(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 110168518 B
(45)授权公告日 2023.07.14

(21)申请号 201780080703.1
(22)申请日 2017.11.06
(65)同一申请的已公布的文献号
申请公布号 CN 110168518 A
(43)申请公布日 2019.08.23
(30)优先权数据
15/345,391 2016.11.07 US
15/705,174 2017.09.14 US
(85)PCT国际申请进入国家阶段日
2019.06.26
(86)PCT国际申请的申请数据
PCT/US2017/060232 2017.11.06
(87)PCT国际申请的公布数据
WO2018/085785 EN 2018.05.11
(73)专利权人 塔谱软件有限责任公司
地址 美国华盛顿州
(72)发明人 金俊 威尔·皮尤 艾萨克·丘嫩
(74)专利代理机构 北京安信方达知识产权代理有限公司 11262
专利代理师 周靖 杨明钊
(51)Int.Cl.
G06F 16/2453(2019.01)
G06F 16/2455(2019.01)
G06F 16/25(2019.01)
G06F 16/26(2019.01)
G06F 3/0484(2022.01)
G06F 9/451(2018.01)
G06F 3/0482(2013.01)
G06T 11/20(2006.01)
G06F 8/38(2018.01)
G06F 16/248(2019.01)
审查员 贾越

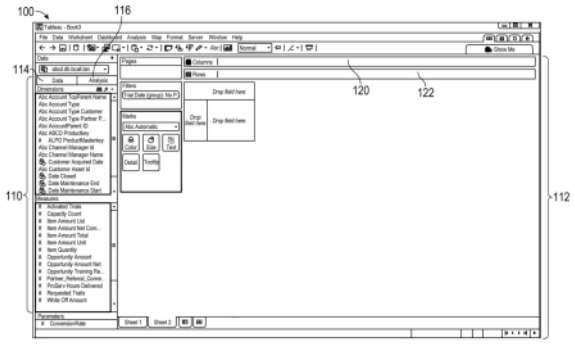
权利要求书3页 说明书34页 附图50页

(54)发明名称

准备和整理用于后续分析的数据的用户界面

(57)摘要

一种计算机系统显示包括流程窗格、工具窗格、配置文件窗格、和数据窗格的用户界面。流程窗格显示识别数据源、操作、和输出数据集的节点/链路流程图。工具窗格包括使用户能够将数据源添加到流程图的数据源选择器，并包括使用户能够将节点插入到流程图中以用于执行特定的转换操作的操作调色板。配置文件窗格显示对应于流程图中的选定节点的方案，包括关于数据字段的信息和关于数据字段的数据值的统计信息，并使用户能够通过与各个数据元素交互来修改流程图。数据窗格显示对应于流程图中的选定节点的数据行，并使用户能够通过与各个数据值交互来修改流程图。



CN 110168518 B

1.一种用于准备用于后续分析的数据的计算机系统,包括:

一个或更多个处理器;

存储器;以及

一个或更多个程序,其被存储在所述存储器中并被配置为由所述一个或更多个处理器执行,所述一个或更多个程序包括指令,所述指令用于:

显示包括数据流程窗格、工具窗格、配置文件窗格、和数据窗格的并行显示的用户界面,其中:

所述数据流程窗格显示包括多个链接节点的流程图,每个节点指定各自的操作和在执行所述各自的操作时生成的各自的中间数据集;

所述工具窗格包括显示用于定位和连接到数据源的可视性的数据源选择器,显示作为节点插入到所述流程图中的一个或多个转换操作的操作调色板、以及用于并入到所述流程图中的其他流程图的调色板;

所述配置文件窗格显示对应于所述流程图中的选定节点的方案,包括显示关于和所述选定节点对应的中间数据集的数据字段的信息和关于所述数据字段的数据值的统计信息的数据元素,其中,与所述配置文件窗格中的所述数据元素中任何一个的用户交互修改所述数据流窗格中显示的所述流程图,包括:

识别要执行的封装所述用户交互的操作;以及

更新所述选定节点或添加新节点,以在所述流程图中包括所述操作;以及

所述数据窗格显示对应于所述流程图中的选定节点的所述中间数据集中的数据行,其中与来自所显示的数据行的各个数据值的用户交互修改在所述数据流窗格中显示的所述流程图。

2.根据权利要求1所述的计算机系统,其中,关于显示在所述配置文件窗格中显示的数据字段的所述信息的所述数据元素包括第一数据字段的数据范围。

3.根据权利要求2所述的计算机系统,其中,响应于对在所述配置文件窗格中的所述第一数据字段的第一数据范围的第一用户动作:

将要执行的所述操作识别为过滤操作;以及

添加将数据过滤到所述第一数据范围的新节点到所述流程图。

4.根据权利要求2所述的计算机系统,其中,所述配置文件窗格中的将所述第一数据字段的所述数据范围映射到指定值的用户交互将执行用户所指定的映射的新节点添加到所述流程图。

5.根据权利要求1所述的计算机系统,其中,响应于与在所述数据窗格中的来自所述显示的数据行的第一数据值的第一用户交互,将数据过滤到所述第一数据值的节点被添加到所述流程图。

6.根据权利要求1所述的计算机系统,其中,响应于对在所述数据窗格中的第一数据字段的第一数据值的用户修改,对所述第一数据字段的数据值等于所述第一数据值的每行数据执行所述修改的新节点被添加到所述流程图。

7.根据权利要求1所述的计算机系统,其中,响应于对在所述数据窗格中的第一数据字段的第一用户动作,将所述第一数据字段分成两个或更多个单独的数据字段的新节点被添加到所述流程图。

8.根据权利要求1所述的计算机系统，其中，响应于在所述数据流程窗格中的将第一节点拖动到所述工具窗格的第一用户动作，新操作被添加到所述操作调色板，所述新操作对应于所述第一节点。

9.根据权利要求1所述的计算机系统，其中，所述配置文件窗格和数据窗格被配置为当在所述数据流程窗格中做出选择时异步地更新。

10.根据权利要求1所述的计算机系统，其中，所述配置文件窗格显示包括数据字段的数据值的分布的一个或更多个直方图。

11.一种非暂时性计算机可读存储介质，所述非暂时性计算机可读存储介质存储被配置为由具有一个或更多个处理器、存储器、和显示器的计算机系统执行的一个或更多个程序，所述一个或更多个程序包括用于执行以下操作的指令：

显示包括数据流程窗格、工具窗格、配置文件窗格、和数据窗格的并行显示的用户界面，其中：

所述数据流程窗格显示包括多个链接节点的流程图，每个节点指定各自的操作和在执行所述各自的操作时生成的各自的中间数据集；

所述工具窗格包括显示用于定位和连接到数据源的可视性的数据源选择器，显示作为节点插入到所述流程图中的一个或多个转换操作的操作调色板、以及用于并入到所述流程图中的其他流程图的调色板；

所述配置文件窗格显示对应于所述流程图中的选定节点的方案，包括显示关于对应于所述选定节点的所述中间数据集的数据字段的信息和关于所述数据字段的数据值的统计信息的数据元素，其中，与所述配置文件窗格中的所述数据元素中任何一个的用户交互修改所述数据流窗格中显示的所述流程图，包括：

识别要执行的封装所述用户交互的操作；以及

更新所述选定节点或添加新节点，以在所述流程图中包括所述操作；以及

所述数据窗格显示对应于所述流程图中的选定节点的所述中间数据集的数据行，其中与来自所述显示的数据行的各个数据值的用户交互修改在所述数据流窗格中显示的所述流程图。

12.根据权利要求11所述的计算机可读存储介质，其中，响应于对在所述配置文件窗格中的第一数据字段的第一数据范围的第一用户动作：

将要执行的所述操作识别为过滤操作；以及

添加将数据过滤到所述第一数据范围的新节点到所述流程图。

13.根据权利要求11所述的计算机可读存储介质，其中，所述配置文件窗格中的将第一数据字段的数据范围映射到指定值的用户交互将执行用户所指定的映射的新节点添加到所述流程图。

14.根据权利要求11所述的计算机可读存储介质，其中，响应于与来自在所述数据窗格中的所述显示的数据行的第一数据值的第一用户交互，将数据过滤到所述第一数据值的新节点被添加到所述流程图。

15.根据权利要求11所述的计算机可读存储介质，其中，响应于对在所述数据窗格中的第一数据字段的第一用户动作，将所述第一数据字段分成两个或更多个单独的数据字段的新节点被添加到所述流程图。

16.一种准备用于后续分析的数据的方法,包括:

在具有显示器、一个或更多个处理器、和存储被配置为由所述一个或更多个处理器执行的一个或更多个程序的存储器的计算机系统处:

显示包括数据流程窗格、工具窗格、配置文件窗格、和数据窗格的并行显示的用户界面,其中:

所述数据流程窗格显示包括多个链接节点的流程图,每个节点指定各自的操作和在执行所述各自的操作时生成的各自的中间数据集;

所述工具窗格包括显示用于定位和连接到数据源的可视性的数据源选择器,显示作为节点插入到所述流程图中的一个或多个转换操作的操作调色板、以及用于并入到所述流程图中的其他流程图的调色板;

所述配置文件窗格显示对应于所述流程图中的选定节点的方案,包括显示关于对应于所述选定节点的所述中间数据集的数据字段的信息和关于所述数据字段的数据值的统计信息的数据元素,其中,与所述配置文件窗格中的所述数据元素中任何一个的用户交互修改所述数据流窗格中显示的所述流程图,包括:

识别要执行的封装所述用户交互的操作;以及

更新所述选定节点或添加新节点,以在所述流程图中包括所述操作;以及

所述数据窗格显示对应于所述流程图中的选定节点的所述中间数据集的数据行,其中与来自所显示的数据行的各个数据值的用户交互修改在所述数据流窗格中显示的所述流程图。

17.根据权利要求16所述的方法,其中,响应于对在所述配置文件窗格中的第一数据字段的第一数据范围的第一用户动作:

将要执行的所述操作识别为过滤操作;以及

添加将数据过滤到所述第一数据范围的新节点到所述流程图。

18.根据权利要求16所述的方法,其中,所述配置文件窗格中的将第一数据字段的所述数据范围映射到指定值的用户交互将执行用户所指定的映射的新节点添加到所述流程图。

19.根据权利要求16所述的方法,其中,响应于与来自在所述数据窗格中的所述显示的数据行的第一数据值的第一用户交互,将数据过滤到所述第一数据值的新节点被添加到所述流程图。

20.根据权利要求16所述的方法,其中,响应于对在所述数据窗格中的第一数据字段的第一用户动作,将所述第一数据字段分成两个或更多个单独的数据字段的新节点被添加到所述流程图。

准备和整理用于后续分析的数据的用户界面

技术领域

[0001] 所公开的实现方式通常涉及数据可视化，且更具体地涉及准备和整理用于由数据可视化应用进行使用的数据的系统、方法和用户界面。

[0002] 背景

[0003] 数据可视化应用使用户能够在视觉上理解数据集，包括分布、趋势、异常值、和对做出业务决策很重要的其他因素。一些数据集是非常大的或复杂的，且包括许多数据字段。可以使用各种工具来帮助理解和分析数据，包括具有多个数据可视化的仪表板。然而，数据经常需要被操纵或改动以将其置于可以容易被数据可视化应用使用的格式中。有时各种ETL(提取/转换/加载)工具被用来构建可用的数据源。

[0004] 当今在ETL和数据准备空间中有两种占优势的模型。数据流程风格系统使用户聚焦于在整个系统中的数据的操作和流程上，这帮助提供在工作的整体结构上的清楚，并使用户控制那些步骤变得更容易。然而，这些系统通常对向用户显示其实际数据工作做得很差，这使用户实际上理解对他们的数据做什么或者需要做什么变得很难。这些系统也可能遭受节点的激增。当每个小操作在图中得到其自己的节点时，甚至中等复杂的流程也可能变成节点和边的令人混乱的老鼠窝(a confusing rat's nest)。

[0005] 另一方面，波特轮风格系统给用户显现对其实际数据的非常具体的电子表格风格界面，并允许用户通过直接的动作来对其数据造型。当用户实际上在这些系统中创造数据流程时，该流程通常被阻塞，使用户理解和控制他们的工作的整体结构变得很难。

[0006] 概述

[0007] 所公开的实现方式具有提供数据流程风格系统和波特轮风格系统的益处的特征，并且更进一步使用户构建数据流程变得更容易。所公开的数据准备应用描述数据流程，但是使节点塌缩到更大的组内，这些组更好地表示用户希望采取的高级动作。这些节点的设计利用在每个步骤由统计数据和相关可视化指导的对实际数据的直接动作。

[0008] 根据一些实现方式，计算机系统准备用于分析的数据。计算机系统包括一个或更多个处理器、存储器、以及存储在存储器中的一个或更多个程序。程序被配置为由一个或更多个处理器执行。程序显示数据准备应用的用户界面。用户界面包括数据流程窗格、工具窗格、配置文件窗格、和数据窗格。数据流程窗格显示标识数据源、操作、和输出数据集的节点/链路流程图。工具窗格包括使用户能够将数据源添加到流程图的数据源选择器，包括使用户能够将节点插入到流程图中用于执行特定的转换操作的操作调色板以及用户可以合并到流程图中的其他流程图的调色板。配置文件窗格显示对应于流程图中的选定节点的方案，包括关于数据字段的信息和关于数据字段的数据值的统计信息，并使用户能够通过与各个数据元素交互来修改流程图。数据窗格显示对应于流程图中的选定节点的数据行，并使用户能够通过与各个数据值交互来修改流程图。

[0009] 在一些实现方式中，关于在配置文件窗格中显示的数据字段的信息包括第一数据字段的数据范围。

[0010] 在一些实现方式中，响应于对在配置文件窗格中的第一数据字段的第一数据范围

的第一用户动作，将数据过滤到第一数据范围的新节点被添加到流程图。

[0011] 在一些实现方式中，配置文件窗格使用户能够将第一数据字段的数据范围映射到指定值，从而将执行用户指定的映射的新节点添加到流程图。

[0012] 在一些实现方式中，响应于与在数据窗格中的第一数据值的第一用户交互，将数据过滤到第一数据值的节点被添加到流程图。

[0013] 在一些实现方式中，响应于对在数据窗格中的第一数据字段的第一数据值的用户修改，对第一数据字段的数据值等于第一数据值的每行数据执行修改的新节点被添加到流程图。

[0014] 在一些实现方式中，响应于对在数据窗格中的第一数据字段的第一用户动作，将第一数据字段分成两个或更多个单独的数据字段的节点被添加到流程图。

[0015] 在一些实现方式中，响应于在数据流程窗格中的将第一节点拖动到工具窗格的第一用户动作，新操作被添加到操作调色板，该新操作对应于第一节点。

[0016] 在一些实现方式中，配置文件窗格和数据窗格被配置为当在数据流程窗格中做出选择时异步地更新。

[0017] 在一些实现方式中，关于在配置文件窗格中显示的数据字段的信息包括显示数据字段的数据值的分布的一个或更多个直方图。

[0018] 根据一些实现方式，过程转换数据。该过程在具有显示器、一个或更多个处理器、和存储器的计算设备处被执行，该存储器存储被配置为由一个或更多个处理器执行的一个或更多个程序。该过程显示包括数据流程窗格和数据窗格的用户界面。该过程接收第一用户输入以在数据流程窗格中构建节点/链路数据转换流程图。流程图中的每个节点指定从相应数据源检索数据的相应操作，指定转换数据的相应操作，或者指定创建相应输出数据集的相应操作。流程图包括具有从第一数据源检索数据的一个或更多个数据源节点和一个或更多个转换操作节点的子树。该过程接收第二用户输入以至少执行子树。根据第二用户输入和对在子树中的节点被配置为命令式地执行的确定，该过程根据由子树中的链路所指定的那样顺序地执行子树中的节点的操作，从而从第一数据源检索数据，转换所检索的数据，形成第一中间数据集，并在数据窗格中显示第一中间数据集。该过程接收第三用户输入以配置子树中的节点来命令式地执行，并且接收第四用户输入以至少执行子树。根据第四用户输入和对子树中的节点被配置为命令式地执行的确定，该过程构造在逻辑上等同于由子树中的节点指定的操作的数据库查询，将数据库查询传输到第一数据源以根据数据库查询从第一数据源检索第二中间数据集，并且在数据窗格中显示第二中间数据集。

[0019] 根据一些实现方式，该过程包括存储第一中间数据集和第二中间数据集。

[0020] 在一些实例中，子树是整个流程图。

[0021] 根据一些实现方式，转换操作节点之一指定过滤由一个节点接收的数据行的过滤转换操作，顺序地执行子树中的节点的操作包括在计算机系统处执行过滤转换操作以过滤掉接收到的数据行，以及根据数据库查询从第一数据源检索第二中间数据集包括在托管第一数据源的远程服务器处应用过滤操作。

[0022] 根据一些实现方式，转换操作节点之一指定联接来自第一数据源的两组数据的联接转换操作，顺序地执行子树中的节点的操作包括执行联接转换操作以在计算机系统处组合两组数据，以及根据数据库查询从第一数据源检索第二中间数据集包括在托管第一数据

源的远程服务器处应用联接操作。

[0023] 根据一些实现方式，数据库查询是用SQL编写的。

[0024] 根据一些实现方式，流程图包括未被包括在子树中的部分，该部分被配置为命令式地执行，第四用户输入指定整个流程图的执行，以及执行流程图包括根据由该部分中的链路指定的那样顺序地执行该部分中的节点的操作，从而访问第二中间数据集，转换第二中间数据集，并且形成最终数据集。

[0025] 根据一些实现方式，过程重构流程图。该过程在具有显示器、一个或更多个处理器、和存储器的计算机系统处被执行，该存储器存储被配置为由一个或更多个处理器执行的一个或更多个程序。该过程包括显示用户界面，该用户界面包括多个窗格，包括数据流程窗格和调色板窗格。数据流程窗格包括具有多个现有节点的流程图，每个节点指定从相应数据源检索数据的相应操作，指定转换数据的相应操作，或指定创建相应输出数据集的相应操作。此外，调色板窗格包括多个流程元素模板。该过程还包括接收第一用户输入以从流程图中选择现有节点或者从调色板窗格选择流程元素模板，并且响应于第一用户输入：(i)显示表示用于放置在流程图中的新节点的可移动图标，其中新节点指定对应于选定现有节点或选定流程元素模板的数据流程操作，以及(ii)根据在新节点的数据流程操作和多个现有节点的操作之间的相关性来在流程图中显示一个或更多个置放(drop)目标。该过程还包括接收第二用户输入以将可移动图标放置在置放目标中的第一置放目标上方，并停止检测第二用户输入。响应于停止检测到第二用户输入，该过程在第一置放目标处将新节点插入流程图中。新节点执行指定的数据流程操作。

[0026] 根据一些实现方式，每个现有节点具有根据指定的相应操作计算的相应中间数据集，以及在第一置放目标处将新节点插入流程图中包括根据指定的数据流程操作来计算新节点的中间数据集。

[0027] 根据一些实现方式，新节点在流程图中被放置在具有第一中间数据集的第一现有节点之后，以及计算新节点的中间数据集包括将数据流程操作应用于第一中间数据集。

[0028] 根据一些实现方式，新节点在流程图中没有原有物(predecessor)，以及计算新节点的中间数据集包括从数据源检索数据以形成中间数据集。

[0029] 根据一些实现方式，该过程还包括，响应于停止检测到第二用户输入，在用户界面的数据窗格中显示来自中间数据集的数据的采样。数据窗格是多个窗格之一。

[0030] 根据一些实现方式，数据流程操作基于第一数据字段的值来过滤数据行，以及显示一个或更多个置放目标包括紧接在现有节点之后显示一个或更多个置放目标，所述现有节点的中间数据集包括第一数据字段。

[0031] 根据一些实现方式，第一用户输入从流程图中选择现有节点，以及在第一置放目标处将新节点插入流程图中创建现有节点的副本。

[0032] 根据一些实现方式，在第一置放目标处将新节点插入流程图中还包括从流程图中移除现有节点。

[0033] 根据一些实现方式，数据流程操作包括以指定顺序执行的多个操作。

[0034] 根据一些实现方式，一种方法在具有显示器的电子设备处执行。例如，电子设备可以是智能电话、平板计算机、笔记本计算机、或台式计算机。该方法实现本文描述的任何计算机系统。

[0035] 在一些实现方式中，一种非暂时性计算机可读存储介质存储被配置为由具有一个或更多个处理器、存储器、和显示器的计算机系统执行的一个或更多个程序。一个或更多个程序包括用于实现如本文所述的准备用于分析的数据的系统的指令。

[0036] 因此，公开了使用户能够分析、准备、和整理数据以及重构现有数据流程的方法、系统、和图形用户界面。

[0037] 附图简述

[0038] 为了对前面提到的系统、方法、和图形用户界面以及提供数据可视化分析方法和数据准备的额外系统、方法、和图形用户界面的更好理解，应结合随附的附图来参考下面的实现方式的描述，其中相似的参考数字在全部附图中指相对应的部分。

[0039] 图1示出了在一些实现方式中使用的图形用户界面。

[0040] 图2是根据一些实现方式的计算设备的框图。

[0041] 图3A和图3B示出了根据一些实现方式的数据准备应用的用户界面。

[0042] 图3C描述了图3A和图3B中所示的用户界面的一些特征。

[0043] 图3D示出了根据一些实现方式的范例流程图。

[0044] 图3E示出了根据一些实现方式的一起工作但以不同频率运行的一对流程。

[0045] 图4A-4V示出了根据一些实现方式使用数据准备应用来构建联接。

[0046] 图5A示出了根据一些实现方式的日志文件的一部分。

[0047] 图5B示出了根据一些实现方式的查找表的一部分。

[0048] 图6A-6C示出了根据一些实现方式的流程的一些操作、输入、和输出。

[0049] 图7A和7B示出了根据一些实现方式的数据准备系统的一些部件。

[0050] 图7C示出了根据一些实现方式关于分析或执行来评估流程。

[0051] 图7D示意性地表示在一些数据准备实现方式中使用的异步子系统。

[0052] 图8A示出了根据一些实现方式的一系列流程操作。

[0053] 图8B示出了根据一些实现方式的类型系统的三个方面。

[0054] 图8C示出了根据一些实现方式的类型环境的属性。

[0055] 图8D示出了根据一些实现方式的基于具有所有已知的数据类型的流程的简单类型检查。

[0056] 图8E示出了根据一些实现方式的对于完全已知的类型的简单类型故障。

[0057] 图8F示出了根据一些实现方式的关于部分流程的简单类型环境计算。

[0058] 图8G示出了根据一些实现方式的打包容器节点的类型。

[0059] 图8H示出了根据一些实现方式的更复杂的类型环境场景。

[0060] 图8I示出了根据一些实现方式重新使用更复杂的类型环境场景。

[0061] 图8J-1、图8J-2和图8J-3指示根据一些实现方式的许多最常用的操作符的属性。

[0062] 图8K和图8L示出了根据一些实现方式的流程和相对应的执行过程。

[0063] 图8M示出了根据一些实现方式运行以在输入和输出节点处的隐含物理模型开始的整个流程。

[0064] 图8N示出了根据一些实现方式运行使具有结果的物理模型具体化的部分流程。

[0065] 图8O示出了根据一些实现方式基于先前的结果来运行流程的一部分。

[0066] 图8P和图8Q示出了根据一些实现方式评估具有固定节点860的流程。

[0067] 图9示出了根据一些实现方式的流程图的一部分。

[0068] 现在将参考其示例在附图中被示出的实现方式。在下面的描述中，阐述了许多具体细节以便提供对本发明的透彻理解。然而，对于本领域中的普通技术人员将明显的是本发明可被实践而不需要这些特定细节。

[0069] 实现方式的描述

[0070] 图1示出用于交互式数据分析的图形用户界面100。根据一些实现方式，用户界面100包括数据选项卡114和分析选项卡116。当数据选项卡被选择114时，用户界面100显示也被称为数据窗格的方案信息区110。方案信息区110提供可以被选择并用于构建数据可视化的所命名的数据元素(例如，字段名称)。在一些实现方式中，字段名称的列表被分成一组维度(例如分类数据)和一组度量(例如数字量)。一些实现方式还包括参数的列表。当分析选项卡116被选择时，用户界面显示分析功能而不是数据元素(未示出)的列表。

[0071] 图形用户界面100还包括数据可视化区112。数据可视化区112包括多个搁架区，例如列搁架区120和行搁架区122。这些也被称为列搁架120和行搁架122。如在这里所示的，数据可视化区112还具有用于显示视觉图形的大空间。因为数据元素还没有被选择，该空间最初没有视觉图形。在一些实现方式中，数据可视化区112具有被称为薄片的多个层。

[0072] 图2是示出根据一些实现方式的可以显示图形用户界面100的计算设备200的框图。计算设备也可以由数据准备(“数据prep”)应用250使用。计算设备200的各种示例包括台式计算机、膝上型计算机、平板计算机以及具有显示器和能够运行数据可视化应用222的处理器的其他计算设备。计算设备200通常包括用于执行存储在存储器214中的模块、程序、和/或指令并从而执行处理操作的一个或更多个处理单元/核心(CPU)202；一个或更多个网络或其它通信接口204；存储器214；以及用于使这些部件互连的一个或更多个通信总线212。通信总线212可以包括互连并控制在系统部件之间的通信的电路。

[0073] 计算设备200包括用户接口206，其包括显示设备208和一个或更多个输入设备或机构210。在一些实现方式中，输入设备/机构包括键盘。在一些实现方式中，输入设备/机构包括“软”键盘，其根据需要显示在显示设备208上，使用户能够“按下”出现在显示器208上的键。在一些实现方式中，显示器208和输入设备/机构210包括触摸屏显示器(也被称为触敏显示器)。

[0074] 在一些实现方式中，存储器214包括高速随机存取存储器，例如DRAM、SRAM、DDR RAM或其他随机存取固态存储器设备。在一些实现方式中，存储器214包括非易失性存储器，例如一个或更多个磁盘存储设备、光盘存储设备、闪存设备、或其他非易失性固态存储设备。在一些实现方式中，存储器214包括远离CPU 202定位的一个或更多个存储设备。存储器214或可选地在存储器214内的非易失性存储器设备包括非暂时性计算机可读存储介质。在一些实现方式中，存储器214或存储器214的计算机可读存储介质存储下面的程序、模块、和数据结构、或其子集：

[0075] • 操作系统216，其包括用于处理各种基本系统服务和执行硬件相关任务的过程；

[0076] • 通信模块218，其用于经由一个或更多个通信网络接口204(有线或无线)和一个或更多个通信网络(例如互联网、其它广域网、局域网、城域网等)将计算设备200连接到其它计算机和设备；

[0077] • web浏览器220(或能够显示网页的其他应用)，其使用户能够通过网络与远程计

算机或设备进行通信；

[0078] • 数据可视化应用222，其为用户提供图形用户界面100以构建视觉图形。例如，用户选择一个或更多个数据源240(其可以存储在计算设备200上或远程地被存储)，从数据源中选择数据字段，并使用所选择的字段来定义视觉图形。在一些实现方式中，用户提供的信息被存储为视觉规范228。数据可视化应用222包括数据可视化生成模块226，其采用用户输入(例如，视觉规范228)，并生成相对应的视觉图形(也称为“数据可视化”或“数据viz”)。数据可视化应用222然后在用户界面100中显示所生成的视觉图形。在一些实现方式中，数据可视化应用222作为独立应用(例如，桌面应用)来执行。在一些实现方式中，数据可视化应用222使用由web服务器提供的网页在web浏览器220或另一应用中执行；以及

[0079] • 由数据可视化应用222使用的零个或多个数据库或数据源240(例如，第一数据源240-1和第二数据源240-2)。在一些实现方式中，数据源被存储为电子表格文件、CSV文件、XML文件、或平面文件、或者被存储在关系数据库中。

[0080] 在一些实例中，计算设备200存储可用于分析和改动用于(例如，由数据可视化应用222)后续分析的数据的数据准备应用250。图3B示出了由数据准备应用250使用的用户界面251的一个示例。数据准备应用250使用户能够构建流程323，如下面更详细描述的。

[0081] 上面所识别的可执行模块、应用、或过程集中的每一者可以被存储在前面提到的存储器设备中的一个或更多个中，并且对应于用于执行上述功能的一组指令。上面所识别的模块或程序(即，指令集)不需要被实现为单独的软件程序、过程、或模块，并且因此在各种实现方式中这些模块的各种子集可以组合或以其他方式重新布置。在一些实现方式中，存储器214存储上面所识别的模块和数据结构的子集。此外，存储器214可以存储上面未描述的附加模块或数据结构。

[0082] 尽管图2示出了计算设备200，但是图2更多地预期作为可能存在的各种特征的功能描述，而不是作为本文所述的实现方式的结构示意图。在实践中且如本领域中的普通技术人员所认识到的，单独示出的项目可以组合并且一些项目可以被分离。

[0083] 图3A和图3B示出了根据一些实现方式的用于准备数据的用户界面。在这些实现方式中，有具有不同的功能的至少五个区。图3A在概念上将此显示为菜单栏区301、左手侧窗格302、流程窗格303、配置文件窗格304、和数据窗格305。在一些实现方式中，配置文件窗格304也被称为方案窗格。在一些实现方式中，“左手侧窗格”302的功能在替代位置上，例如在菜单窗格301下方或在数据窗格305下方。

[0084] 该界面给用户提供多个精简、协调的视图，其帮助用户看到并理解他们需要做什么。这个新颖的用户界面向用户显现其流程和其数据的多个视图以帮助他们不仅采取动作而且发现他们需要采取什么动作。流程窗格303中的流程图组合并总结动作，使流程变得更可读，并且与在配置文件窗格304和数据窗格305中的实际数据的视图协调。数据窗格305提供在逻辑流程中的每个点处的数据的代表性样本，且配置文件窗格提供数据的域的直方图。

[0085] 在一些实现方式中，菜单栏301具有文件菜单，该文件菜单具有创建新数据流程规范、保存数据流程规范、和加载先前创建的数据流程规范的选项。在一些实例中，流程规范被称为流程。流程规范描述如何操纵来自一个或更多个数据源的输入数据以创建目标数据集。目标数据集通常用在使用数据可视化应用的后续数据分析中。

[0086] 在一些实现方式中，左手侧窗格302包括最近数据源连接的列表以及连接到新数据源的按钮。

[0087] 在一些实现方式中，流程窗格303包括流程规范的视觉表示(流程图或流程)。在一些实现方式中，流程是显示数据源、被执行的操作、和流程的目标输出的节点/链路图。

[0088] 一些实现方式通过将流程的部分处理为声明性查询来提供流程的灵活执行。也就是说，不是让用户指定每个计算细节，而是用户指定目标(例如，输入和输出)。执行流程的过程优化了选择提高性能的执行策略的计划。实现方式还允许用户选择性地禁止该行为以控制执行。

[0089] 在一些实现方式中，配置文件窗格304显示在流程窗格303中选择的节点的方案和相关统计和/或可视化。一些实现方式支持同时选择多个节点，但是其他实现方式支持一次只选择单个节点。

[0090] 在一些实现方式中，数据窗格305显示在流程窗格303中选定节点的行级数据。

[0091] 在一些实现方式中，用户使用菜单栏中的“文件->新流程”选项来创建新流程。用户还可以将数据源添加到流程。在一些实例中，数据源是关系数据库。在一些实例中，一个或更多个数据源是基于文件的，例如CSV文件或电子表格文件。在一些实现方式中，用户使用在左手侧窗格302中的文件连接可视性(affordance)来将基于文件的源添加到流程。这打开提示用户选择文件的文件对话框。在一些实现方式中，左手侧窗格302还包括使用户能够连接到数据库(例如，SQL数据库)的数据库连接可视性。

[0092] 当用户在流程窗格303中选择节点(例如，表)时，该节点的方案被显示在配置文件窗格304中。在一些实现方式中，配置文件窗格304包括统计或可视化，例如字段的数据值的分布(例如，作为直方图或饼形图)。在使流程窗格303中的多个节点的选择成为可能的实现方式中，关于每个选定节点的方案被显示在配置文件窗格304中。

[0093] 此外，当在流程窗格303中选择节点时，该节点的数据被显示在数据窗格305中。数据窗格305通常以行和列来显示数据。

[0094] 实现方式使利用流程窗格303、配置文件窗格304、或数据窗格305编辑流程变得容易。例如，一些实现方式使在这三个窗格中的任一个中的节点/表上的右键单击操作成为可能，并基于在该表中的现有列上的标量计算来添加新列。例如，标量操作可以是计算三个数字列之和的数学操作、连接来自两列(其为字符串)的字串数据的字串操作、或者将字符串列转换为日期列的转换操作(当日期在数据源中被编码为字符串时)。在一些实现方式中，右键单击菜单(从在流程窗格303、配置文件窗格304、或数据窗格305中的表/节点访问)提供“创建所计算的字段...”的选项。选择该选项初启创建计算的对话框。在一些实现方式中，计算限于标量计算(例如，不包括聚合、自定义详细级别计算、和表计算)。当新列被创建时，用户界面在流程窗格303中添加所计算的节点，将新节点连接到其前项，并选择该新节点。在一些实现方式中，当流程图中的节点的数量变大时，流程窗格303添加滚动框。在一些实现方式中，流程图中的节点可以被分组在一起并被标记，被分层地显示(例如，最初显示高级流程，向下搜索(drill)以查看选定节点的细节)。

[0095] 用户还可以通过与流程窗格303、配置文件窗格304、或数据窗格305交互来移除列(例如，通过右键点击该列并选择“移除列”选项)。移除列导致将节点添加到流程窗格303，适当地连接新节点，并选择新节点。

[0096] 在流程窗格303中，用户可以选择节点并选择“输出为”以创建新的输出数据集。在一些实现方式中，这使用右键点击来执行。这将初启让用户选择目标文件名和目录(或数据库和表名)的文件对话框。完成该向流程窗格303添加新节点，但实际上并没有创建目标数据集。在一些实现方式中，目标数据集具有两个部件，包括包含数据的第一文件(表格数据提取或TDE)、和指向该数据文件的相对应的索引或指针条目(表格数据源或TDS)。

[0097] 实际的输出数据文件在流程运行时被创建。在一些实现方式中，用户通过从菜单栏301中选择“文件->运行流程”来运行流程。注意，单个流程可以产生多个输出数据文件。在一些实现方式中，流程图在其运行时提供视觉反馈。

[0098] 在一些实现方式中，菜单栏301包括在“文件”菜单上的“保存”或“另存为”的选项，其使用户能够保存流程。在一些实现方式中，流程被保存为“.loom”文件。该文件包含在加载时重新创建流程所需的所有事物。当流程被保存时，它可以使用在“文件”菜单中的“加载”的菜单选项来稍后被重新加载。这初启让用户加载以前的流程的文件选择器对话框。

[0099] 图3B示出了用于数据准备的用户界面，其显示在每个窗格中的用户界面元素。菜单栏311包括一个或更多个菜单，例如文件菜单和编辑菜单。虽然编辑菜单是可用的，但是对流程的更多改变通过与流程窗格313、配置文件窗格314、或数据窗格315进行交互来被执行。

[0100] 在一些实现方式中，左手侧窗格312包括数据源调色板/选择器，其包括用于定位和连接到数据的可视性。这组连接器包括只提取连接器，包括立方体。实现方式可以向支持它的任何数据源发出自定义SQL表达式。

[0101] 左手侧窗格312还包括操作调色板，其显示可以放置到流程中的操作。这包括任意联接(任意类型的和具有各种谓词)、联合、将行转换为列、重命名和限制列、标量计算的投影、过滤、聚合、数据类型转换、数据解析、合并、融合、拆分、聚合、值替换、和采样。一些实现方式还支持操作员创建集合(例如，将数据字段的数据值划分成集合)、进仓(例如，将数据字段的数字数据值分组成一组范围)、和表计算(例如，计算每行的数据值(例如，总数的百分比)，这些数据值不仅依赖于行中的数据值，而且还依赖于表中的其他数据值)。

[0102] 左手侧窗格312还包括可以全部或部分地合并到当前流程中的其他流程的调色板。这使用户能够重新使用流程的部件来创建新的流程。例如，如果已经创建了使用10个步骤的组合来擦掉某一类型的输入的流程的一部分，则该10个步骤流程部分可以在同一流程中或在完全分离的流程中被保存和重新使用。

[0103] 流程窗格313显示当前流程的视觉表示(例如，节点/链路流程图)323。流程窗格313提供用于用记载(document)该过程的流程的概述。在许多现有产品中，流程过于复杂，这阻碍了理解。所公开的实现方式通过合并节点来促进理解，保持整体流程更简单和更简洁。如上面所提到的，当节点的数量增加时，实现方式通常添加滚动框。通过将多个相关节点合并成也被称为容器节点的超级节点来减小对滚动条的需要。这使用户能够更在概念上看到整个流程，并且只在必要时允许用户钻研细节。在一些实现方式中，当“超级节点”被扩展时，流程窗格313仅显示在超级节点内的节点，并且流程窗格313具有识别流程的哪个部分正在显示的标题。实现方式通常实现多个分层级别。复杂的流程可能包括节点嵌套的几个等级。

[0104] 如上所述，配置文件窗格314包括关于在流程窗格313中的当前选择的节点(或多

个节点)处的数据的方案信息。如在这里所示的,方案信息提供关于数据的统计信息,例如每个字段的数据分布的直方图324。用户可以直接与配置文件窗格交互以修改流程323(例如,通过基于数据字段的值来选择用于过滤数据行的数据字段)。配置文件窗格314还给用户提供关于当前选择的节点(或多个节点)的相关数据以及指导用户的工作的可视化。例如,直方图324示出了每列的域的分布。一些实现方式使用刷擦来显示这些域如何与彼此交互。

[0105] 这里的一个示例通过使用户能够直接操纵在流程中的数据来示出该过程多么不同于一般实现方式。考虑过滤特定数据行的两种替代方式。在这种情况下,用户想要将加利福尼亚州排除在考虑之外。使用一般工具,用户选择“过滤器”节点,将过滤器放置到流程中某个位置处,然后初启输入计算公式(例如“state_name<> ‘CA’”)的对话框。在这里的所公开的实现方式中,用户可以在配置文件窗格314(例如,显示字段值“CA”以及有多少行具有该字段值)中和在数据窗格315(例如,具有“CA”作为state_name的值的单独行)中看到数据值。在一些实现方式中,用户可以右键点击在配置文件窗格314中(或在数据窗格315中)的状态名称的列表中的“CA”,并从下拉菜单中选择“排除”。用户与数据本身而不是与数据交互的流程元素交互。实现方式为计算、联接、联合、聚合等提供类似的功能。这种方法的另一个好处是,结果是立即的。当“CA”被过滤掉时,过滤器立即适用。如果操作花费一些时间来完成,则操作将异步地被执行,且当工作在后台中运行时用户能够继续工作。

[0106] 数据窗格315显示对应于在流程窗格313中的所选择的一个或更多个节点的数据行。每列315对应于数据字段之一。用户可以直接与数据窗格中的数据交互以修改流程窗格313中的流程323。用户还可以直接与数据窗格交互以修改各个字段值。在一些实现方式中,当用户对一个字段值做出改变时,用户界面将相同的改变应用于在其值(或模式)匹配用户刚刚改变的值的同一列中的所有其他值。例如,如果用户在州数据列中针对一个字段值将“WA”改变为“Washington”,则一些实现方式将同一列中的所有其他“WA”值更新为“Washington”。一些实现方式更进一步更新该列以将该列中的任何州缩写替换为完整的州名(例如,用“Oregon”替换“OR”)。在一些实现方式中,在对整个列应用全局改变之前,用户被提示确认。在一些实现方式中,对在一列中的一个值的改变也可以(自动或伪自动地)应用于其他列。例如,数据源可以包括居住的州和入账的州。然后,对州的格式的改变可以应用于两者。

[0107] 在数据窗格315中对数据的采样被选择以向用户提供有价值的信息。例如,一些实现方式选择显示数据字段的值的全范围(包括异常值)的行。作为另一个示例,当用户选择了具有数据的两个或多个表的节点时,一些实现方式选择行以帮助联接这两个表。选择在数据窗格315中显示的行以显示在两个表之间匹配的行以及不匹配的行。这可能在确定哪些字段用于联接和/或确定使用什么类型的联接(例如,内、左外、右外、或全外)时是有用的。

[0108] 图3C示出了在用户界面中显示的一些特征以及什么由这些特征显示。如上面在图3B中所示的,流程图323总是显示在流程窗格313中。配置文件窗格314和数据窗格315也总是被示出,但是这些窗格的内容基于哪个或哪些节点在流程窗格313中被选择而改变。在一些实例中,在流程窗格313中的对节点的选择初启一个或更多个节点特定窗格(在图3A或图3B中未示出)。在被显示时,除了其他窗格还有节点特定窗格。在一些实现方式中,节点特定

窗格被显示为可移动的浮动弹出窗口。在一些实现方式中，节点特定窗格被显示在用户界面中的固定位置处。如上面所提到的，左手侧窗格312包括用于选择或打开数据源的数据源调色板/选择器以及用于选择可应用于流程图323的操作的操作调色板。一些实现方式还包括“其他流程调色板”，其使用户能够将另一个流程的全部或部分导入当前流程323内。

[0109] 流程图323中的不同节点执行不同的任务，且因此节点内部信息是不同的。此外，一些实现方式根据节点是否被选择来显示不同的信息。例如，未选定的节点包括简单的描述或标签，而选定节点显示更详细的信息。一些实现方式还显示操作的状态。例如，一些实现方式根据节点的操作是否已经被执行来不同地显示流程图323中的节点。此外，在操作调色板中，一些实现方式根据操作对当前选择的节点使用是否可用的来不同地显示操作。

[0110] 流程图323提供了一种容易、形象的方式来理解数据如何得到处理，并且保持过程以对用户合乎逻辑的方式被组织。尽管用户可以直接在流程窗格313中编辑流程图323，但是对操作的改变通常以更直接的方式完成，直接对在配置文件窗格314或数据窗格315中的数据或方案进行操作(例如，右键点击在配置文件窗格中的数据字段的统计数据以添加列或从流程中移除列)。

[0111] 不是为每个微小的操作显示节点，用户能够将操作一起分组到更少数量的更重要的节点内。例如，可以在一个节点而不是三个单独的节点中实现联接，接着移除两列。

[0112] 在流程窗格313中，用户可以执行各种任务，包括：

[0113] • 改变节点选择。这驱使什么数据显示在用户界面的其余部分中。

[0114] • 钉住(Pin)流程操作。这允许用户指定流程的一些部分必须首先发生，并且不能被重新排序。

[0115] • 拆分和组合操作。用户可以容易重新组织操作以匹配发生了什么事的逻辑模型。例如，用户可能想要建造被称为“标准化医院代码”的一个节点，其包含许多操作和特殊案例。用户可以最初创建各个操作，然后将代表各个操作的节点合并到超级节点“标准化医院代码”中。相反，在创建了包含许多各个操作的节点之后，用户可以选择拆分一个或更多个操作(例如，以创建可以更普遍地被重新使用的节点)。

[0116] 配置文件窗格314提供了用于使用户弄清楚转换的结果是否是他们预期它们将要成为的结果的快速方式。异常值和不正确的值通常基于与节点中的其他两个值的比较或基于在其他节点中的值的比较而在视觉上“弹出”。配置文件窗格帮助用户查出数据问题，无论这些问题是否由不正确的转换或脏数据引起。除了帮助用户找到坏数据以外，配置文件窗格还允许直接交互以修复所发现的问题。在一些实现方式中，配置文件窗格314异步地更新。当节点在流程窗格中被选择时，用户界面开始填充随着时间推移而变得更好的部分值(例如，数据值分布直方图)。在一些实现方式中，配置文件窗格包括警告用户是否完成的指示器。对于非常大的数据集，一些实现方式仅基于样本数据来构建配置文件。

[0117] 在配置文件窗格314中，用户可以执行各种任务，包括：

[0118] • 调查数据范围和相关性。用户可以使用配置文件窗格314以使用直接导航来聚焦于某些数据或列关系。

[0119] • 过滤输入/输出数据或数据范围。用户可以通过直接交互来向流程323添加过滤操作。这导致在流程窗格313中创建新节点。

[0120] • 转换数据。用户可以直接与配置文件窗格314交互，以便将值从一个范围映射到

另一个值。这在流程窗格313中创建新节点。

[0121] 数据窗格315提供了用于使用户查看和修改由流程产生的行的方式。通常，数据窗格选择对应于选定节点的行的采样(例如，10、50、或100行而不是一百万行的样本)。在一些实现方式中，行被采样，以便显示各种特征。在一些实现方式中，行用统计方法(例如每第n行)被采样。

[0122] 数据窗格315通常是用户清理数据的地方(例如，当源数据是不干净时)。像配置文件窗格一样，数据窗格异步地更新。当节点被首次选择时，数据窗格315中的行开始出现，并且采样随着时间的推移而变得更好。大多数数据集将只有数据的子集是在这里可用的(除非数据集很小)。

[0123] 在数据窗格315中，用户可以执行各种任务，包括：

[0124] • 导航的分类。用户可以基于列将数据窗格中的数据加以分类，这对流程没有影响。目的是帮助在数据窗格中导航数据。

[0125] • 导航的过滤器。用户可以过滤在视图中的数据，这不将过滤器添加到流程。

[0126] • 将过滤器添加到流程。用户还可以创建应用于流程的过滤器。例如，用户可以为特定数据字段选择各个数据值，然后根据该值来采取行动以过滤数据(例如，排除该值或仅包括该值)。在这种情况下，用户交互在数据流程323中创建新节点。一些实现方式使用户能够选择在单个列中的多个数据值，且然后基于选定值的集合来构建过滤器(例如，排除该集合或被限制于仅仅该集合)。

[0127] • 修改行数据。用户可以直接修改行。例如，将在特定行中的特定字段的数据值从3改变为4。

[0128] • 将一个值映射到另一个值。用户可以修改特定列的数据值，并在对特定列的具有该值的所有行中传播该改变。例如，对于代表州的整列用“NY”代替“N.Y.”。

[0129] • 拆分列。例如，如果用户看到日期被格式化为“14-Nov-2015”，用户可以将该字段分为日、月、和年的三个单独的字段。

[0130] • 合并列。用户可以合并两个或更多个列以创建单个组合列。

[0131] 节点特定窗格显示对流程中的选定节点独有的信息。因为节点特定窗格在大多数时间是不需要的，所以用户界面通常不用用户界面指定只用于此用途的区。替代地，一般使用在用户界面的其他区之上浮置的弹出窗口根据需要来显示节点特定窗格。例如，一些实现方式使用节点特定窗格来为联接、联合、将行转换成列、将列转换成行、运行Python脚本、解析日志文件、或将JSON对象转换为表格形式而提供特定的用户界面。

[0132] 数据源调色板/选择器使用户能够从各种数据源引入数据。在一些实现方式中，数据源调色板/选择器在左手侧窗格312中。用户可以使用数据源调色板/选择器来执行各种任务，包括：

[0133] • 建立数据源连接。这使用户能够从数据源中拉入数据，数据源可以是SQL数据库、数据文件(例如CSV或电子表格)、非关系数据库、web服务、或其他数据源。

[0134] • 设置连接属性。用户可以指定连接到数据源所需的证书和其他属性。对于一些数据源，属性包括对特定数据的选择(例如，在数据库中的特定表或来自工作手册文件的特定工作表)。

[0135] 在许多情况下，如上所示，用户基于用户与配置文件窗格314和数据窗格315的交

互来调用对在流程中节的点的操作。此外，左手侧窗格312提供操作调色板，其允许用户调用某些操作。例如，一些实现方式包括在操作调色板中的“调用Python脚本”的选项。此外，当用户创建他们想要重新使用的节点时，他们可以将它们保存为操作调色板上的可用操作。操作调色板提供已知操作的列表(包括用户定义的操作)，并允许用户使用用户界面手势(例如，拖动和置放)来将操作合并到流程中。

[0136] 一些实现方式提供了“其他流程调色板/选择器”，其允许用户容易重新使用他们已经构建的流程或其他人已经构建的流程。另一个流程调色板提供了用户可以从起开始的或将其合并的其他流程的列表。一些实现方式除了选择整个流程之外还支持选择其他流程的部分。用户可以使用用户界面手势(例如拖动和置放)来合并其他流程。

[0137] 节点内部确切地指定在节点中正在发生了什么操作。有足够的信息以使用户能够“重构”流程或更详细地理解流程。用户可以确切地查看什么在节点中(例如，什么操作被执行)，并且可以将操作从该节点中移除、移动到另一个节点内。

[0138] 一些实现方式包括项目模型，这允许用户将多个流程一起分组到“项目”或“工作簿”内。对于复杂的流程，用户可以将整体流程分成更可理解的组成部分。

[0139] 在一些实现方式中，操作状态被显示在左手侧窗格312中。因为许多操作在后台中异步地被执行，所以操作状态区向用户指示什么操作在进行中以及进度的状态(例如，1％完成、50％完成、或100％完成)。操作状态显示在后台中什么操作正在发生，使用户能够取消操作，使用户能够刷新数据，并使用户能够使部分结果运行至完成。

[0140] 流程例如图3B中的流程323代表从原始数据源通过转换而流到目标数据集的行的流水线。例如，图3D示出了简单的示例流程338。该流程基于涉及车辆的交通事故。相关数据被存储在事故表和车辆表中的SQL数据库中。在该流程中，第一节点340从事故表中读取数据，以及第二节点344从车辆表中读取数据。在该示例中，事故表被标准化(342)，并且一个或更多个关键字段被识别(342)。类似地，关于车辆数据一个或更多个关键字段被识别(346)。使用共享键来联接两个表(348)，并将结果写入到目标数据集(350)。如果事故表和车辆表都在同一个SQL数据库中，备选方案是创建在一个查询中从两个表中读取数据的单个节点。该查询可以指定要选择哪些数据字段，以及数据是否应该被一个或更多个过滤器(例如WHERE子句)限制。在一些实例中，如流程338中所指示的，数据被检索并在本地被联接，因为用于联接表的数据需要被修改。例如，车辆表的主键可以具有整数数据类型，而事故表可以使用零填充字符字段来指定所涉及的车辆。

[0141] 流程抽象(如在图3D中所示的流程抽象)是大多数ETL和数据准备产品所共有的。这个流程模型给了用户对他们的转换的逻辑控制。这种流程通常被解释为命令式程序，并在很少修改或不修改的情况下由平台执行。也就是说，用户已经提供了特定的细节来定义对执行的物理控制。例如，在这个流程上工作的一般ETL系统将确切地如所指定的那样从数据库中拉下两个表，如所指定的那样塑造数据，在ETL引擎中联接表，且然后将结果写出到目标数据集。对物理计划的完全控制可能是有用的，但是妨碍系统修改或优化计划以提高性能的能力(例如，在SQL server处执行前面的流程)。在大多数时间客户不需要执行细节的控制，因此这里的实现方式使操作能够命令式地被表达。

[0142] 这里的一些实现方式跨越从完全声明性查询到命令式程序的范围。一些实现方式利用内部分析查询语言(AQL)和联合评估器。默认地，流只要可能就被解释为单个声明性查

询规范。这个声明性查询被转换成AQL并移交给查询评估器，查询评估器最终分享操作符、分发、并执行它们。在上面图3D中的示例中，整个流程可以被设计为单个查询。如果两个表来自同一个服务器，则这个整个操作可能被推送到远程数据库，实现显著的性能益处。灵活性不仅使优化和分发流程执行成为可能，而且还实现对照即时数据源(例如，来自事务数据库，且不仅仅是数据仓库)的查询的执行。

[0143] 当用户想要控制流程的实际执行顺序时(例如，由于性能原因)时，用户可以钉住操作。钉住是告诉流程执行模块不要使移动操作经过计划中的那个点。在一些实例中，用户可能想要临时对顺序实行极端控制(例如，在流程创作或调试期间)。在这种情况下，所有的操作符都可以被钉住，且流程确切地以用户指定的顺序被执行。

[0144] 注意，并不是所有流程都可以分解成单个AQL查询，如图3E所示。在该流程中，存在每小时运行(362)的每小时置放352，并且数据在附加(356)到中转数据库(staging database)之前被标准化(354)。然后，在每天基础上(364)，来自中转数据库的数据被聚集(358)并作为目标数据集被写(360)出。在这种情况下，每小时的时间表和每天的时间表必须保持作为单独的块。

[0145] 图4A-4V示出了根据一些实现方式的将联接添加到流程的一些方面。如图4A所示，用户界面包括左窗格312、流程区域313、配置文件区域314、和数据网格315。在图4A-4V的示例中，用户首先使用在左窗格312中的连接调色板来连接到SQL数据库。在这种情况下，数据库包括由国家公路交通安全管理局提供的死亡事故分析报告系统(FARS)数据。如图4B所示，用户从可用表的列表402中选择“事故”表404。在图4C中，用户将事故表图标406拖动到流程区域313。一旦将表图标406置放在流程区域313中，就创建代表表的节点408，如图4D所示。此时，事故表的数据被加载，并且事故表的配置文件信息被显示在配置文件窗格314中。

[0146] 配置文件窗格314提供每个列(包括状态列410)的分布数据，如图4E所示。在一些实现方式中，在配置文件窗格中的每列数据都显示直方图以显示数据的分布。例如，加利福尼亚州、佛罗里达州、和佐治亚州有大量事故，而特拉华州有少量事故。配置文件窗格使利用在每列的顶部处的键图标412来识别作为键或部分键的列变得容易。如图4F所示，一些实现方式使用三个不同的图标来指定列是否是数据库键、系统键414、或“几乎是”系统键416。在一些实现方式中，当与一个或更多个其他列结合的列是系统键时，该列几乎是系统键。在一些实现方式中，如果空值行被排除，如果列是系统键，则列几乎是系统键。在他的示例中，“ST案例”和“案例号”都几乎是系统键。

[0147] 在图4G中，用户已经选择了在左窗格312中的“个人”表418。在图4H中，用户将个人表418拖动到流程区域313，流程区域313在被拖动时显示为可移动图标419。在将个人表图标419置放到流程区域313内之后，在流程区域中创建个人节点422，如图4I所示。在这个阶段，在事故节点408和个人节点422之间没有连接。在该示例中，两个节点被选择，因此配置文件窗格314分成两个部分：第一部分420显示事故节点408的配置文件信息，而第二部分421显示个人节点422的配置文件信息。

[0148] 图4J提供了流程窗格313和配置文件窗格314的放大视图。配置文件窗格314包括选项424以显示联接列候选项(即，联接来自两个节点的数据的可能性)。在选择该选项后，作为连接候选项的数据字段被显示在配置文件窗格314中，如图4K所示。因为现在联接候选项被显示，配置文件窗格314显示选项426以隐藏联接列候选项。在该示例中，配置文件窗格

314指示(430)在个人表中的列ST案例可以与在事故表中的ST案例字段联接。配置文件窗格还指示(428)在事故表中有三个额外的联接候选项,并且指示(432)在个人表中有两个额外的联接候选项。在图4L中,用户点击(433)提示图标,且作为响应,配置文件窗格将两个候选列放置成相邻于彼此,如图4M所示。事故表的AT案例列的标题434现在指示它可以与个人表的ST案例列联接。

[0149] 图4N示出了联接多个节点的数据的替代方法。在该示例中,用户已经将事故表数据408和人口表数据441加载到流程区域313中。通过简单地将人口节点441拖动到事故节点408的顶部上,联接被自动创建并联接体验窗格442被显示,联接体验窗格442使用户能够复查和/或修改该联接。在一些实现方式中,联接体验被放置在配置文件窗格314中;在其他实现方式中,联接体验临时代替配置文件窗格314。当联接被创建时,新节点440被添加到流程,其用图形显示在两个节点408和441之间的连接的创建。

[0150] 如图4O所示,联接体验442包括具有各种图标的工具栏区域448。当联接候选图标450被选择时,界面识别在每个表中的哪些字段是联接候选项。一些实现方式包括收藏夹图标452,其显示突显的“收藏夹(favorite)”数据字段(例如,由用户先前选择的、由用户先前识别为重要的、或者由用户通常先前选择的)。在一些实现方式中,收藏夹图标452用于将某些数据字段指定为收藏夹。因为存在有限的空间用于显示在配置文件窗格314和数据窗格315中的列,所以一些实现方式使用关于收藏夹数据字段的信息来选择哪些列被默认地显示。

[0151] 在一些实现方式中,“显示键”图标454的选择使界面识别哪些数据列是键或由多个数据字段组成的键的部分。一些实现方式包括数据/元数据切换图标456,其将显示从显示关于数据的信息切换到显示关于元数据的信息。在一些实现方式中,数据总是被显示,并且除了数据以外,元数据图标456切换是否元数据被显示。一些实现方式包括数据网格图标458,其切换数据网格315的显示。在图4O中,数据网格当前被显示,因此选择数据网格图标458将使数据网格不显示。实现方式通常也包括搜索图标460,其初启搜索窗口。默认地,搜索适用于数据和元数据(例如,数据字段的名称以及在字段中的数据值)。一些实现方式包括高级搜索的选项以更精确地指定什么被搜索。

[0152] 在联接体验442的左侧上是一组联接控件,包括联接类型464的规范。如在本领域中已知的,联接通常是左外联接、内联接、右外联接、或全外联接。这些由联接图标464用图形示出。当前的联接类型被突显,但是用户可以通过选择不同的图标来改变联接类型。

[0153] 一些实现方式提供联接子句概述466,其显示在联接的两侧上的字段的名称、以及在联接的两侧上的数据字段的数据值的直方图。当在联接中有多个数据字段时,一些实现方式显示所有相关的数据字段;其他实现方式包括用户界面控件(未示出)以在联接中的全部数据字段中滚动。一些实现方式还包括概述控件468,其示出来自每个表的多少行基于联接条件的类型被联接。在该控件中的部分的选择确定在配置文件窗格314和数据网格315中显示什么。

[0154] 图4P、图4Q、和图4R示出了联接控制区域462的替代用户界面。在每种情况下,联接类型都出现在顶部处。在每种情况下,存在被包括在联接中的数据字段的视觉表示。在这里有在联接中的两个数据字段,其是ST案例和年。在这些替代方案中的每一个中,还有用图形示出来自每个表的行的小部分被联接的区段。图4Q的上部分出现在下面的图4U中。

[0155] 图4R包括显示两个表如何相关的下部分。拆分条472代表在事故表中的行，以及拆分条474代表人口表。在中间的大条477代表通过在两个表之间的内联接而连接的行。因为当前选择的联接类型是左外联接，所以联接结果集476还包括代表没有链接到人口表的任何行的事故表的行的部分478。在底部处的是另一个矩形480，其表示未链接到事故表的任何行的人口表的行。因为当前联接类型是左外联接，所以部分480不被包括在结果集476中(在底部矩形480中的行将被包括在右外联接或全外联接中)。用户可以选择该图的任何部分，以及选定部分被显示在配置文件窗格和数据窗格中。例如，用户可以选择“左外部分”矩形478，且然后看在数据窗格中的行以查看这些行是否与用户的分析相关。

[0156] 图4S示出了使用图4R中所示的联接控制界面元素(包括联接控制选择器464)的联接体验。在这里，左外联接图标482被突显，如在图4T的放大视图中更清楚地示出的。在本示例中，第一个表是事故表，以及第二个表是因素表。如图4U所示，界面显示了被联接的行486的数量和未被联接的行488的数量。这个示例有大量未联接的行。用户可以选择未联接的条488以初启在图4V中的显示。通过在配置文件中刷擦和在数据网格中过滤，由于因素表在2010年之前没有条目的事实，用户能够看到空值是左外联接和不匹配值的结果。

[0157] 所公开的实现方式支持有助于各种场景的许多特征。上面已经描述了许多特性，但是下面的一些场景说明了这些特征。

[0158] 场景1:事件日志收集

[0159] Alex在IT行业工作，且他的工作之一是从在他们的基础设施中的机器收集和准备日志以产生用于在IT组织中的各种进行调试和分析的共享数据集。

[0160] 机器运行Windows，且Alex需要收集应用日志。已经有每天晚上运行并将日志的CSV导出转储到共享目录的代理;每天的数据都被输出到单独的目录，并且它们以指示机器名称的格式被输出。在图5A中示出来自应用日志的片段。

[0161] 这具有一些令人感兴趣的特性：

[0162] • 第1行包含标题信息。一般来说，情况可以是这样或可以不是这样。

[0163] • 每行数据有六列，但标题有五列。

[0164] • 在这里分隔符明确地是“,”。

[0165] • 最后一列可能使用了带引号的多行字符串。注意，在这里第3-9行是一行的所有部分。还注意，该字段使用双引号来指示应该逐字被解释的引号。

[0166] Alex创建一个流程，该流程读取在给定目录中的所有CSV文件，并对它们执行交错联合(例如，如果数据字段存在于至少一个CSV文件中，则创建该数据字段，但是当相同的数据字段存在于两个或更多个CSV文件中时，只创建该数据字段的一个实例)。CSV输入例程在读取五列时完成了相当好的工作，但是在读取第六列中的引号时表现失常，将它们作为几列读取。

[0167] Alex接着：

[0168] • 选择数据窗格中的列，并将它们归并在一起。

[0169] • 添加它来自于的机器名称，其取自文件名。他通过在数据的示例中选择机器名称并选择“提取为新列”来实现此。系统从这个动作推断出一个模式。

[0170] • 通过右键点击并选择“添加标识符”来为每行生成唯一标识符。

[0171] • 编辑正好在数据窗格中的列名称和类型。

[0172] 所有这些都通过对在数据窗格315中的数据的直接动作来完成，但是导致逻辑被插入到流程窗格313中的流程中。

[0173] Alex然后将他的目标数据仓库拖到流程窗格中，并将输出连接起来以将这些记录附加到高速缓存，其将包含他的日志的完整记录。

[0174] 最后，Alex的流程查询该目标数据集以找到前一天报告的机器的集合，将此与今天的机器进行比较，并向Alex输出具有未报告的预期机器的列表的警报。

[0175] 注意，Alex可以用不同的方式实现同样的结果。例如：

[0176] • Alex可以创建两个单独的流程：一个执行摄取；而一个比较每天的机器与前一天的机器，并接着用结果警告Alex。

[0177] • Alex可以创建在一个阶段中执行摄取的流程。当此完成时，Alex可以执行第二个流程，该第二个流程查询数据库并将每天与前一天进行比较并警告Alex。

[0178] • Alex可以创建具有将目标作为输入和输出的流程。该流程将执行摄取，将它写到数据库，并进一步聚合以找到当天的机器。它还查询目标以获得前一天的结果，执行比较，并发出警报。

[0179] Alex知道机器应该通宵报告，所以Alex每天早晨做的第一件事就是运行他的流程。然后，他将早晨的其余时间用来检查未报告的机器。

[0180] 场景2:收集和整合FARS

[0181] Bonnie为一家保险公司工作，且想要引入死亡事故分析报告系统(FARS)数据作为她的分析的组成部分。FARS数据是通过FTP可得到的，且Bonnie需要弄清楚如何得到它并把它拼凑起来。她决定使用数据准备应用250来这样做。

[0182] Bonnie看一眼FARS公布的一组格式，并决定使用DBF文件。这些DBF文件散布在FTP站点周围，并且只在压缩的ZIP压缩文件中是可用的。Bonnie探索树形视图，并选择她想要下载的文件。当数据下载时，Bonnie开始在她的流程中的下一步骤。她选择文件的集合并选择“提取”，这增加了将文件解压缩到由年份标记的单独目录中的步骤。

[0183] 当数据开始进入时，Bonnie开始看到问题：

[0184] • 最初几年有三个文件，其对应于三个表:事故、个人、和车辆。这些在后来的几年中存在，但是也有更多的表。

[0185] • 这些文件没有统一的名称。例如，事故文件在1975-1982年和1994-2014年被命名为“accident.dbf”，但在中间的年份被命名为“accYYYY.dbf”(其中YYYY是四数位年份)。

[0186] • 即使表名称是相同的，它们的结构也随着时间的推移而改变。后来的表包括在较早的数据中不存在的额外列。

[0187] Bonnie以在所有年份存在的事故表开始。她选择文件，右键点击，并然后选择“联合”，其执行交错联合并保留列。她对在所有年份存在的其他三个表且然后对其余表重复这一过程。当她完成时，她的流程的最后阶段产生19个单独的表。

[0188] 一旦她有了此，她就试图把数据拼凑在一起。看起来普通的连接键应该是被称为ST_CASE的列，但是仅仅通过看事故表的配置文件窗格，她可以知道这在任何地方都不是键列。ST_CASE不是键，但是通过点击年份，她可以容易地看到每年只有一个ST_CASE。年份和ST_CASE一起看起来是很好的连接键。

[0189] 她以个人表开始。在她对此可以联接之前，她需要在她的每个表中的年份，且它不

在那里。但是因为文件路径有年份，她可以选择在数据窗格中的该数据，且然后选择“提取为新列”。系统为此推断正确的模式，并提取每行的年份。然后，她选择在她的流程中的两个表，选择在一个表中的年份和案例列，并将它们拖到另一个表，创建联接。

[0190] 现在她有了键，她继续创建联接以拉平FARS数据。当她完成时，她将数据作为TDE(表格数据提取)公布到她的表格服务器，所以她的团队可以使用它。

[0191] 场景3:FARS清理

[0192] Colin是在与Bonnie相同的部门处的另一名员工。一些人试图使用Bonnie的流程产生的数据，但它包括许多具有隐含意义的值。发现Bonnie已经转移到另一家公司，他们转向Colin。

[0193] 看流程，Colin可以容易地看到它的整体逻辑，且也看到具有隐含意义的编码数据。当他发现包含关于具有隐含意义代码的查找表(LUT)的200页PDF手册时，这个过程看起来令人生畏。图5B显示了在PDF中的示例查找表。一些是比较简单的，而一些是明显比较复杂的。

[0194] Colin以一些更重要的表开始。他发现他可以选择在PDF文件中的表并将它粘贴到流程窗格313中。在一些情况下，表中的数据并不是完全正确的，但是它完成合理的工作，且Colin然后可以手动地修补数据窗格315中的结果，节省了他相当多的时间。当他工作时，他立即看到了他的结果。如果表未对齐，他马上看到此。

[0195] 最终，Colin引入似乎与他的团队执行的分析特别相关的十几个LUT，并公布结果，所以他的团队可以使用数据。当人们请求更多关于特定列的信息时，Colin可以进一步增大他的流程以引入额外的LUT。

[0196] 场景4:发现数据错误

[0197] Danielle是一家大型软件公司的开发人员，正在查看代表构建时间的数据。Danielle对数据的格式有很多控制，并以一种很好的可消费CSV格式产生它，但是想要简单地加载它并将它附加到她创建的数据库。

[0198] 当她载入数据时，她扫描配置文件视图314。对她来说，一些事情立即看起来很奇怪:存在具有负时间的若干构造。显然在这里出了问题，且她想要调试这个问题，但她也想要把数据聚集到一起用于分析。

[0199] 她在配置文件视图中选择负时间，并点击“仅保持”以仅保留错误的行。她添加目标以使这些流动到文件中。她将使用那些原始行来指导她的调试。

[0200] 回到她的流程，她正好在过滤器之前添加另一个分支。她再次选择负值(例如，在配置文件窗格314或数据窗格315中)，且然后简单地按下“删除”。这用空值代替值，空值是真实值不是已知的良好指示器。她继续进行她的简单流程的其余部分，将构建数据附加到数据库，且稍后她将查看负值。

[0201] 场景5:跟踪车辆零件

[0202] Earl为一家汽车制造商工作，并负责维护显示工厂中的每个车辆和主要零件的当前状态的数据集。数据被报告给若干操作暂存器，但是这些操作暂存器相当大。有几十万个零件，且作为自动化设施，当每个车辆或零件通过工厂前进时，几千条记录为每个车辆或零件机械地被创建。这些操作暂存器还包含与零件状态无关的许多记录，但包含其他操作信息(例如，“在阀134中的压力为500kPa”)。存在对关于每个零件的快速、简洁的记录的商业

需要。

[0203] Earl将三个关系操作暂存器中的每一个的表拖到流程窗格313中。它们中的两个将数据存储为包含日志记录的单个表。第三个关系操作暂存器具有小星形方案,Earl通过拖动和置放来快速拉平这个小星形方案以创建联接。

[0204] 接下来,通过额外的拖动和置放,Earl能够快速执行表的交错联合。在结果中,他可以将列拖放在一起,且界面为他合并结果。

[0205] 零件标识号是稍微有问题的:一个系统具有在值中的连字符。Earl采用在数据窗格315中的值之一,选择连字符,并按下删除。界面推断出从该列中删除连字符的规则,并将移除该列中的所有数据的连字符的规则插入流程中。

[0206] Earl不想要大多数状态代码,因为它们与他当前的项目无关。他只想要与零件相关的状态代码。他引入具有关于状态代码的信息的表,并将它置放在他的流程的最后一个节点上,导致关于状态代码的新联接。他现在只选择具有等于“零件”的“目标类型”的那些行,并选择“仅保持”以过滤掉其他值。该过滤在配置文件窗格314或数据窗格315中完成。

[0207] 最后,Earl只想要每个零件的最后一个值。通过直接的手势,他按日期对数据窗格中的数据进行排序,按零件号分组,并添加一个“前n个”表计算以便只对每个零件进行最终更新。

[0208] Earl运行他的流程,并发现它花费四个小时来运行。但他知道他可如何使这加速。他可以记录他运行他的流程的最后时间,并且只在每个随后的运行时合并新的记录。然而,为了完成此,他需要更新在他的累积集合中的现有行,并且只有当它们代表新的零件时才添加行。他需要“融合”操作。

[0209] Earl使用零件号来识别匹配,并提供当匹配出现或不出现时的动作。使用更新逻辑,Earl的流程只花费15分钟来运行。在时间上的节省让公司更紧密地随时跟踪零件在仓库中哪里以及它们的状态是什么样的。

[0210] Earl然后将这个工作推送到服务器,所以它可以被调度并在中央运行。他还可以在自己的桌上型机器上创建被调度的任务,该桌上型机器使用命令行界面来运行流程。

[0211] 场景6:投资经纪人

[0212] Gaston在一个团队中从事于投资经纪人,负责采用由IT产生的数据并消化它,使得数据可以被与客户一起工作的各种团队使用。IT产生各种数据集,其显示客户的投资搭配的部分——债券头寸、股票头寸等——但独自不是Gaston的客户所需要的东西。

[0213] 由Hermine领导的一个团队需要所有的客户位置数据被聚集在一起,使得她的团队可以在他们的客户拜访时回答他们的问题。数据准备没有那么复杂。

[0214] Gaston对IT产生的夜间数据库删除做一些改动,将它们联合在一起,并进行一些简单的检查以确保数据看起来正常。然后,他将它向下过滤到仅仅Hermine的团队需要的东西,并创建TDE用于使她的团队使用。

[0215] 使用他们的以前的工具,Gaston必须记住每天早晨到达并运行流程。但是使用新的数据准备应用250,可以声明性地处理这个流程。他向Hermine发送她的团队使用的TDS,所以Hermine的团队制作的每个数据可视化都直接对照数据库来运行。这意味着Gaston不必担心刷新数据,而且它快速执行。

[0216] 由Ian领导的另一个团队使用类似的数据来完成他的客户的账户的绩效审查。为

了产生这个数据,Gaston重新使用他为Hermine所做的工作,但是将数据过滤到Ian的团队的账户,且然后执行额外的流程以将数据与各种索引和绩效指标联接,使得Ian的团队可以执行他们的分析。这项工作很昂贵,而且并不看起来在现场执行得很好。如果他运行流程,则它花费几个小时来完成——但Ian的团队每月只需要这一次。Gaston在服务器上建立循环日历项目以每月运行它一次。

[0217] 场景7:剔除客户数据

[0218] Karl是一家大型软件公司的战略账户经理。他试图使用表格来可视化关于在行业会议处的出席者、他们为谁工作、他们的代表是谁,他们是活跃的还是有希望的客户、他们的公司是小公司是大公司等的信息。

[0219] Karl有会议出席者的列表,但他以前走过这条路。上一次他在这个位置上时,他花了8个小时来清理列表,以及在他完成时花了15分钟构建可视化。这一次他使用数据准备应用250来加速和自动化该过程。

[0220] Karl首先想要清理公司名称。细查这些数据,他看到了他所预期的东西:同一家公司经常以多种不同的格式被列出,且它们中的一些被拼错。他调用在操作调色板上提供的模糊重复数据消除例程以识别潜在的副本。他复查结果并校正几个案例,其中算法是过于急切的。他还发现算法被遗漏的几个案例,所以他将它们分组。这产生具有规范公司名称的客户列表。

[0221] 然后,他试图将他的数据与保持在他的表格服务器上的数据源中的公司的列表联接。他发现每家公司都有多个列表。多个不同的公司可能有相同的名称,且单个公司可能有基于地区的多个帐户。

[0222] 为了解决此,Karl使用他找到的关于LinkedIn™的REST连接器,并将它传递到在他的数据中的每个电子邮件地址以检索每个人的国家和州。该过程采用他拥有的信息(例如,个人的姓名、个人的公司、和个人的职位),并使用LinkedIn的搜索功能来为每个条目提出最佳结果。然后,他将公司和位置数据联接到在他的服务器中的数据以找到正确的帐户。

[0223] Karl发现他的联接并不总是起作用。他选择的规范公司名称并不总是与在他的账户数据库中的东西相匹配。他将他的联接转换为模糊联接,复查模糊匹配,并进一步手动地校正结果。

[0224] 现在他已经清理了他的数据,他在表格中打开它以创建他的数据可视化。

[0225] 流程的常用特征包括:

[0226] • 多个级别的联合、联接、和聚合,其需要用户具有对操作的逻辑顺序的精确控制。

[0227] • 由用户安排和注释以提高理解的布局。

[0228] • 当数据穿过流程前进时,对在数据的结构内的清晰性的需要。

[0229] • 重新使用流程的一部分以产生两个不同的输出。

[0230] • 有时在单独的团队中的为两个或更多其他用户准备数据的作者。

[0231] • 调度流程以自动运行。

[0232] 数据准备应用有时被分类为ETL(提取、转换、和加载)系统。这三个阶段中的每个都执行不同类型的任务。

[0233] 在提取阶段中,用户从一个或更多个可用数据源拉出数据。通常,用户执行这些任

务：

[0234] • 简单地移动文件。例如，用户可以在其他处理之前从FTP源检索文件。

[0235] • 摄取在结构(例如关系、半结构化、或非结构化)、格式(例如结构化存储、CSV文件、或JSON文件)、和源(例如来自文件系统或来自形式数据库(formal database))方面广泛变化的数据。

[0236] • 读取整个源或它的选定部分。部分读取是常见的，以拉出比上次摄取更新的或自从上次摄取以来改变的数据或由于性能原因而对数据块采样或拉出数据块。

[0237] 在转换阶段中，用户以多种方式转换数据。通常，转换包括这些任务：

[0238] • 清理数据以修复错误、处理缺失或重复的值、协调应该是相同的变量值、使值符合标准，等等。

[0239] • 通过标量和表计算、聚合、行和列的过滤、将行转换成列(将列转换成行)、或外部数据的合并(例如，通过地理编码)来增加或丰富数据。

[0240] • 通过联合或联接(包括模糊连接)来组合多个源。

[0241] • 使已放置在一起(在行中或在列中)用于单独处理的多种类型的数据去交错。

[0242] • 提取数据的配置文件或关于数据的度量以更好地理解它。

[0243] 在加载阶段中，用户存储结果，使得结果可被分析。这包括：

[0244] • 将数据写到表格数据提取(TDE)、格式化文件(例如CSV或Excel)、或外部数据库。

[0245] • 按计划创建快照。

[0246] • 用新的或修改的结果追加或更新数据。

[0247] 一旦用户构造了准备数据的流程，用户就常常需要：

[0248] • 安排流程在指定时间或与其他流程合作地运行。

[0249] • 与其他人共享流程的结果。

[0250] • 与其他人共享流程本身，使得其他人可以检查、修改、克隆、或管理它。这包括与IT共享流程或数据，使得IT可以改进和管理它。

[0251] 所公开的系统250给用户控制。在许多情况下，数据准备应用为用户做出智能选择，但是用户总是能够断言控制。控制常常有两个不同的方面：对操作的逻辑排序的控制，其用于确保结果是正确的并且与用户的期望语义相匹配；以及物理控制，其主要用于确保性能。

[0252] 所公开的数据准备应用250也提供自由。然而用户它们期望可以组装和重新组装他们的数据产生部件，以便实现他们需要的数据的形状。

[0253] 所公开的数据准备应用250提供增量交互和即时反馈。当用户采取动作时，系统通过关于用户的数据的样本的即时结果以及通过视觉反馈来提供反馈。

[0254] 一般，ETL工具使用命令式语义。也就是说，用户指定每个操作的细节和执行操作的顺序。这给了用户完全的控制。相比而言，SQL数据库引擎评估声明性查询，并能够基于由查询请求的数据来选择最佳执行计划。

[0255] 所公开的实现方式支持命令式和声明性操作，且用户可以在粒度的不同级别处在这两个执行选项之间进行选择。例如，用户可能想要在开始时实行流程的完全控制，同时了解新数据集。稍后，当用户对结果感到舒适时，用户可以将控制的全部或部分交给数据准备

应用,以便优化执行速度。在一些实现方式中,用户可以为每个流程指定默认行为(命令式或声明性),并无视在各个节点上的默认行为。

[0256] 所公开的实现方式可以将数据写到许多不同的目标数据库,包括TDE、SQL服务器、Oracle、Redshift、平面文件等。在一些实例中,流程在目标系统中创建新的数据集。在其他实例中,流程通过追加新行、更新现有行、插入行、或删除行来修改现有数据集。

[0257] 当运行流程时错误可能出现。错误可以包括瞬态系统问题、在数据中的潜在的已知错误条件(用户可以为了错误条件而对校正动作编码)以及作者没有考虑的隐含约束。所公开的实现方式通常在可能时自动处理这些错误条件。例如,如果在过去遇到相同的错误条件,一些实现方式重新应用已知的解决方案。

[0258] 虽然流程在本质上是数据转换,但是实现方式使用户能够用声明性建模信息来注释他们的输出,以解释输出可以如何被使用、查看、确认(validated)、或组合。示例包括:

[0259] • 影响值如何在表格中被显示(例如默认着色或格式)的注释。

[0260] • 在字段上的指示单位或行数的注释。

[0261] • 别名和组的创建。

[0262] • 函数约束,例如在表之间的主键和外键。

[0263] • 域约束,例如要求字段是正的。

[0264] 所公开的实现方式通常包括这些部件:

[0265] • 前端区域,用户可以与之交互以查看、构建、编辑、和运行流程。

[0266] • 抽象流程语言(AFL)。这是一种内部语言,其表示在流程中的所有逻辑,包括到源的连接、计算和其他转换、建模操作、以及对行做什么,其是流程的结果。

[0267] • 执行引擎。引擎解释并执行AFL程序。在一些实现方式中,引擎在本地运行。查询可能被推送到远程服务器,但是结果和进一步的处理将使用本地资源来完成。在服务器环境中,服务器为流程提供共享的分布式执行环境。该服务器可以调度和执行来自许多用户的流程,并且可以自动分析和向外扩展AFL流程。

[0268] • 目录服务器,其允许向其他人公布流程。

[0269] 一些数据可视化应用能够执行数据准备流程,并且可以使用TDE或其他所创建的数据集来构造数据可视化。

[0270] 所公开的实现方式还可以导入由其他应用创建的(例如,在ETL工具中创建的)一些数据流程。

[0271] 实现方式使用户能够:

[0272] • 连接到数据源并从数据源读取数据,如图6B所示。

[0273] • 构建以任意顺序和组合来组合所支持的操作(见图6A)的流程。

[0274] • 见数据将如何在它们的流程的每个步骤(例如,在配置文件窗格和数据窗格中)转换的合理例子。

[0275] • 在流程的每一步骤制作数据的可视化。

[0276] • 在本地执行完整的流程以产生输出,例如TDE或CSV输出(见图6C)。

[0277] • 将流水线或TDE结果公布到目录服务器。

[0278] • 将在数据准备中创建的TDS(表格数据源)作为显式流程导入。

[0279] 在对已配置的服务器的访问的情况下,用户可以:

[0280] • 与其他人共享TDE。

[0281] • 以适当的安全性与其他用户共享数据准备流水线(流程)。

[0282] • 在服务器环境中执行数据准备流水线以手动地或按计划产生TDE。

[0283] 一个节点的输出可以指向多于一个的后面的节点。在这里有两个基本情况。在第一种情况下,流程分叉且不回到一起。当流程不汇聚时,有来自流程的多个输出。在这种情况下,每个分支实际上是由在树中的所有原有物组成的单独查询。当可能时,实现方式优化此,使得流程的共享部分不被执行多于一次。

[0284] 在第二种情况下,流程确实汇聚。在语义上,这意味着行流经两条路径。再次,流程执行通常不加倍执行前身。注意,单个流程可以同时具有这两种情况。

[0285] 用户界面:

[0286] • 使用户能够在流程中创建分叉。当新节点被添加时,用户可以指定新节点是否在选定节点处创建分叉或作为中间节点插在操作的现有序列中。例如,如果当前存在从节点A到节点B的路径并且用户选择在A处插入新节点,则用户可以选择创建到新节点的第二路径或者在A和B之间插入新节点。

[0287] • 使用户能够运行流程的各个输出而不是整个流程。

[0288] 用户可以将过滤器添加到任意复杂性的流程。例如,用户可以点击以在流程中的某个点处添加过滤器,且然后输入充当谓词的计算。在一些实现方式中,计算表达式限于标量函数。但是,一些实现方式实现更复杂的表达式,如聚合、表计算、或详细级别表达式。

[0289] 用户可以编辑任何过滤器,即使它由系统推断出。特别是,所有过滤器都被表示为表达式。

[0290] 配置文件窗格314和数据窗格315提供了创建过滤器的简单方式。例如,一些实现方式使用户能够在数据窗格中为列选择一个或更多个数据值,然后右键点击并选择“仅保持“或“排除“。这将在当前选择的节点处将过滤器插到流程中。系统推断出表达式以实现过滤器,并该表达式被保存。如果用户需要以后修改过滤器,这么做很容易,无论以后的时间是马上还是一年以后。

[0291] 在配置文件窗格314中,用户可以选择为数据字段指定值的范围的存储桶(bucket)。例如,对于分类字段,范围通常被指定为值的列表。对于数值字段,该范围通常被指定为具有上限或下限的连续范围。用户可以选择存储桶,并容易创建选择(或排除)其字段的值落在该范围内的所有行的过滤器。

[0292] 当用户基于在一列中的多个值或一列的多个存储桶创建过滤器时,过滤器表达式使用OR。也就是说,如果行匹配任何一个值或范围,则该行与表达式匹配。

[0293] 用户还可以在数据窗格中基于在单行中的多个数据值创建过滤器。在这种情况下,过滤器表达式使用AND。也就是说,只有匹配所有指定值的行才匹配表达式。这也可以应用于在配置文件窗格中的存储桶。在这种情况下,行必须关于每个选定的存储桶范围匹配。

[0294] 一些实现方式还允许基于包括两行或更多行以及包括两列或更多列的多个数据值来创建过滤器。在这种情况下,所创建的表达式在析取范式中,每个析取项对应于具有选定数据值的行之一。一些实现方式也将相同的技术应用于在配置文件窗口中的范围选择。

[0295] 注意,在这些情况中的每一种下,用户在视觉上选择数据值或存储桶,然后用简单的手势(例如,右键点击加上菜单选择)创建过滤器,该过滤器将行限制到仅仅选定值或排

除选定值。用户不必弄清楚如何在正确的布尔逻辑中写表达式。

[0296] 如上面关于图4A-4V所示的，用户可以创建联接。根据声明性执行是否被实现，联接可以被推送到远程服务器用于执行，如下面在图9中所示的。

[0297] 一些实现方式提供流程简化或简缩版本作为节点和注释。在一些实现方式中，用户可以在全视图或简缩视图之间切换，或者切换各个节点以隐藏或暴露在节点内的细节。例如，单个节点可能包括十几个操作以对某些源文件执行清理。在具有清理步骤的实验的几次迭代之后，它们运行良好，且用户通常不想看到细节。细节仍然在那里，但是用户能够通过只查看节点的简缩版本来隐藏混乱。

[0298] 在一些实现方式中，不散开的操作节点被一起合拢到关于节点的注释内。例如联接和拆分的操作将用额外的节点中断流程。在一些实现方式中，简缩视图的布局是自动的。在一些实现方式中，用户可以在简缩视图中重新布置节点。

[0299] 配置文件窗格和数据窗格都提供了关于与在流程窗格中的当前选择的节点相关联的行的集合的有用信息。例如，配置文件窗格显示数据中的各种数据值的基数(例如，直方图显示多少行具有每个数据值)。为多个数据字段显示值的分布。由于在配置文件窗格中显示的数据的数量，数据的检索通常异步地被执行。

[0300] 在一些实现方式中，用户可以点击在配置文件窗格中的数据值，并查看其他项目的比例刷擦。当用户选择特定的数据值时，用户界面：

[0301] • 指示选择。

[0302] • 使用比例刷擦来指示与在该表中的其他列的相关性。

[0303] • 过滤或突显相关数据窗格以仅显示其值与该选择匹配的行。(这过滤在数据窗格中的所显示的数据，且不在流程窗格中创建过滤节点。)

[0304] • 当存在在配置文件窗格中选择的多个值时，所有选定的值被指示且数据窗格被相应地过滤(即，过滤到与所述值中的任何一个匹配的行)。

[0305] 在一些实现方式中，行不被显示在数据窗格中，除非由用户特别请求。在一些实现方式中，数据窗格总是自动被填充，过程异步地继续进行。一些实现方式对于选定节点基于行的基数来应用不同的标准。例如，一些实现方式在基数低于阈值时显示行，且如果基数高于阈值，不显示行或者异步地继续进行。一些实现方式指定两个阈值，将一组行指定为小的、大的、或非常大的。在一些实现方式中，界面显示小基数的行，继续进行以异步地显示大基数的行，并且不显示非常大的基数的结果。当然，数据窗格只能显示少量的行，这些行通常是通过采样(例如，每第n行)被选择。在一些实现方式中，数据窗格实现无限滚动以适应未知数量的数据。

[0306] 所公开的数据准备应用提供了用户界面在本机读取、修改、和操作的文档模型。该模型描述到用户的流程，同时为UI提供形式体系。该模型可以转换成使用AQL和联合评估器来运行的表格模型。该模型还实现中间结果的可靠缓存和重新使用。

[0307] 如图7A所示，数据模型包括三个子模型，每个子模型都描述了在它的适当评估阶段中的流程。第一子模型是“Loom Doc”702。(一些实现方式将数据准备应用称为“Loom”。)

[0308] Loom Doc 702是描述用户看到并与之直接交互的流程的模型。Loom Doc 702包含执行所有ETL操作和类型检查所需的所有信息。一般，Loom Doc 702不包括纯粹为了渲染或编辑流程所需的信息。Loom doc 702被构造为流程。每个操作具有：

[0309] • 描述其将如何执行其操作的一组属性。

[0310] • 描述要对什么数据执行操作的零个或更多个输入。

[0311] • 描述从该操作产生的数据的零个或更多个输出。

[0312] 有四种主要类型的操作:输入操作、转换操作、输出操作、和容器操作。

[0313] 输入操作执行ETL的“提取”部分。它们将流程绑定到数据源,并被配置为从该源拉出数据并将该数据暴露于流程。输入操作包括加载CSV文件或连接到SQL数据库。输入操作的节点通常具有零个输入和至少一个输出。

[0314] 转换操作执行ETL的“转换”部分。它们提供对数据流的“函数”操作并转换它。转换操作的示例包括“将计算创建为‘[HospitalName]-[Year]’”、“过滤具有hospitalId＝’HarbourView’的行”等。转换节点具有至少一个输入和至少一个输出。

[0315] 输出操作提供ETL的“加载”部分。它们在有用进来的数据流实际上更新下游数据源的副作用下操作。这些节点只有一个输入且没有输出(没有对在流程中的后续节点的“输出”)。

[0316] 容器操作将其他操作分组到逻辑组中。这些用于帮助使流程更容易进行记载。容器操作作为在流程窗格中的“节点”被暴露于用户。每个容器节点包含其他流程元素(例如,一系列常规节点)以及文档的字段。容器节点可以有任何数量的输入和任何数量的输出。

[0317] 数据流表示越过流程从一个节点移动到另一个节点的数据的实际行。在逻辑上,这些可以被视为行,但是在操作上数据流可以以任何数量的方式被实现。例如,一些流程被简单地向下编译成AQL(分析查询语言)。

[0318] 可扩展操作是数据准备应用不直接知道如何评估的操作,因此它调用第三方进程或代码。这些操作是不作为联合评估器的一部分运行的操作。

[0319] 逻辑模型704是包含所有实体、字段、关系、和约束的模型。它通过在流程上运行而被构造,并定义在流程的任何部分处被构造的模型。在逻辑模型中的字段是结果中的列。在逻辑模型中的实体代表结果中的表,但是一些实体包括其他实体。例如,并集具有作为其他实体的结果的实体。在逻辑模型中的约束表示附加约束,例如过滤器。在逻辑模型中的关系表示跨越实体的关系,提供足够的信息来组合它们。

[0320] 物理模型706是第三个子模型。物理模型包括用于缓存的元数据,包识别流程是否需要重新运行的信息以及如何关于流程直接查询结果数据库。元数据包括:

[0321] • 此时逻辑模型的散列。

[0322] • 每个根数据源的时间戳以及它上次何时被查询。

[0323] • 描述结果数据所在的位置的路径或URI。

[0324] 这个数据用于优化流程以及实现结果的更快导航。

[0325] 物理模型包括对用于创建该物理模型的逻辑模型的引用(例如指向文件或数据暂存器的指针)。物理模型706还包括表格数据源(TDS),其识别将被用于评估模型的数据源。通常,这从逻辑模型704生成。

[0326] 物理模型还包括将被用于从指定数据源提取数据的AQL(分析查询语言)查询。

[0327] 如图7A所示,loom doc 702被编译(722)以形成逻辑模型704,并且逻辑模型704被评估(724)以形成物理模型。

[0328] 图7B示出了由一些实现方式使用的文件格式710。文件格式710被用在本地和远程

执行中。注意，文件格式包含数据和流程。在一些实例中，流程可以通过完成复制/粘贴来创建数据。在这些情况下，数据成为流程的一部分。文件格式保持UI状态与下层流程分离。一些显示与应用一起被保存。布局的其他部分是用户特定的，且被存储在应用之外。文件格式可以被版本化。

[0329] 文件格式有多文档格式。在一些实现方式中，文件格式有三个主要部分，如图7B所示。在一些实现方式中，文件格式710包括编辑信息712。该部分负责使编辑体验在设备和编辑会话之间继续。该部分存储评估流程不需要的、但为用户重新构建UI所需的数据的任何片段。编辑信息712包括包含持久撤销缓冲区的撤销历史，该持久撤销缓冲区允许用户在编辑会话已经关闭和重新打开之后撤销操作。编辑信息还包括UI状态，例如什么窗格是可见的，流程节点的x/y坐标，其不会反映在流程如何运行中。当用户重新打开UI时，用户看到以前在那里的内容，使重新继续工作变得更容易。

[0330] 文件格式710包括Loom Doc 702，如上面参考图7A所述。这是所需的文件格式的唯一部分。该部分包含流程。

[0331] 文件格式710还包括本地数据714，其包含运行流程所需的任何表或本地数据。可以通过用户交互(例如将HTML表粘贴到数据准备应用中)或者当流程使用需要被上传到服务器用于评估的本地CSV文件时来创建这个数据。

[0332] 评估子系统在图7C中被示出。评估子系统提供了评估流程的可靠方式。评估子系统还提供了对早期运行的结果进行操作或者在流程的操作的顶部上分层放置操作的简单方式。此外，评估子系统提供了在运行流程的随后部分时可以重新使用由流程的一个部分产生的结果的自然方式。评估子系统还提供了对照所缓存的结果来运行的快速方式。

[0333] 存在用于评估流程的两个基本的上下文，如图7C所示。当运行(740)流程时，该过程评估该流程并将结果倾注到输出节点内。如果在调试模式中运行，则过程写出在临时数据库中的结果，其可用于导航、分析、和更快地运行部分流程。

[0334] 在导航和分析(730)时，用户审查数据集。这可以包括查看数据分布、查找脏数据等。在这些场景中，评估器通常避免运行整个流程，且替代地，对照从以前运行以前的流程创建的临时数据库直接运行更快的查询。

[0335] 这些过程利用在缓存周围的良好的元数据，以便确保智能缓存决策是可能的。

[0336] 一些实现方式包括异步子系统，如图7D所示。异步子系统向用户提供非阻止行为。如果用户正在完成不需要收回行的一系列操作，那么用户在获取行时不被阻止。异步子系统提供增量结果。用户常常不需要完整集合的数据来开始验证或试图理解结果。在这些情况下，异步子系统在它们到达时给出最佳结果。异步子系统还为正在进行的查询提供可靠的“取消”操作。

[0337] 在一些实现方式中，异步模型包括四个主要部件：

[0338] • 浏览器层。该层从它开始的异步任务中获得UUID和更新版本。然后它使用UUID用于获得更新。

[0339] • REST API。该层在线程池中开始任务。线程池中的任务在它们获得更新时更新状态服务。当浏览器层想要知道是否有更新时，它调用REST API过程以获得最新状态。

[0340] • AqlAPI。这个层被调用，好像它是有回调的同步调用一样。只有当底层请求完成时，调用才结束。但是，回调允许用已经处理的行更新状态服务。这实现了向用户提供渐进

的进展。

[0341] • 联合评估器。AqlApi调用到联合评估器内，联合评估器提供另一层异步，因为它作为新的进程运行。

[0342] 取消操作的实现取决于取消出现的位置。在浏览器层中，容易发送取消请求并接着停止对结果的轮询。在REST API中，容易向正在运行的线程发送取消事件。

[0343] 一些实现方式使得在流程已经被创建之后“重构”流程变得安全和容易。目前，ETL工具允许人们产生最初看起来相当简单的流程，但是当它们变得更大时变得不可能改变。这是因为人们很难理解其变化将如何影响流程，以及因为很难使相当多的行为爆发成与业务需求相关的部分。这当中的很多是由用户界面引起的，但是底层语言需要提供由UI所需的信息。

[0344] 所公开的实现方式使用户能够创建容易被重构的流程。这意味着用户能够容易采用操作或节点：

[0345] • 使操作四处移动，在逻辑上将它们重新排序。实现方式提供关于这些操作是否产生错误的直接反馈。例如，假设用户有具有ADD_COLUMN->FILTER的流程。用户可以将FILTER节点拖到ADD_COLUMN节点之前，除非FILTER使用被添加的列。如果FILTER使用新列，则界面立即引起错误，告诉用户问题。

[0346] • 使多个操作和节点塌缩到新节点(其可以被重复使用)内。这个新节点将具有它接受的“类型”和它返回的“类型”。例如，假设用户有包含JOIN_TABLES->ALTER_COLUMN->ALTER_COLUMN->ALTER_COLUMN的流程的片段。实现方式使用户能够将这四个步骤组合成一个节点，并给该节点分配有意义的名称，例如“FIXUP_CODES”。新节点将两个表作为输入并返回一个表。输入表的类型将包括它们联接到的列以及以在ALTER_COLUMNS中被使用结束的任何列。输出表的类型是由于操作而产生的类型。

[0347] • 从节点拆分操作。这是用户可以在即时操作期间重新组织被有组织地添加到节点的操作的场合。例如，假设用户有巨大的节点——其具有在它内的20个操作，并且用户想要将与修复医院代码相关的10个操作拆分到其自己的节点中。用户可以选择这些节点，并将它们拉出。如果在节点中存在依赖于正在被移除的操作的其他操作，则系统显示错误并建议在FixupHospitalCodes节点之后对创建新节点的修复。

[0348] • 将操作内联到现有节点中。在用户完成一些清理后，可能有属于流程的另一部分的一些工作。例如，当用户清理保险代码时，她发现医院代码的一些问题并清理它。然后，她想要将医院代码清理移动到FixupHospitalCodes节点。这使用容易的拖动/置放操作来实现。如果用户试图在它依赖于的操作之前将操作置放在流程中的一个位置上，则界面提供所提议的置放位置不起作用的即时视觉反馈。

[0349] • 改变类型，并立即查明它是否中断了流程的部分。用户可以使用流程，然后决定改变一个列的类型。即使在运行流程之前，实现方式也立即通知用户关于任何问题。

[0350] 在一些实现方式中，当用户重构流程时，系统通过识别置放目标来帮助。例如，如果用户选择节点并开始在流程窗格中拖动它，则一些实现方式显示节点可以被移动的位置(例如，通过突显)。

[0351] 所公开的数据准备应用使用具有三个方面的语言：

[0352] • 表达式语言。这是用户如何定义计算。

[0353] • 数据流程语言。这是用户如何定义流程的输入、转换、关系、和输出。这些操作直接改变数据模型。在这种语言中的类型是实体(表)和关系而不是仅仅单独列。用户不直接看到这种语言,但通过在UI中创建节点和操作来间接地使用它。示例包括联接表和移除列。

[0354] • 控制流程语言。这些是可能发生在数据流程周围但实际上不是数据流程的操作。例如,从文件共享复制压缩文档(zip),且然后将它解压缩,采用写出的TDE,且然后将它复制到共享,或者通过数据源的任意列表运行数据流程。

[0355] 这些语言是不同的,但是分层放置在彼此的顶部上。表达式语言由流程语言使用,流程语言又可以由控制流程语言使用。

[0356] 语言描述了在逻辑上从左转到右的操作的流程,如图8A所示。然而,由于流程被评估的方式,实际实现方式可以为了更好的性能而重新安排操作。例如,在数据被提取时将过滤器移动到远程数据库可以大大提高整体执行速度。

[0357] 数据流程语言是大多数人将其与数据准备应用相关联的语言,因为它描述了直接影响ETL的流程和关系。语言的个这部分具有两个主要组成部分:模型和节点/操作。这不同于标准ETL工具。不是流程直接对数据进行操作(例如,使实际行从“过滤”操作流动到“添加字段”操作),所公开的流程定义指定它想要创建什么的逻辑模型和定义其想要如何具体化逻辑模型的物理模型。这个抽象在优化方面提供了更多的余地。

[0358] 模型是基本名词。它们描述了方案和正在被操作于的数据的关系。如上面所提到的,有逻辑模型和单独的物理模型。逻辑模型提供在给定点处关于流程的基本“类型”。其描述了描述正被转换的数据的字段、实体、和关系。这个模型包括诸如集合和组的东西。逻辑模型指定需要什么,但没有任何具体化。该模型的核心部分是:

[0359] • 字段:这些是将在输出中的数据字段中被转变(或者有助于这样做的计算)的实际字段。每个字段都与实体和表达式相关联。字段不一定都需要是可见的。有3种类型的字段:物理字段、所计算的字段和临时字段。物理字段被具体化为结果数据集。这些可以是正确的字段或计算。所计算的字段被写到结果TDS作为所计算的字段,因此它们将从不被具体化。临时字段被写入以更好地将对物理字段的计算作为因素计入。它们没有以任何方式被写出。如果临时字段由所计算的字段引用,则语言将发出警告并将该字段处理为所计算的字段。

[0360] • 实体:这些是描述逻辑模型的命名空间的对象。实体由进入的表的方案创建的,或者可以由通过关系关联在一起的实体的集合组成。

[0361] • 关系:这些是描述不同的实体如何与彼此相关的对象。它们可用于将多个实体组合成一个新的复合实体。

[0362] • 约束:这些描述了添加到实体的约束。约束包括实际上限制实体的结果的过滤器。一些约束被实施。所实施的约束由上游源保证,例如唯一约束或非空约束。一些约束被断言。这些是被认为是真的约束。每当发现数据违反此约束时,都以某种方式通知用户。

[0363] 流程可以包括在逻辑模型中的一个或更多个分叉。将流程分叉对每个分叉使用相同的逻辑模型。然而,对于分叉的每一侧都有在覆盖层下面的新实体。这些实体基本上传递到原始实体,除非列在它们上面被投影或移除。

[0364] 创建新实体的一个原因是保持对实体间的任何关系的跟踪。当字段中没有一个改变时,这些关系将继续是有效的。然而,如果字段被修改,则它将是在新实体上的新字段,因

此该关系将被已知不再起作用。

[0365] 一些实现方式允许钉住节点或操作。流程描述了一组操作的逻辑排序，但是系统可以通过使物理排序变得不同来自由地优化处理。然而，用户可能想要确保逻辑和物理排序是完全相同的。在这些情况下，用户可以“钉住”节点。当节点被钉住时，系统确保在钉住之前的操作物理上发生在钉住之后的操作之前。在一些情况下，这将导致某种形式的具体化。然而，只要有可能，系统就经历此。

[0366] 物理模型描述了逻辑模型在特定点处的具体化。每个物理模型都具有对用来生成它的逻辑模型的反向引用。物理模型对于缓存、增量流程运行、和加载操作是重要的。物理模型包括对包含流程的结果的任何文件的引用，其是到目前为止描述逻辑模型的唯一散列。物理模型还指定为运行生成的TDS(表格数据源)和AQL(分析查询语言)。

[0367] 节点和操作是基本动词。模型中的节点包括定义数据如何被成形、计算、和过滤的操作。为了与UI语言保持一致，术语“操作”指在流程中做某事的“节点”之一。节点用于指包含操作并映射到用户在UI中的流程窗格中看到的内容的容器。每个专用节点/操作都具有与它相关联的描述其将如何操作的属性。

[0368] 有四种基本类型的节点：输入操作、转换操作、输出操作、和容器节点。输入操作从某个外部源创建逻辑模型。示例包括导入CSV的操作。输入操作代表在ETL(提取)中的E。转换操作将逻辑模型转换成新的逻辑模型。转换操作接受逻辑模型并返回新的逻辑模型。转换节点代表在ETL(转换)中的T。示例是向现有逻辑模型添加列的投影操作。输出操作接受逻辑模型，并将它具体化为某个其他数据暂存器。例如，采用逻辑模型并将它的结果具体化为TDE的操作。这些操作代表在ETL(加载)中的L。容器节点是关于组成如何在流程间进行的基本抽象，且也提供当节点在UI中被显示时应当被显示的东西的抽象。

[0369] 如图8B所示，类型系统由三个主要概念组成：

[0370] • 操作是原子动作，每个原子动作具有输入和输出以及所需的一组字段。

[0371] • 所需字段是操作所需的字段。可以通过评估对空类型环境的操作、然后收集任何被“假定”的字段来确定所需字段。

[0372] • 类型环境是确定如何查找流程中的给定点的类型的结构。在流程图中的每个“边”代表一个类型环境。

[0373] 在两个阶段中执行类型检查。在类型环境创建阶段中，系统在流程的方向上穿过流程运行。系统弄清楚每个节点需要什么类型以及它们输出什么类型的环境。如果流程是抽象的(例如，它实际上没有连接到任何输入节点)，则空类型环境被使用。类型细化是第二阶段。在这个阶段中，系统从第一阶段起采用类型环境，并使它们“向后”流动以查看在类型环境创建中发生的任何类型变窄是否造成了类型冲突。在此阶段中，系统还为整个子流程创建一组所需字段。

[0374] 每个操作具有与它关联的类型环境。该环境包含可访问的所有字段及其类型。如图8C所示，类型环境有五个属性。

[0375] 环境可以是“开放的”或“封闭的”。当环境是开放的时，它假设可以有它不知道的字段。在这种情况下，任何不知道的字段都将被假定为任何类型。这些字段将被添加到AssumedTypes字段。当环境是封闭的时，它假设它知道所有的字段，所以任何不知道的字段都是失效的(failure)。

[0376] 所有已知类型都在Types成员中。这是从字段名称到它们的类型的映射。类型可以是另一种类型环境,或它可以是字段。字段是最基本的类型。

[0377] 每个字段由两个部分组成。basicTypes是描述字段的类型的可能集合的一组类型。如果这个集合只有一个元素,那么我们知道它有什么类型。如果集合为空,则存在类型错误。如果集合有多于一个元素,那么有几种可能的类型。如果需要,系统可以解析并进一步完成类型变窄。derivedFrom是对参与导出这个属性的字段的引用。

[0378] 在范围中的每个字段具有潜在的一组类型。每种类型可以是布尔、字符串、整数、小数、日期、DateTime、Double、几何形状、和持续时间的任何组合。还有“任何”类型,其可以是任何东西的类型的速记。

[0379] 在开放式类型环境的情况下,可能存在已知不存在的字段。例如,在“removeField”操作之后,系统可能不知道在类型环境(因为它是开放的)中的所有字段,但是系统确实知道刚刚被移除的字段不在那里。类型环境属性“NotPresent”用于识别这样的字段。

[0380] AssumedTypes属性是被添加的类型的列表,因为它们被引用而不是被定义。例如,如果有在开放式类型环境中评估的表达式[A]+[B],则系统假设有两个字段:A和B。AssumedTypes属性允许系统保持对什么以这种方式被添加的跟踪。这些字段可以被积累用于进一步的类型筛选以及用于能够确定容器的所需字段。

[0381] “上一个”类型环境属性是对类型环境的引用,这个属性从类型环境导出。它在整个流程中查找类型不一致的向后遍历期间用于类型细化阶段。

[0382] 也可以组成类型环境。这发生在采用多个输入的操作中。当类型环境被合并时,它将每个类型环境映射到它的类型集合中的一个值。然后将进一步的类型解析指派给各个类型环境。然后常常通过某种方式“拉平”类型环境以创建只具有作为类型的字段的新类型环境由操作符将这个类型环境转换为输出类型环境。

[0383] 这由联接和联合操作符使用,以便在它们自己的表达式中精确地使用来自不同环境的所有字段,并且有将环境映射到输出类型环境的方式。

[0384] 由输入节点创建的类型环境是由它正在读取的数据源返回的方案。对于SQL数据库,这将是它正在提取的表、查询、所存储的过程、或视图的方案。对于CSV文件,这将是从文件中拉出的方案,无论用户使什么类型与列相关。每一列及其类型都转变成字段/类型映射。此外,类型环境被标记为封闭的。

[0385] 转换节点的类型环境是它的输入的环境。如果它具有多个输入,则它们将被合并以创建操作的类型环境。输出是基于操作符的单个类型环境。图8J-1至图8J-3中的表列出了许多操作。

[0386] 容器节点可具有多个输入,因此它的类型环境将是复合类型环境,其将适当的子类型环境按规定路线发送到适当的输出节点。当容器被拉出以被重新使用时,它为每个输入解析空类型环境以确定它的相依性。

[0387] 在一些实现方式中,容器节点是能够有多于一个输出的唯一类型的节点。在这种情况下,它可能有多个输出类型环境。这不应该与使输出分支(其可能对任何节点发生)混淆。然而,在使输出分支的情况下,每个输出边具有相同的类型环境。

[0388] 存在几种情况,其中当系统发现关于字段的冲突的需求时,类型错误被标记。未解

析的字段在此阶段不被视为错误,因为此阶段可能出现在具有无界输入的流程上。但是,如果用户试图运行流程,未解析的变量将是被报告的问题。

[0389] 许多输入都有对类型的特定定义。例如,特定定义包括使用CHAR(10)而不是VARCAR(2000),字段使用什么排序规则,或小数类型具有什么标度和精度。一些实现方式并不将这些作为类型系统的一部分来跟踪,而是将它们作为运行时间信息的一部分来跟踪。

[0390] UI和中间层能够查明运行时间类型。该信息能够通过常规回调流动,以及嵌在tempdb的类型中(例如,在系统从缓存的运行填充的情况下)。UI向用户显示更具体的已知类型,但不基于它们来进行类型检查。这使OutputNodes(其使用更具体的类型)的创建成为可能,同时允许系统的其余部分使用更简化的类型。

[0391] 图8D示出了基于具有所有已知的数据类型的流程的简单类型检查。图8E示出了具有完全已知的类型的简单类型失效。图8F示出了部分流程的简单类型环境计算。图8G示出了打包容器节点的类型。图8H示出了更复杂的类型环境场景。图8I示出了重新使用更复杂的类型环境场景。

[0392] 一些实现方式推断数据类型,并使用推断出的数据类型用于优化数据流程或使数据流程生效。这对于基于文本的数据源(例如XLS或CSV文件)是特别有用的。基于数据元素如何在流程中较晚地被使用,有时可以推断数据类型,推断出的数据类型可以在流程中较早地被使用。在一些实现方式中,作为文本字符串接收的数据元素可以在从数据源检索之后立即被分派为适当的数据类型。在一些实例中,推断数据类型是递归的。也就是说,通过推断一个数据元素的数据类型,系统能够推断一个或更多个附加数据元素的数据类型。在一些实例中,数据类型推断能够在不确定确切的数据类型(例如,确定数据元素是数字,但是不能确定它是整数还是浮点数)的情况下排除一个或更多个数据类型。

[0393] 大多数类型错误在类型检查阶段中被发现。这正好在计算初始类型环境之后到来,并基于关于每种类型知道什么来细化范围。

[0394] 这个阶段开始于所有终端类型环境。对于每个类型环境,系统向后回到其先前的环境。该过程向后退回,直到它到达封闭的环境或没有先前环境的环境为止。然后,该过程检查在每个环境中的类型以确定任何字段是否在类型方面不同。如果是这样并且它们的交集为空,则该过程引起类型错误。如果任何字段在类型方面不同并且交集不为空,则流程将类型设置为交集和任何受影响的节点,其使它们的类型环境被重新计算。此外,任何被“假定”的类型都被添加到先前的类型环境,并且类型环境被重新计算。

[0395] 存在被跟踪的几个细微之处。首先,字段名称本身不一定是唯一的,因为用户可以用具有不同类型的某物盖写字段。因此,该过程使用从一个类型回到用于生成它的类型的指针,从而避免被解析到在图中的不同部分处的相同名称的不相关事物欺骗。例如,假设字段A具有类型[int,decimal],但是然后有将字段A变成字符串的进行投影的节点。回到A的较早版本且说该类型不起作用将是个错误。相反,此时的退回将不使A退回而经过addField操作。

[0396] 类型检查一次使一个变量变窄。在上面的步骤中,在重新计算已知变量之前,类型检查只应用于一个变量。这在存在具有多个签名的重载函数(例如Function1(string,int)和Function1(int,string))的情况下是安全的。假设这被称为Function1([A],[B])。该过程确定类型是A:[String,int]和B:[String,,int]。然而,将类型解析为A:[String]和B:

[String]是无效的,因为如果A是字符串,则B需要是int。一些实现方式通过在每个类型变窄之后重新运行类型环境计算来处理这种类型的相依性。

[0397] 一些实现方式通过只在实际上具有包括变窄的变量的所需字段的节点上进行工作来优化要做什么工作。在这里有少许细微之处,即变窄的A可能以也使B变窄结束。采取上面的Function1例子。在这些情况下,系统需要知道B何时改变且也检查它的变窄。

[0398] 当看操作符将如何起作用时,最好从在这里被识别为“打开”、“多输入”、“输入类型”和“结果类型”的四个主要属性方面来考虑它们。

[0399] 当操作流经列时,它被指定为打开。例如,“过滤”是打开操作,因为在输入中的任何列也将在输出中。Group by(按…分组)不是打开的,因为任何未聚合或分组的列都不在结果类型中。

[0400] “多输入”属性指定此操作是否采用多个输入实体。例如,联接是多输入的,因为它需要两个实体并使它们成为一个实体。联合是多输入的另一种操作。

[0401] “输入类型”属性指定节点所需的类型。对于多输入操作,这是复合类型,其中每个输入包含它自己的类型。

[0402] “结果类型”属性指定由此操作产生的输出类型。

[0403] 图8J-1、图8J-2、和图8J-3中的表指示许多最常用操作符的属性。

[0404] 在许多情况下,当需要改变时,流程随着时间的过去而被创建。当流程通过有机进化而增长时,其可以变得巨大而复杂。有时,用户需要修改流程以处理变化的需要或者重新组织流程,使得它更容易理解。在许多ETL工具中,流程的这样的重构是困难的或不可能的。

[0405] 在这里的实现方式不仅使重构成为可能,还帮助用户这么做。在技术层面上,系统可以获得任何节点(或节点序列)的RequireFields,且然后在具有可适应该节点的类型环境的任何点处点亮置放目标。

[0406] 另一个场景涉及重新使用在流程中的现有节点。例如,假设用户想要采用一系列操作并建立自定义节点。自定义节点操作来“标准化保险代码”。用户可以创建容器节点,其具有在其中的多个操作。然后,系统可以为它计算所需字段。用户可以使用保存命令或者将容器节点拖到左手侧窗格312来保存将来要使用的节点。现在,当一个人从在左手侧窗格中的调色板中选择节点时,系统点亮在流程中的置放目标,且用户可以将节点置放到置放目标之一上(例如,就像上面的重构示例一样)。

[0407] ETL可能变得混乱,所以这里的实现方式支持各种系统扩展。扩展包括:

[0408] • 用户定义的流程操作。用户可以用输入、输出、和转换操作扩展数据流程。这些操作可以使用自定义逻辑或分析来修改行的内容。

[0409] • 控制流程脚本。用户可以构建进行非数据流程操作的脚本,例如从共享下载文件、解压缩文件、为目录中的每个文件运行流程,等等。

[0410] • 命令行脚本。用户可以从命令行运行它们的流程。

[0411] 这里的实现方式采用从人们如何使用所提供的可扩展性方面来说是与语言无关的方法。

[0412] 第一个扩展允许用户构建配合到流程内的自定义节点。对创建扩展节点有两个部分:

[0413] • 定义输出的类型。例如,“进来的每件事物以及新列‘foo’”。

[0414] • 提供脚本或可执行程序以实际上运行转换。

[0415] 一些实现方式定义了允许用户定义的扩展的两种节点类型。“ScriptNode”是一个节点，其中用户可以写脚本以操纵行，并将它们传递回。该系统提供API函数。然后，用户可以将转换(或输入或输出)节点写为脚本(例如，用Python或Javascript)。“ShellNode”是一个节点，其中用户可以定义要运行的可执行程序并将行输送到可执行程序内。然后，可执行程序将结果写出到stdout，将错误写到stderr，并在它被完成后退出。

[0416] 当用户为流程创建扩展时，内部处理更加复杂。不是将每件事物都编译成一条AQL语句，该过程将评估分成围绕自定义节点的两个部分，并将来自第一部分的结果引导到节点内。这在图8K和8L中示出，其中用户定义的节点850将流程分成两个部分。在流程评估期间，用户定义的脚本节点852从流程的第一部分接收数据，并为流程的第二部分提供输出。

[0417] 除了以某种方式修改流动的数据的定制之外，用户还可以写控制流程如何运行的脚本。例如，假设用户需要从共享(其具有每天被公布给它的电子表格)中拉出数据。已定义的流程已经知道如何处理CSV或Excel文件。用户可以写控制脚本，其在远程共享上迭代，拉下相关文件，然后在这些文件上运行。

[0418] 存在许多常见的操作，例如模式联合，用户可以将这些操作添加到数据流程节点中。然而，当技术继续发展时，总是存在获取或存储不被系统定义的数据流程节点适应的数据的方式。这些是控制流程脚本是可适用的情况。这些脚本作为流程的一部分运行。

[0419] 如上面所提到的，也可以从命令行调用流程。这将允许人们将脚本嵌在其他过程或隔夜工作中。

[0420] 实现方式具有提供许多有用的特征的流程评估过程。这些特征包括：

[0421] • 自始至终一直运行流程。

[0422] • 使流程分裂，以便确保顺序或“钉住”操作。

[0423] • 使流程分裂，以允许第三方代码运行。

[0424] • 运行流程，但不是一直回到上游数据源，而是从先前运行的流程的输出运行它。

[0425] • 预运行流程的一些部分以填充本地高速缓存。

[0426] 评估过程基于在逻辑模型和物理模型之间的相互作用来工作。任何具体化的物理模型都可以是流程的起点。然而，语言运行时间提供抽象以定义流程的哪些子部分要运行。一般来说，运行时间不确定何时运行子流程，与何时运行全流程。这由其他部件确定。

[0427] 图8M示出了运行整个流程是以在输入和输出节点处的隐含物理模型开始的。图8N示出了运行部分流程具体化具有结果的物理模型。图8O示出了基于先前结果来运行流程的一部分。

[0428] 虽然物理模型可以被重新排序以优化处理，但是逻辑模型对用户隐藏这些细节，因为它们通常是不相关的。流程评估器使节点看起来像以它们在流程中显示的顺序而被评估。如果节点被钉住，它将实际上使流程在那里具体化，保证在左边的部分在右边的部分之前评估。在分叉的流程中，公共预流程只运行一次。这个过程是幂等的，意味着输入操作符可以由于失效而被再次调用并且不失败。注意，没有下面的要求：回来的数据与它第一次的状态完全相同(即，当在上游数据源中的数据在第一次和第二次尝试之间改变时)。

[0429] 转换操作符的执行没有副作用。另一方面，提取操作符通常有副作用。直到流程的下一次运行为止，在流程中在它之前修改数据源的任何操作未被看到。加载操作符通常没

有副作用,但存在例外。事实上,一些加载操作符需要副作用。例如,从共享拉下文件并解压缩它们被认为是副作用。

[0430] 一些实现方式关于列名称是大小写敏感的,但一些实现方式不是。一些实现方式提供了用户可配置的参数以指定列名称是否是大小写敏感的。

[0431] 一般来说,所缓存的对象的视图总是及时“前进”。

[0432] 图8P和图8Q示出了评估具有钉住的节点860的流程。在流程评估期间,首先执行在钉住之前的节点以创建用户节点结果862,并且在流程的后面的部分中使用用户节点结果862。注意,钉住并不阻止重新安排在每个部分内的执行。钉住的节点实际上是逻辑检查点。

[0433] 除了由用户钉住的节点之外,一些节点根据它们执行的操作而固有地被钉住。例如,如果节点对自定义代码(例如,Java进程)进行调出,则逻辑操作不能跨越节点移动。自定义代码是“黑盒”,因此它的输入和输出必须是定义明确的。

[0434] 在一些实例中,四处移动操作可以提高性能,但产生降低一致性的副作用。在一些情况下,用户可以使用钉住作为保证一致性的方式,但以性能为代价。

[0435] 如上面所提到的,用户可以直接在数据网格315中编辑数据值。在一些实例中,系统根据用户的编辑推断出一般规则。例如,用户可以将字符串“19”添加到数据值“75”以创建“1975”。基于数据和用户编辑,系统可以推断下列的规则:用户想要填写字符串以形成缺少世纪的两字符年份的4字符年份。在一些实例中,推断仅基于改变本身(例如,前置“19”),但是在其他实例中,系统还使推断基于列中的数据(例如,该列具有在范围“74”-“99”中的值)。在一些实现方式中,在将规则应用于列中的其他数据值之前,用户被提示确认该规则。在一些实现方式中,用户也可以选择将相同的规则应用于其他列。

[0436] 用户对数据值的编辑可以包括如刚刚所述的添加到当前数据值、移除字符串的一部分、用另一个子串替换某个子串、或者这些的任何组合。例如,可以用多种格式(例如(XXX)YYY-ZZZZ)指定电话号码。用户可以编辑一个特定的数据值以移除括号和破折号并添加点以创建XXX.YYY.ZZZZ。系统可以基于编辑数据值的单个实例来推断规则,并将规则应用于整个列。

[0437] 作为另一个示例,数字字段也可以使规则被推断出。例如,如果用户用零替换负值,则系统可以推断出所有负值都应该归零。

[0438] 在一些实现方式中,根据共享的规则当在数据网格315的单个列中编辑两个或更多个数据值时规则被推断出。

[0439] 图9示出了逻辑流程323可以根据操作是被指定为命令式的还是声明性的而以不同的方式被执行。在这个流程中,有两个输入数据集:数据集A902和数据集B 904。在这个流程中,直接从数据源中检索这些数据集。根据该流程,使用联接操作906来组合两个数据集902和904,以产生中间数据集。在联接操作之后,流程323应用过滤器908,过滤器908创建另一中间数据集,其具有比由联接操作906创建的第一中间数据集更少的行。

[0440] 如果在该流程中的所有节点都被指定为命令式的,那么执行该流程确切地进行节点指定的事情:数据集902和904从它们的数据源中被检索,这些数据集在本地被组合,且然后行的数量通过过滤器减少。

[0441] 如果在这个流程中的节点被指定为具有声明性执行(其通常是默认的),则执行优化器可以重新组织物理流程。在第一种场景中,假设数据集902和904来自不同的数据源,并

且过滤器908仅适用于在数据集A902中的字段。在这种情况下，可以将过滤器推回到检索数据集A902的查询，因而减少被检索和处理的数据的数量。这在数据集A902从远程服务器被检索和/或过滤器删除大量行时是特别有用的。

[0442] 在第二种场景中，再次假设声明性执行，但是假设数据集A902和数据集B 902都从同一数据源被检索(例如，这些数据集中的每一个对应于在同一数据库服务器上的同一数据库中的表)。在这种情况下，流优化器可以将整个执行推回到远程服务器，构建联接两个表并包括应用由过滤器节点908指定的过滤操作的WHERE子句的单个SQL查询。这个执行灵活性可以大大减少整体执行时间。

[0443] 用户构建数据流程并随着时间的推移改变数据流程，因此一些实现方式提供了增量流程执行。每个节点的中间结果被保存，并仅在必要时被重新计算。

[0444] 为了确定节点是否需要被重新计算，一些实现方式使用流程散列和矢量时钟。在流程323中的每个节点都具有它自己的流称散列和矢量时钟。

[0445] 给定节点的流称散列是识别在流程中直到并包括给定节点的所有操作的散列值。如果流程定义的任何方面改变(例如，添加节点、移除节点、或改变在任何节点处的操作)，则散列将是不同的。注意，流程散列只跟踪流程定义，且不看底层数据。

[0446] 矢量时钟跟踪由节点使用的数据的版本。它是矢量，因为给定节点可能使用来自多个源的数据。数据源包括由直到并包括给定节点的任何节点访问的任何数据源。矢量包括每个数据源的单调递增的版本值。在一些情况下，单调递增的值是来自数据源的时间戳。注意，该值对应于数据源，而不是当数据由在流程中的任何节点处理时。在一些情况下，数据源可以提供单调递增的版本值(例如，数据源具有编辑时间戳)。如果数据源不能提供像这样的版本号，数据准备应用250计算替代值(例如，查询何时被发送到数据源或从数据源被检索)。一般来说，优选地有指示数据何时最后一次改变的版本值，而不是指示数据准备应用何时最后一次查询数据的值。

[0447] 通过使用流程散列和矢量时钟，数据准备应用250限制需要被重新计算的节点的数量。

[0448] 在本文的发明的描述中使用的术语仅仅为了描述特定实现方式的目的，且并没有被规定为限制本发明。如在本发明和所附的权利要求的描述中所使用的，单数形式“一(a)”、“一(an)”、和“所述(the)”意欲也包括复数形式，除非上下文另有明确指示。还将理解的是，如在本文中所使用的术语“和/或”指的是并包括一个或更多个相关联的所列出的项目的任何和所有的可能的组合。将要进一步理解的是，术语“包括(comprises)”和/或“包括(comprising)”当在本说明书中被使用时，指定所陈述的特征、步骤、操作、元素、和/或部件的存在，但不排除一个或更多个其它特征、步骤、操作、元素、部件、和/或其组的存在或添加。

[0449] 为了解释的目的，前面的描述已经参考具体实现方式进行了描述。然而，上面的说明性讨论并非被规定为详尽的或将本发明限制到所公开的精确形式。鉴于上面的教导，许多修改和变形是可能的。实现方式被选择和描述，以便最佳地解释本发明的原理及其实际应用，以从而使本领域中的技术人员能够在有适于所设想的特定用途的各种修改的情况下最佳地利用本发明和各种实现方式。

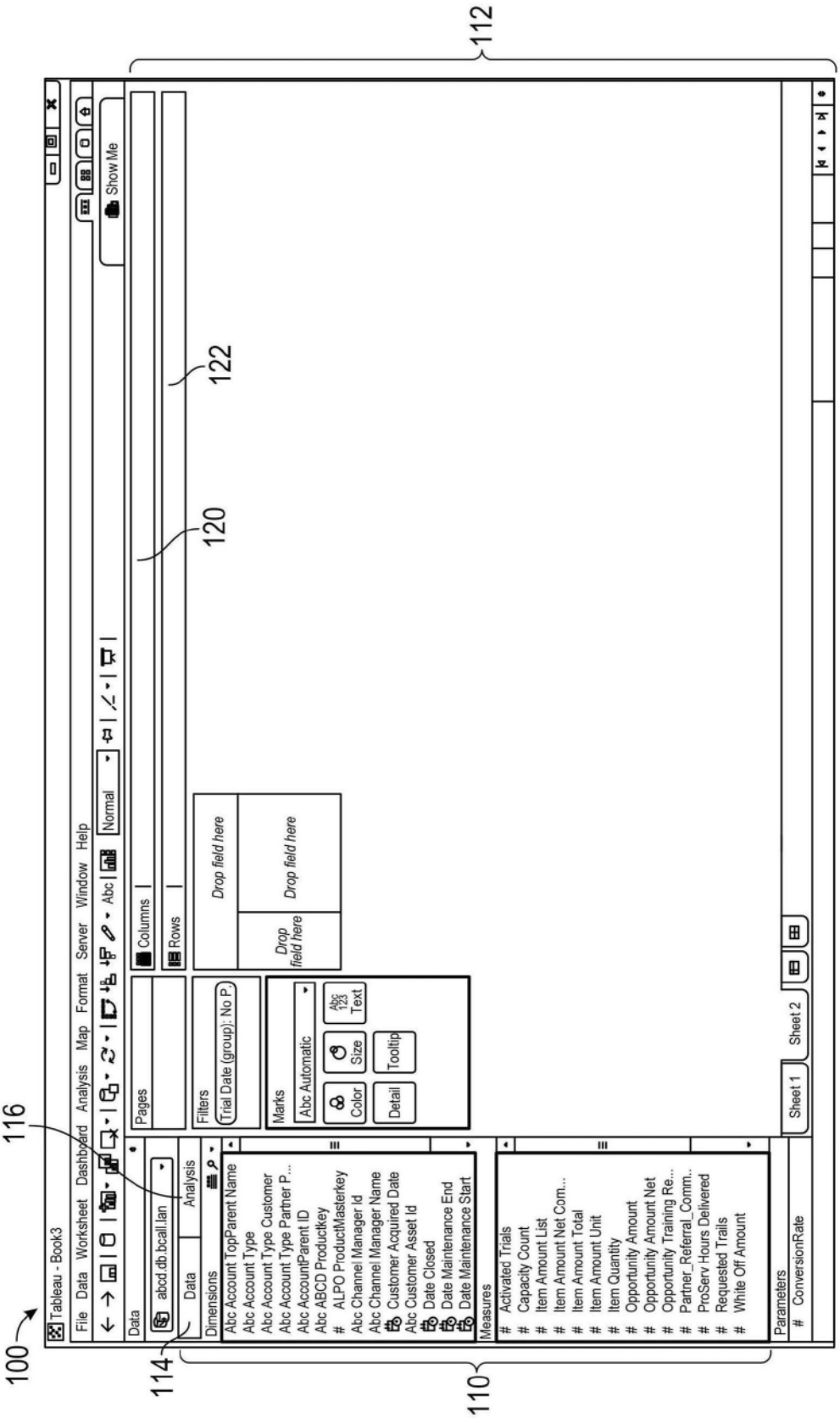

100
116
114
110
112
120
122
Tableau - Book3
File Data Worksheet Dashboard Analysis Map Format Server Window Help
Show Me
Data
abcd.db.bcall.lan
Data
Analysis
Dimensions
Abc Account TopParent Name
Abc Account Type
Abc Account Type Customer
Abc Account Type Partner P...
Abc AccountParent ID
Abc ABCD Productkey
ALPO ProductMasterkey
Abc Channel Manager Id
Abc Channel Manager Name
Customer Acquired Date
Abc Customer Asset Id
Date Closed
Date Maintenance End
Date Maintenance Start
Measures
Activated Trials
Capacity Count
Item Amount List
Item Amount Net Com...
Item Amount Total
Item Amount Unit
Item Quantity
Opportunity Amount
Opportunity Amount Net
Opportunity Training Re...
Partner_Referral_Comm...
ProServ Hours Delivered
Requested Trails
White Off Amount
Parameters
ConversionRate
Pages
Filters
Trial Date (group): No P...
Marks
Abc Automatic
Color
Size
Abc 123 Text
Detail
Tooltip
Columns
Rows
Drop field here
Drop field here
Drop field here
Sheet 1
Sheet 2

图1

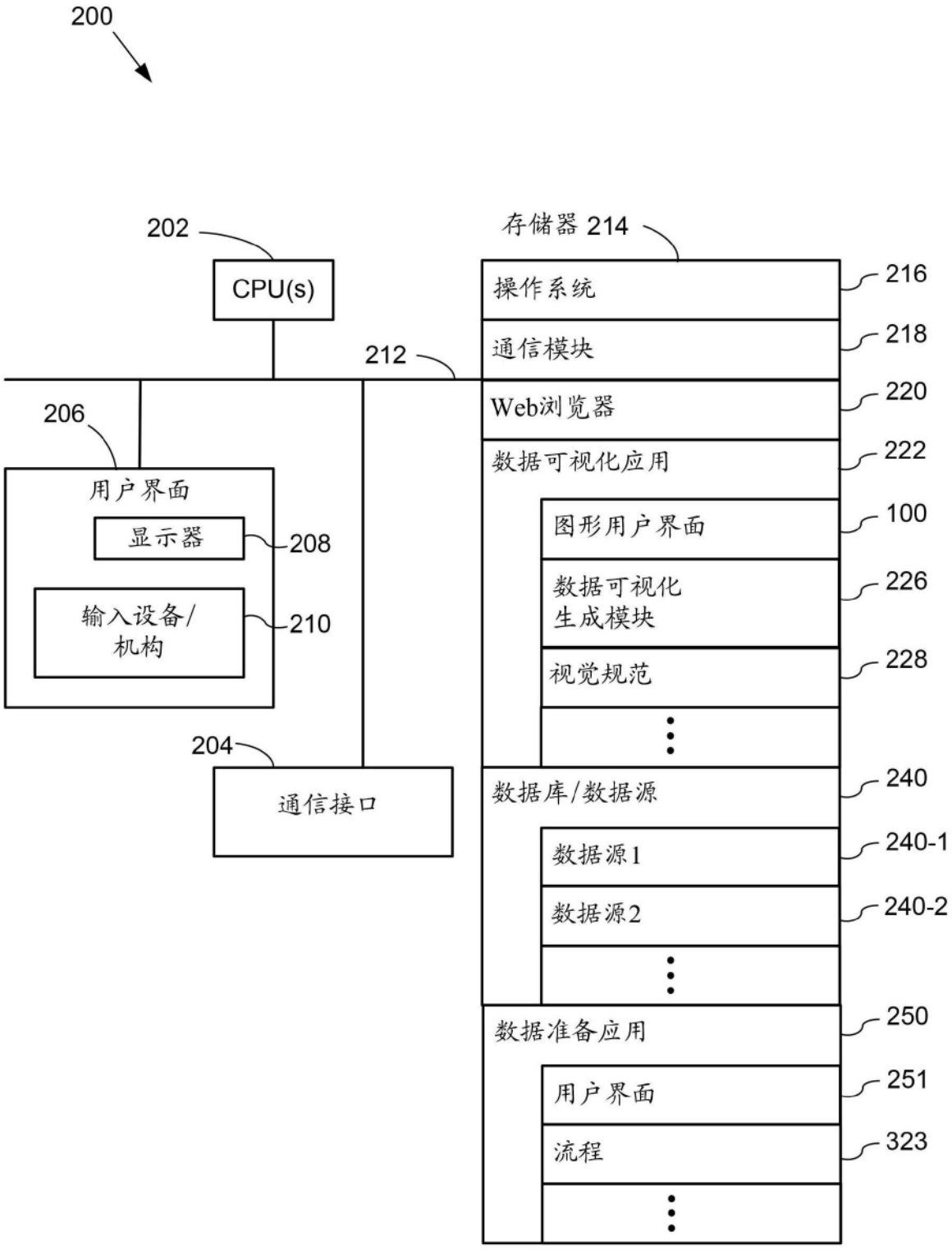

200
202
CPU(s)
存储器 214
操作系统
216
通信模块
218
212
Web浏览器
220
206
数据可视化应用
222
用户界面
显示器
208
图形用户界面
100
数据可视化
生成模块
226
输入设备/
机构
210
视觉规范
228
204
通信接口
数据库/数据源
240
数据源1
240-1
数据源2
240-2
数据准备应用
250
用户界面
251
流程
323

图2

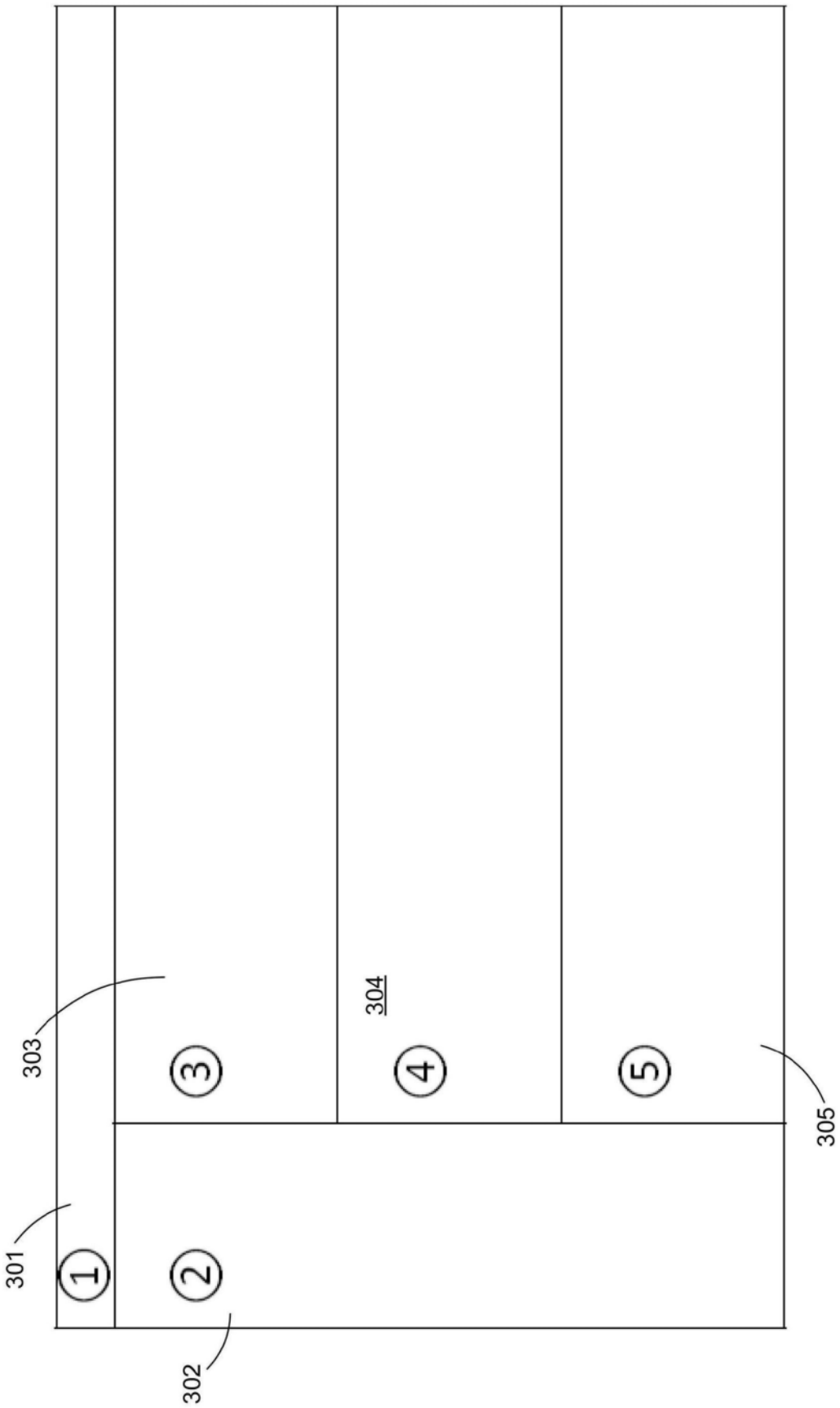


图3A

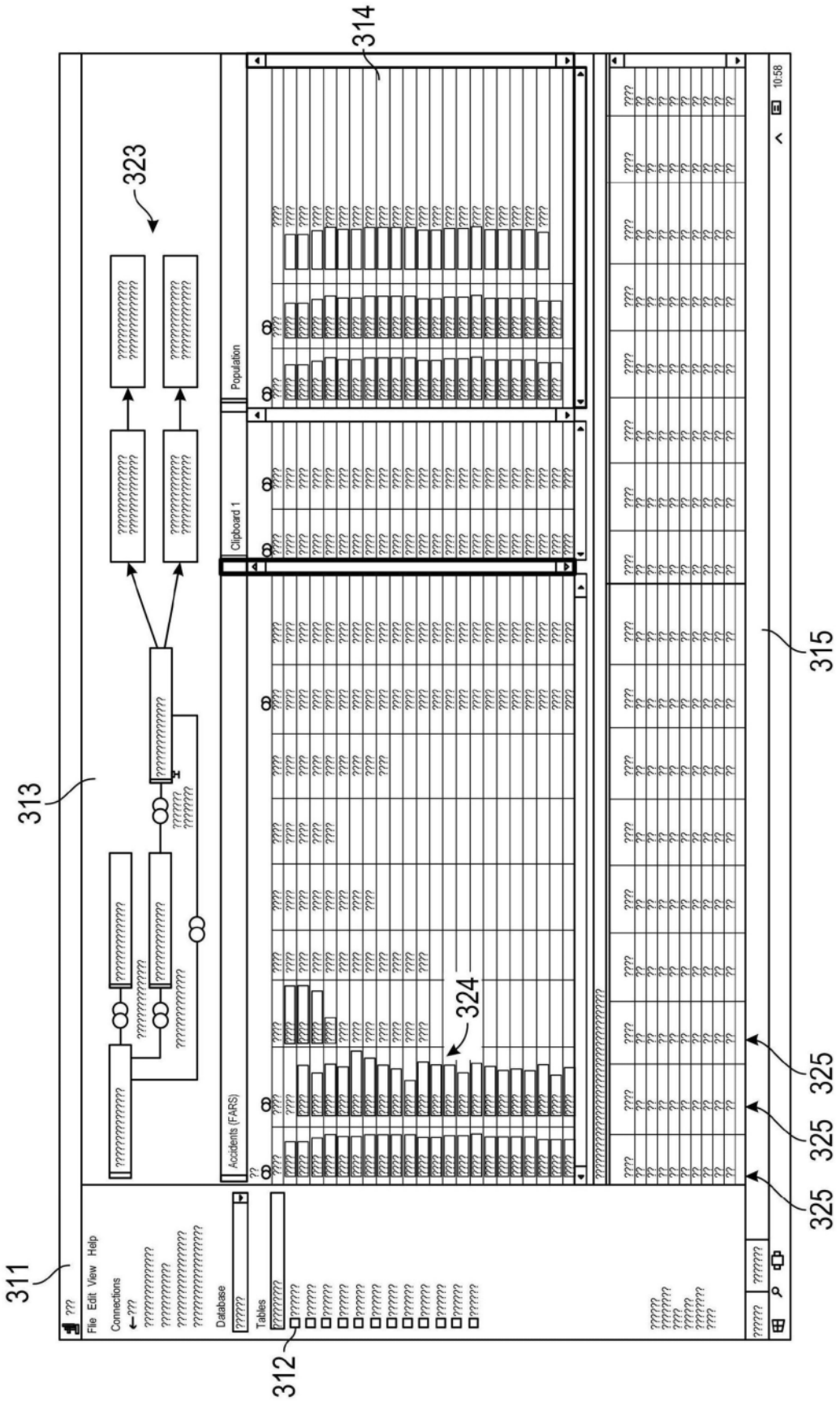

311
312
313
314
315
323
324
325
325
325
File Edit View Help
Connections
Database
Tables
Accidents (FARS)
Clipboard 1
Population
10:58

图3B

概念/抽象	总是相关	总是相关，但内容基于节点来改变	基于节点而相关
流程图	XXXXXX		
配置文件窗格		XXXXXX	
数据窗格		XXXXXX	
节点特定窗格			XXXXXX
数据源调色板/选择器	XXXXXX		
操作“调色板”	XXXXXX		
其它流程“调色板”	XXXXXX		
节点内部		XXXXXX	
项目模型	XXXXXX		
操作状态	XXXXXX		

图3C

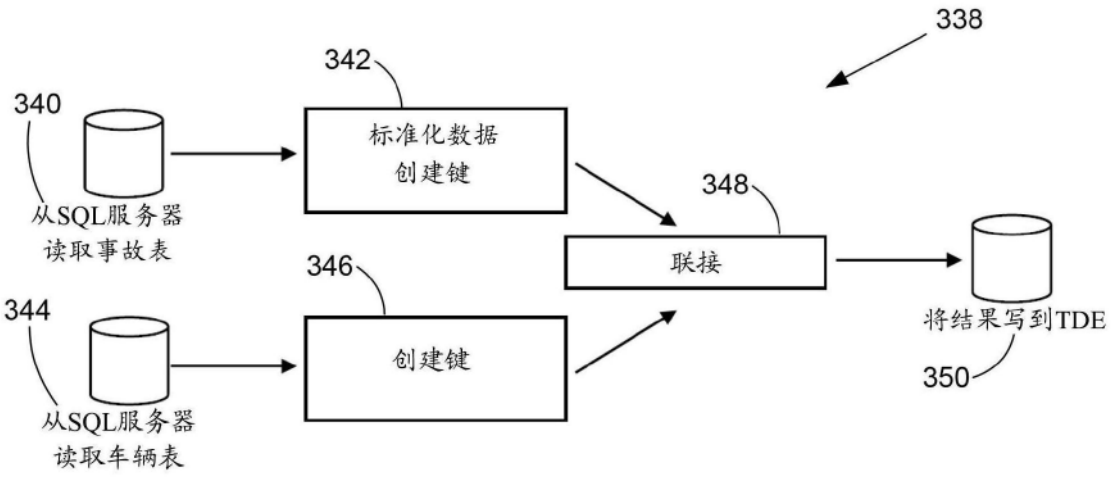


图3D

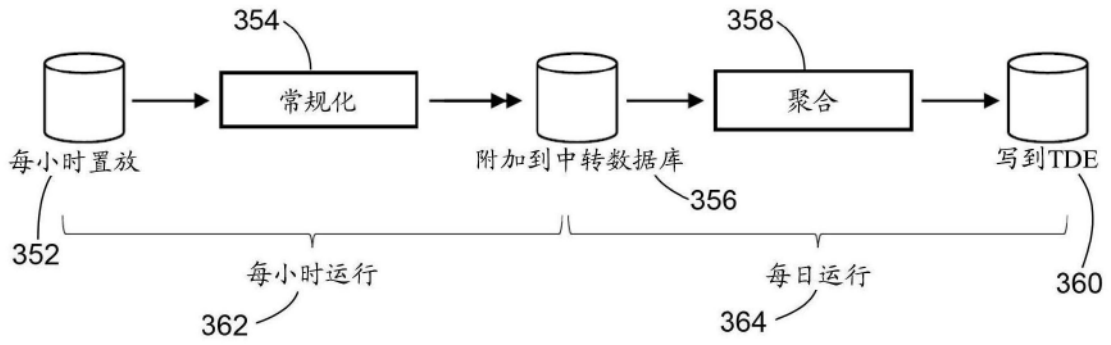


图3E

图4A

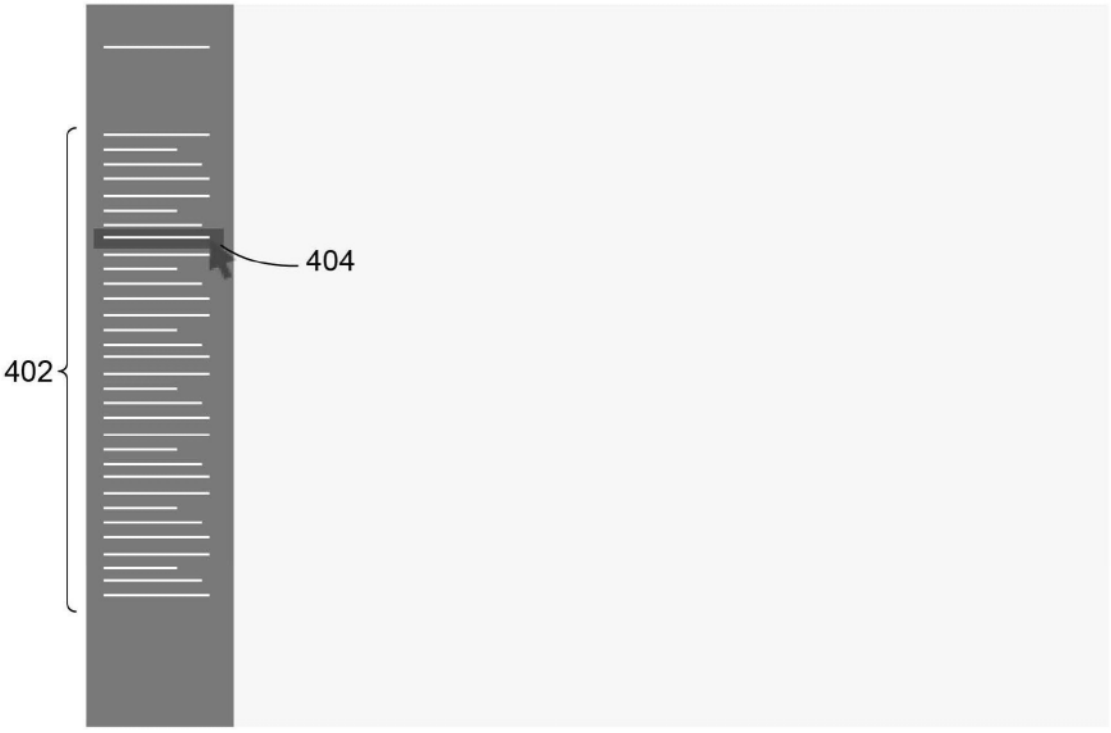


图4B

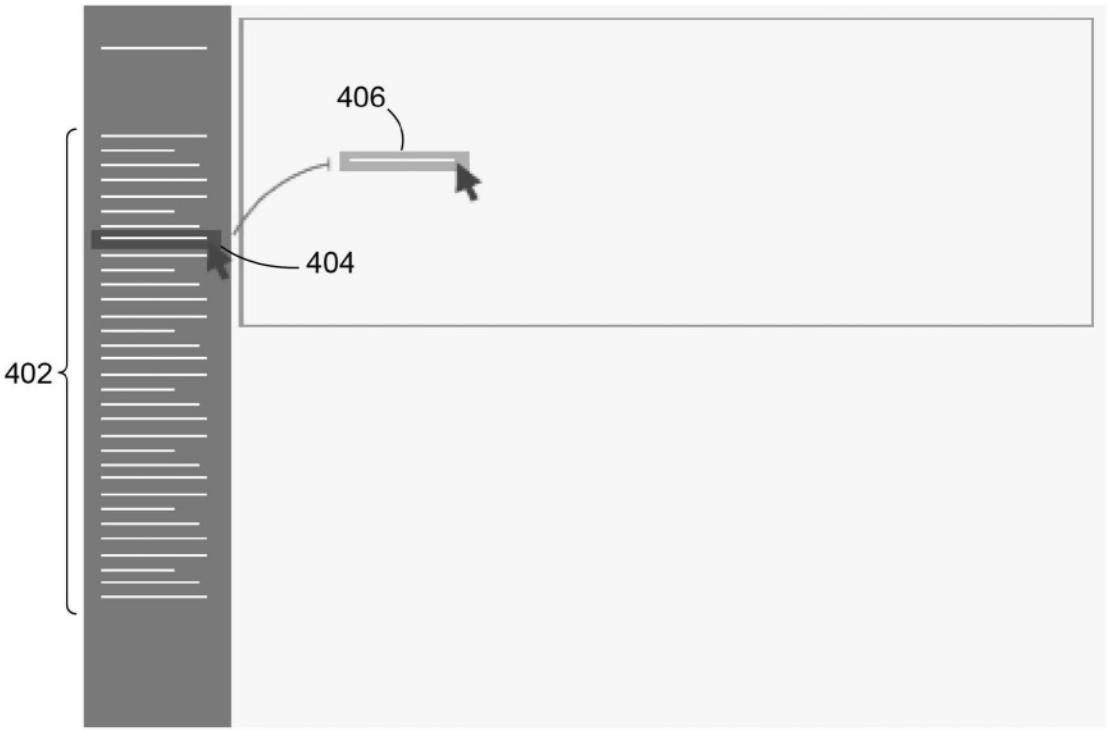


图4C

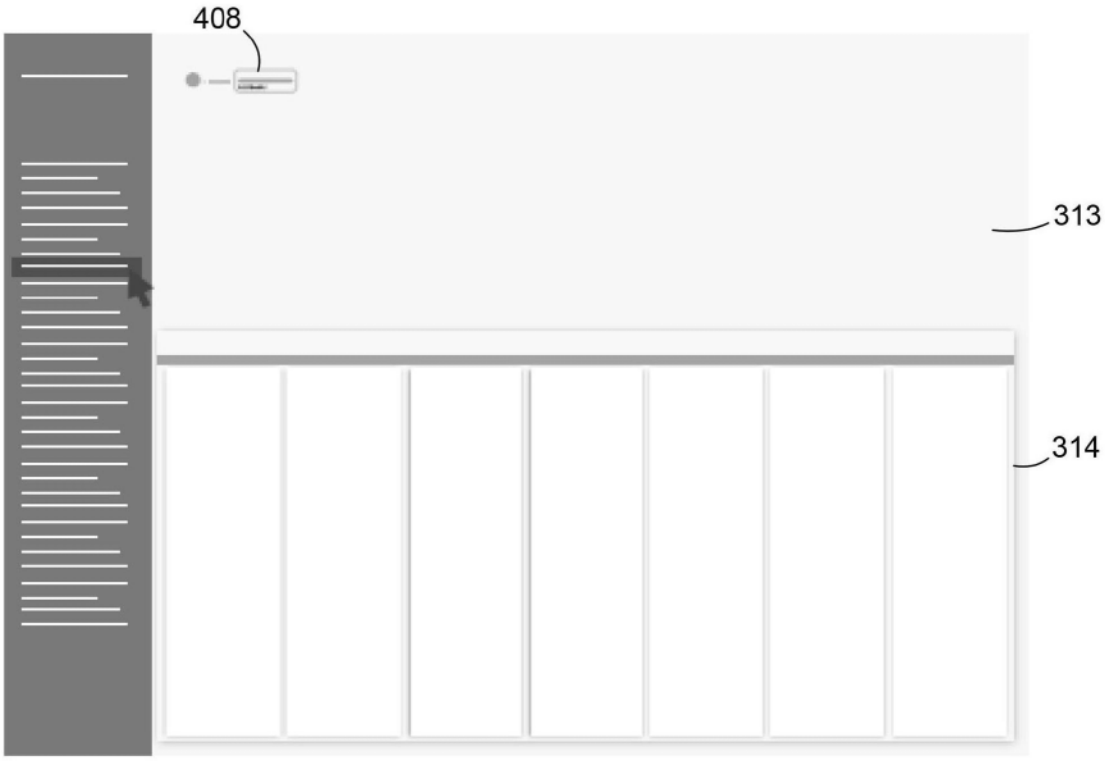


图4D

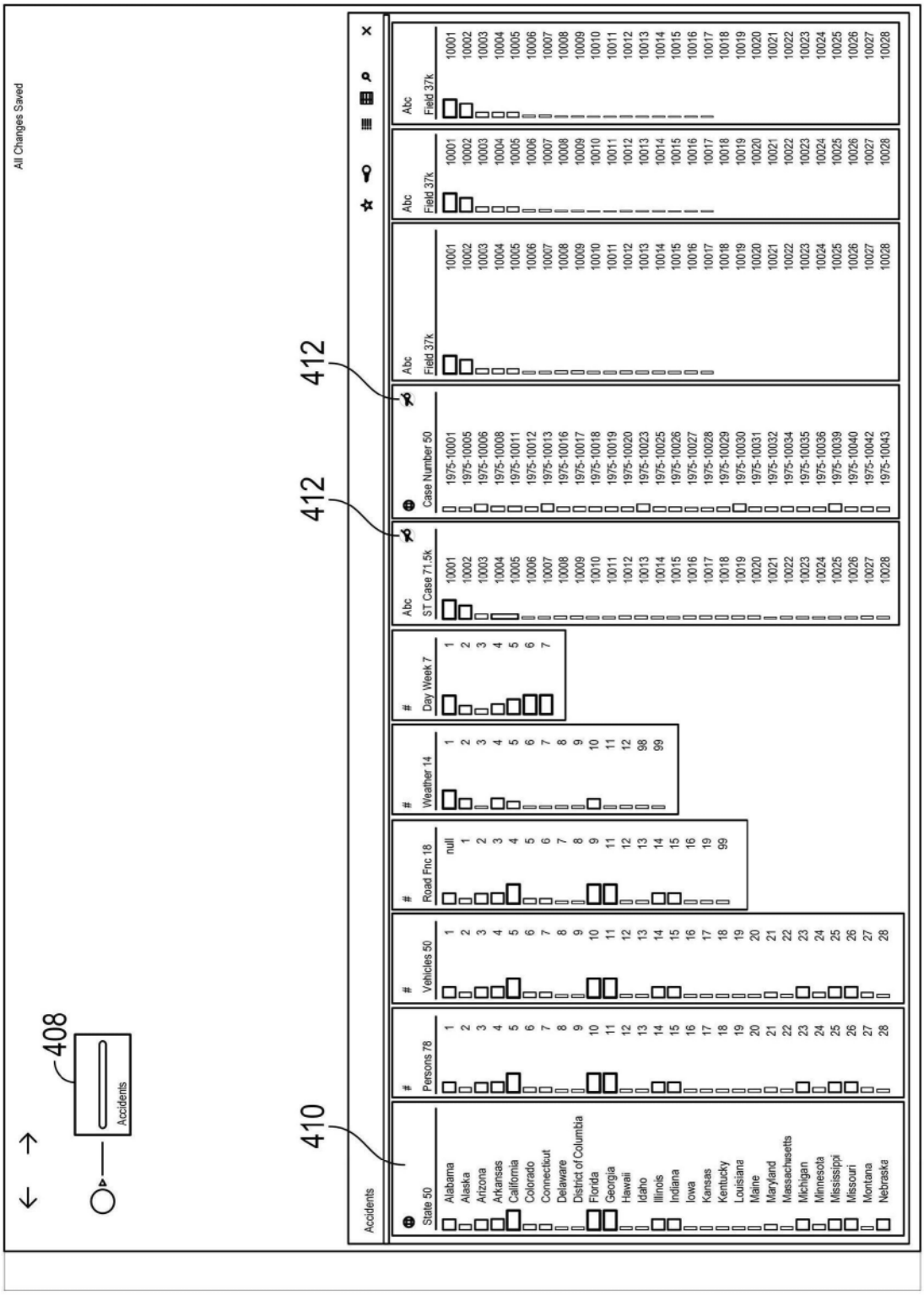

All Changes Saved
408
Accidents
410
412
412
Accidents
State 50
Persons 78
Vehicles 50
Road Fnc 18
Weather 14
Day Week 7
ST Case 71.5k
Case Number 50
Field 37k

图4E

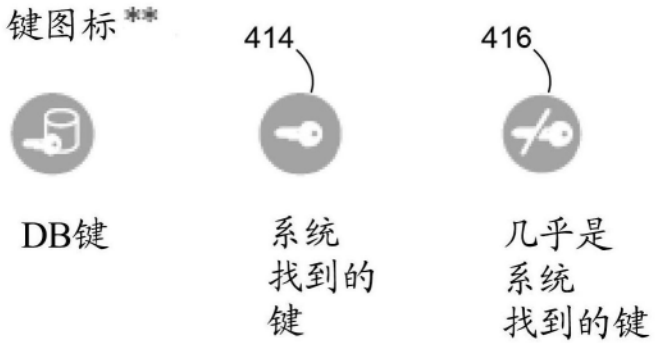


图4F

图4G

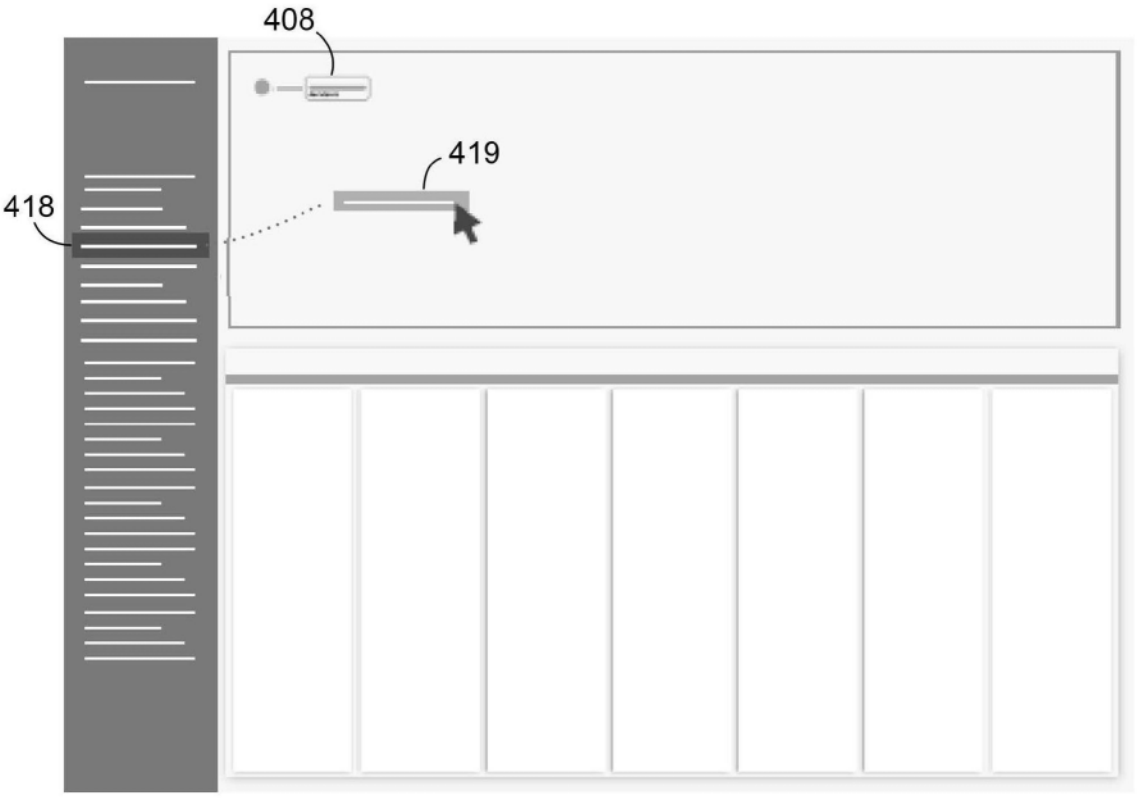


图4H

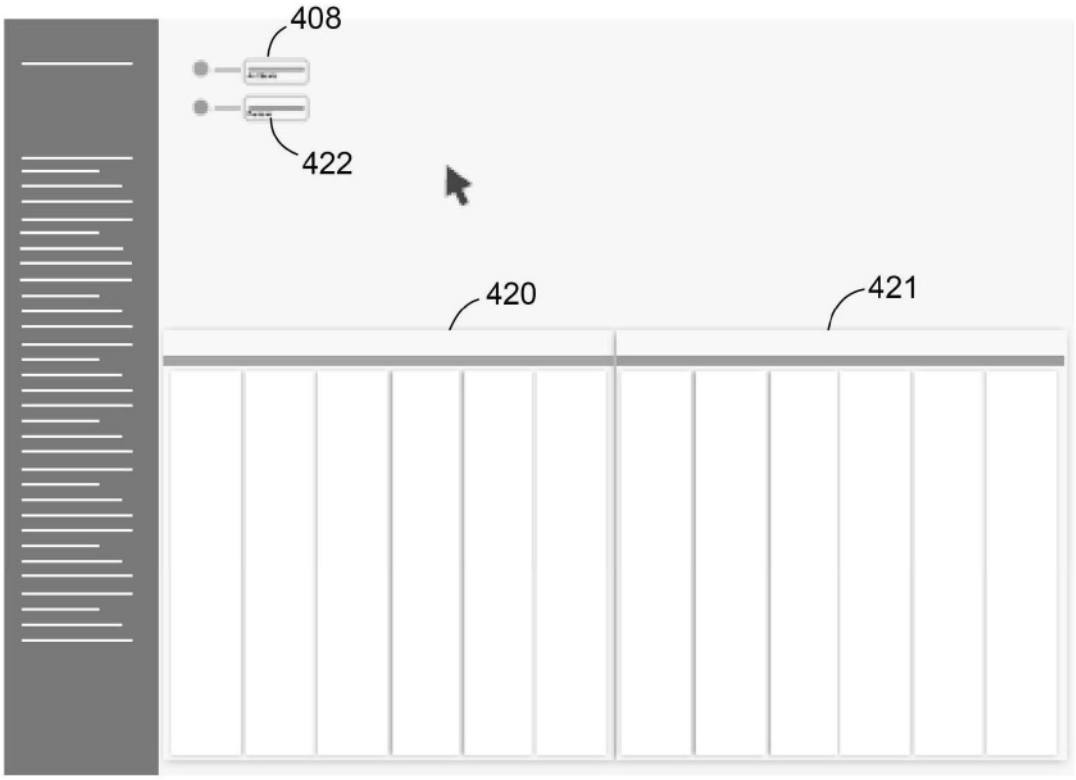


图4I

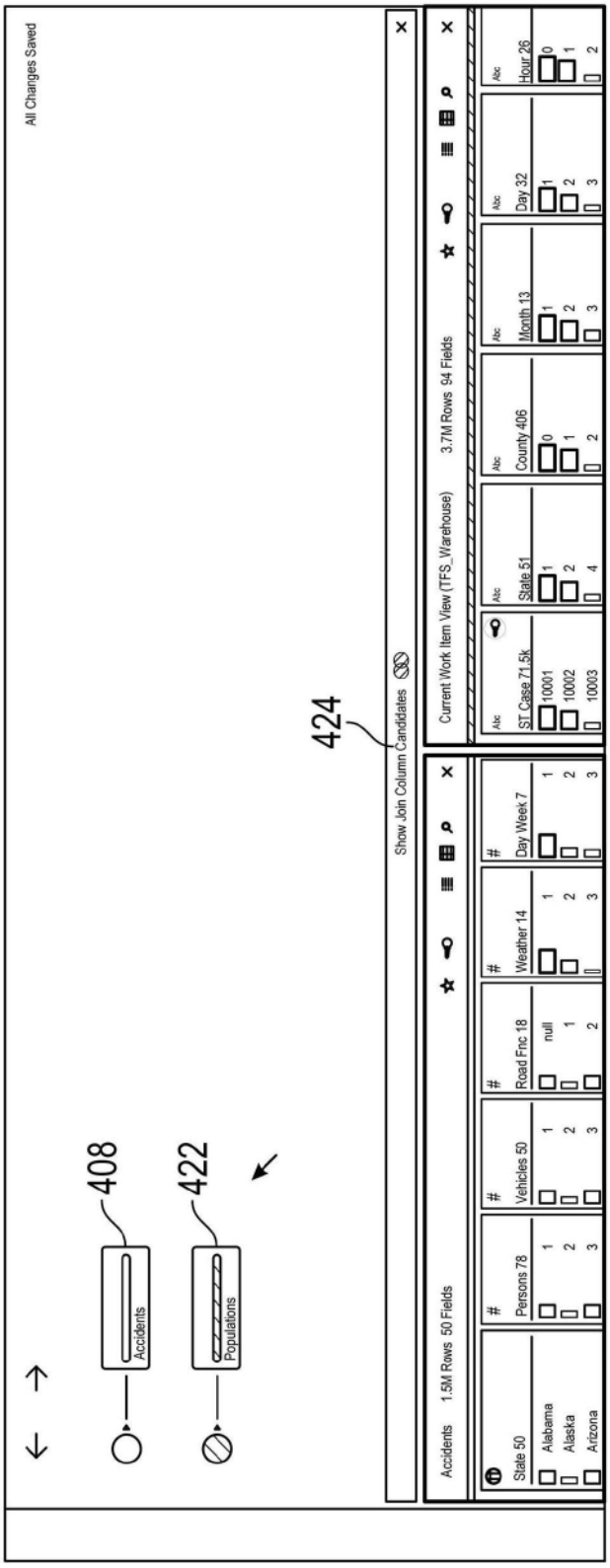

All Changes Saved
408
Accidents
422
Populations
424
Show Join Column Candidates
Accidents 1.5M Rows 50 Fields
State 50
Alabama
Alaska
Arizona
Persons 78
Vehicles 50
Road Fnc 18
Weather 14
Day Week 7
Current Work Item View (TFS_Warehouse) 3.7M Rows 94 Fields
ST Case 71.5k
10001
10002
10003
State 51
County 406
Month 13
Day 32
Hour 26

图4J

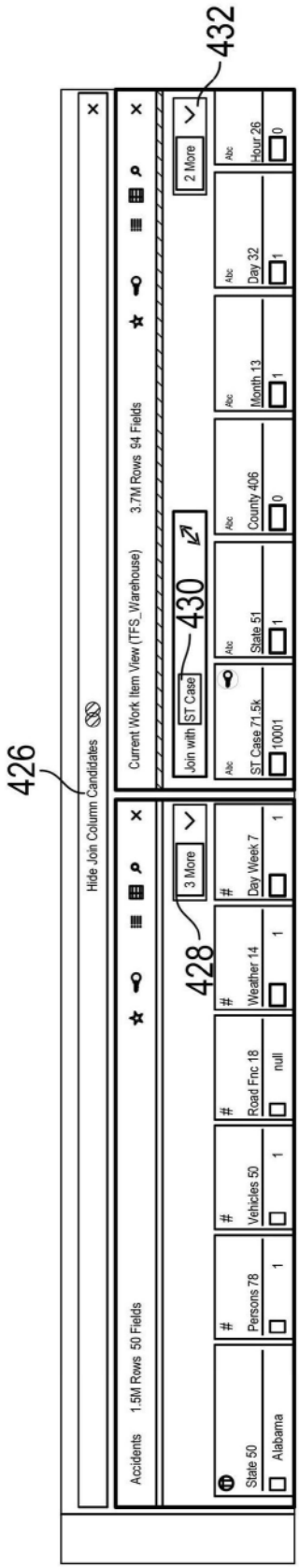

426
Hide Join Column Candidates
Accidents 1.5M Rows 50 Fields
State 50
Alabama
Persons 78
1
Vehicles 50
1
Road Fnc 18
null
Weather 14
1
Day Week 7
1
3 More
428
Current Work Item View (TFS_Warehouse) 3.7M Rows 94 Fields
Join with ST Case
430
Abc ST Case 71.5k
10001
Abc State 51
1
Abc County 406
0
Abc Month 13
1
Abc Day 32
1
2 More
432
Abc Hour 26
0

图4K

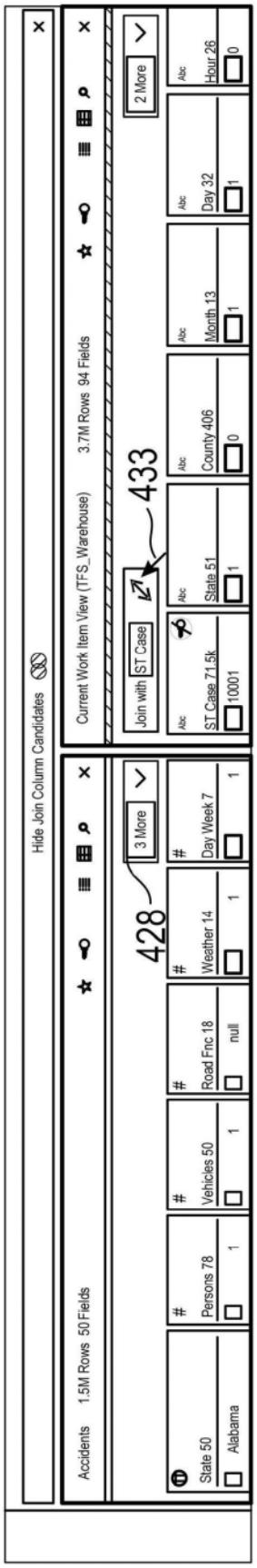

Hide Join Column Candidates
Accidents 1.5M Rows 50 Fields
State 50
Alabama
Persons 78
Vehicles 50
Road Fnc 18
null
Weather 14
3 More
Day Week 7
428
Current Work Item View (TFS_Warehouse) 3.7M Rows 94 Fields
Join with ST Case
ST Case 71.5k
10001
State 51
433
County 406
Month 13
Day 32
2 More
Hour 26

图4L

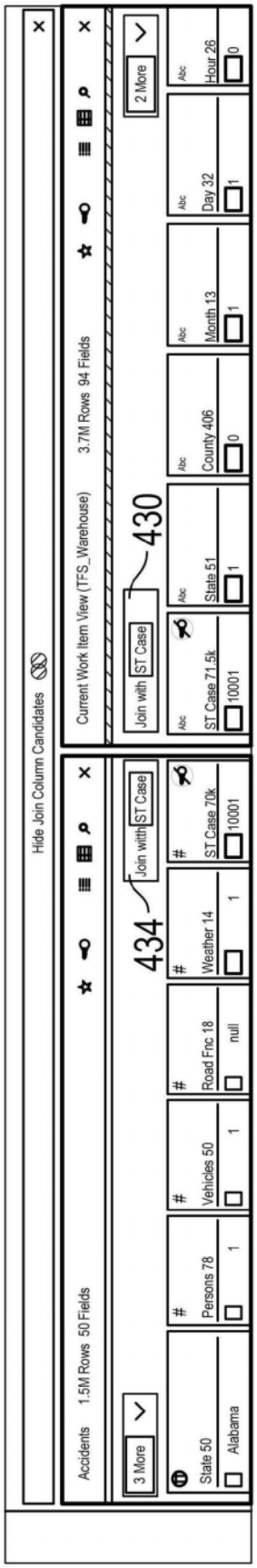

Hide Join Column Candidates
Accidents 1.5M Rows 50 Fields
3 More
State 50
Alabama
Persons 78
1
Vehicles 50
1
Road Fnc 18
null
Weather 14
1
ST Case 70k
10001
Join with ST Case
434
Current Work Item View (TFS_Warehouse) 3.7M Rows 94 Fields
Join with ST Case
430
Abc ST Case 71.5k
10001
Abc State 51
1
Abc County 406
0
Abc Month 13
1
Abc Day 32
1
2 More
Abc Hour 26
0

图4M

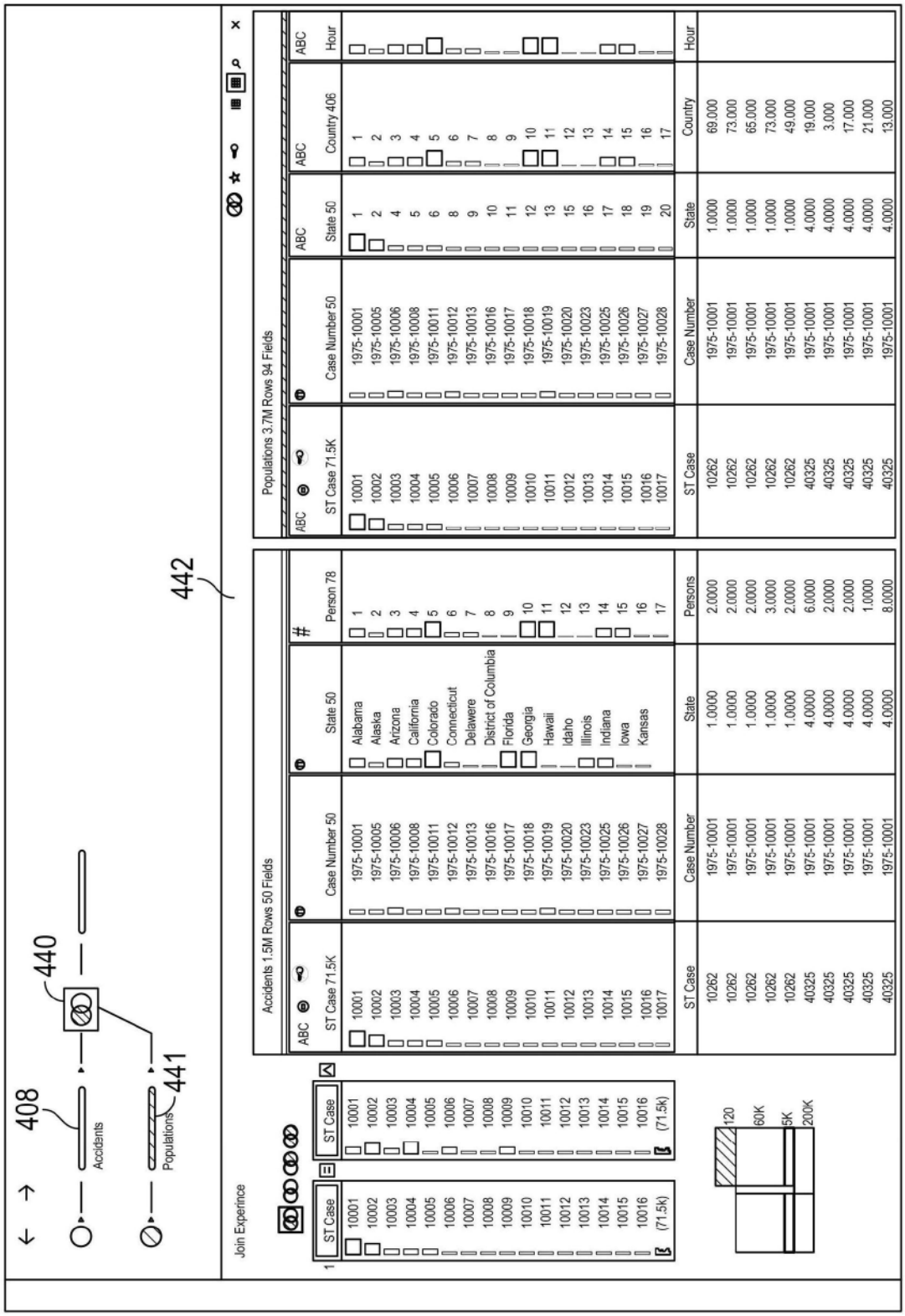

408
440
Accidents
441
Populations
442
Join Experince
Accidents 1.5M Rows 50 Fields
Populations 3.7M Rows 94 Fields
ST Case
Case Number 50
State 50
Person 78
Country 406
Hour
Case Number
State
Persons
Country
120
60K
5K
200K

图4N

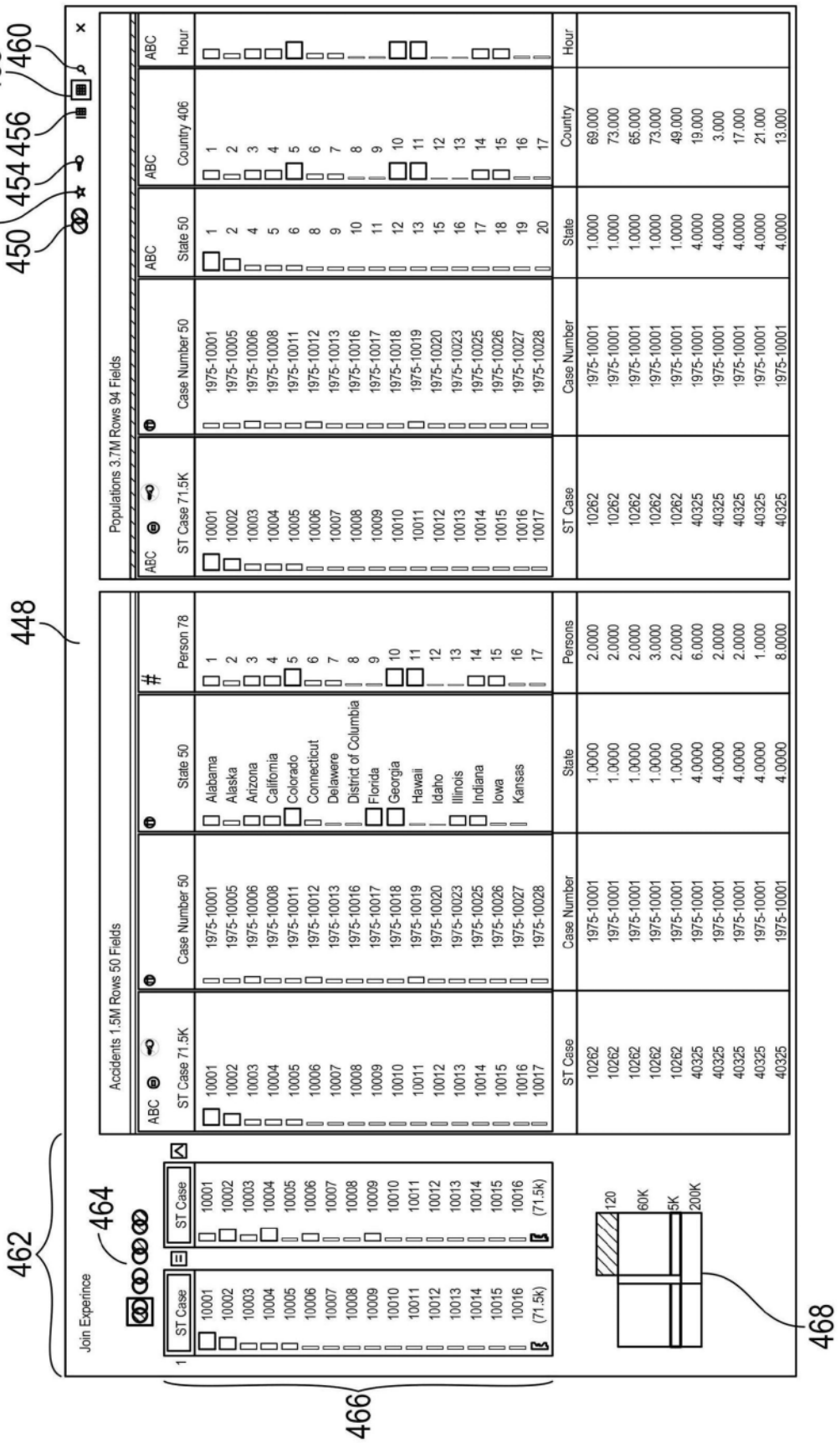

458
460
452
454
456
450
448
462
464
466
468
Join Experince
Accidents 1.5M Rows 50 Fields
Populations 3.7M Rows 94 Fields
ST Case 71.5K
Case Number 50
State 50
Person 78
Persons
Country 406
Country
Hour
120
60K
5K
200K

图40

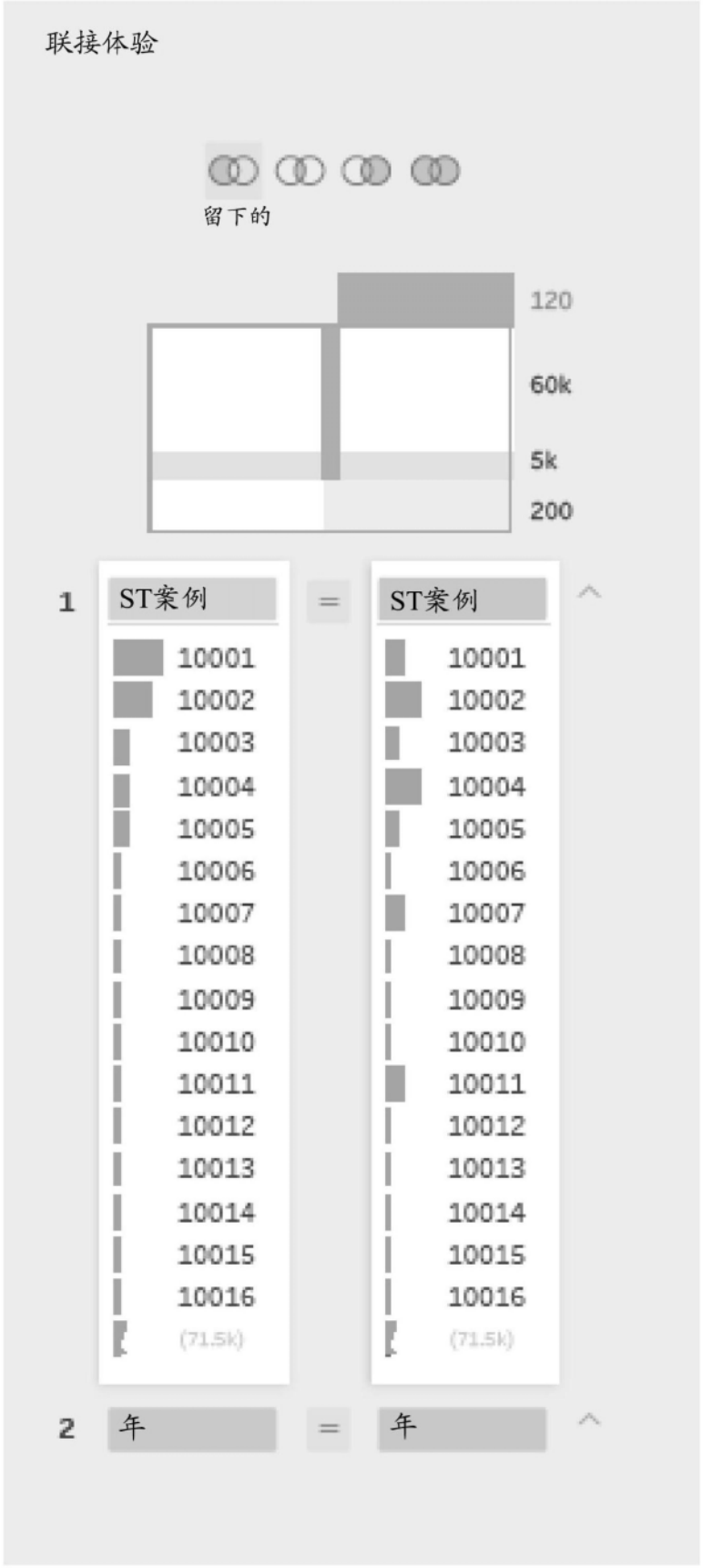

联接体验
留下的
120
60k
5k
200
1
ST案例
=
ST案例
10001
10002
10003
10004
10005
10006
10007
10008
10009
10010
10011
10012
10013
10014
10015
10016
(71.5k)
2
年
=
年

图4P

图4Q

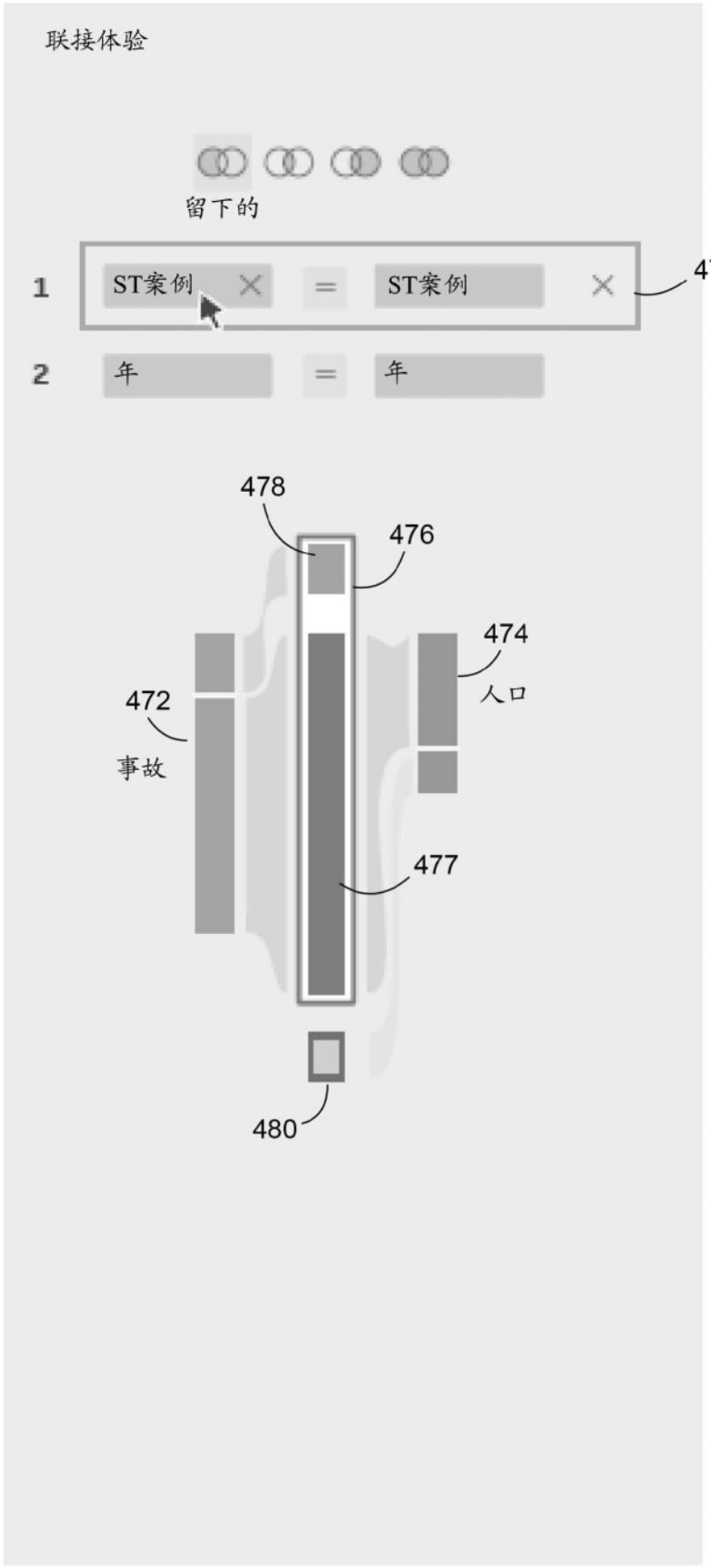


图4R

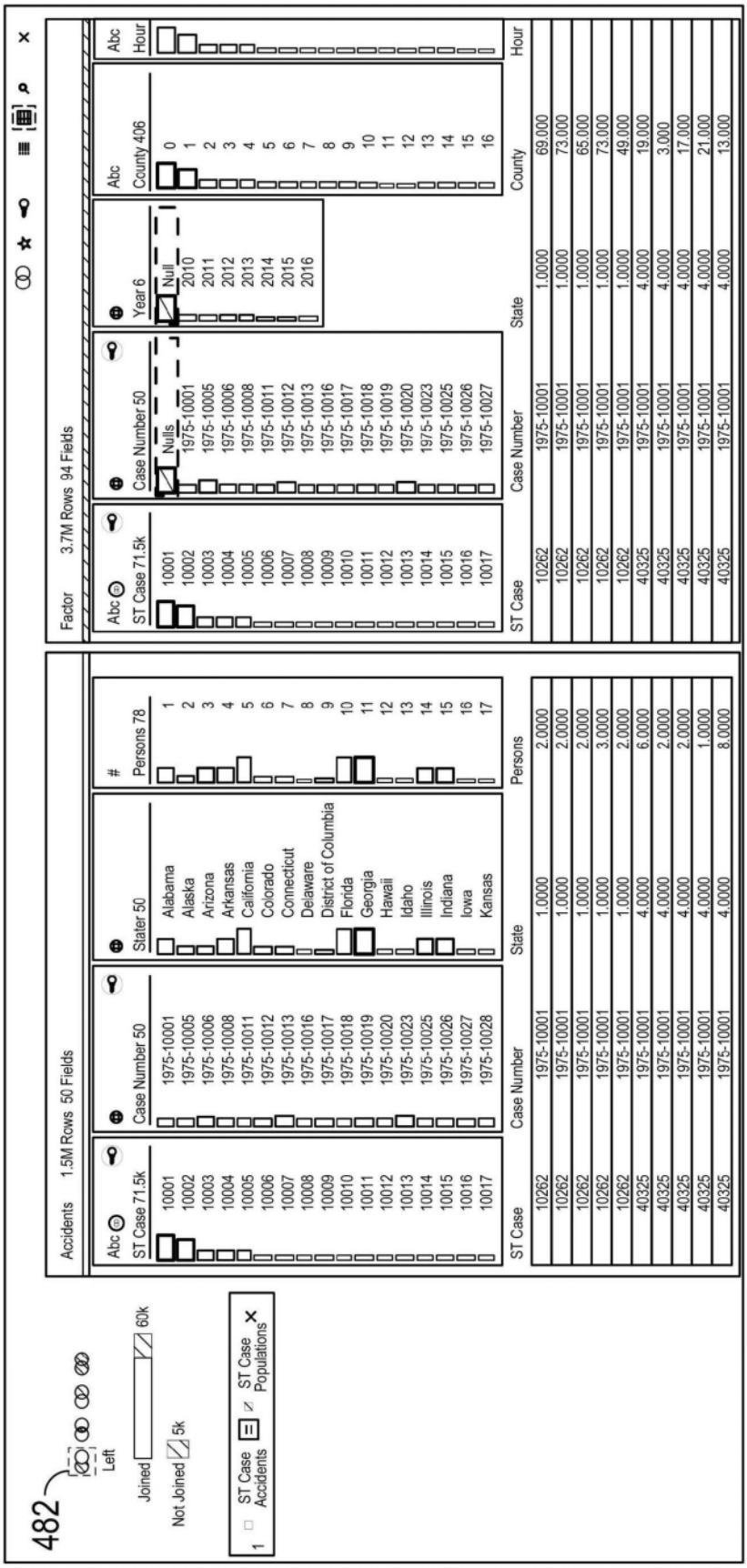

482
Left
Joined
60k
Not Joined
5k
ST Case
Accidents
ST Case
Populations
Accidents 1.5M Rows 50 Fields
Abc
ST Case 71.5k
Case Number 50
Stater 50
#
Persons 78
ST Case
Case Number
State
Persons
Factor 3.7M Rows 94 Fields
Abc
ST Case 71.5k
Case Number 50
Year 6
Abc
County 406
Abc
Hour
ST Case
Case Number
State
County
Hour

图4S

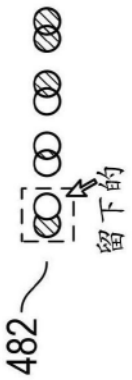


图4T

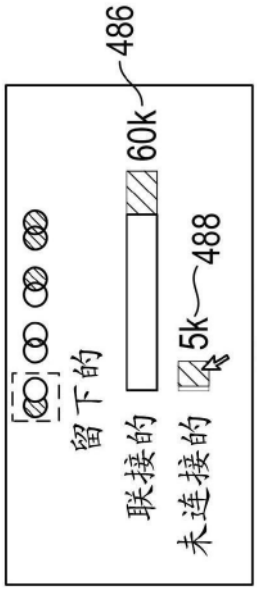


图4U

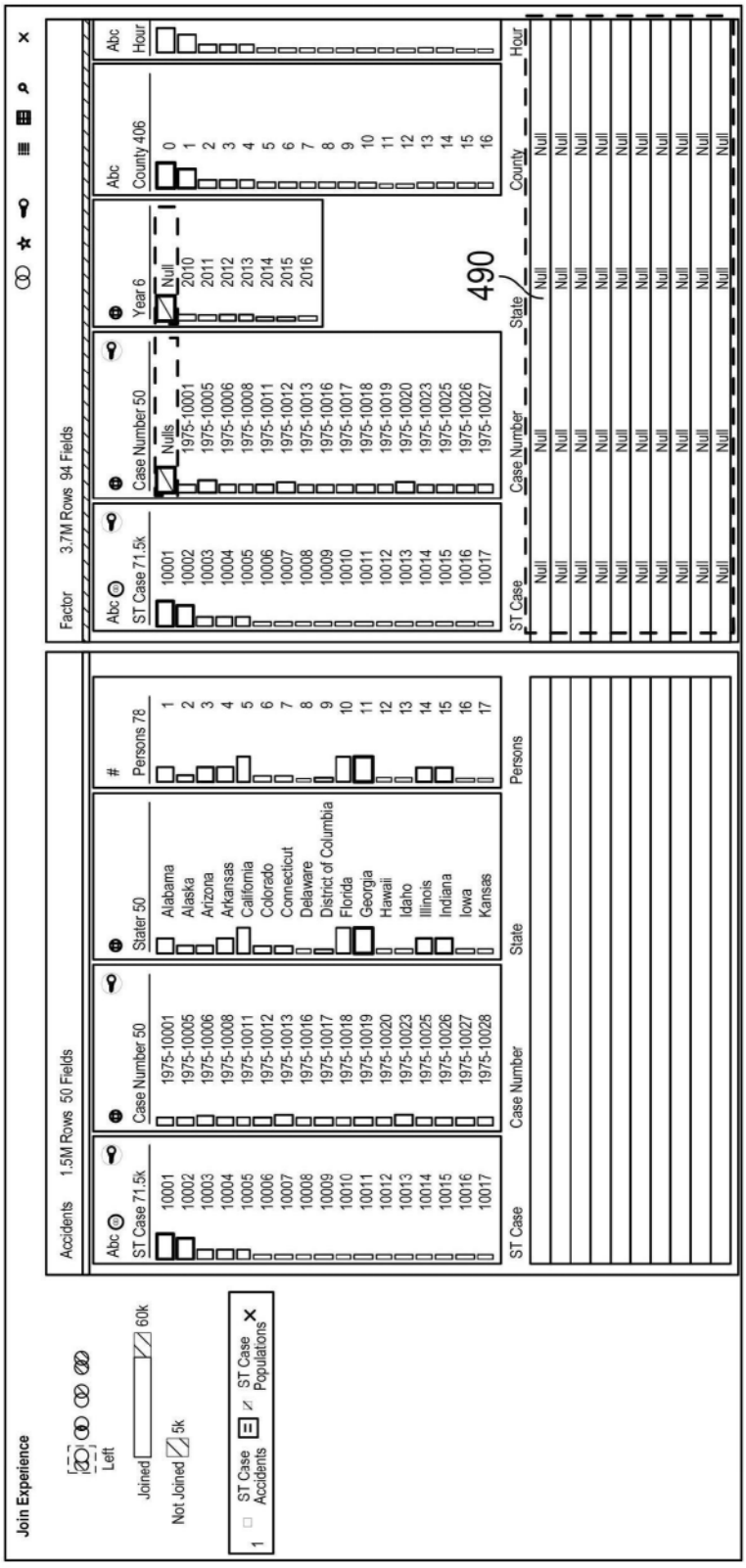

Join Experience
Left
Joined
Not Joined
60k
5k
ST Case Accidents
ST Case Populations
Accidents 1.5M Rows 50 Fields
Abc ST Case 71.5k
Case Number 50
Stater 50
Persons 78
Alabama
Alaska
Arizona
Arkansas
California
Coloradc
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
ST Case
Case Number
State
Persons
Factor 3.7M Rows 94 Fields
Nulls
Year 6
County 406
Hour
County
490
Null

图4V

```
01> Level,Date and Time,Source,Event ID,Task Category
02> Information,11/4/2015 11:12:41 AM,Service Control Manager,7036,None,The Microsoft
    Software Shadow Copy Provider service entered the running state.
03> Information,11/4/2015 10:37:07 AM,Service Control Manager,7045,None,"A service was
    installed in the system.
04>
05> Service Name:  Sophos Web Intelligence Update
06> Service File Name:  ""C:\ProgramData\Sophos\Web Intelligence\swi_update_64.exe""
07> Service Type:  user mode service
08> Service Start Type:  auto start
09> Service Account:  LocalSystem"
10> Information,11/4/2015 10:31:43 AM,Service Control Manager,7036,None,The Microsoft
    Software Shadow Copy Provider service entered the stopped state.
11> Information,11/4/2015 10:28:43 AM,Service Control Manager,7036,None,The Volume Shadow
    Copy service entered the stopped state.
12> ...
```

图5A

SAS 名称：ROAD_FNC

属性代码

1975-1980

这个数据元素被包括在格式中，但未被初始化。不使用它。

1981-1986

1 主要干线-洲际公路
2 主要干线-其它城市高速公路和快速公路
3 主要干线-其它
4 次要干线
5 城市收集器
6 主要乡村收集器
7 次要乡村收集器
8 本地道路或街道
9 未知的

1987-以后

乡村

01 主要干线-洲际公路
02 主要干线-其它
03 次要干线
04 主要收集器
05 次要收集器
06 本地道路或街道
09 未知的

城市的

11 主要干线-洲际公路
12 主要干线-其它城市高速公路和快速公路
13 其它主要干线
14 次要干线
15 收集器
16 本地道路或街道
19 未知的

99 未知的

图5B

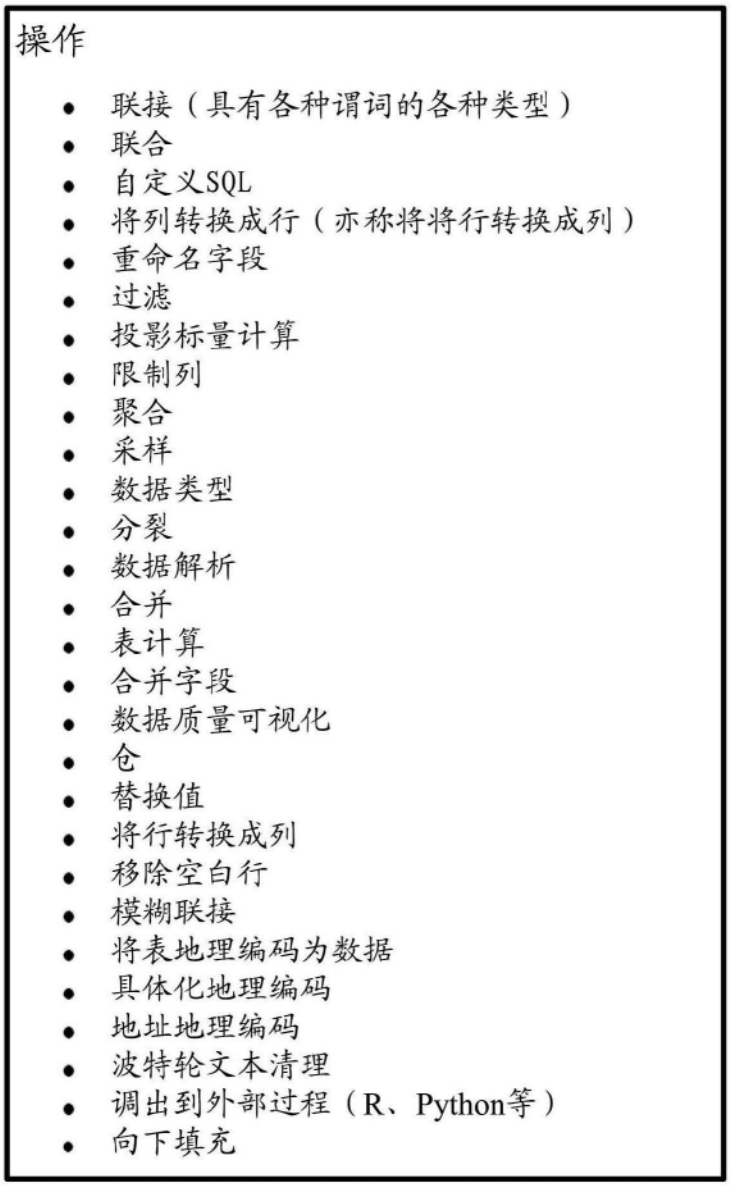

操作
联接（具有各种谓词的各种类型）
联合
自定义SQL
将列转换成行（亦称将将行转换成列）
重命名字段
过滤
投影标量计算
限制列
聚合
采样
数据类型
分裂
数据解析
合并
表计算
合并字段
数据质量可视化
仓
替换值
将行转换成列
移除空白行
模糊联接
将表地理编码为数据
具体化地理编码
地址地理编码
波特轮文本清理
调出到外部过程（R、Python等）
向下填充

图6A

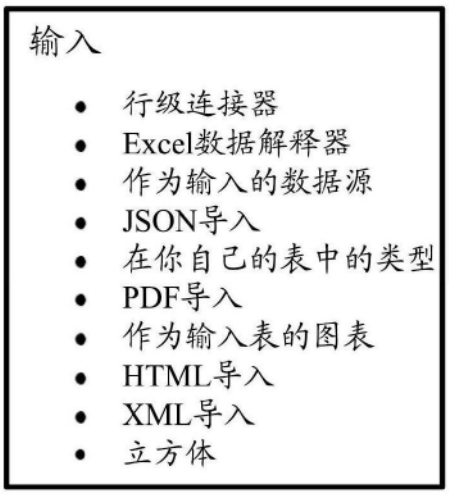

输入
行级连接器
Excel数据解释器
作为输入的数据源
JSON导入
在你自己的表中的类型
PDF导入
作为输入表的图表
HTML导入
XML导入
立方体

图6B

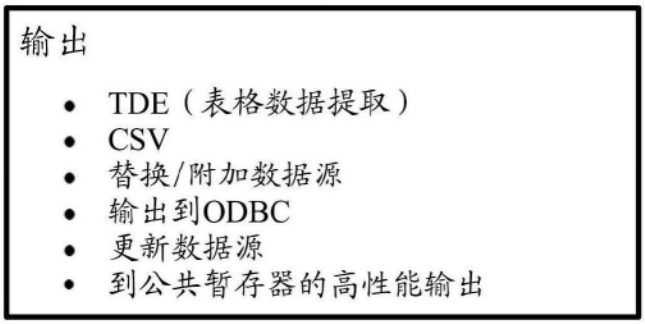

输出
TDE（表格数据提取）
CSV
替换/附加数据源
输出到ODBC
更新数据源
到公共暂存器的高性能输出

图6C

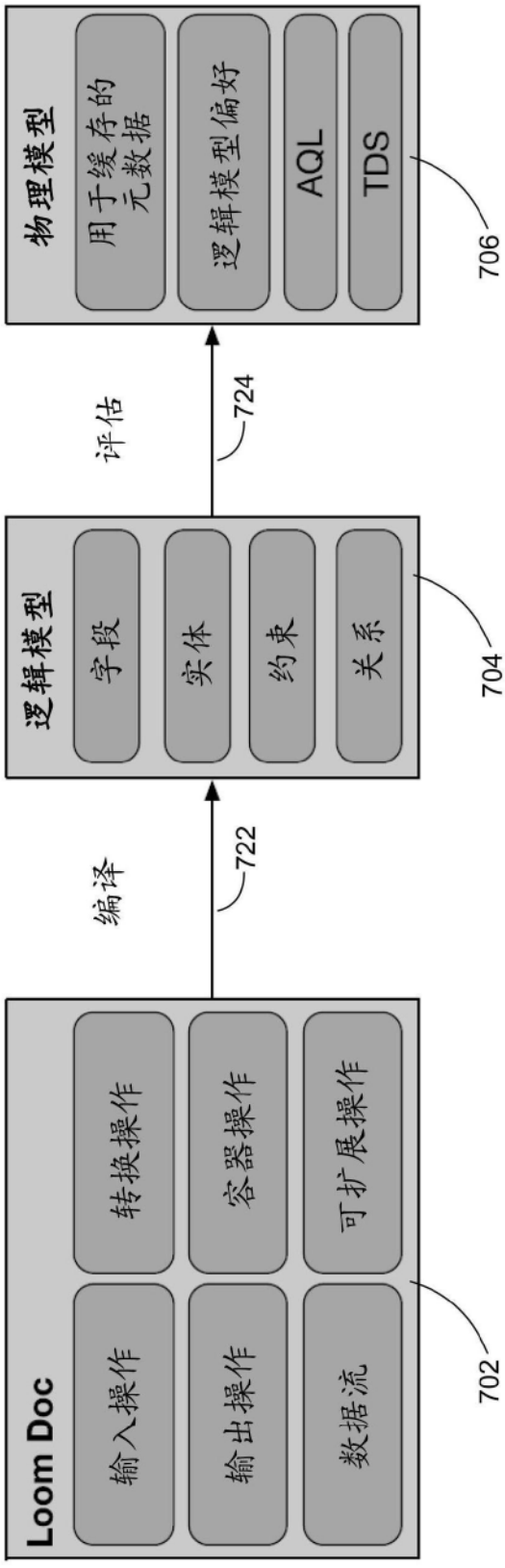

Loom Doc
输入操作
输出操作
数据流
转换操作
容器操作
可扩展操作
702
编译
722
逻辑模型
字段
实体
约束
关系
704
评估
724
物理模型
用于缓存的元数据
逻辑模型偏好
AQL
TDS
706

图7A

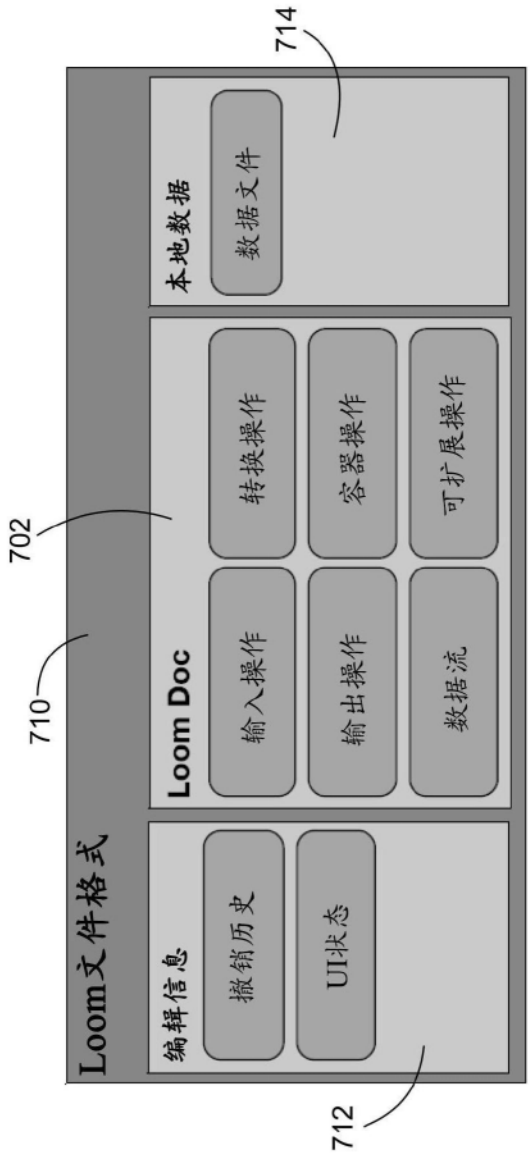

714
Loom文件格式
编辑信息
撤销历史
UI状态
712
710
Loom Doc
输入操作
输出操作
数据流
转换操作
容器操作
可扩展操作
702
本地数据
数据文件

图7B

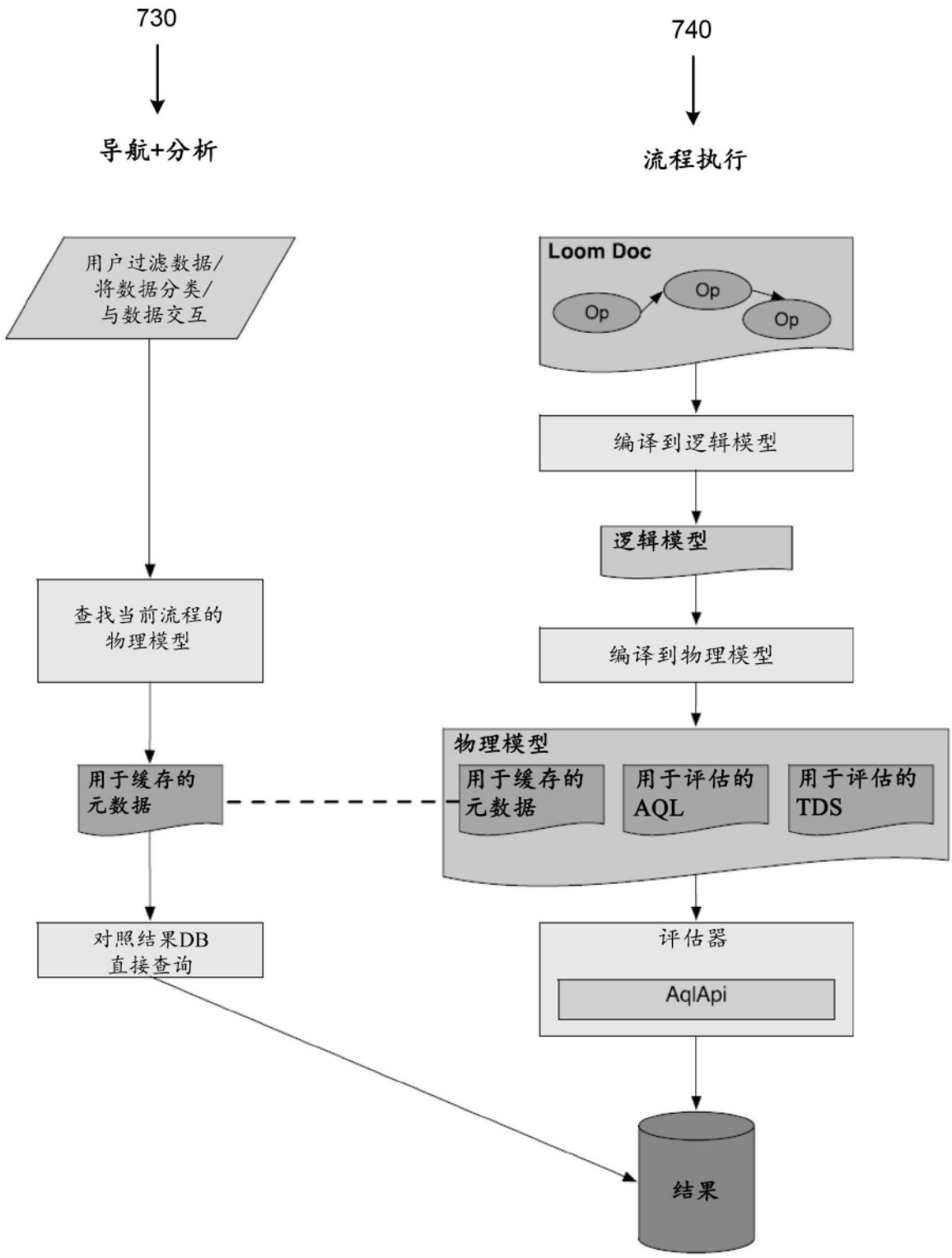

730
导航+分析
740
流程执行
用户过滤数据/
将数据分类/
与数据交互
Loom Doc
Op
Op
Op
编译到逻辑模型
逻辑模型
查找当前流程的
物理模型
编译到物理模型
物理模型
用于缓存的
元数据
用于缓存的
元数据
用于评估的
AQL
用于评估的
TDS
对照结果DB
直接查询
评估器
AqlApi
结果

图7C

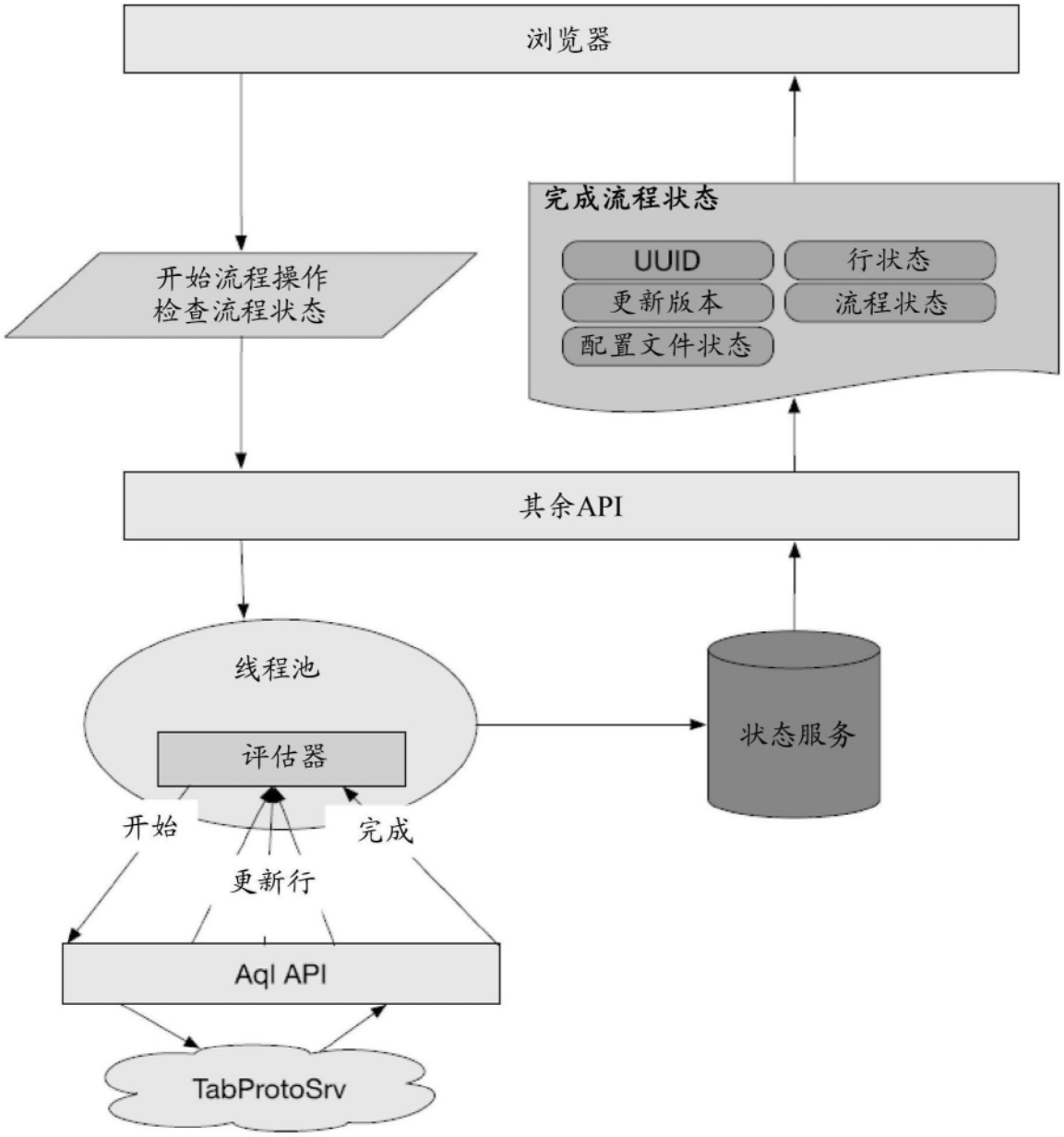

浏览器
完成流程状态
UUID
行状态
更新版本
流程状态
配置文件状态
开始流程操作
检查流程状态
其余API
线程池
评估器
状态服务
开始
完成
更新行
Aql API
TabProtoSrv

图7D

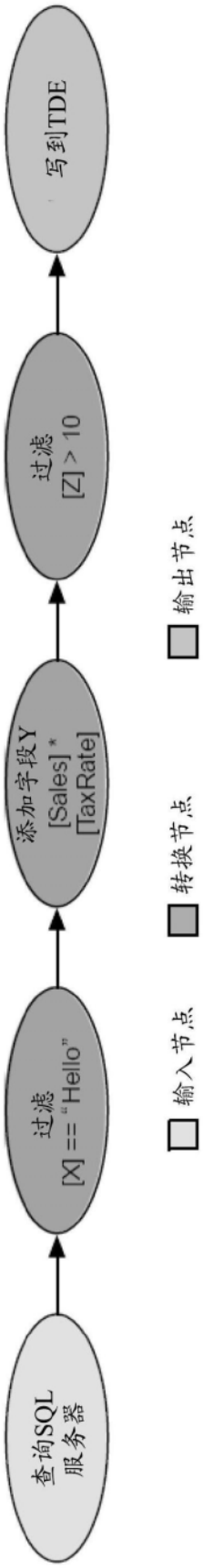

查询SQL
服务器
过滤
[X] == "Hello"
添加字段Y
[Sales] *
[TaxRate]
过滤
[Z] > 10
写到TDE
输入节点
转换节点
输出节点

图8A

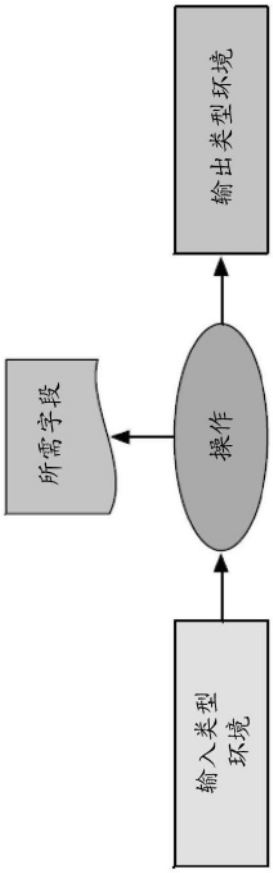


图8B

类型环境

属性	描述
开放的	是否查找未被列出的应被考虑为错误或被解析为可以是任何类型的潜在字段的字段
类型	从字段名称转到类型的映射
不存在	我们知道的一组字段名称不存在（即使类型是封闭的）
AssumedTypes	所有类型被添加，同时处理类型环境，因为它们被引用，而不是因为它们由于操作而被显式地添加
上一个	用于创建此的类型环境的指针

图8C

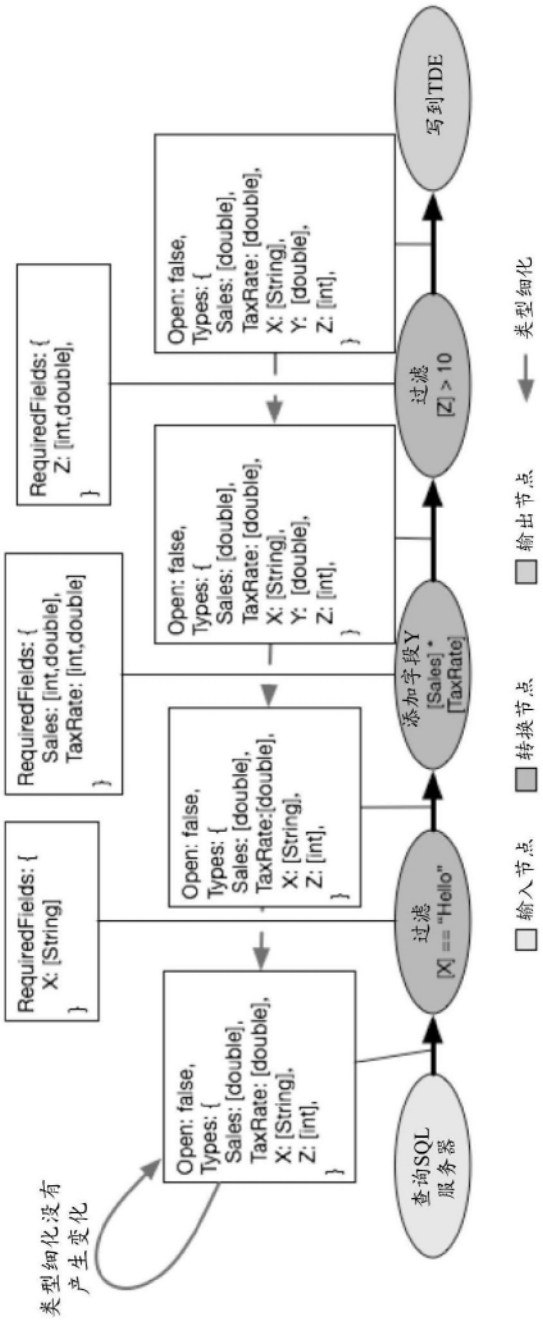

查询SQL
服务器
过滤
[X] == "Hello"
添加字段Y
[Sales] * [TaxRate]
过滤
[Z] > 10
写到TDE
RequiredFields: { X: [String] }
RequiredFields: { Sales: [int,double], TaxRate: [int,double] }
RequiredFields: { Z: [int,double] }
Open: false, Types: { Sales: [double], TaxRate: [double], X: [String], Z: [int], }
Open: false, Types: { Sales: [double], TaxRate:[double], X: [String], Z: [int], }
Open: false, Types: { Sales: [double], TaxRate: [double], X: [String], Y: [double], Z: [int], }
Open: false, Types: { Sales: [double], TaxRate: [double], X: [String], Y: [double], Z: [int], }
类型细化没有产生变化
输入节点
转换节点
输出节点
类型细化

图8D

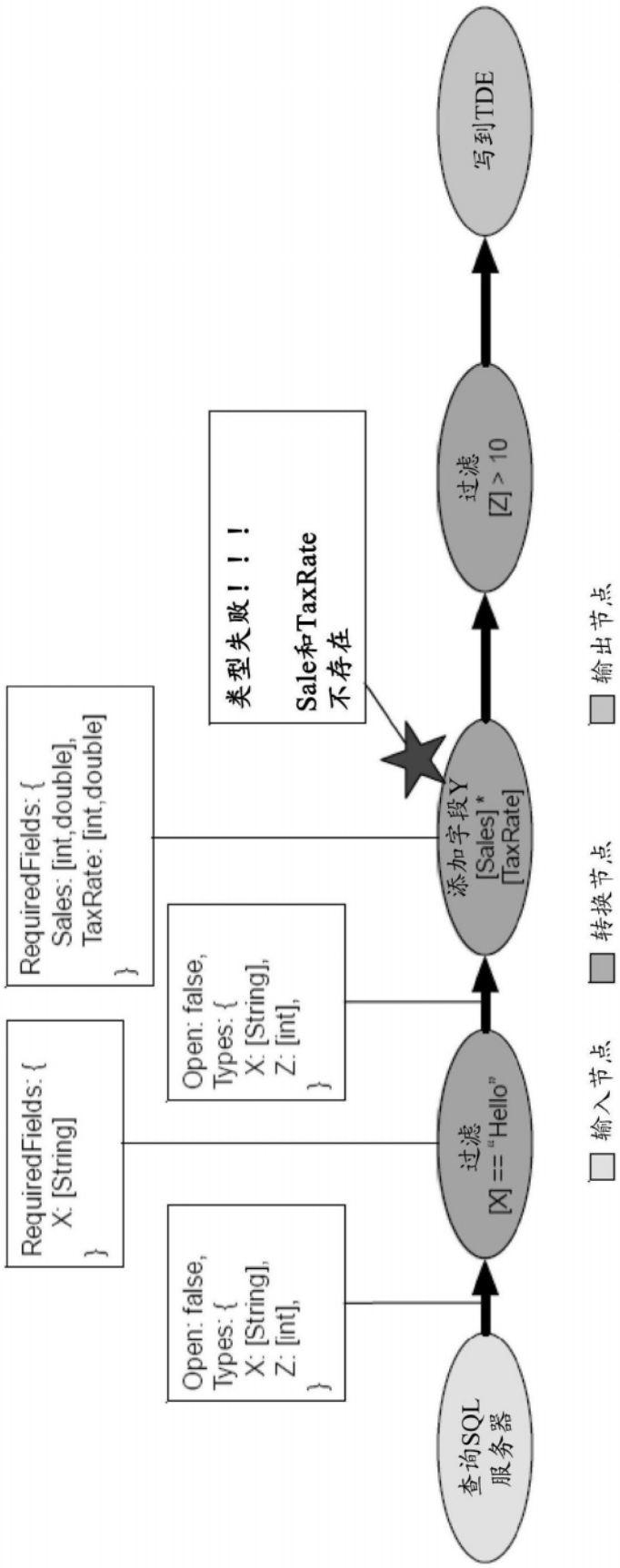

查询SQL
服务器
过滤
[X] == "Hello"
添加字段Y
[Sales] *
[TaxRate]
过滤
[Z] > 10
写到TDE
Open: false,
Types: {
X: [String],
Z: [int],
}
RequiredFields: {
X: [String]
}
Open: false,
Types: {
X: [String],
Z: [int],
}
RequiredFields: {
Sales: [int,double],
TaxRate: [int,double]
}
类型失败！！！
Sale和TaxRate
不存在
输入节点
转换节点
输出节点

图8E

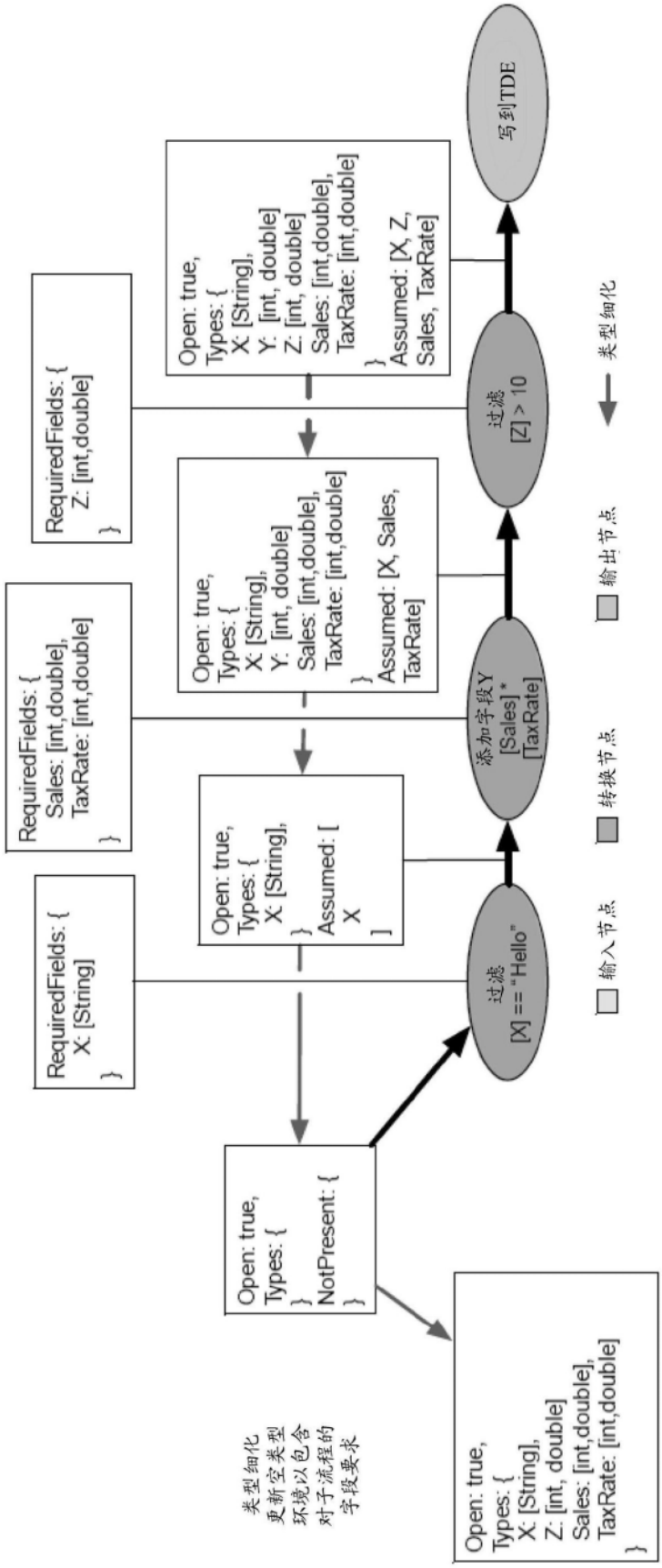

写到TDE
过滤
[Z] > 10
添加字段Y
[Sales]*
[TaxRate]
过滤
[X] == "Hello"
Open: true,
Types: {
X: [String],
Y: [int, double]
Z: [int, double]
Sales: [int,double],
TaxRate: [int,double]
}
Assumed: [X, Z,
Sales, TaxRate]
RequiredFields: {
Z: [int,double]
}
Open: true,
Types: {
X: [String],
Y: [int, double]
Sales: [int,double],
TaxRate: [int,double]
}
Assumed: [X, Sales,
TaxRate]
RequiredFields: {
Sales: [int,double],
TaxRate: [int,double]
}
Open: true,
Types: {
X: [String],
}
Assumed: [
X
]
RequiredFields: {
X: [String]
}
Open: true,
Types: {
}
NotPresent: {
}
Open: true,
Types: {
X: [String],
Z: [int, double]
Sales: [int,double],
TaxRate: [int,double]
}
输入节点
转换节点
输出节点
类型细化
类型细化
更新空类型
环境以包含
对子流程的
字段要求

图8F

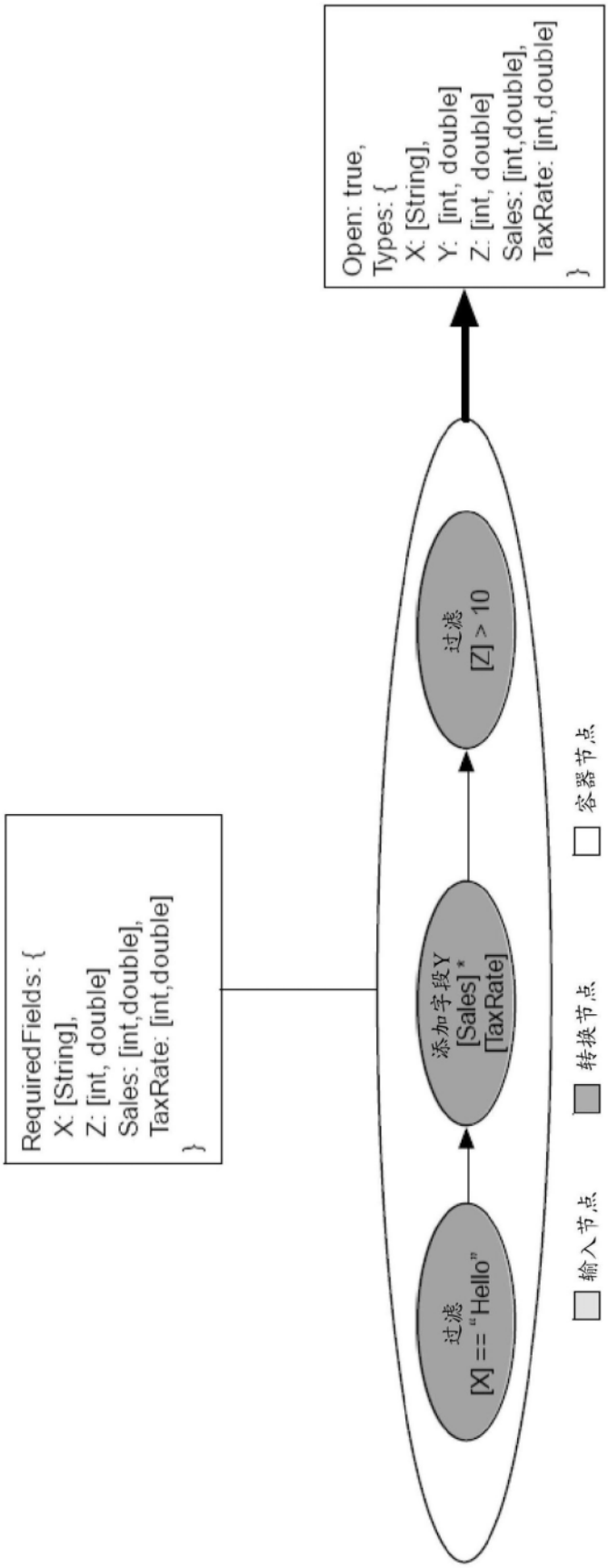

RequiredFields: {
X: [String],
Z: [int, double]
Sales: [int,double],
TaxRate: [int,double]
}
过滤
[X] == "Hello"
添加字段Y
[Sales] *
[TaxRate]
过滤
[Z] > 10
Open: true,
Types: {
X: [String],
Y: [int, double]
Z: [int, double]
Sales: [int,double],
TaxRate: [int,double]
}
输入节点
转换节点
容器节点

图8G

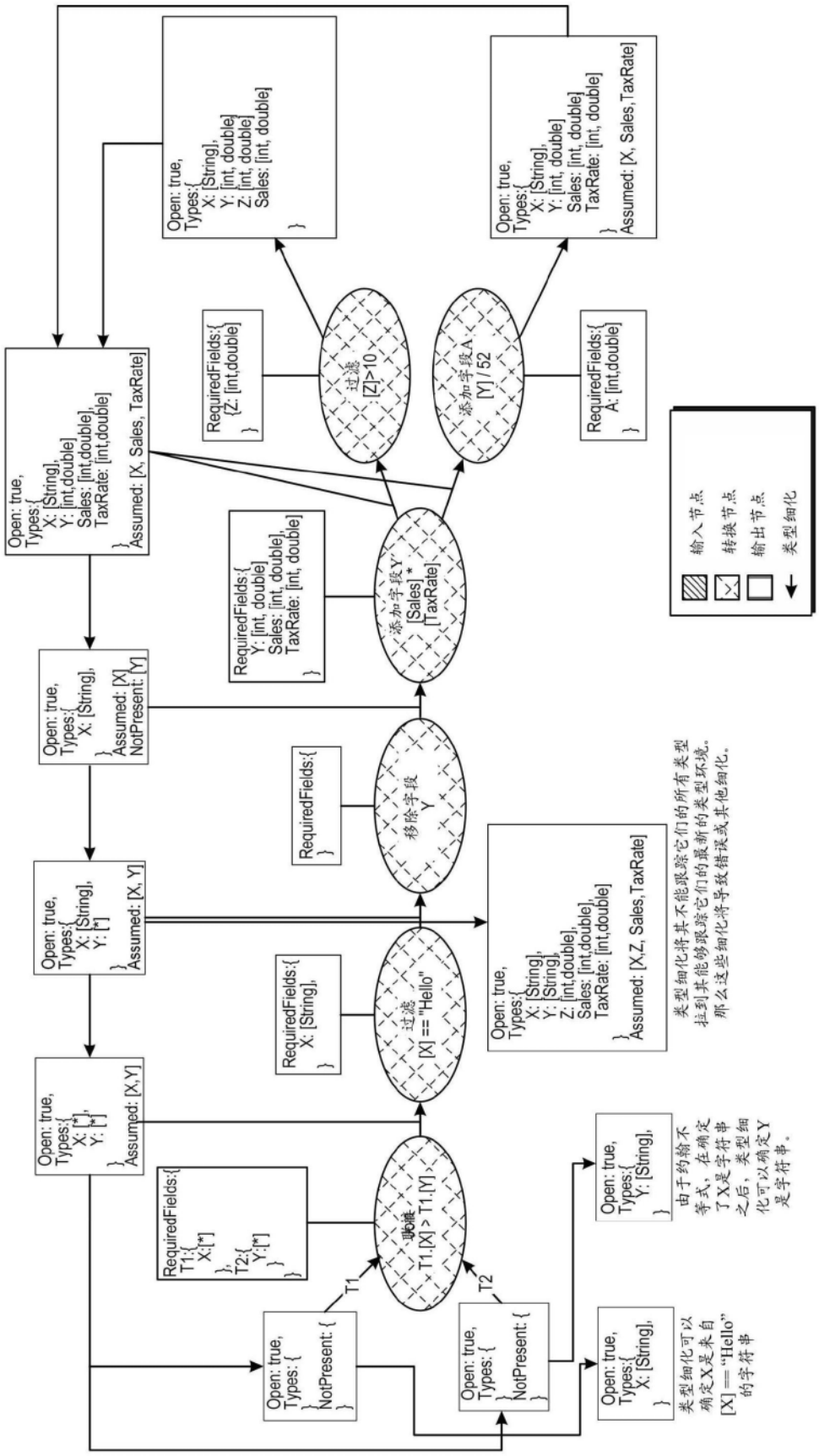

Open: true,
Types:{
X: [String],
Y: [int,double]
Sales: [int,double],
TaxRate: [int,double]
}
Assumed: [X, Sales, TaxRate]
RequiredFields:{
{Z: [int,double]
}
过滤
[Z]>10
Open: true,
Types:{
X: [String],
Y: [int, double]
Z: [int, double]
Sales: [int, double]
}
添加字段A
[Y] / 52
RequiredFields:{
A: [int,double]
}
Open: true,
Types:{
X: [String],
Y: [int, double]
Sales: [int, double]
TaxRate: [int, double]
}
Assumed: [X, Sales, TaxRate]
RequiredFields:{
Y: [int, double]
Sales: [int, double],
TaxRate: [int, double]
}
添加字段'Y'
[Sales] *
[TaxRate]
输入节点
转换节点
输出节点
类型细化
Open: true,
Types:{
X: [String],
Assumed: [X]
NotPresent: [Y]
}
RequiredFields:{
}
移除字段
Y
Open: true,
Types:{
X: [String],
Y: [*]
}
Assumed: [X, Y]
Open: true,
Types:{
X: [String],
Y: [String],
Z: [int,double],
Sales: [int,double],
TaxRate: [int,double]
}
Assumed: [X,Z, Sales,TaxRate]
类型细化将其不能跟踪它们的所有类型
拉到其能够跟踪它们的最新的类型环境。
那么这些细化将导致错误或其他细化。
RequiredFields:{
X: [String],
}
过滤
[X] == "Hello"
Open: true,
Types:{
X: [*],
Y: [*]
}
Assumed: [X,Y]
RequiredFields:{
T1:{
X:[*]
},
T2:{
Y:[*]
}
}
过滤
T1.[X] > T1.[Y]
T1
T2
Open: true,
Types: {
}
NotPresent: {
}
Open: true,
Types: {
}
NotPresent: {
}
Open: true,
Types:{
Y: [String],
}
由于约翰不
幸式，在确定
了X是字符串
之后，类型细
化可以确定Y
是字符串。
Open: true,
Types:{
X: [String],
}
类型细化可以
确定X是来自
[X] == "Hello"
的字符串

图8H

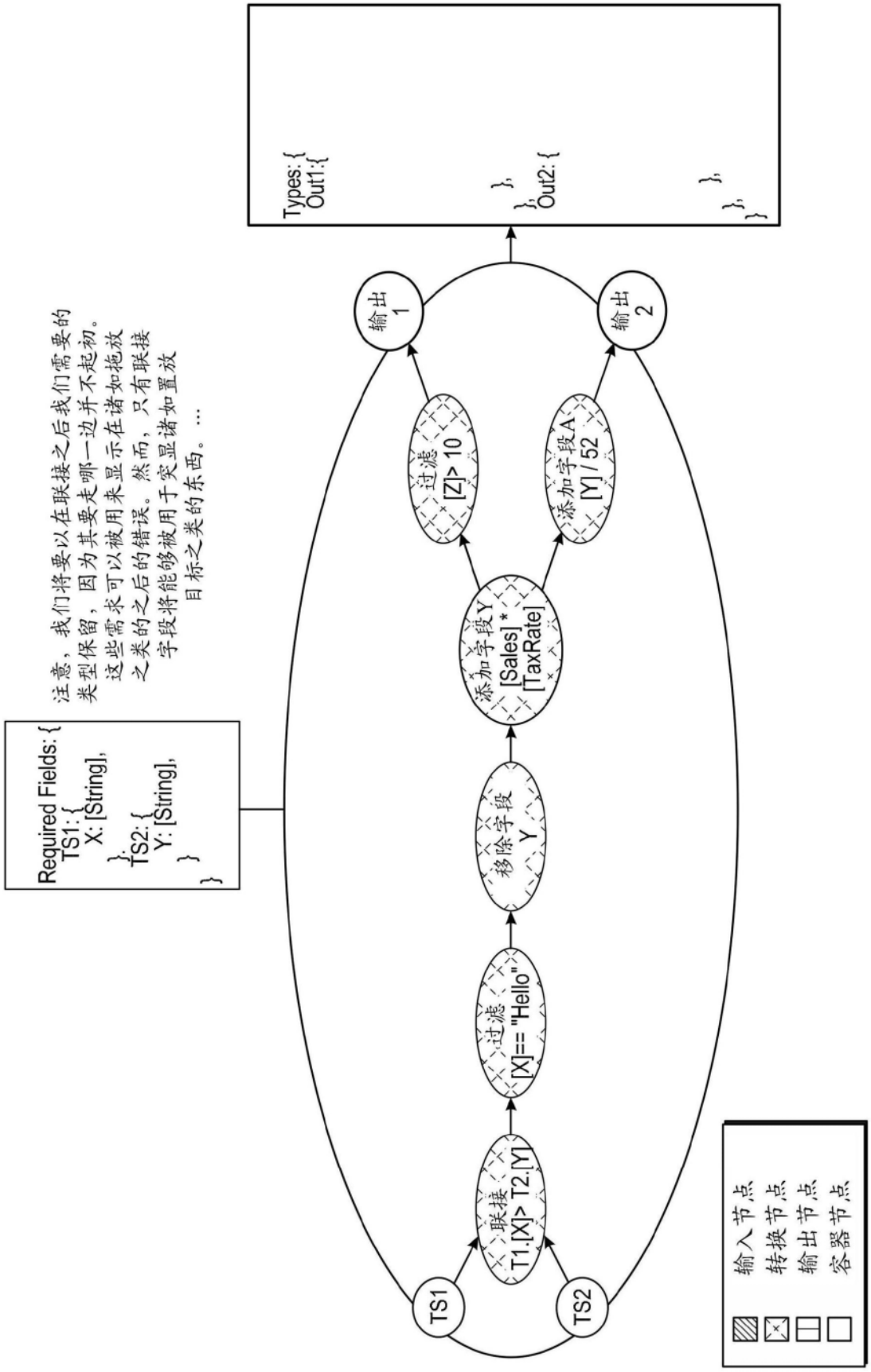

Required Fields: {
TS1: {
X: [String],
},
TS2: {
Y: [String],
}
}
注意，我们将要以在联接之后我们需要的类型保留，因为其要走哪一边并不起初。这些需求可以被用来显示在诸如拖放之类的之后的错误。然而，只有联接字段将能够被用于突显诸如置放目标之类的东西。...
Types: {
Out1:{
},
},
Out2: {
},
},
}
TS1
TS2
联接 T1.[X]> T2.[Y]
过滤 [X]== "Hello"
移除字段 Y
添加字段Y [Sales] * [TaxRate]
过滤 [Z]> 10
添加字段A [Y] / 52
输出 1
输出 2
输入节点
转换节点
输出节点
容器节点

图8I

名称	是 开放的	多输入	输入类型	产生的类型
过滤 (所有过滤 类型)	真	假	*在表达式中使用的 所有字段	*以输入类型开始 *基于用于过滤的表达式 使任何字段类型变窄
添加列	真	假	*在表达式中使用的 所有字段 注意：目前我们将不 要求被创建字段不存在。 我们将简单地具有盖写 初始字段的新字段。	*以输入类型开始 *添加新字段或更新 现有字段 *基于用于创建字段的 表达式使任何字段 类型变窄
联接	真	假	*被联接的所有字段	*合并输入表。这将使具 有相同名称的所联接的 列塌缩并将基于下面列出 的规则来重命名关于名称 冲突的任何列 *使所联接的字段的 类型变窄（如果可能）
聚合	假	假	*所有字段被聚合 *所有字段被分组	*以被分组的所有字段 开始 *添加所有聚合字段 *将字段标记为“封闭的”
联合	真	真		*所有进入的映射后的 列的联合

图8J-1

名称	是 开放的	多输入	输入类型	产生的类型
表格行 转换成列	真	假	*字段从列转换成行	*以输入类型开始 *从类型移除从列转换成行的列并标记为“不存在” *添加新的从列转换成行的列 *添加从列转换成行产生的列
表格列 转换成行	真	假	*字段从行转换成列 *在从行转换成列的计算中使用的任何字段	*以输入类型开始 *从类型中移除从行转换成列的字段并标记为“不存在” *添加从行转换成列的值作为字段
重命名 字段	真	假	*包含将被重命名的字段 *不包含将被重命名的字段	*以输入类型开始 *从类型中移除被重命名的字段并标记为“不存在” *添加具有与原始字段相同的类型的新名称的字段。
移除列	真	假	*包含待移除的所有字段	*以输入类型开始 *从类型中移除该字段并标记为“不存在”

图8J-2

名称	是 开放的	多输入	输入类型	产生的类型
限制列	假	假		*以输入类型开始 *移除不在列表上 包括的所有列 *将类型标记为“封闭的”
转换列	真	假	*包含待转换的字段	*以输入类型开始 *将选定列的类型改变 到新类型
结合	真	假	*包含待结合的字段和 在结合中引用的任何 其它字段	*以输入类型开始 *添加具有产生的 类型的新结合列
合并列	真	假	*包含待合并的所有字段	*以输入类型开始 *添加将合并字段连成 串的新字段 *从类型中移除合并的 字段并标记为“不存在”
分裂列	真	假	*包含将要分裂的 所有字段 *不包含由于分裂而 产生的列	*以输入类型开始 *以适当的类型添加 由于分裂而得到的 新字段
映射值	真	假	*包含将不被映射的列	*与输入类型相同
输入操作	假	假		*为输入中的每个字段 创建字段 *将类型标记为“封闭的”

图8J-3

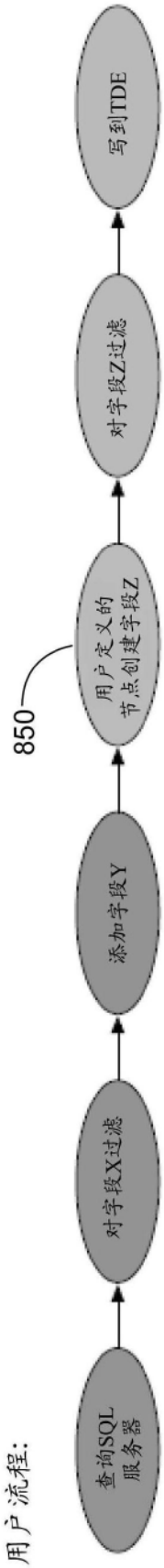


图8K

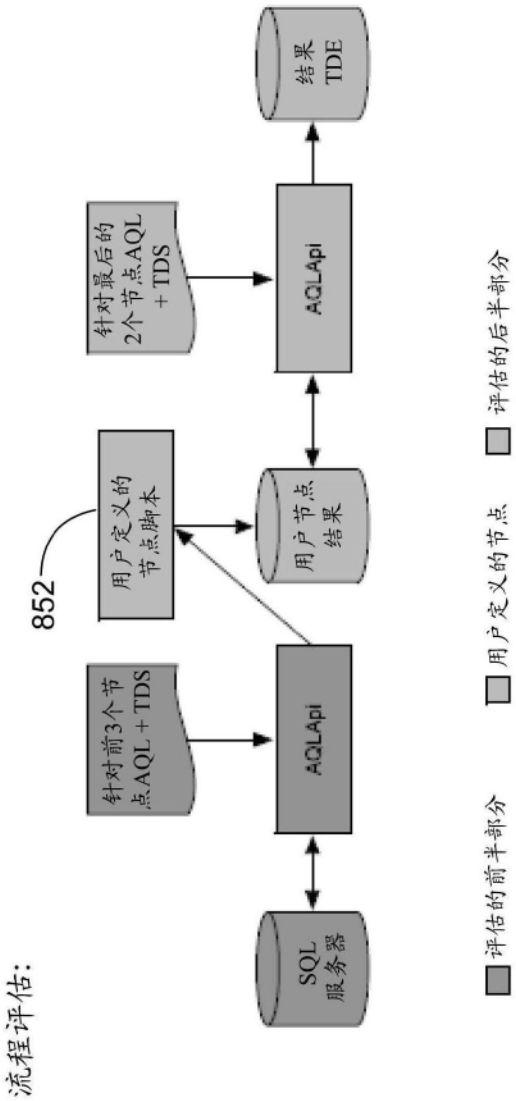

流程评估:
SQL 服务器
AQLApi
针对前3个节点AQL + TDS
852
用户定义的节点脚本
用户节点结果
AQLApi
针对最后的2个节点AQL + TDS
结果 TDE
评估的前半部分
用户定义的节点
评估的后半部分

图8L

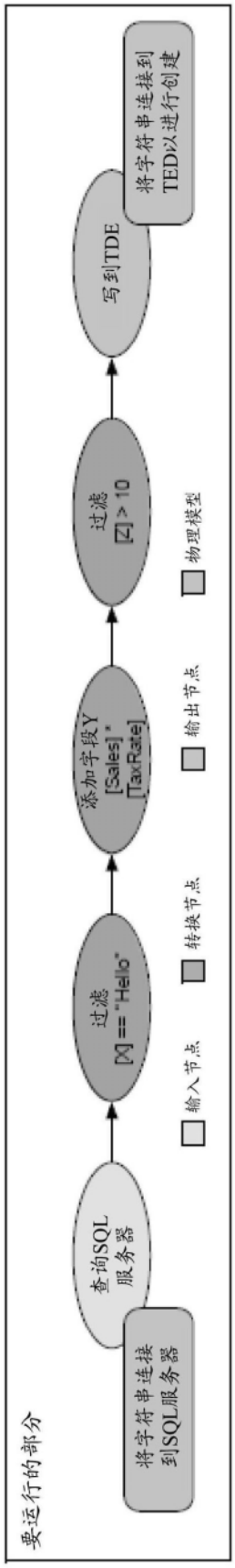

要运行的部分
将字符串连接到SQL服务器
查询SQL服务器
过滤 [X] == "Hello"
添加字段'Y' [Sales]*[TaxRate]
过滤 [Z] > 10
写到TDE
将字符串连接到TED以进行创建
输入节点
转换节点
输出节点
物理模型

图8M

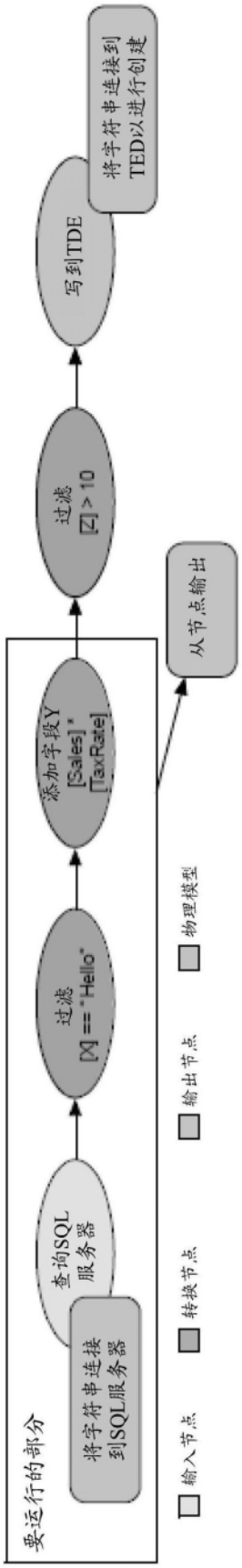

将字符串连接到TED以进行创建
写到TDE
过滤 [Z] > 10
添加字段Y [Sales]*[TaxRate]
过滤 [X] == "Hello"
查询SQL服务器
将字符串连接到SQL服务器
要运行的部分
从节点输出
输入节点
转换节点
输出节点
物理模型

图8N

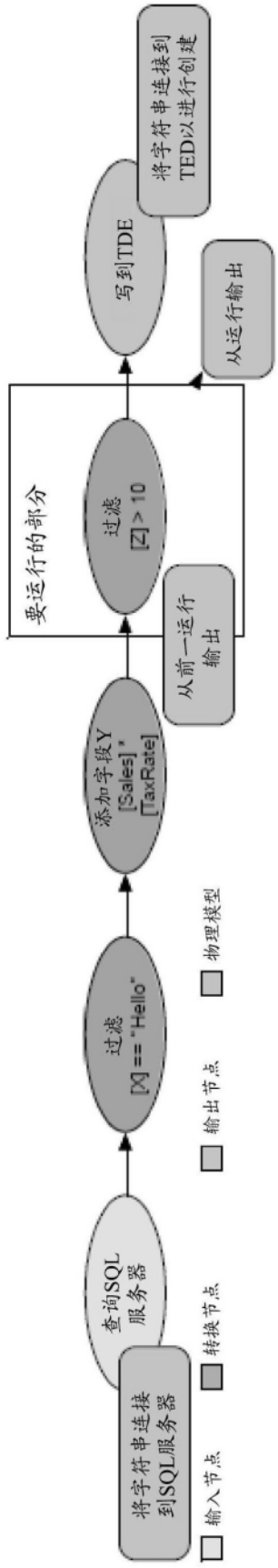

将字符串连接到SQL服务器
查询SQL服务器
过滤 [X] == "Hello"
添加字段Y [Sales]' [TaxRate]
从前一运行输出
要运行的部分
过滤 [Z] > 10
写到TDE
从运行输出
将字符串连接到TDE以进行创建
输入节点
转换节点
输出节点
物理模型

图80

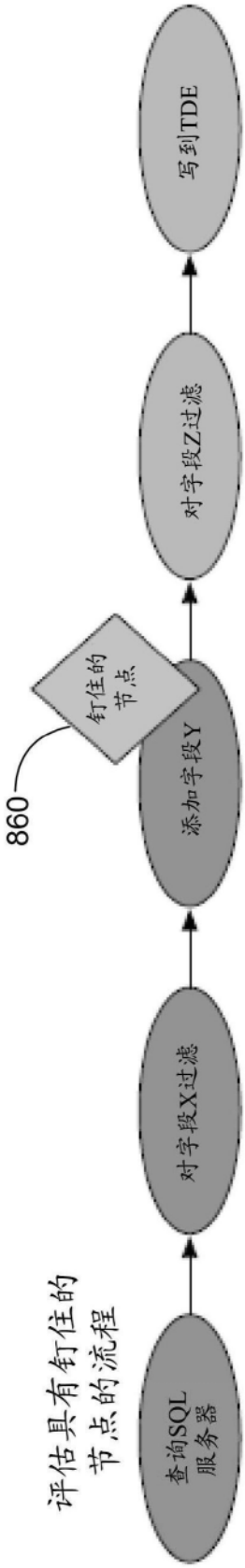

评估具有钉住的
节点的流程
查询SQL
服务器
对字段X过滤
添加字段Y
860
钉住的
节点
对字段Z过滤
写到TDE

图8P

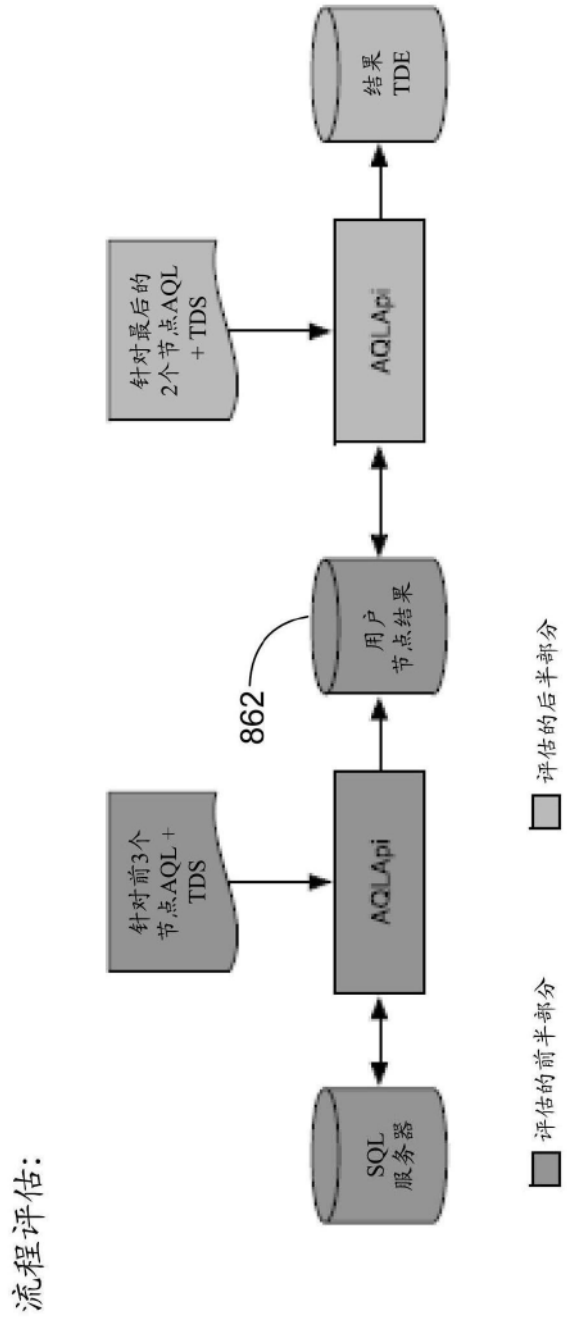

流程评估:
SQL
服务器
AQLApi
针对前3个
节点AQL +
TDS
862
用户
节点结果
AQLApi
针对最后的
2个节点AQL
+ TDS
结果
TDE
评估的前半部分
评估的后半部分

图8Q

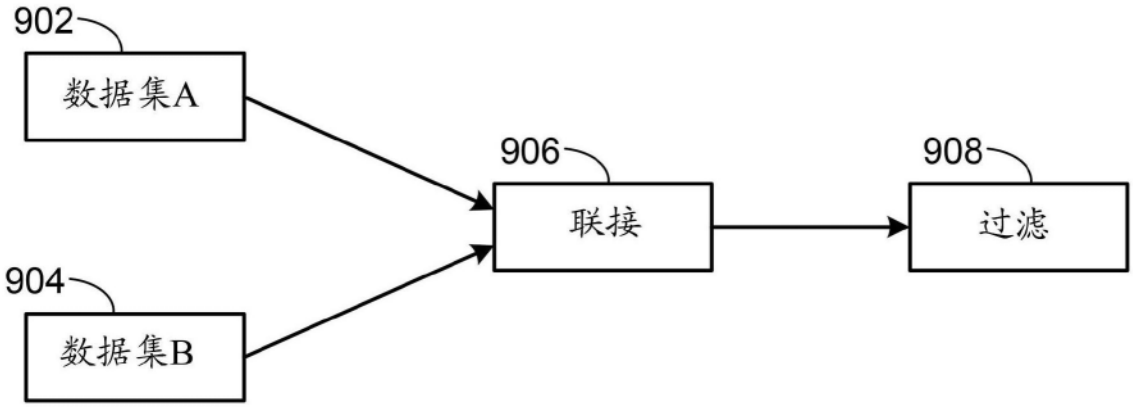


图9