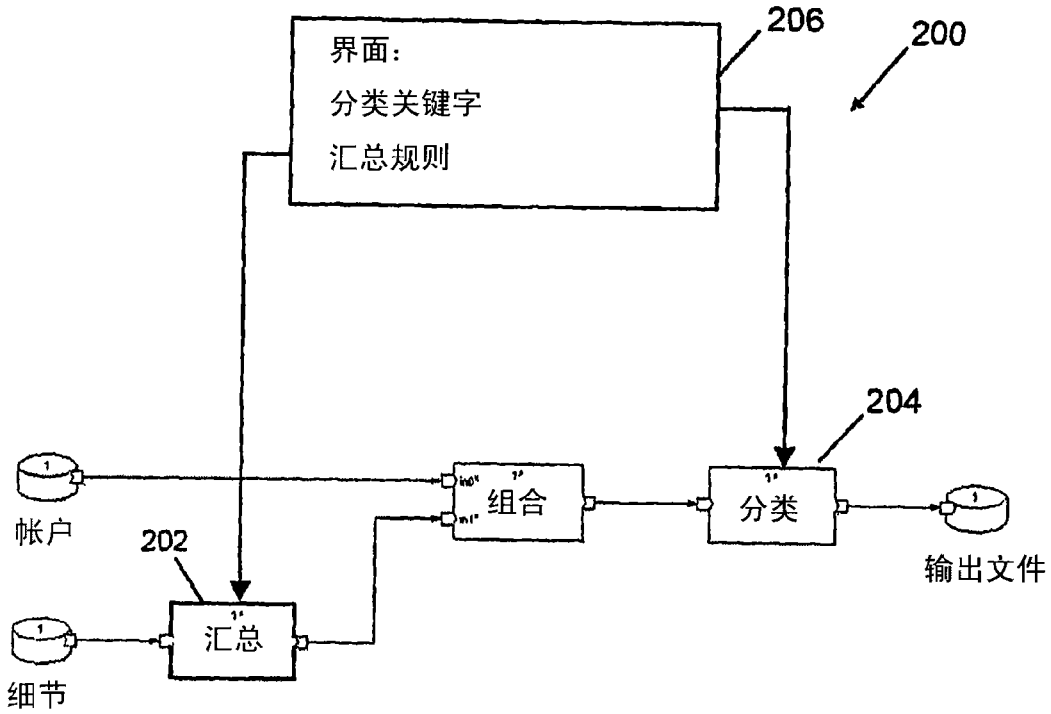

界面：
分类关键字
汇总规则
206
200
204
帐户
202
组合
分类
输出文件
汇总
细节

图2

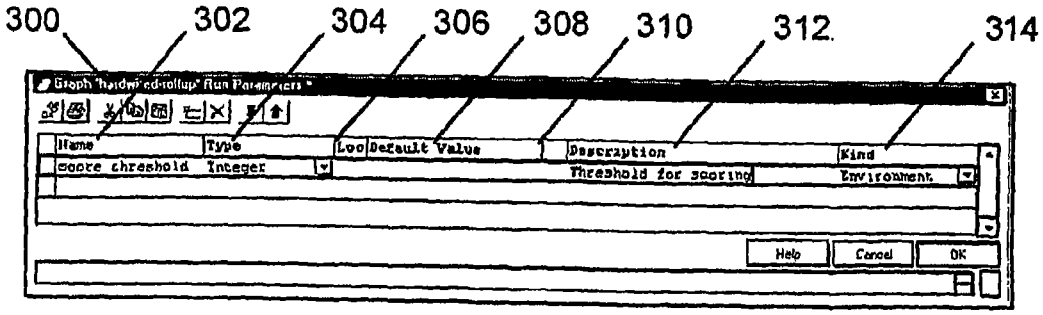

300
302
304
306
308
310
312
314
Name
Type
Loc
Default Value
Description
Kind
score threshold
Integer
Threshold for scoring
Environment
Help
Cancel
OK

图3

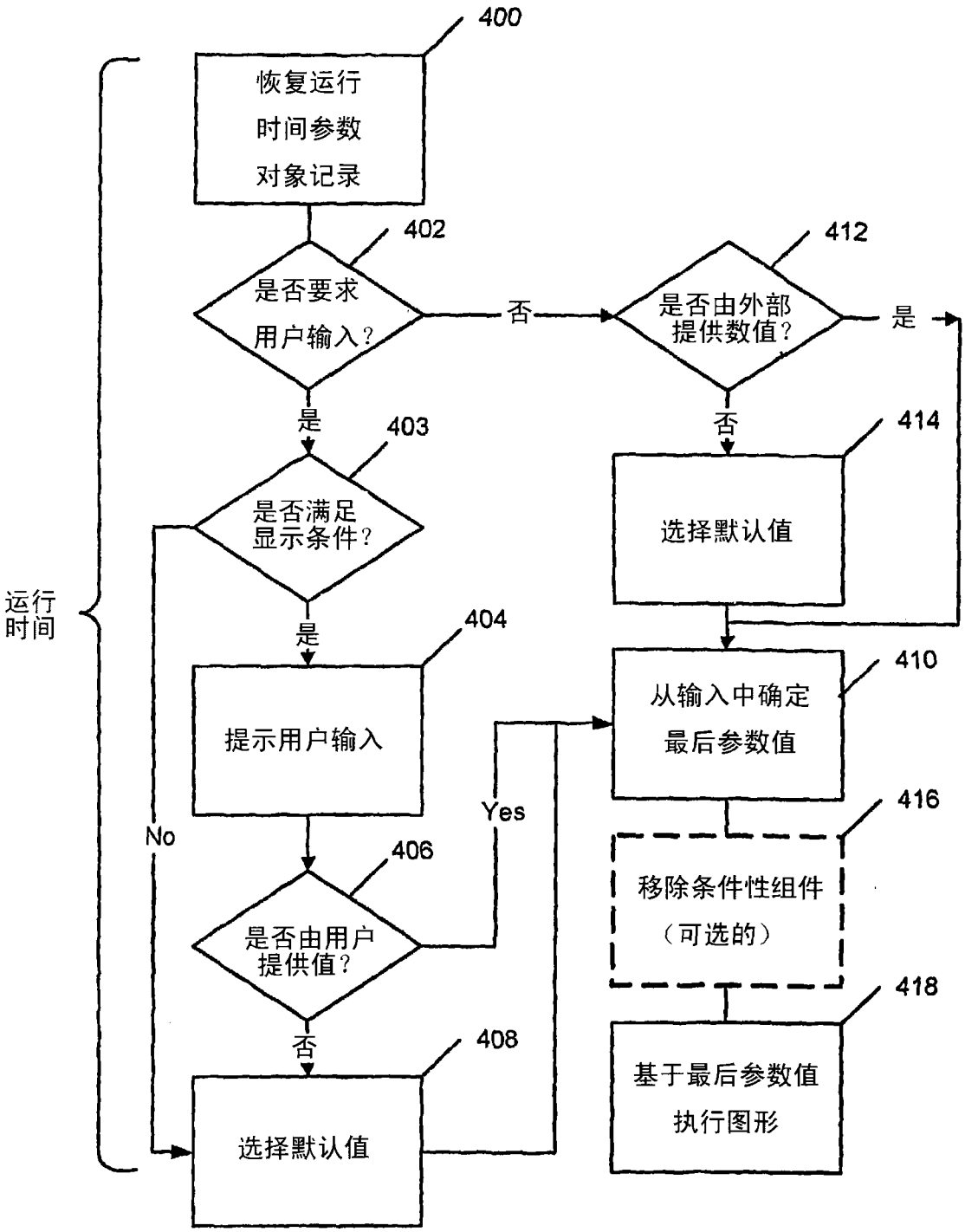

400
恢复运行
时间参数
对象记录
402
是否要求
用户输入?
否
412
是否由外部
提供数值?
是
是
403
是否满足
显示条件?
否
414
选择默认值
运行
时间
是
404
提示用户输入
410
从输入中确定
最后参数值
416
移除条件性组件
(可选的)
Yes
No
406
是否由用户
提供值?
418
基于最后参数值
执行图形
否
408
选择默认值

图4

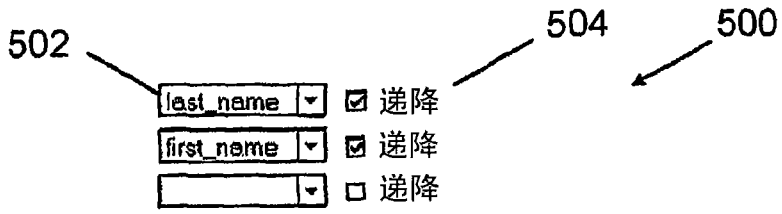

502
504
500
last_name
递降
first_name
递降
递降

图 5

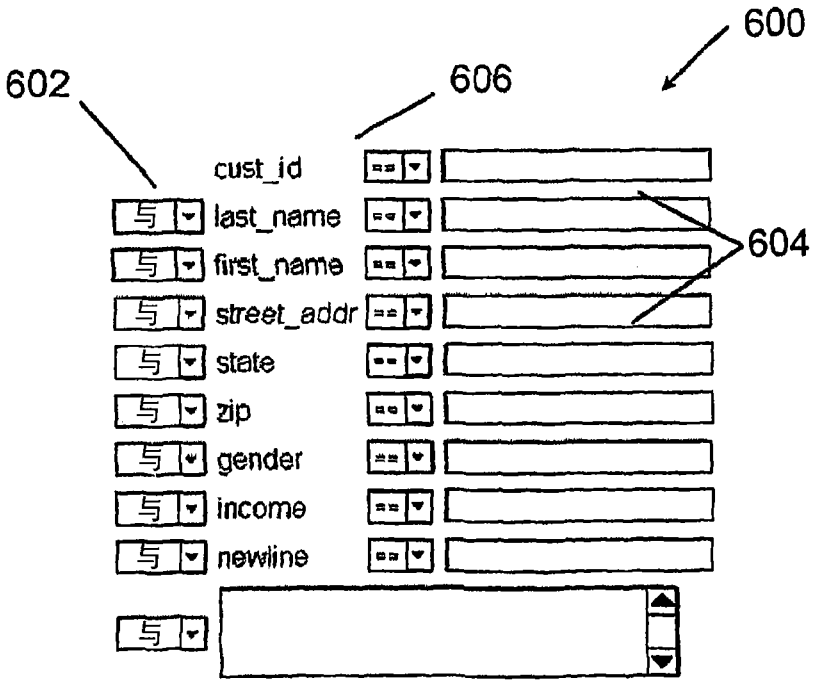

600
602
606
604
cust_id
与
last_name
first_name
street_addr
state
zip
gender
income
newline

图6

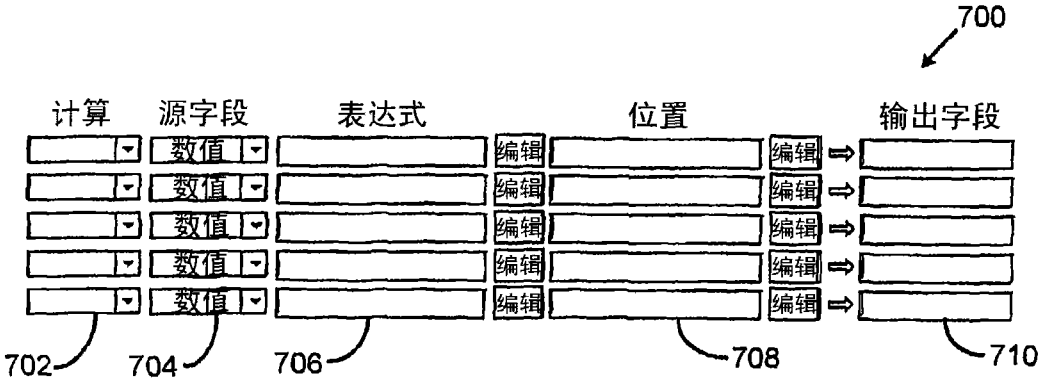

700
计算
源字段
表达式
位置
输出字段
数值
编辑
702
704
706
708
710

图 7

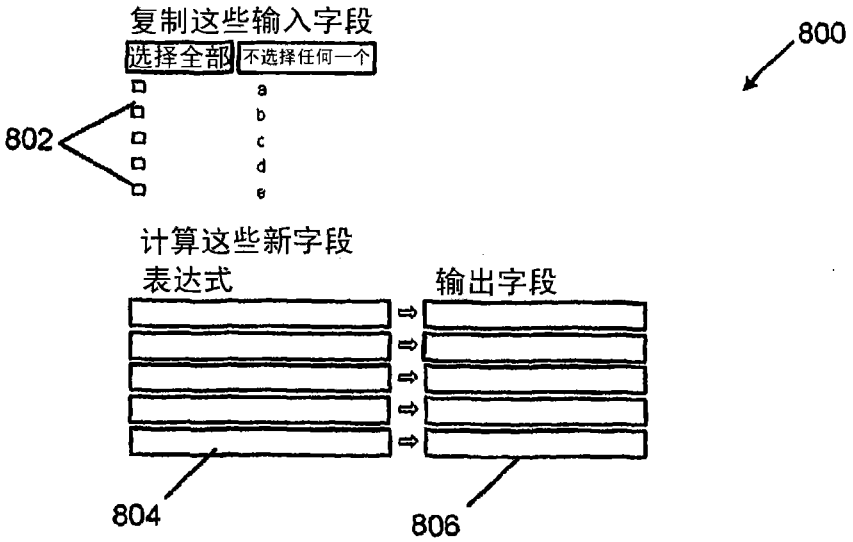

复制这些输入字段
选择全部
不选择任何一个
800
802
a
b
c
d
e
计算这些新字段
表达式
输出字段
804
806

图 8

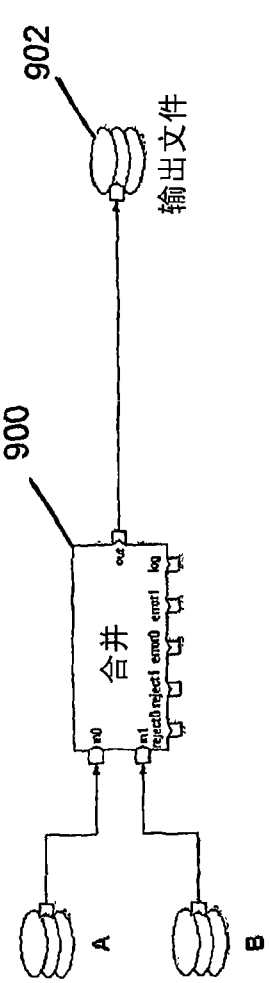


图9A
组合A和B

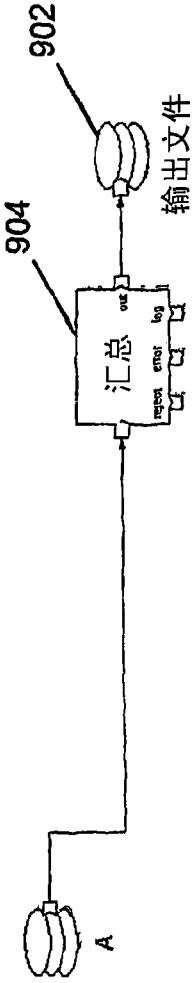


图9B
汇集A

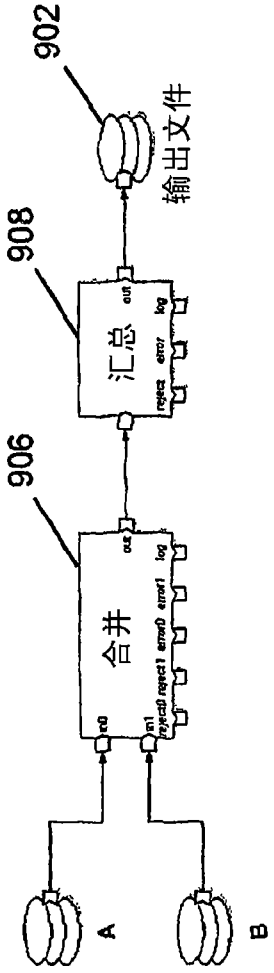


图9C
组合A和B，然后汇集

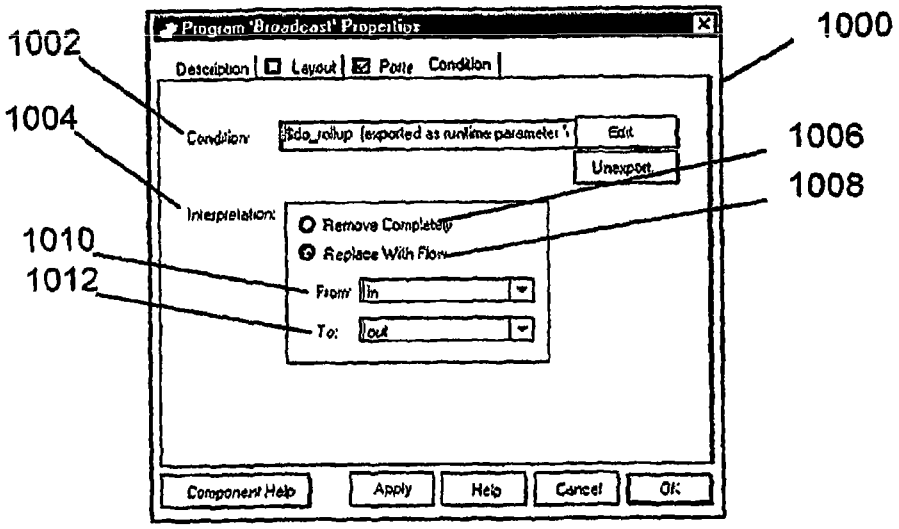

1000
1002
1004
1006
1008
1010
1012
Program 'Broadcast' Properties
Description
Layout
Post
Condition
Condition:
Edit
Unexport
Interpretation:
Remove Completely
Replace With Flow
From:
In
To:
Out
Component Help
Apply
Help
Cancel
OK

图 10

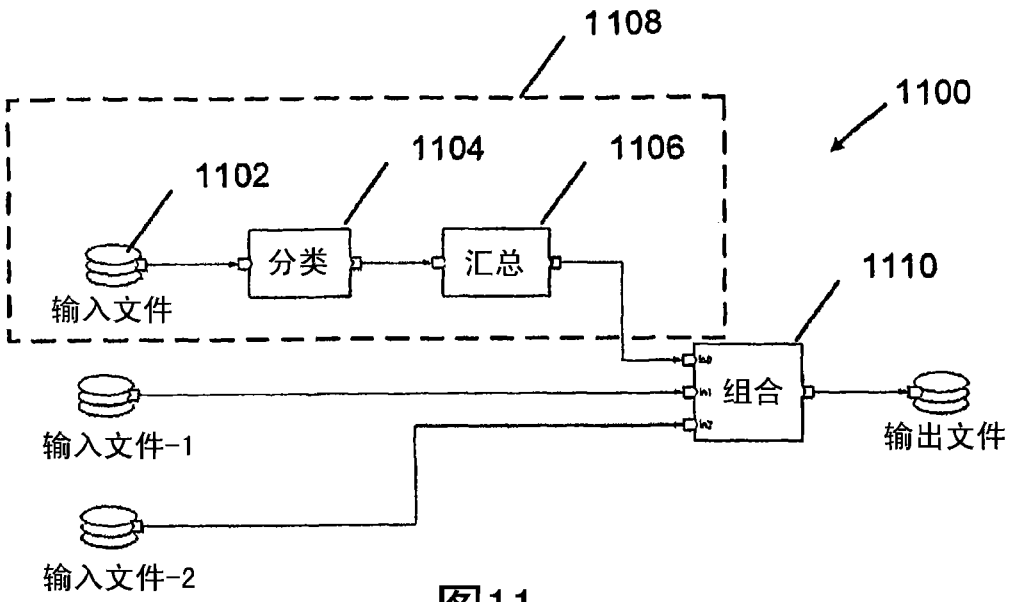


图11

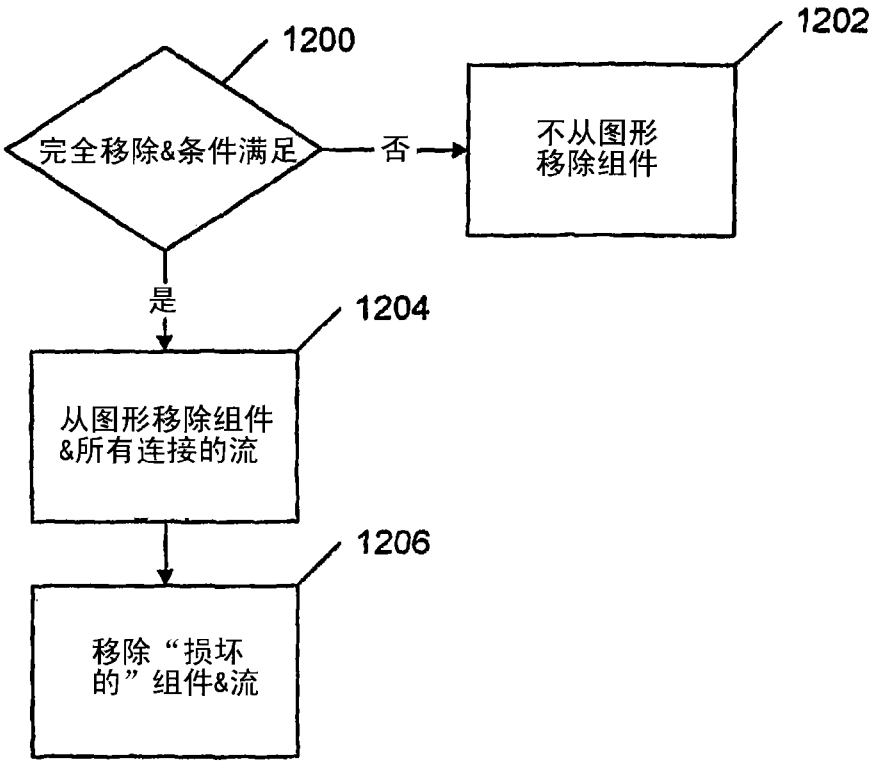

1200
完全移除&条件满足
否
1202
不从图形
移除组件
是
1204
从图形移除组件
&所有连接的流
1206
移除“损坏
的”组件&流

图12

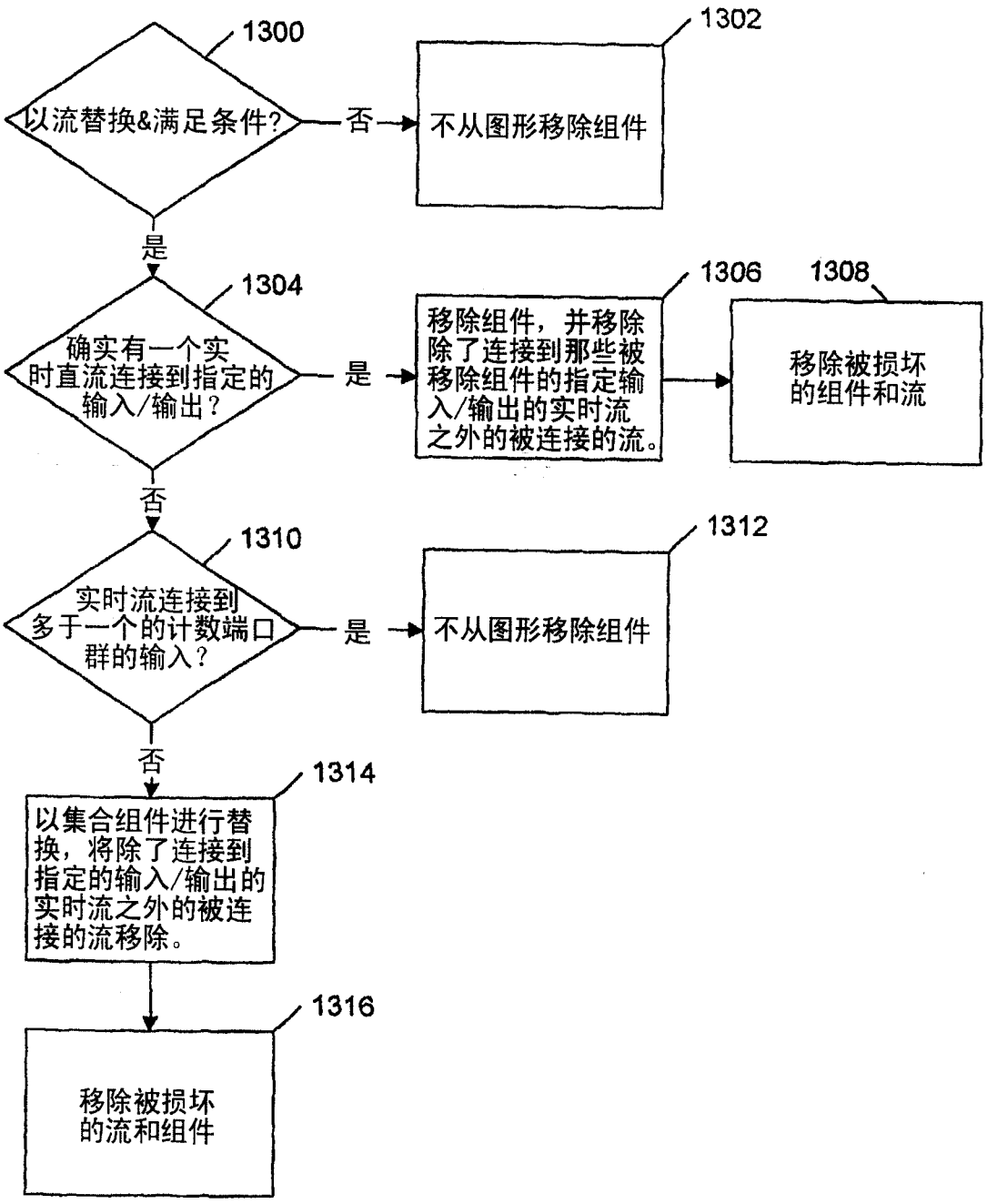

1300
以流替换&满足条件?
否
1302
不从图形移除组件
是
1304
确实有一个实时直流连接到指定的输入/输出?
是
1306
移除组件，并移除除了连接到那些被移除组件的指定输入/输出的实时流之外的被连接的流。
1308
移除被损坏的组件和流
否
1310
实时流连接到多于一个的计数端口群的输入?
是
1312
不从图形移除组件
否
1314
以集合组件进行替换，将除了连接到指定的输入/输出的实时流之外的被连接的流移除。
1316
移除被损坏的流和组件

图13

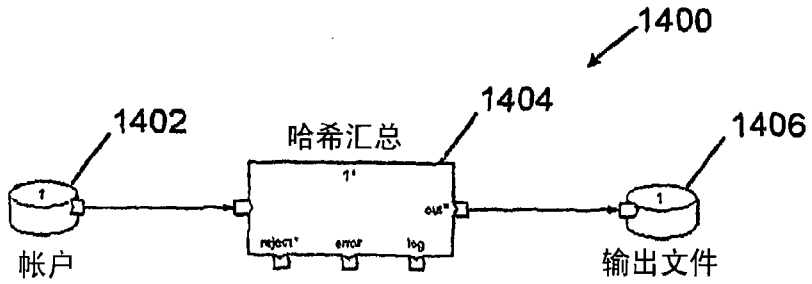

1400
1402
哈希汇总
1404
1406
帐户
输出文件

图14

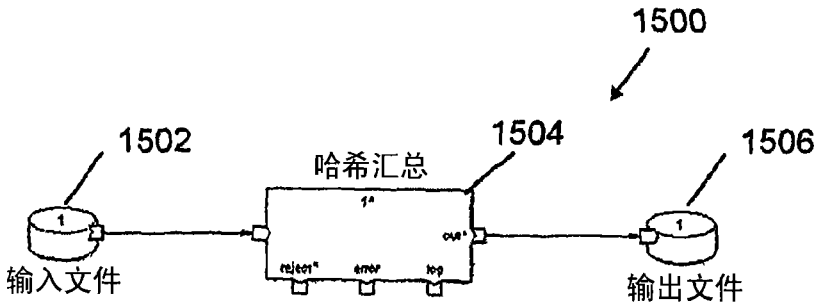

1500
1502
哈希汇总
1504
1506
输入文件
输出文件

图15

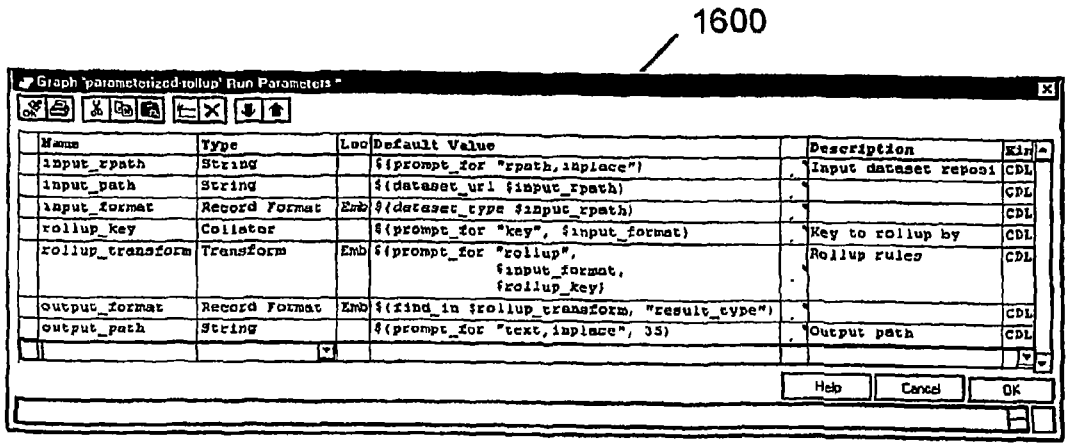

1600
Graph 'parameterized-rollup' Run Parameters
Name
Type
Loc
Default Value
Description
input_xpath
String
${prompt_for "xpath,inplace"}
Input dataset reposi
input_path
String
${dataset_url $input_xpath}
input_format
Record Format
Emb
${dataset_type $input_xpath}
rollup_key
Collator
${prompt_for "key", $input_format}
Key to rollup by
rollup_transform
Transform
Emb
${prompt_for "rollup", $input_format, $rollup_key}
Rollup rules
output_format
Record Format
Emb
${find_in $rollup_transform, "result_type"}
output_path
String
${prompt_for "text,inplace", 35}
Output path
Help
Cancel
OK

图16

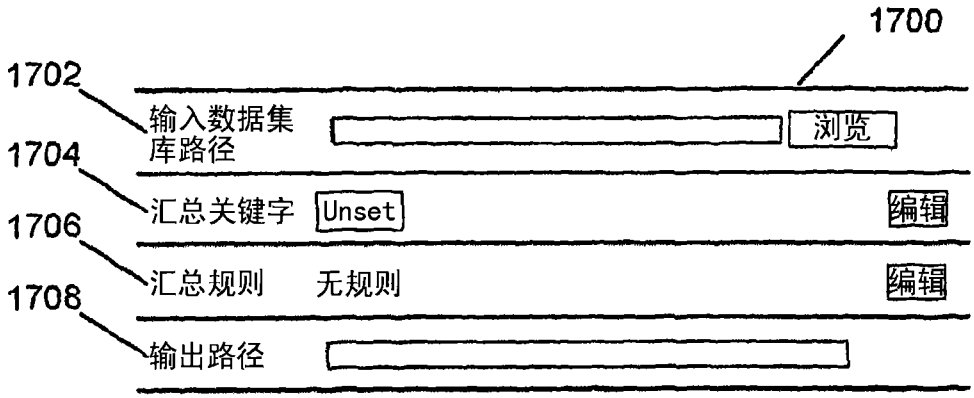

1700
1702
输入数据集
库路径
浏览
1704
汇总关键字
Unset
编辑
1706
汇总规则
无规则
编辑
1708
输出路径

图17A

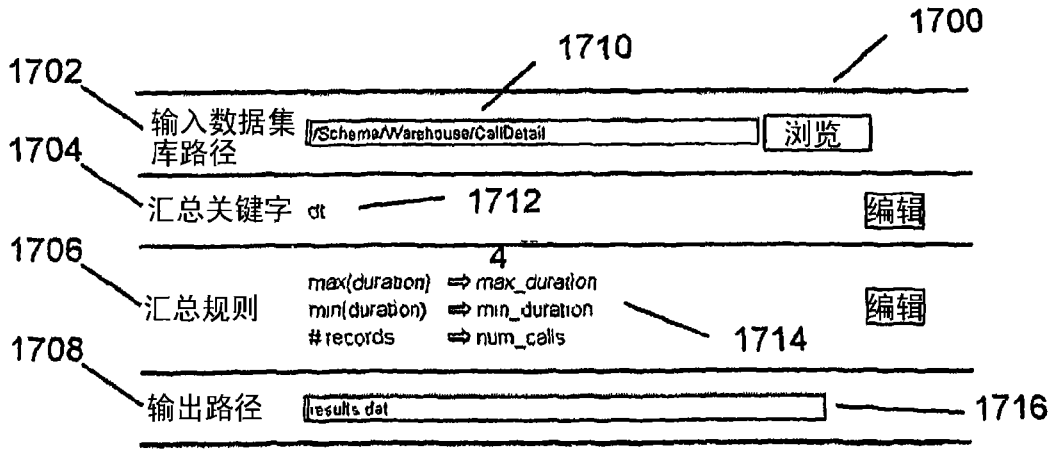

1700
1710
1702
输入数据集
库路径
/Schema/Warehouse/CallDetail
浏览
1704
汇总关键字
dt
1712
编辑
1706
4
汇总规则
max(duration) ⇨ max_duration
min(duration) ⇨ min_duration
records ⇨ num_calls
1714
编辑
1708
输出路径
results.dat
1716

图17B

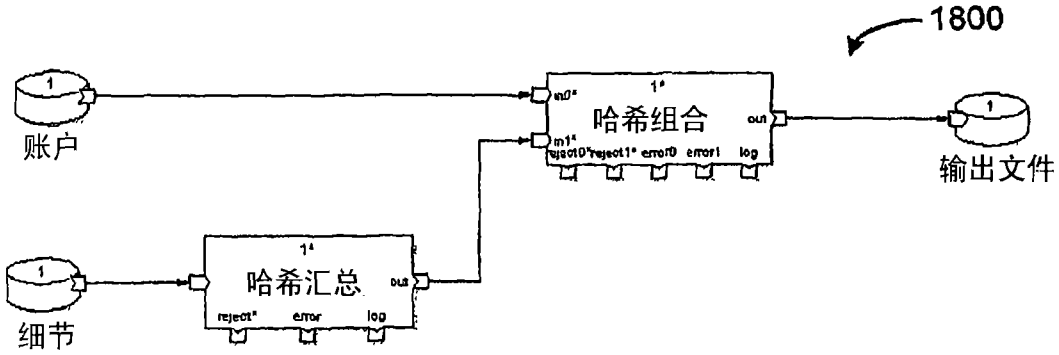

1800
账户
哈希组合
输出文件
哈希汇总
细节
reject*
error
log

图18

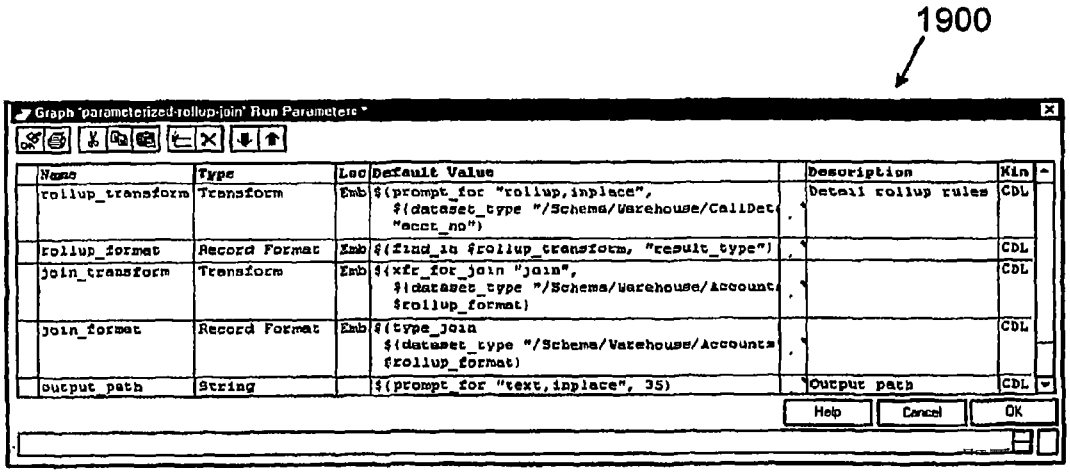

1900
Graph 'parameterized-rollup-join' Run Parameters
Name
Type
Loc
Default Value
Description
Kin
rollup_transform
Transform
Emb
Detail rollup rules
rollup_format
Record Format
Emb
join_transform
Transform
Emb
join_format
Record Format
Emb
output_path
String
Output path
CDL
Help
Cancel
OK

图19

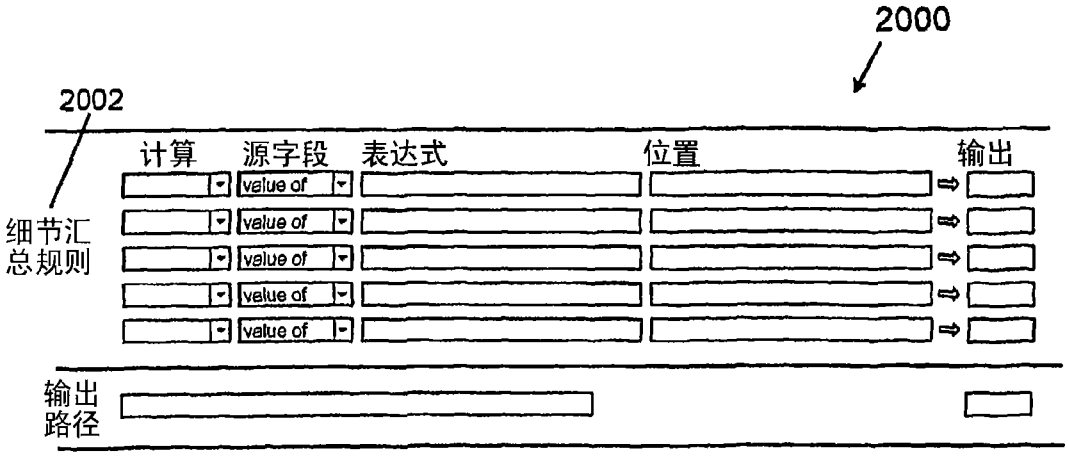

2000
2002
计算
源字段
表达式
位置
输出
value of
细节汇总规则
输出路径

图20

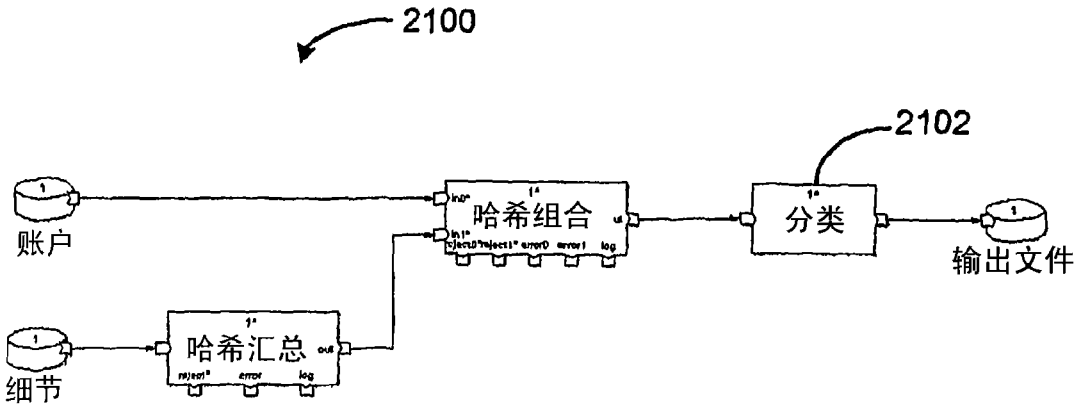

2100
2102
账户
哈希组合
分类
输出文件
细节
哈希汇总

图21

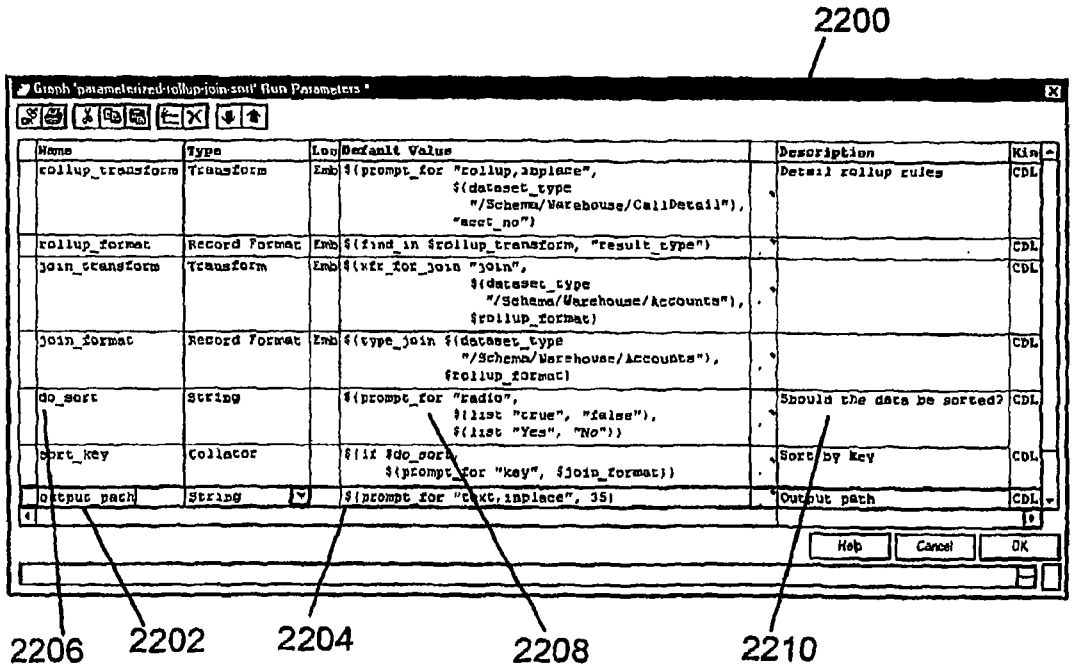

2200
Graph 'parameterized-rollup-join-sort' Run Parameters
Name
Type
Default Value
Description
rollup_transform
Transform
Detail rollup rules
rollup_format
Record Format
join_transform
Transform
join_format
Record Format
do_sort
String
Should the data be sorted?
sort_key
Collator
Sort by key
output_path
String
Output path
Help
Cancel
OK
2206
2202
2204
2208
2210

图22

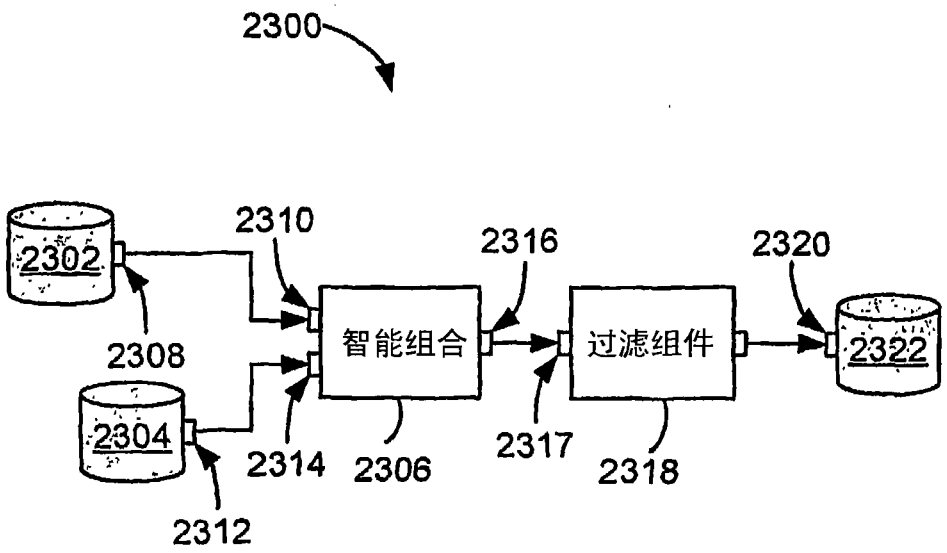

2300
2302
2310
2316
2320
2308
智能组合
过滤组件
2322
2304
2314
2306
2317
2318
2312

图23A

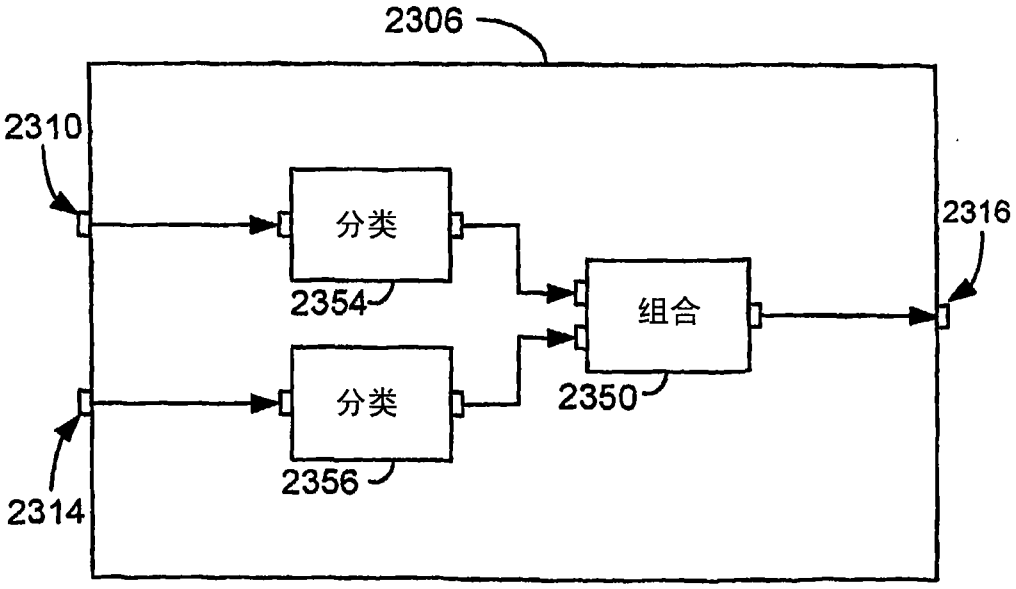

2306
2310
分类
2316
2354
组合
分类
2350
2356
2314

图23B

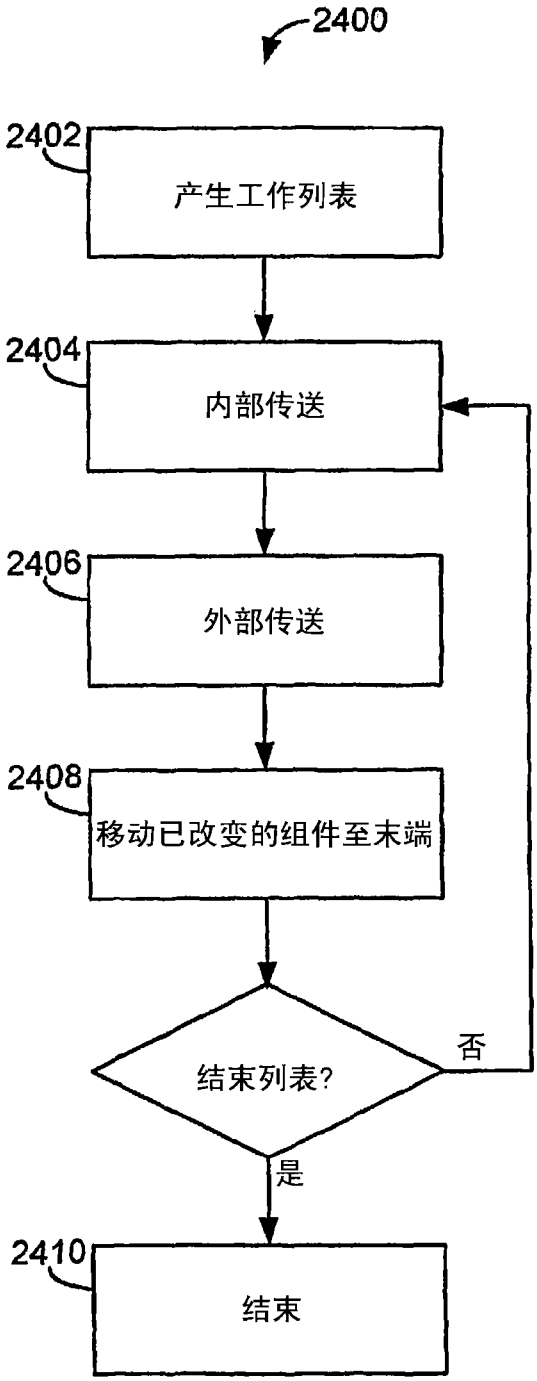

2400
2402
产生工作列表
2404
内部传送
2406
外部传送
2408
移动已改变的组件至末端
结束列表?
否
是
2410
结束

图24

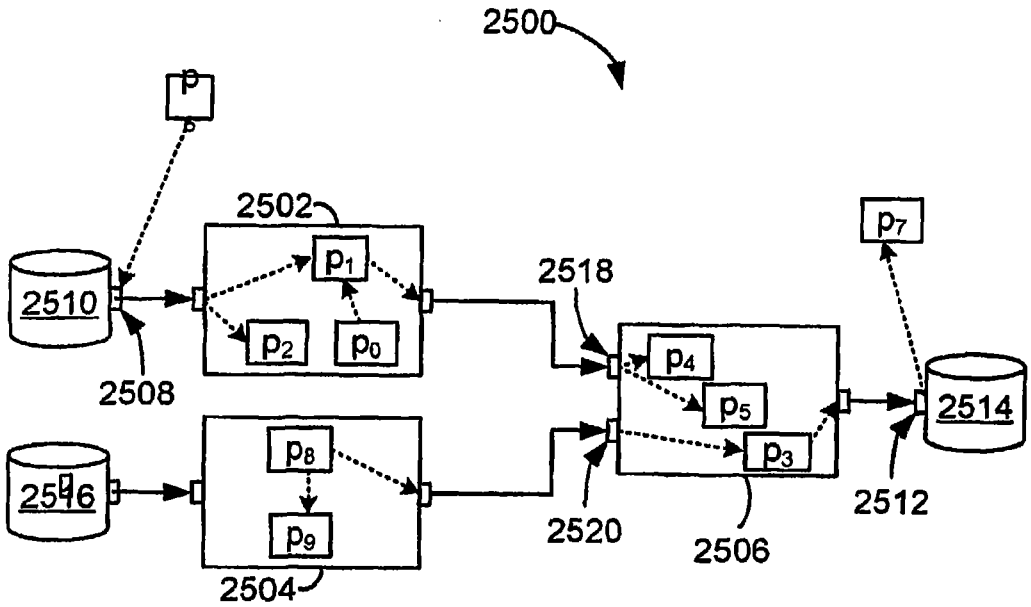


图25A

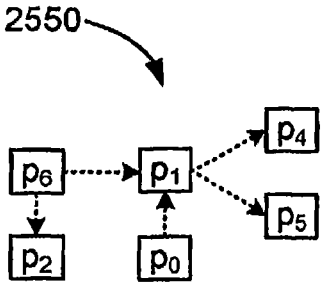


图25B

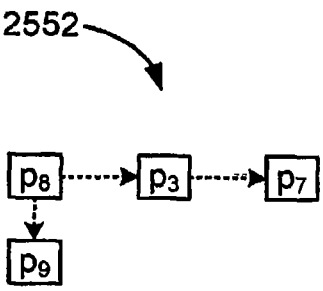


图25C

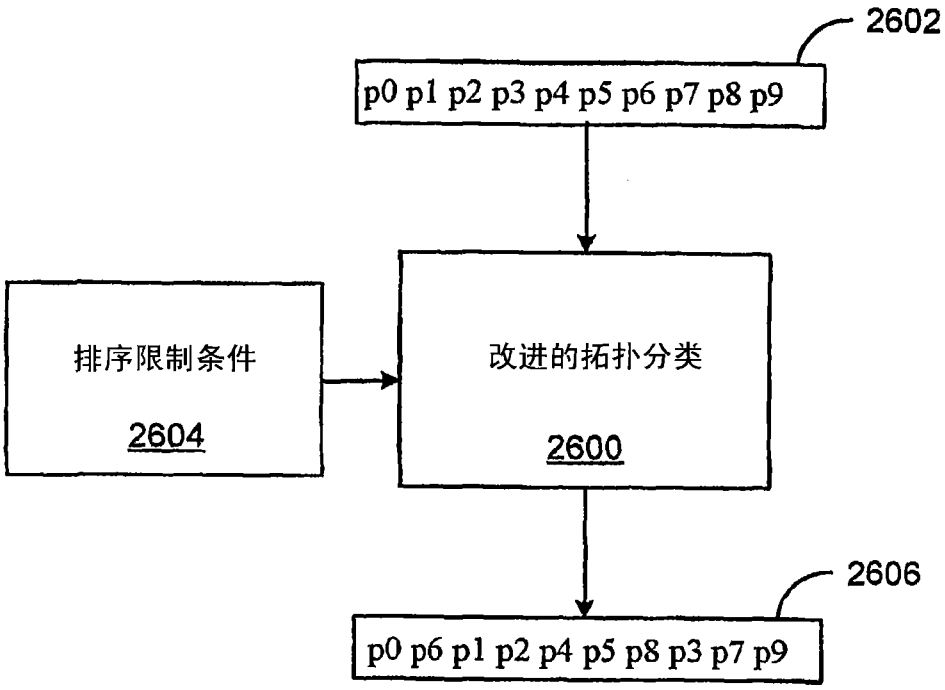

2602
p0 p1 p2 p3 p4 p5 p6 p7 p8 p9
排序限制条件
2604
改进的拓扑分类
2600
2606
p0 p6 p1 p2 p4 p5 p8 p3 p7 p9

图26