Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

CA 3153691 C 2024/06/11

(11)(21) 3 153 691

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2020/08/07
(87) Date publication PCT/PCT Publication Date: 2021/03/18
(45) Date de délivrance/Issue Date: 2024/06/11
(85) Entrée phase nationale/National Entry: 2022/03/07
(86) N° demande PCT/PCT Application No.: US 2020/045461
(87) N° publication PCT/PCT Publication No.: 2021/050182
(30) Priorité/Priority: 2019/09/13 (US16/570,969)

(51) Cl.Int./Int.Cl. *G06F 16/26* (2019.01),
G06F 16/22 (2019.01)

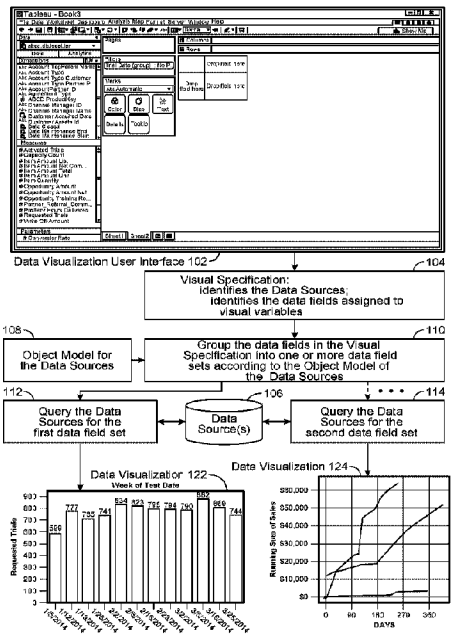
(72) Inventeurs/Inventors:
EUBANK, CHRISTIAN, US;
TALBOT, JUSTIN, US

(73) Propriétaire/Owner:
TABLEAU SOFTWARE, LLC, US

(74) Agent: FASKEN MARTINEAU DUMOULIN LLP

(54) Titre : UTILISATION D'UNE AGREGATION DE MESURES APPROPRIEES POUR GENERER DES VISUALISATIONS DE DONNEES D'ENSEMBLES DE DONNEES MULTI-FAITS

(54) Title: UTILIZING APPROPRIATE MEASURE AGGREGATION FOR GENERATING DATA VISUALIZATIONS OF MULTI-FACT DATASETS



(57) Abrégé/Abstract:
A computer receives a visual specification, which specifies a data source, visual variables, and data fields from the data source. Each visual variable is associated with either data fields (e.g., dimension and/or measures) or filters. The computer obtains a data model encoding the data source as a tree of related logical tables. Each logical table includes logical fields, each of which corresponds to either a data field or a calculation that spans logical tables. The computer generates a dimension subquery for the dimensions and the filters. The computer also generates, for each measure, an aggregated measure sub query grouped by the dimensions. The computer forms a final query by joining the dimension sub query to each of the aggregated measure subqueries. The computer subsequently executes the final query and displays a data visualization according to the results of the final query.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 March 2021 (18.03.2021)

WIPO | PCT

(10) International Publication Number
WO 2021/050182 A1

(51) International Patent Classification:
G06F 16/242 (2019.01) *G06F 16/26* (2019.01)

(21) International Application Number:
PCT/US2020/045461

(22) International Filing Date:
07 August 2020 (07.08.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
16/570,969 13 September 2019 (13.09.2019) US

(71) Applicant: **TABLEAU SOFTWARE, INC.** [US/US]; 1621 N. 34th Street, Seattle, WA 98103 (US).

(72) Inventors: **EUBANK, Christian**; 1621 N. 34th Street, Seattle, WA 98103 (US). **TALBOT, Justin**; 1621 N. 34th Street, Seattle, WA 98103 (US).

(74) Agent: **SANKER, David, V.** et al.; Morgan Lewis & Bockius LLP, 1400 Page Mill Road, Palo Alto, CA 94304 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) **Title:** UTILIZING APPROPRIATE MEASURE AGGREGATION FOR GENERATING DATA VISUALIZATIONS OF MULTI-FACT DATASETS

(57) **Abstract:** A computer receives a visual specification, which specifies a data source, visual variables, and data fields from the data source. Each visual variable is associated with either data fields (e.g., dimension and/or measures) or filters. The computer obtains a data model encoding the data source as a tree of related logical tables. Each logical table includes logical fields, each of which corresponds to either a data field or a calculation that spans logical tables. The computer generates a dimension subquery for the dimensions and the filters. The computer also generates, for each measure, an aggregated measure sub query grouped by the dimensions. The computer forms a final query by joining the dimension sub query to each of the aggregated measure subqueries. The computer subsequently executes the final query and displays a data visualization according to the results of the final query.

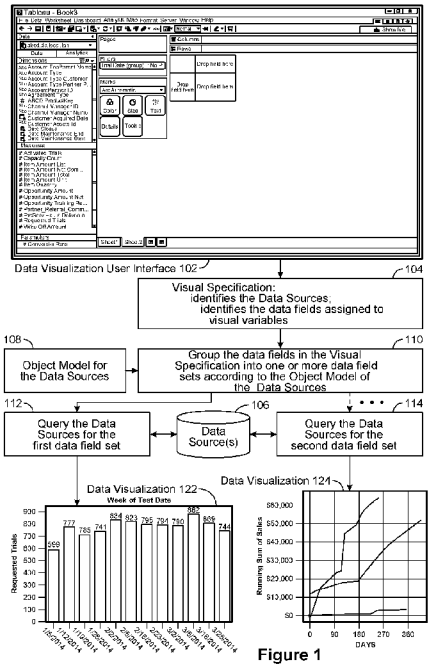


Figure 1

WO 2021/050182 A1

Utilizing Appropriate Measure Aggregation for Generating Data Visualizations of Multi-fact Datasets

TECHNICAL FIELD

[0001] The disclosed implementations relate generally to data visualization and more specifically to interactive visual analysis of a data set using an object model of the data set.

BACKGROUND

[0002] Data visualization applications enable a user to understand a data set visually, including distribution, trends, outliers, and other factors that are important to making business decisions. Some data elements are computed based on data from the selected data set. For example, data visualizations frequently use sums to aggregate data. Some data visualization applications enable a user to specify a "Level of Detail" (LOD), which can be used for the aggregate calculations. However, specifying a single Level of Detail for a data visualization is insufficient to build certain calculations.

[0003] Some data visualization applications provide a user interface that enables users to build visualizations from a data source by selecting data fields and placing them into specific user interface regions to indirectly define a data visualization. See, for example, U.S. Patent Application Serial No. 10/453,834, filed June 2, 2003, entitled "Computer Systems and Methods for the Query and Visualization of Multidimensional Databases," now U.S. Patent No. 7,089,266. However, when there are complex data sources and/or multiple data sources, it may be unclear what type of data visualization to generate (if any) based on a user's selections.

[0004] In addition, some systems construct queries that yield data visualizations that are not what a user expects. In some cases, some rows of data are omitted (e.g., when there is no corresponding data in one of the fact tables). In some cases, numeric aggregated fields produce totals that are overstated because the same data value is being counted multiple times. These problems can be particularly problematic because an end user may not be aware of the problem and/or not know what is causing the problem.

Date Reçue/Date Received 2023-07-20

SUMMARY

[0005] Generating a data visualization that combines data from multiple tables can be challenging, especially when there are multiple fact tables. In some cases, it can help to construct an object model of the data before generating data visualizations. In some instances, one person is a particular expert on the data, and that person creates the object model. By storing the relationships in an object model, a data visualization application can leverage that information to assist all users who access the data, even if they are not experts.

[0006] An object is a collection of named attributes. An object often corresponds to a real-world object, event, or concept, such as a Store. The attributes are descriptions of the object that are conceptually at a 1:1 relationship with the object. Thus, a Store object may have a single [Manager Name] or [Employee Count] associated with it. At a physical level, an object is often stored as a row in a relational table, or as an object in JSON.

[0007] A class is a collection of objects that share the same attributes. It must be analytically meaningful to compare objects within a class and to aggregate over them. At a physical level, a class is often stored as a relational table, or as an array of objects in JSON.

[0008] An object model is a set of classes and a set of many-to-one relationships between them. Classes that are related by 1-to-1 relationships are conceptually treated as a single class, even if they are meaningfully distinct to a user. In addition, classes that are related by 1-to-1 relationships may be presented as distinct classes in the data visualization user interface. Many-to-many relationships are conceptually split into two many-to-one relationships by adding an associative table capturing the relationship.

[0009] Once an object model is constructed, a data visualization application can assist a user in various ways. In some implementations, based on data fields already selected and placed onto shelves in the user interface, the data visualization application can recommend additional fields or limit what actions can be taken to prevent unusable combinations. In some implementations, the data visualization application allows a user considerable freedom in selecting fields, and uses the object model to build one or more data visualizations according to what the user has selected.

[0010] In accordance with some implementations, a method generates data visualizations. The method is performed at a computer having one or more processors and memory. The memory stores one or more programs configured for execution by the one or more processors. The computer receives a visual specification, which specifies a data source,

a plurality of visual variables, and a plurality of data fields from the data source. Each of the visual variables is associated with either (i) a respective one or more of the data fields or (ii) one or more filters, and each of the data fields is identified as either a dimension or a measure. The computer obtains a data model (or object model) encoding the data source as a tree of logical tables. Each logical table has its own physical representation and includes a respective one or more logical fields. Each logical field corresponds to either a data field or a calculation that spans one or more logical tables. Each edge of the tree connects two logical tables that are related. The computer generates a dimension subquery based on logical tables that supply the data fields for the dimensions and the filters. The computer also generates, for each measure, based on the logical tables that supply the data fields for the respective measure and the filters, an aggregated measure subquery grouped by the dimensions. The computer forms a final query by joining, using the dimensions, the dimension subquery to each of the aggregated measure subqueries. The computer subsequently executes the final query against the data source to retrieve tuples that comprise distinct ordered combinations of data values for the data fields. The computer then builds and displays a data visualization according to the data fields in the tuples and according to the visual variables to which each of the data fields is associated.

[0011] In some implementations, the computer generates each aggregated measure subquery by performing a sequence of operations. The computer computes a measure sub-tree of the tree of logical tables. The measure sub-tree is a minimum sub-tree required to supply the data fields for a respective measure. The computer also computes a dimension-filter sub-tree of the tree of logical tables. The dimension-filter sub-tree is a minimum sub-tree required to supply all of the physical inputs for the dimensions and the filters. When the dimension-filter sub-tree does not share any logical table with the measure sub-tree, the computer adds a neighboring logical table from the measure sub-tree to the dimension-filter sub-tree. The computer compiles the measure sub-tree to obtain a measure join tree and compiles the dimension-filter sub-tree to obtain a dimension-filter join tree. The computer layers calculations and filters over the measure join tree and the dimension-filter join tree to obtain an updated measure sub-tree and an updated dimension-filter sub-tree, respectively. The computer de-duplicates the updated dimension-filter sub-tree by applying a group-by operation that uses the dimensions and linking fields, which include (i) keys from relationships between the logical tables and (ii) data fields of calculations shared with the measure sub-tree, to obtain a de-duplicated dimension-filter sub-tree. The computer combines the de-duplicated

dimension-filter sub-tree with the updated measure sub-tree to obtain the aggregated measure subquery.

[0012] In some implementations, the computer compiles the measure sub-tree by inner joining logical tables in the measure sub-tree to obtain the measure join tree.

[0013] In some implementations, the computer computes the dimension-filter sub-tree by performing a sequence of operations. The computer inner joins logical tables in the dimension-filter sub-tree that are shared with the measure sub-tree, and left-joins (also referred to as left outer joins) logical tables in the dimension-filter sub-tree that are not shared with the measure sub-tree, to obtain the dimension-filter join tree.

[0014] In some implementations, the computer combines the de-duplicated dimension-filter sub-tree with the updated measure sub-tree by performing a sequence of operations. The computer determines if the de-duplicated dimension-filter sub-tree contains a filter. When the de-duplicated dimension-filter sub-tree contains a filter, the computer inner-joins the updated measure-sub-tree with the de-duplicated dimension-filter sub-tree. When the de-duplicated dimension-filter sub-tree does not contain a filter, the computer left outer-joins the updated measure-sub-tree with the de-duplicated dimension-filter sub-tree.

[0015] In some implementations, the computer determines if the keys indicate a many-to-one relationship or a one-to-one relationship between a first logical table and a second logical table. When the keys indicate a many-to-one relationship between the first logical table and the second logical table, the computer includes the first table and the second table in the measure sub-tree, thereby avoiding the group-by in the de-duplication operation for the first logical table and the second logical table.

[0016] In some implementations, when the dimension-filter sub-tree joins against the measure sub-tree exclusively along many-to-one and one-to-one links, the computer replaces tables shared by the measure sub-tree and the dimension-filter sub-tree with the de-duplicated dimension-filter sub-tree.

[0017] In some implementations, the computer generates the dimension subquery by inner-joining a first one or more logical tables in the tree of logical tables. Each logical table of the first one or more logical tables supplies the data fields for a dimension and/or a filter.

[0018] In some implementations, the computer forms the final query by joining the dimensions subquery and the aggregated measure subqueries on the dimensions using outer joins, and applying a COALESCE after each outer join.

[0019] In some implementations, when the visualization has no dimensions, the computer performs a full join between the aggregated measure subqueries to form the final query.

[0020] In accordance with some implementations, a system for generating data visualizations includes one or more processors, memory, and one or more programs stored in the memory. The programs are configured for execution by the one or more processors. The programs include instructions for performing any of the methods described herein.

[0021] In accordance with some implementations, a non-transitory computer readable storage medium stores one or more programs configured for execution by a computer system having one or more processors and memory. The one or more programs include instructions for performing any of the methods described herein.

[0022] Thus methods, systems, and graphical user interfaces are provided for interactive visual analysis of a data set.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] For a better understanding of the aforementioned implementations of the invention as well as additional implementations, reference should be made to the Description of Implementations below, in conjunction with the following drawings in which like reference numerals refer to corresponding parts throughout the figures.

[0024] Figure 1 illustrates conceptually a process of building a data visualization in accordance with some implementations.

[0025] Figure 2 is a block diagram of a computing device according to some implementations.

[0026] Figure 3 is a block diagram of a data visualization server according to some implementations.

[0027] Figure 4 provides an example data visualization user interface according to some implementations.

[0028] Figure 5A illustrates an example data model (or object model), in accordance with some implementations.

[0029] Figure 5B illustrates a data visualization, in accordance with some implementations.

[0030] Figure 5C illustrates a data visualization, in accordance with some implementations.

[0031] Figure 6A illustrates an example data model or object model, in accordance with some implementations.

[0032] Figure 6B illustrates a data visualization, in accordance with some implementations.

[0033] Figure 7 illustrates an example query, in accordance with some implementations.

[0034] Figure 8A illustrates a data visualization, in accordance with some implementations.

[0035] Figure 8B illustrates an example query, in accordance with some implementations.

[0036] Figure 9A illustrates a data visualization, in accordance with some implementations.

[0037] Figure 9B illustrates an example query, in accordance with some implementations.

[0038] Figures 10A and 10B illustrates example queries, in accordance with some implementations.

[0039] Figure 11A illustrates a data visualization, in accordance with some implementations.

[0040] Figure 11B illustrates an example query, in accordance with some implementations.

[0041] Figures 11C-11F illustrate data visualizations, in accordance with some implementations.

[0042] Figure 12A illustrates a data visualization, in accordance with some implementations.

[0043] Figure 12B-12D illustrates example queries, in accordance with some implementations.

[0044] Figures 13A-13D provide a flowchart of a method for generating data visualizations using an object model, in accordance with some implementations.

[0045] Like reference numerals refer to corresponding parts throughout the drawings.

[0046] Reference will now be made in detail to implementations, examples of which are illustrated in the accompanying drawings. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one of ordinary skill in the art that the present invention may be practiced without these specific details.

DESCRIPTION OF IMPLEMENTATIONS

[0047] Some implementations of an interactive data visualization application use a data visualization user interface 102 to build a visual specification 104, as shown in Figure 1. The visual specification identifies one or more data source 106, which may be stored locally (e.g., on the same device that is displaying the user interface 102) or may be stored externally (e.g., on a database server or in the cloud). The visual specification 104 also includes visual variables. The visual variables specify characteristics of the desired data visualization indirectly according to selected data fields from the data sources 106. In particular, a user assigns zero or more data fields to each of the visual variables, and the values of the data fields determine the data visualization that will be displayed.

[0048] In most instances, not all of the visual variables are used. In some instances, some of the visual variables have two or more assigned data fields. In this scenario, the order of the assigned data fields for the visual variable (e.g., the order in which the data fields were assigned to the visual variable by the user) typically affects how the data visualization is generated and displayed.

[0049] Some implementations use an object model 108 (sometimes called a data model) to build the appropriate data visualizations. In some instances, an object model applies to one data source (e.g., one SQL database or one spreadsheet file), but an object model may encompass two or more data sources. Typically, unrelated data sources have distinct object

models. In some instances, the object model closely mimics the data model of the physical data sources (e.g., classes in the object model corresponding to tables in a SQL database). However, in some cases the object model is more normalized (or less normalized) than the physical data sources. An object model groups together attributes (e.g., data fields) that have a one-to-one relationship with each other to form classes, and identifies many-to-one relationships among the classes. In the illustrations below, the many-to-one relationships are illustrated with arrows, with the "many" side of each relationship vertically lower than the "one" side of the relationship. The object model also identifies each of the data fields (attributes) as either a dimension or a measure. In the following, the letter "D" (or "d") is used to represent a dimension, whereas the latter "M" (or "m") is used to represent a measure. When an object model 108 is constructed, it can facilitate building data visualizations based on the data fields a user selects. Because a single object model can be used by an unlimited number of other people, building the object model for a data source is commonly delegated to a person who is a relative expert on the data source,

[0050] As a user adds data fields to the visual specification (e.g., indirectly by using the graphical user interface to place data fields onto shelves), the data visualization application 222 (or web application 322) groups (110) together the user-selected data fields according to the object model 108. Such groups are called data field sets. In many cases, all of the user-selected data fields are in a single data field set. In some instances, there are two or more data field sets. Each measure m is in exactly one data field set, but each dimension d may be in more than one data field set.

[0051] The data visualization application 222 (or web application 322) queries (112) the data sources 106 for the first data field set, and then generates a first data visualization 122 corresponding to the retrieved data. The first data visualization 122 is constructed according to the visual variables 282 in the visual specification 104 that have assigned data fields 284 from the first data field set. When there is only one data field set, all of the information in the visual specification 104 is used to build the first data visualization 122. When there are two or more data field sets, the first data visualization 122 is based on a first visual sub-specification consisting of all information relevant to the first data field set. For example, suppose the original visual specification 104 includes a filter that uses a data field f. If the field f is included in the first data field set, the filter is part of the first visual sub-specification, and thus used to generate the first data visualization 122.

[0052] When there is a second (or subsequent) data field set, the data visualization application 222 (or web application 322) queries (114) the data sources 106 for the second (or subsequent) data field set, and then generates the second (or subsequent) data visualization 124 corresponding to the retrieved data. This data visualization 124 is constructed according to the visual variables 282 in the visual specification 104 that have assigned data fields 284 from the second (or subsequent) data field set.

[0053] Figure 2 is a block diagram illustrating a computing device 200 that can execute the data visualization application 222 or the data visualization web application 322 to display a data visualization 122. In some implementations, the computing device displays a graphical user interface 102 for the data visualization application 222. Computing devices 200 include desktop computers, laptop computers, tablet computers, and other computing devices with a display and a processor capable of running a data visualization application 222. A computing device 200 typically includes one or more processing units/cores (CPUs) 202 for executing modules, programs, and/or instructions stored in the memory 214 and thereby performing processing operations; one or more network or other communications interfaces 204; memory 214; and one or more communication buses 212 for interconnecting these components. The communication buses 212 may include circuitry that interconnects and controls communications between system components. A computing device 200 includes a user interface 206 comprising a display 208 and one or more input devices or mechanisms 210. In some implementations, the input device/mechanism includes a keyboard; in some implementations, the input device/mechanism includes a "soft" keyboard, which is displayed as needed on the display 208, enabling a user to "press keys" that appear on the display 208. In some implementations, the display 208 and input device / mechanism 210 comprise a touch screen display (also called a touch sensitive display). In some implementations, the display is an integrated part of the computing device 200. In some implementations, the display is a separate display device.

[0054] In some implementations, the memory 214 includes high-speed random-access memory, such as DRAM, SRAM, DDR RAM or other random-access solid-state memory devices. In some implementations, the memory 214 includes non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid-state storage devices. In some implementations, the memory 214 includes one or more storage devices remotely located from the CPUs 202. The memory 214, or alternatively the non-volatile memory devices within the memory 214, comprises a non-

transitory computer-readable storage medium. In some implementations, the memory 214, or the computer-readable storage medium of the memory 214, stores the following programs, modules, and data structures, or a subset thereof:

- an operating system 216, which includes procedures for handling various basic system services and for performing hardware dependent tasks;
- a communication module 218, which is used for connecting the computing device 200 to other computers and devices via the one or more communication network interfaces 204 (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on;
- a web browser 220 (or other client application), which enables a user to communicate over a network with remote computers or devices;
- a data visualization application 222, which provides a graphical user interface 102 for a user to construct visual graphics (e.g., an individual data visualization or a dashboard with a plurality of related data visualizations). In some implementations, the data visualization application 222 executes as a standalone application (e.g., a desktop application). In some implementations, the data visualization application 222 executes within the web browser 220 (e.g., as a web application 322);
- a graphical user interface 102, which enables a user to build a data visualization by specifying elements visually, as illustrated in Figure 4 below;
- in some implementations, the user interface 102 includes a plurality of shelf regions 250, which are used to specify characteristics of a desired data visualization. In some implementations, the shelf regions 250 include a columns shelf 230 and a rows shelf 232, which are used to specify the arrangement of data in the desired data visualization. In general, fields that are placed on the columns shelf 230 are used to define the columns in the data visualization (e.g., the x-coordinates of visual marks). Similarly, the fields placed on the rows shelf 232 define the rows in the data visualization (e.g., the y-coordinates of the visual marks). In some implementations, the shelf regions 250 include a filters shelf 262, which enables a user to limit the data viewed according to a selected data field (e.g., limit the data to rows for which a certain field has a specific value or has values in a specific range). In some implementations, the shelf regions 250 include a marks shelf 264, which is used to

specify various encodings of data marks. In some implementations, the marks shelf 264 includes a color encoding icon 270 (to specify colors of data marks based on a data field), a size encoding icon 272 (to specify the size of data marks based on a data field), a text encoding icon (to specify labels associated with data marks), and a view level detail icon 228 (to specify or modify the level of detail for the data visualization);

- visual specifications 104, which are used to define characteristics of a desired data visualization. In some implementations, a visual specification 104 is built using the user interface 102. A visual specification includes identified data sources 280 (i.e., specifies what the data sources are), which provide enough information to find the data sources 106 (e.g., a data source name or network full path name). A visual specification 104 also includes visual variables 282, and the assigned data fields 284 for each of the visual variables. In some implementations, a visual specification has visual variables corresponding to each of the shelf regions 250. In some implementations, the visual variables include other information as well, such as context information about the computing device 200, user preference information, or other data visualization features that are not implemented as shelf regions (e.g., analytic features);
- one or more object models 108, which identify the structure of the data sources 106. In an object model, the data fields (attributes) are organized into classes, where the attributes in each class have a one-to-one correspondence with each other. The object model also includes many-to-one relationships between the classes. In some instances, an object model maps each table within a database to a class, with many-to-one relationships between classes corresponding to foreign key relationships between the tables. In some instances, the data model of an underlying source does not cleanly map to an object model in this simple way, so the object model includes information that specifies how to transform the raw data into appropriate class objects. In some instances, the raw data source is a simple file (e.g., a spreadsheet), which is transformed into multiple classes;
- a data visualization generator 290, which generates and displays data visualizations according to visual specifications. In accordance with some implementations, the data visualization generator 290 uses an object model 108 to generate queries 294

(e.g., dimension subqueries, aggregated measure subqueries, and/or final queries) and/or optimize queries using query optimizers 292. The details of the query generation and optimization techniques are described below in reference to Figures 5A-11, according to some implementations.

- visualization parameters 236, which contain information used by the data visualization application 222 other than the information provided by the visual specifications 104 and the data sources 106; and
- zero or more databases or data sources 106 (e.g., a first data source 106-1), which are used by the data visualization application 222. In some implementations, the data sources can be stored as spreadsheet files, CSV files, XML files, flat files, JSON files, tables in a relational database, cloud databases, or statistical databases.

[0055] Each of the above identified executable modules, applications, or set of procedures may be stored in one or more of the previously mentioned memory devices, and corresponds to a set of instructions for performing a function described above. The above identified modules or programs (i.e., sets of instructions) need not be implemented as separate software programs, procedures, or modules, and thus various subsets of these modules may be combined or otherwise re-arranged in various implementations. In some implementations, the memory 214 stores a subset of the modules and data structures identified above. In some implementations, the memory 214 stores additional modules or data structures not described above.

[0056] Although Figure 2 shows a computing device 200, Figure 2 is intended more as functional description of the various features that may be present rather than as a structural schematic of the implementations described herein. In practice, and as recognized by those of ordinary skill in the art, items shown separately could be combined and some items could be separated.

[0057] Figure 3 is a block diagram of a data visualization server 300 in accordance with some implementations. A data visualization server 300 may host one or more databases 328 or may provide various executable applications or modules. A server 300 typically includes one or more processing units/cores (CPUs) 302, one or more network interfaces 304, memory 314, and one or more communication buses 312 for interconnecting these components. In some implementations, the server 300 includes a user interface 306, which includes a display 308 and one or more input devices 310, such as a keyboard and a mouse. In some implementations,

the communication buses 312 includes circuitry (sometimes called a chipset) that interconnects and controls communications between system components.

[0058] In some implementations, the memory 314 includes high-speed random-access memory, such as DRAM, SRAM, DDR RAM, or other random-access solid-state memory devices, and may include non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid-state storage devices. In some implementations, the memory 314 includes one or more storage devices remotely located from the CPU(s) 302. The memory 314, or alternatively the non-volatile memory devices within the memory 314, comprises a non-transitory computer-readable storage medium.

[0059] In some implementations, the memory 314, or the computer-readable storage medium of the memory 314, stores the following programs, modules, and data structures, or a subset thereof:

- an operating system 316, which includes procedures for handling various basic system services and for performing hardware dependent tasks;
- a network communication module 318, which is used for connecting the server 300 to other computers via the one or more communication network interfaces 304 (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on;
- a web server 320 (such as an HTTP server), which receives web requests from users and responds by providing responsive web pages or other resources;
- a data visualization web application 322, which may be downloaded and executed by a web browser 220 on a user's computing device 200. In general, a data visualization web application 322 has the same functionality as a desktop data visualization application 222, but provides the flexibility of access from any device at any location with network connectivity, and does not require installation and maintenance. In some implementations, the data visualization web application 322 includes various software modules to perform certain tasks. In some implementations, the web application 322 includes a user interface module 324, which provides the user interface for all aspects of the web application 322. In some implementations, the user interface module 324 specifies shelf regions 250, as described above for a computing device 200;

- the data visualization web application also stores visual specifications 104 as a user selects characteristics of the desired data visualization. Visual specifications 104, and the data they store, are described above for a computing device 200;
- one or more object models 108, as described above for a computing device 200;
- a data visualization generator 290, which generates and displays data visualizations according to user-selected data sources and data fields, as well as one or more object models 108, which describe the data sources 106. The operation of the data visualization generator is described above with respect to a computing device 200;
- in some implementations, the web application 322 includes a data retrieval module 326, which builds and executes queries to retrieve data from one or more data sources 106. The data sources 106 may be stored locally on the server 300 or stored in an external database 328. In some implementations, data from two or more data sources may be blended. In some implementations, the data retrieval module 326 uses a visual specification 104 to build the queries, as described above for the computing device 200 in Figure 2;
- in some implementations, the memory stores visualization parameters 236, as described above fore the computing device 200; and
- one or more databases 328, which store data used or created by the data visualization web application 322 or data visualization application 222. The databases 328 may store data sources 106, which provide the data used in the generated data visualizations. Each data source 106 includes one or more data fields 330. In some implementations, the database 328 stores user preferences. In some implementations, the database 328 includes a data visualization history log 334. In some implementations, the history log 334 tracks each time the data visualization renders a data visualization.

[0060] The databases 328 may store data in many different formats, and commonly includes many distinct tables, each with a plurality of data fields 330. Some data sources comprise a single table. The data fields 330 include both raw fields from the data source (e.g., a column from a database table or a column from a spreadsheet) as well as derived data fields, which may be computed or constructed from one or more other fields. For example, derived data fields include computing a month or quarter from a date field, computing a span of time between two date fields, computing cumulative totals for a quantitative field, computing

percent growth, and so on. In some instances, derived data fields are accessed by stored procedures or views in the database. In some implementations, the definitions of derived data fields 330 are stored separately from the data source 106. In some implementations, the database 328 stores a set of user preferences for each user. The user preferences may be used when the data visualization web application 322 (or application 222) makes recommendations about how to view a set of data fields 330. In some implementations, the database 328 stores a data visualization history log 334, which stores information about each data visualization generated. In some implementations, the database 328 stores other information, including other information used by the data visualization application 222 or data visualization web application 322. The databases 328 may be separate from the data visualization server 300, or may be included with the data visualization server (or both).

[0061] In some implementations, the data visualization history log 334 stores the visual specifications 104 selected by users, which may include a user identifier, a timestamp of when the data visualization was created, a list of the data fields used in the data visualization, the type of the data visualization (sometimes referred to as a "view type" or a "chart type"), data encodings (e.g., color and size of marks), the data relationships selected, and what connectors are used. In some implementations, one or more thumbnail images of each data visualization are also stored. Some implementations store additional information about created data visualizations, such as the name and location of the data source, the number of rows from the data source that were included in the data visualization, version of the data visualization software, and so on.

[0062] Each of the above identified executable modules, applications, or sets of procedures may be stored in one or more of the previously mentioned memory devices, and corresponds to a set of instructions for performing a function described above. The above identified modules or programs (i.e., sets of instructions) need not be implemented as separate software programs, procedures, or modules, and thus various subsets of these modules may be combined or otherwise re-arranged in various implementations. In some implementations, the memory 314 stores a subset of the modules and data structures identified above. In some implementations, the memory 314 stores additional modules or data structures not described above.

[0063] Although Figure 3 shows a data visualization server 300, Figure 3 is intended more as a functional description of the various features that may be present rather than as a

structural schematic of the implementations described herein. In practice, and as recognized by those of ordinary skill in the art, items shown separately could be combined and some items could be separated. In addition, some of the programs, functions, procedures, or data shown above with respect to a server 300 may be stored or executed on a computing device 200. In some implementations, the functionality and/or data may be allocated between a computing device 200 and one or more servers 300. Furthermore, one of skill in the art recognizes that Figure 3 need not represent a single physical device. In some implementations, the server functionality is allocated across multiple physical devices that comprise a server system. As used herein, references to a "server" or "data visualization server" include various groups, collections, or arrays of servers that provide the described functionality, and the physical servers need not be physically collocated (e.g., the individual physical devices could be spread throughout the United States or throughout the world).

[0064] Figure 4 shows a data visualization user interface 102 in accordance with some implementations. The user interface 102 includes a schema information region 410, which is also referred to as a data pane. The schema information region 410 provides named data elements (e.g., field names) that may be selected and used to build a data visualization. In some implementations, the list of field names is separated into a group of dimensions and a group of measures (typically numeric quantities). Some implementations also include a list of parameters. The graphical user interface 102 also includes a data visualization region 412. The data visualization region 412 includes a plurality of shelf regions 250, such as a columns shelf region 230 and a rows shelf region 232. These are also referred to as the column shelf 230 and the row shelf 232. In addition, this user interface 102 includes a filters shelf 262, which may include one or more filters 424.

[0065] As illustrated here, the data visualization region 412 also has a large space for displaying a visual graphic. Because no data elements have been selected yet in this illustration, the space initially has no visual graphic.

[0066] A user selects one or more data sources 106 (which may be stored on the computing device 200 or stored remotely), selects data fields from the data source(s), and uses the selected fields to define a visual graphic. The data visualization application 222 (or web application 322) displays the generated graphic 122 in the data visualization region 412. In some implementations, the information the user provides is stored as a visual specification 104.

[0067] In some implementations, the data visualization region 412 includes a marks shelf 264. The marks shelf 264 allows a user to specify various encodings 426 of data marks. In some implementations, the marks shelf includes a color encoding icon 270, a size encoding icon 272, a text encoding icon 274, and/or a view level detail icon 228, which can be used to specify or modify the level of detail for the data visualization.

[0068] In some implementations, data visualization platforms enable users to build visualizations through drag and drop actions using a single logical table, even when the data comes from multiple physical tables. The logical table can be constructed by physical modeling, which can include pivots, joins, and unions. Tables combined through physical modeling represent logical tables themselves. In some data visualization platforms, such as Tableau, a query generation model automatically maps user actions to underlying queries of data from the physical tables.

[0069] In some implementations, an analyst creates an object model, an example of which is shown in Figure 5A, which has six logical tables. For the example in Figure 5A, each table has its own measure granularity, which is better modeled as a logical table, regardless of the actual physical storage of the data. The example object model includes a Line Items table 502 that has a join 514 to an Orders table 504 and another join 516 to a Products table 506. The example also shows the Orders table 504 having a join 518 to an Addresses table 508 and another join 520 to a Customers table 510. The Addresses table 508 has a join 522 to a States table 512.

[0070] Suppose a user creates the visualization 530 shown in Figure 5B. In this case, Sales 524 is a measure from the Line Items table 502 and Population 526 is a measure from the States table 512. While it is possible to derive the correct result for Sales, the data visualization has overstated measure values for Population. For this example, the population is indicated as billions of people 528 for some states. The reason for this duplication is that the data visualization framework queried all the tables joined together. The granularity of this join tree is that of Line Items 502. In other words, each row represents a line item and will contain a column containing the population of the state in which the line item occurred. Thus, SUM(Population) will yield the state's population multiplied by the number of line items for that state. This problem occurs because the six conceptually logical tables were treated as a single logical table.

[0071] One solution to fix the measure duplication is to use Level of Detail calculations. For instance, the calculation {Fixed [State (States)]: MIN([Population])} can be rewritten to get Population aggregated to its native granularity. Figure 5C illustrates a data visualization 540 after using the Level of Detail calculations 532, in accordance with some implementations. As shown, the population is correctly displayed in millions 536 (instead of in billions as incorrectly displayed in Figure 5B). A key downside to this approach, however, is that it requires the analyst to become aware of the duplication as well as understand the semantics of these calculations. In addition, the default axis label 534 is a complex pression rather than "Population." Disclosed implementations provide an alternative solution that is performed by the data visualization application automatically.

[0072] To overcome at least some of these problems, some implementations include a method for mapping drag and drop actions to more granular logical models. Instead of a single logical table, some implementations operate over a tree of logical tables where each node is a logical table (with its own physical representation), and each edge is a link between two tables.

[0073] Some implementations handle situations where primary keys for one or more logical tables are unknown or cannot be ascertained (without more complex analysis). In other words, the primary keys for the logical tables are missing. Primary keys are a powerful tool for recovering a table's granularity.

[0074] Some implementations handle multiple relationship cardinalities between logical tables. Relationships may be many-to-one, one-to-one, or many-to-many. Some implementations treat unknown relationship types as many-to-many. Some implementations use relationship information to recover primary keys. For instance, the fields in the "one" side of a relation contain the primary key.

[0075] In the following description, logical fields refer to either data fields that arise from underlying representations inside logical tables (e.g., fields from the physical database tables backing a logical table), or calculations with inputs that span logical tables.

[0076] Figure 6A illustrates a object model as a logical tree, according to some implementations. As indicated by the numerals, the tables correspond to the tables in the example shown in Figure 5A. Each logical table has a many-to-one relationship with its neighbor on the right. Sometimes, the relationships have unknown cardinalities.

[0077] Some implementations map user actions to visualizations with proper measure aggregation. Some implementations leverage the logical tree structure to generate a

visualization (an example of which is shown in Figure 6B) with measures aggregated at their native granularities. Thus, for the example described above in reference to Figure 5A, an analyst can obtain the proper Population values without having to add new calculations.

[0078] Some implementations calculate full domain values. To illustrate, suppose the full set of states is contained within the States table 512. For the visualization in Figure 6B, not all states may have had sales. As a result, if all the tables are joined together using an inner join, a visualization framework can drop states without sales (e.g., Alaska and Hawaii). One solution is to use a partial or full outer join to keep all the states. Some implementations generate a visualization that contains the full domain by first querying the logical tables that are necessary to compute the dimension values.

[0079] Some implementations ensure measure values are represented or preserved even as new dimensions are added. For instance, due to a missing or malformed foreign key, a sale may not have a state. If the tables were inner joined together, the sales values would get dropped. Some implementations avoid this problem by querying the tables needed to get the full measure values and using left joins to ensure that missing dimensions do not cause measures to get dropped. For the example above, Sales without states are encoded by the "Null" state.

[0080] Some implementations query fewer tables than would be necessary with solutions that do not use the tree of logical tables. For the example above, alternate frameworks would have queried a join tree of all six tables shown in Figure 6A. With the techniques disclosed herein, a data visualization framework can recognize that only the Line Items table 502, the Orders table 504, the Addresses table 508, and the States tables 512 have to be queried, since these are the logical tables that contain the dimension and measure values or are along the join paths for these tables.

[0081] Some implementations leverage or incorporate primary key and cardinality information when such information is available, although the techniques yield correct results even in the worst case (e.g., when all the links or relationships between the logical tables are many-to-many, or when there are no known primary keys for the logical tables). Some implementations incorporate such information to generate simpler queries.

[0082] Some implementations map a description of a visualization to a high-level query representation that includes dimensions, measures, and filters. Traditional implementations assume a single logical table while converting this representation into a lower-level query

representation. The techniques described herein, on the other hand, generate queries that encode the semantics of a tree of logical tables. Some implementations generate a subquery that contains the dimensions and a subquery for each aggregated measure (grouped by the dimensions). Some implementations join these subqueries together on the dimensions, as further described below.

Generating Subqueries

[0083] To generate a dimension subquery, some implementations join all the logical tables that contain a dimension field, or join these tables together, and group by the set dimensions. When generating the measure subquery, some implementations generate a flat table at the granularity of the measure that contains the measure's input fields (in the case of a logical measure) and the dimensions. Some implementations apply the aggregate with a group by on the dimensions. In the following discussions, a non-logical field (i.e., a non-calculated field) is sometimes called a physical field, and logical tables are sometimes called tables.

[0084] For each measure, some implementations generate the flat table at the measure's granularity using an algorithm. The algorithm includes collecting the physical input fields for the dimensions, the measure, and the filters. The algorithm also includes computing a minimum sub-tree (called the physical sub-tree) needed for all the physical input fields. The algorithm further includes computing another minimum subtree (called the measure-sub-tree) needed to supply all the physical inputs for the measure.

[0085] The algorithm further includes partitioning the sub-tree into sub-tree components. The trees emanating from the measure subtree are called the dimension-filter subtrees. At this point, the measure and dimension-filter subtrees are disjoint. It is possible that logical fields or filters will span into or across the measure subtree. In that case, the algorithm includes creating a dimension-measure subtree that merges one or more dimension-filter subtrees with the minimum set of tables (e.g., neighboring tables) from the measure subtree.

[0086] The algorithm also includes assigning the logical fields and filters to the subtrees that contains all their inputs. The algorithm further includes layering the logical fields and filters on top of the join tree consisting of all the tables in the subtree joined together. Some implementations inner join tables that are in the measure subtree and left outer join the other tables along the paths emanating from the measure tables.

[0087] Some implementations de-duplicate each dimension (and, if applicable, the dimension/measure) subtree on the dimensions and using linking fields. The structure of the de-duplication step is a "Group By" on a set of fields and a MAX on the rest of the fields. Some implementations use a set of linking fields that include (i) physical fields (sometimes called physical input fields or data fields) from the measure tables needed to compute logical dimension fields and filters, and (ii) relationship fields that link this subtree against the measure subtree. Some implementations left outer join all the subtrees, starting from the measure subtree, together. If a dimension subtree has a filter, then some implementations add a constant calculation to the dimension subtree and add a filter on top of the join to determine that this calculation is not null.

[0088] The following example illustrates an application of the algorithm described above, according to some implementations. For the visualization example discussed above in reference to Figures 5A-5C and 6A-6B, the pre-aggregate sub-query for States and Population is simply the States table 512. The query for State and Sales is shown in Figure 7, in accordance with some implementations. The physical subtree consists of the Line Items table 502 (for the measure), the States table 512 (for the dimension), as well as the Orders table 504, and the Addresses table 508, since these tables are needed for supplying the physical input fields. For this example, the measure subtree is Line Items and there is a single dimension subtree that contains the other three tables. Some implementations group (704) the dimension subtree by the dimension (States) and the key (indicated as 'PK' for primary key) from the spanning relationship between the Line Items table 502 and the dimension subtree. In this case, the dimension subtree joins (702) via the link between the Line Items table 502 and the Orders table 504. Some implementations join the measure and dimension subtrees via this link. The joins 702, 706, and 708 are left outer joins, emanating from the measure subtree, to ensure that rows in the Line Items table with missing states are not lost.

[0089] Referring next to Figures 8A and 8B, suppose a user created a calculation that spanned logical tables, such as [Full City Name] = [City Name] + ", " + [State Name], where [City Name] comes from the Addresses table 508 and [State Name] came from the States table 512. Figure 8A illustrates a data visualization 800 in this scenario, in accordance with some implementations. The query for the visualization is shown in Figure 8B, in accordance with some implementations. The physical join tree includes the States table 512 (for the measure and dimension) and the Addresses table 508 (for the dimension).

[0090] For this example, the measure subtree is States, and the dimension subtree includes a logical dimension field 804 that spans the Addresses table 508 and the States table 512. Thus, this example shown an instance of a dimension-measure subtree. Some implementations join (802) this subtree to the measure subtree using the relationships between the measure tables and rest of the tables in the dimension-measure subtree. Some implementations de-duplicate (806) the dimension-measure subtree using the dimension (Full City Name), the joining relationship (State foreign key (FK)), as well as the physical input of the dimension that falls in the measure part of the dimension-measure subtree (State Name). Subsequently, some implementations join (808) using the key from the relationship and the physical input field.

[0091] Referring next to Figures 9A and 9B, suppose a user wants to create a visualization 900 of sum of Sales, by Ship Mode, filtered to the "Technology" category. Some implementations generate the query shown in Figure 9B. The physical join tree is the Line Items table 502 (for the measure), the Orders table 504 (for the dimension), and the Products table 506 (for the filter). The measure subtree is the Line Items table 502. In this case, there are two disjoint dimension subqueries: Orders (for the dimension) and Products (for the filter 906). Some implementations left outer join (902) the measure subquery (sometimes called the measure subtree) and a de-duplicated Ship Mode dimension subquery 904 on the keys from the relationship. Next, some implementations left join (912) this result with the de-duplicated dimension subquery 908. Some implementations add a sentinel calculation (e.g., the sentinel 910) on top of filters (e.g., the filters 906 and 914). Some implementations add a filter to only keep rows for which the sentinel value is non-null to ensure that the filter is respected, given the left outer joins. Some implementations swap the order in which the dimension and dimension-measure subgraphs are joined while obtaining the same results.

[0092] Figures 10A and 10B illustrate examples of optimized queries, in accordance with some implementations. Some implementations generate optimized queries when information about relationship cardinalities and/or primary keys is available.

[0093] In some implementations, when computing the measure subtree component, the system pulls in tables that can be reached via chains of many-to-one or one-to-one links. To illustrate, for the States and Line Items subquery described above in reference to Figure 7, some implementations leverage the fact that all the links are many-to-one to expand the measure subtree to include all the tables. This query, illustrated in Figure 10A, is simpler than before.

In particular, unlike the query in Figure 7, the query shown in Figure 10A does not have any "Group-By" operations (e.g., the operation 704). Some implementations perform this optimization without knowing all the cardinalities of the edges. For example, the optimization requires only knowing that the three relationships between the tables 502, 504, 508, and 512, are many-to-one, but it does not require knowing any information about the rest of the tree (the object model shown in the Figure 5A). Even with partial information, some implementations generate optimized queries. For example, if it were known that the Line Items – Orders link was many-to-one, some implementations perform partial optimization by including the Line Items and Orders tables in the measure subtree, and the Addresses table joined to the States tables in the measure subtree.

[0094] In some implementations, if the dimension/measure subtree joins against the measure subtree exclusively along many-to-one and one-to-one links, then when computing the measure subtree, the set of tables shared by the measure and dimension/measure subtree is replaced with the de-duplicated dimension/measure subtree. To illustrate, for the query in the example described above in reference to Figure 8B, if it is known that the relationship between Addresses and States is many-to-one, some implementations simplify the query as shown in Figure 10B. This optimization is based on the fact that the dimension-measure subtree links to the measure subtree exclusively via many-to-one links (in this case Addresses-States).

Combining Subqueries

[0095] Given a dimension subtree and a subtree for each aggregated measure, some implementations combine these queries to form a final query using outer joins. Some implementations join on the dimensions in the visualization and, after each join, apply a COALESCE on the left and right instances of each dimension. Figure 11A illustrates a data visualization 1100 that displays sums of Population and Sales (described above), grouped by Region and Category. Some implementations combine the subqueries to form a final query 1102, as shown in Figure 11B, for the visualization shown in Figure 11A. In some implementations, each subquery (e.g., the subqueries 1104 and 1106) has a different domain. Figure 11C illustrates a data visualization 1120 for the dimension subquery (for the example in Figure 11B), in accordance with some implementations. Figure 11D illustrates a data visualization 1130 for the subquery for Sum of Sales (with the dimensions), according to some implementations. Figure 11E illustrates a data visualization 1140 for the subquery for Sum of Population (with the dimensions), according to some implementations.

[0096] In some implementations, outer joins ensure that all combinations of the dimensions that appear in at least one subquery are represented. The coalesces ensure that after a join, that all non-null values for each dimension are represented. For instance, (Region, Category) = (Central, Null) only appears on the right side of the outer join when joining in the Population subquery against the rest of the query. If the left side version of the dimensions is chosen, this would result in the erroneous result ((Null, Null). Similarly, if the right side version of the dimensions is chosen, then that would result in (Null, Furniture) from the Sum of Sales subquery.

[0097] Some implementations perform full joins between the measure subqueries for visualizations without dimensions (e.g., for the visualization 1150 shown in Figure 11F corresponding to sums of Sales and Population).

[0098] Some implementations generate visualizations based on the object model for complex queries. To illustrate, suppose a user created a calculation that spanned logical tables such as [Tax Adjusted Sales] = [Sales] * [Sales Tax Rate]. Here, [Sales] comes from the Line Items table 502 and [Sales Tax Rate] comes from the States table 512. Suppose the query also includes a filter predicate calculation of [Segment] = 'Home Office' AND [Region] = 'East', where [Segment] from the Customers table 510 and [Region] comes from the Addresses table 508. Now, further suppose that the user wants to create the visualization SUM([Tax Adjusted Sales]) grouped by Category, where the filter predicate is true. Figure 12A illustrates a data visualization 1200 generated using the techniques described herein, according to some implementations.

[0099] Assuming the relationship cardinalities are unknown, some implementations generate a final query 1202 shown in Figure 12B to compute the measure at its native granularity. For this example, the physical join tree includes the Line Items table 502, the Orders table 504, the Addresses table 508, and the States table 512 (for the measure), the Products table 506 (for a dimension) and the Customers table 510 (which we need for the filter). The measure is a logical field that spans the Line Items, Orders, Addresses and States tables. The measure subtree comprises these tables inner joined together. Some implementations also layer on the definition for [Tax Adjusted Sales].

[00100] Some implementations start with two dimension subtrees: Products and Customers. Since the filter on the predicate spans from Customers across to Addresses, some implementations generate a dimension-measure subtree of Orders, Addresses and Customers.

For the subtree corresponding to Category, some implementations group Products by Category and the linking key (Product PK) and left join this to the measure subtree. For the dimension-measure subtree, some implementations inner join the tables from the measure subtree together and the left join Customers. Some implementations add the filter predicate logical field and apply the filter. The filter predicate is a calculation with a physical input in the measure subtree [Region]. Therefore, some implementations de-duplicate this subtree on the relationship fields from the Customer-Orders link (since that is the link at which the dimension-only objects links to the measure objects) as well as [Region]. Some implementations join the dimension-measure subtree against the rest of the query using these fields as well.

[00101] Some implementations simplify the query described above in reference to Figure 12B. If it is known that the Orders-Customers link is many-to-one, some implementations simplify the query by eliminating the need of the dimension-subquery (because it is known that Customers can be safely joined to the measure subtree without impacting the granularity, as described above in reference to Figure 10A). Figure 12C illustrates an example of the optimized query 1204, according to some implementations. Similarly, if it is known that the Line Items-Products link is many-to-one, some implementations apply similar logic to reduce or simplify the query even further as shown by the optimized query 1206 in Figure 12D.

[00102] Figures 13A-13D provide a flowchart of a method 1300 for generating (1302) data visualizations using an object model according to the techniques described above, in accordance with some implementations. The method 1300 is performed (1304) at a computing device 200 having one or more processors and memory. The memory stores (1306) one or more programs configured for execution by the one or more processors.

[00103] The computer receives (1308) a visual specification 104, which specifies one or more data sources 106, a plurality of visual variables 282, and a plurality of data fields 284 from the one or more data sources 106. Each of the visual variables 282 is associated with either (i) a respective one or more of the data fields 284 or (ii) one or more filters, and each of the data fields 284 is identified as either a dimension or a measure. In some implementations, the visual specification 104 includes one or more additional visual variables that are not associated with any data fields 330 from the one or more data sources 106. In some implementations, each of the visual variables 282 is one of: rows attribute, columns attribute, filter attribute, color encoding, size encoding, shape encoding, or label encoding.

[00104] The computer obtains (1310) a data model encoding the data source as a tree of logical tables. Each logical table has its own physical representation and includes a respective one or more logical fields. Each logical field corresponds to either a data field or a calculation that spans one or more logical tables. Each edge of the tree connects two logical tables that are related. The computer generates (1312) a dimension subquery based on logical tables that supply the data fields for the dimensions and the filters. In some implementations, the computer generates the dimension subquery by inner-joining (1314) a first one or more logical tables in the tree of logical tables, wherein each logical table of the first one or more logical tables supplies the data fields for a dimension or a filter.

[00105] The computer also generates (1316), for each measure, based on the logical tables that supply the data fields for the respective measure and the filters, an aggregated measure subquery grouped by the dimensions.

[00106] Referring next to Figure 13B, the computer forms (1318) a final query by joining, using the dimensions, the dimension subquery to each of the aggregated measure subqueries. In some implementations, the computer forms the final query by joining (1320) the dimensions subquery and the aggregated measure subqueries on the dimensions using outer joins, and applying a COALESCE after each outer join. In some implementations, when the visualization has no dimensions, the computer performs (1322) a full join between the aggregated measure subqueries to form the final query. Some implementations use a special table (called Table Dee in some Tableau products) with an empty schema and a single row to represent visualizations without dimensions, and determine if a given visualization has no dimensions by checking if a base table in the dimensions subquery is the special table. Some implementations also use the special table for constant values. For example, SUM(1) is evaluated to the value 1 projected on top of the special table. Since the special table has one row, SUM(1) evaluates to the value 1. In some implementations, if none of a logical field's inputs belong to any table, the field is evaluated using the special table.

[00107] The computer subsequently executes (1324) the final query against the data source to retrieve tuples that comprise distinct ordered combinations of data values for the data fields. The computer then builds and displays (1326) a data visualization (e.g., in the graphical user interface 102 for the computing device 200) according to the data fields in the tuples and according to the visual variables to which each of the data fields is associated.

[00108] Referring next to Figure 13C, in some implementations, the computer generates each aggregated measure subquery by performing a sequence of operations. The computer computes (1328) a measure sub-tree of the tree of logical tables. The measure sub-tree is a minimum sub-tree required to supply the data fields for a respective measure. In some implementations, the computer compiles (1330) the measure sub-tree by inner joining logical tables in the measure sub-tree to obtain the measure join tree. Using an inner-join helps ensure that the order of joins does not matter, thereby providing consistent semantics when computing granularity. Inner joins provide native-level granularity for measures. There will be multiple tables in the measure sub-tree only if it is a calculation. Most of the time, though, the measure sub-tree includes a single table. When a calculated field spans multiple tables, the field's native granularity is its inputs joined together. The computer also computes (1332) a dimension-filter sub-tree from the tree of logical tables. The dimension-filter sub-tree is a minimum sub-tree required to supply all the physical inputs for the dimensions and the filters. (For a non-calculated dimension field, the physical input is the dimension field itself. For a calculated dimension, the physical inputs are all the data fields needed to calculate the dimension.) In some implementations, the computer computes the dimension-filter sub-tree by inner joining (1334) logical tables in the dimension-filter sub-tree that are shared with the measure sub-tree, and left-joining logical tables in the dimension-filter sub-tree that are not shared with the measure sub-tree, to obtain the dimension-filter join tree. Suppose there is a calculated measure that spans multiple tables in a many-to-one relationship. There needs to be separate instances of that calculation for every distinct combination of rows that can be combined. The inner-join produces that set of rows.

[00109] When the dimension-filter sub-tree does not share any logical table with the measure sub-tree, the computer adds (1336) a neighboring logical table from the measure sub-tree to the dimension-filter sub-tree. The computer compiles (1338) the measure sub-tree to obtain a measure join tree and compiles the dimension-filter sub-tree to obtain a dimension-filter join tree. Referring next to Figure 13D, the computer layers (1340) calculations and filters over the measure join tree and the dimension-filter join tree to obtain an updated measure sub-tree and an updated dimension-filter sub-tree, respectively. The computer de-duplicates (1342) the updated dimension-filter sub-tree by applying a Group-By operation (GB) that uses the dimensions and linking fields that include (i) keys from relationships (e.g., the primary key is equal to the foreign key) between the logical tables and (ii) data fields of calculations shared with the measure sub-tree, to obtain a de-duplicated dimension-filter sub-tree. Suppose there

is a dimension that is a calculation. Suppose further that the calculation has physical input fields (sometimes called data fields) that are in the measure part of the dimension-filter sub-tree. Those fields are also linking fields. The Group-By operation finds unique set of dimensions. Some implementations use the linking fields to join back. In some instances, when there are calculations that share fields with the measure sub-tree, some implementations recover the unique rows that the measure came from. In such instances, the joins acts like quasi-packing together primary keys for the measure sub-tree.

[00110] The computer subsequently combines (1344) the de-duplicated dimension-filter sub-tree with the updated measure sub-tree to obtain the aggregated measure subquery.

[00111] In situations when there are primary keys, some implementations do not use a dimension-filter sub-tree. In such cases, some implementations combine primary keys of all the tables of the measure sub-tree.

[00112] In some implementations, the computer combines the de-duplicated dimension-filter sub-tree with the updated measure sub-tree by performing a sequence of operations. The computer determines if the de-duplicated dimension-filter sub-tree contains a filter. When the de-duplicated dimension-filter sub-tree contains a filter, the computer (1346) inner-joins the updated measure-sub-tree with the de-duplicated dimension-filter sub-tree. When the de-duplicated dimension-filter sub-tree does not contain a filter, the computer left outer-joins (1348) the updated measure-sub-tree with the de-duplicated dimension-filter sub-tree.

[00113] In some implementations, the computer determines if the keys indicate a many-to-one relationship or a one-to-one relationship between a first logical table and a second logical table. When the keys indicate many-to-one relationship between the first logical table and the second logical table, the computer includes (1350) the first table and the second table in the measure sub-tree, thereby avoiding the Group-By in the de-duplication operation for the first logical table and the second logical table.

[00114] In some implementations, when the dimension-filter sub-tree joins against the measure sub-tree exclusively along many-to-one and one-to-one links, the computer replaces (1352) tables shared by the measure sub-tree and the dimension-filter sub-tree with the de-duplicated dimension-filter sub-tree.

[00115] The terminology used in the description of the invention herein is for the purpose of describing particular implementations only and is not intended to be limiting of the invention. As used in the description of the invention and the appended claims, the singular

forms "a," "an," and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will also be understood that the term "and/or" as used herein refers to and encompasses any and all possible combinations of one or more of the associated listed items. It will be further understood that the terms "comprises" and/or "comprising," when used in this specification, specify the presence of stated features, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, steps, operations, elements, components, and/or groups thereof.

[00116] The foregoing description, for purpose of explanation, has been described with reference to specific implementations. However, the illustrative discussions above are not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations are possible in view of the above teachings. The implementations were chosen and described in order to best explain the principles of the invention and its practical applications, to thereby enable others skilled in the art to best utilize the invention and various implementations with various modifications as are suited to the particular use contemplated.

What is claimed is:

1. A method of generating data visualizations, comprising:

at a computer having a display, one or more processors and memory storing one or more programs configured for execution by the one or more processors:

receiving a visual specification, which specifies a data source, a plurality of visual variables, and a plurality of data fields from the data source, wherein each of the visual variables is associated with either (i) a respective one or more of the data fields or (ii) one or more filters, and each of the data fields is identified as either a dimension or a measure;

retrieving a stored data model encoding the data source as a tree of logical tables, each logical table having its own physical representation and including a respective one or more logical fields, each logical field corresponding to either a data field or a calculation that spans one or more logical tables, wherein each edge of the tree connects two logical tables that are related;

generating a dimension subquery based on logical tables that supply the data fields for the dimensions and the filters;

generating, for each measure, based on the logical tables that supply the data fields for the respective measure and the filters, an aggregated measure subquery grouped by the dimensions;

forming a final query by joining, using the dimensions, the dimension subquery to each of the aggregated measure subqueries;

executing the final query against the data source to retrieve tuples that comprise distinct ordered combinations of data values for the data fields; and

building and displaying a data visualization according to the data fields in the tuples and according to the visual variables to which each of the data fields is associated.

2. The method of claim 1, wherein generating each aggregated measure subquery comprises:

computing a measure sub-tree of the tree of logical tables, wherein the measure sub-tree is a minimum sub-tree required to supply the data fields for a respective measure;

computing a dimension-filter sub-tree of the tree of logical tables, wherein the dimension-filter sub-tree is a minimum sub-tree required to supply all the physical inputs for the dimensions and the filters;

Date Reçue/Date Received 2023-07-20

in accordance with a determination that the dimension-filter sub-tree does not share any logical table with the measure sub-tree, adding a neighboring logical table from the measure sub-tree to the dimension-filter sub-tree;

compiling the measure sub-tree to obtain a measure join tree and compiling the dimension-filter sub-tree to obtain a dimension-filter join tree;

layering calculations and filters over the measure join tree and the dimension-filter join tree to obtain an updated measure sub-tree and an updated dimension-filter sub-tree, respectively;

de-duplicating the updated dimension-filter sub-tree by applying a group-by operation that uses the dimensions and linking fields that include (i) keys from relationships between the logical tables and (ii) data fields of calculations shared with the measure sub-tree, to obtain a de-duplicated dimension-filter sub-tree; and

combining the de-duplicated dimension-filter sub-tree with the updated measure sub-tree to obtain the aggregated measure subquery.

3. The method of claim 2, wherein compiling the measure sub-tree comprises inner joining logical tables in the measure sub-tree to obtain the measure join tree.

4. The method of claim 2, wherein computing the dimension-filter sub-tree comprises inner joining logical tables in the dimension-filter sub-tree that are shared with the measure sub-tree, and left-joining logical tables in the dimension-filter sub-tree that are not shared with the measure sub-tree, to obtain the dimension-filter join tree.

5. The method of claim 2, wherein combining the de-duplicated dimension-filter sub-tree with the updated measure sub-tree comprises:

in accordance with a determination that the de-duplicated dimension-filter sub-tree contains one or more filters, inner-joining the updated measure-sub-tree with the de-duplicated dimension-filter sub-tree; and

in accordance with a determination that the de-duplicated dimension-filter sub-tree contains no filters, left outer-joining the updated measure-sub-tree with the de-duplicated dimension-filter sub-tree.

6. The method of claim 2, further comprising:

Date Reçue/Date Received 2023-07-20

determining if the keys indicate a many-to-one relationship or a one-to-one relationship between a first logical table and a second logical table; and

in accordance with a determination that the keys indicate many-to-one relationship between the first logical table and the second logical table, including the first logical table and the second logical table in the measure sub-tree, thereby avoiding the group-by in the de-duplication for the first logical table and the second logical table.

7. The method of claim 2, further comprising:

in accordance with a determination that the dimension-filter sub-tree joins against the measure sub-tree exclusively along many-to-one and one-to-one links, replacing tables shared by the measure sub-tree and the dimension-filter sub-tree with the de-duplicated dimension-filter sub-tree.

8. The method of claim 1, wherein generating the dimension subquery comprises inner-joining a first one or more logical tables in the tree of logical tables, wherein each logical table of the first one or more logical tables supplies the data fields for the dimensions or the filters.

9. The method of claim 1, wherein forming the final query comprises joining the dimensions subquery and the aggregated measure subqueries on the dimensions using outer joins, and applying a COALESCE each outer join.

10. The method of claim 1, wherein forming the final query comprises, in accordance with a determination that the visualization has no dimensions, performing a full outer join between the aggregated measure subqueries.

11. A computer system for generating data visualizations, comprising:

one or more processors; and

memory;

wherein the memory stores one or more programs configured for execution by the one or more processors, and the one or more programs comprising instructions for:

receiving a visual specification, which specifies a data source, a plurality of visual variables, and a plurality of data fields from the data source, wherein each of the visual variables is associated with either (i) a respective one or more of the data fields or (ii) one or more filters, and each of the data fields is identified as either a dimension or a measure;

Date Reçue/Date Received 2023-07-20

retrieving a stored data model encoding the data source as a tree of logical tables, each logical table having its own physical representation and including a respective one or more logical fields, each logical field corresponding to either a data field or a calculation that spans one or more logical tables, wherein each edge of the tree connects two logical tables that are related;

generating a dimension subquery based on logical tables that supply the data fields for the dimensions and the filters;

generating, for each measure, based on the logical tables that supply the data fields for the respective measure and the filters, an aggregated measure subquery grouped by the dimensions;

forming a final query by joining, using the dimensions, the dimension subquery to each of the aggregated measure subqueries;

executing the final query against the data source to retrieve tuples that comprise distinct ordered combinations of data values for the data fields; and

building and displaying a data visualization according to the data fields in the tuples and according to the visual variables to which each of the data fields is associated.

12. The computer system of claim 11, wherein generating each aggregated measure subquery comprises:

computing a measure sub-tree of the tree of logical tables, wherein the measure sub-tree is a minimum sub-tree required to supply the data fields for a respective measure;

computing a dimension-filter sub-tree of the tree of logical tables, wherein the dimension-filter sub-tree is a minimum sub-tree required to supply all the physical inputs for the dimensions and the filters;

in accordance with a determination that the dimension-filter sub-tree does not share any logical table with the measure sub-tree, adding a neighboring logical table from the measure sub-tree to the dimension-filter sub-tree;

compiling the measure sub-tree to obtain a measure join tree and compiling the dimension-filter sub-tree to obtain a dimension-filter join tree;

layering calculations and filters over the measure join tree and the dimension-filter join tree to obtain an updated measure sub-tree and an updated dimension-filter sub-tree, respectively;

de-duplicating the updated dimension-filter sub-tree by applying a group-by operation that uses the dimensions and linking fields that include (i) keys from relationships between

Date Reçue/Date Received 2023-07-20

the logical tables and (ii) the physical input fields of the calculations shared with the measure sub-tree, to obtain a de-duplicated dimension-filter sub-tree; and

combining the de-duplicated dimension-filter sub-tree with the updated measure sub-tree to obtain the aggregated measure subquery.

13. The computer system of claim 12, wherein compiling the measure sub-tree comprises inner joining logical tables in the measure sub-tree to obtain the measure join tree.

14. The computer system of claim 12, wherein compiling the dimension-filter sub-tree comprises inner joining logical tables in the dimension-filter sub-tree that are shared with the measure sub-tree, and left-joining logical tables in the dimension-filter sub-tree that are not shared with the measure sub-tree, to obtain the dimension-filter join tree.

15. The computer system of claim 12, wherein combining the de-duplicated dimension-filter sub-tree with the updated measure sub-tree comprises:

in accordance with a determination that the de-duplicated dimension-filter sub-tree contains one or more filters, inner-joining the updated measure-sub-tree with the de-duplicated dimension-filter sub-tree; and

in accordance with a determination that the de-duplicated dimension-filter sub-tree contains no filters, left outer-joining the updated measure-sub-tree with the de-duplicated dimension-filter sub-tree.

16. The computer system of claim 12, wherein the one or more programs further comprise instructions for:

determining if the keys indicate a many-to-one relationship or a one-to-one relationship between a first logical table and a second logical table; and

in accordance with a determination that the keys indicate many-to-one relationship between the first logical table and the second logical table, including the first logical table and the second logical table in the measure sub-tree, thereby avoiding the group-by in the de-duplication for the first logical table and the second logical table.

17. The computer system of claim 12, wherein the one or more programs further comprise instructions for:

Date Reçue/Date Received 2023-07-20

in accordance with a determination that the dimension-filter sub-tree joins against the measure sub-tree exclusively along many-to-one and one-to-one links, replacing tables shared by the measure sub-tree and the dimension-filter sub-tree with the de-duplicated dimension-filter sub-tree.

18. The computer system of claim 11, wherein generating the dimension subquery comprises inner-joining a first one or more logical tables in the tree of logical tables, wherein each logical table of the first one or more logical tables supplies the data fields for the dimensions or the filters.

19. The computer system of claim 11, wherein forming the final query comprises joining the dimensions subquery and the aggregated measure subqueries on the dimensions using outer joins, and applying a COALESCE after each outer join.

20. A non-transitory computer readable storage medium storing one or more programs configured for execution by a computer system having a display, one or more processors, and memory, the one or more programs comprising instructions for:

receiving a visual specification, which specifies a data source, a plurality of visual variables, and a plurality of data fields from the data source, wherein each of the visual variables is associated with either (i) a respective one or more of the data fields or (ii) one or more filters, and each of the data fields is identified as either a dimension or a measure;

retrieving a stored data model encoding the data source as a tree of logical tables, each logical table having its own physical representation and including a respective one or more logical fields, each logical field corresponding to either a data field or a calculation that spans one or more logical tables, wherein each edge of the tree connects two logical tables that are related;

generating a dimension subquery based on logical tables that supply the data fields for the dimensions and the filters;

generating, for each measure, based on the logical tables that supply the data fields for the respective measure and the filters, an aggregated measure subquery grouped by the dimensions;

forming a final query by joining, using the dimensions, the dimension subquery to each of the aggregated measure subqueries;

executing the final query against the data source to retrieve tuples that comprise distinct ordered combinations of data values for the data fields; and

Date Reçue/Date Received 2023-07-20

building and displaying a data visualization according to the data fields in the tuples and according to the visual variables to which each of the data fields is associated.

Date Reçue/Date Received 2023-07-20

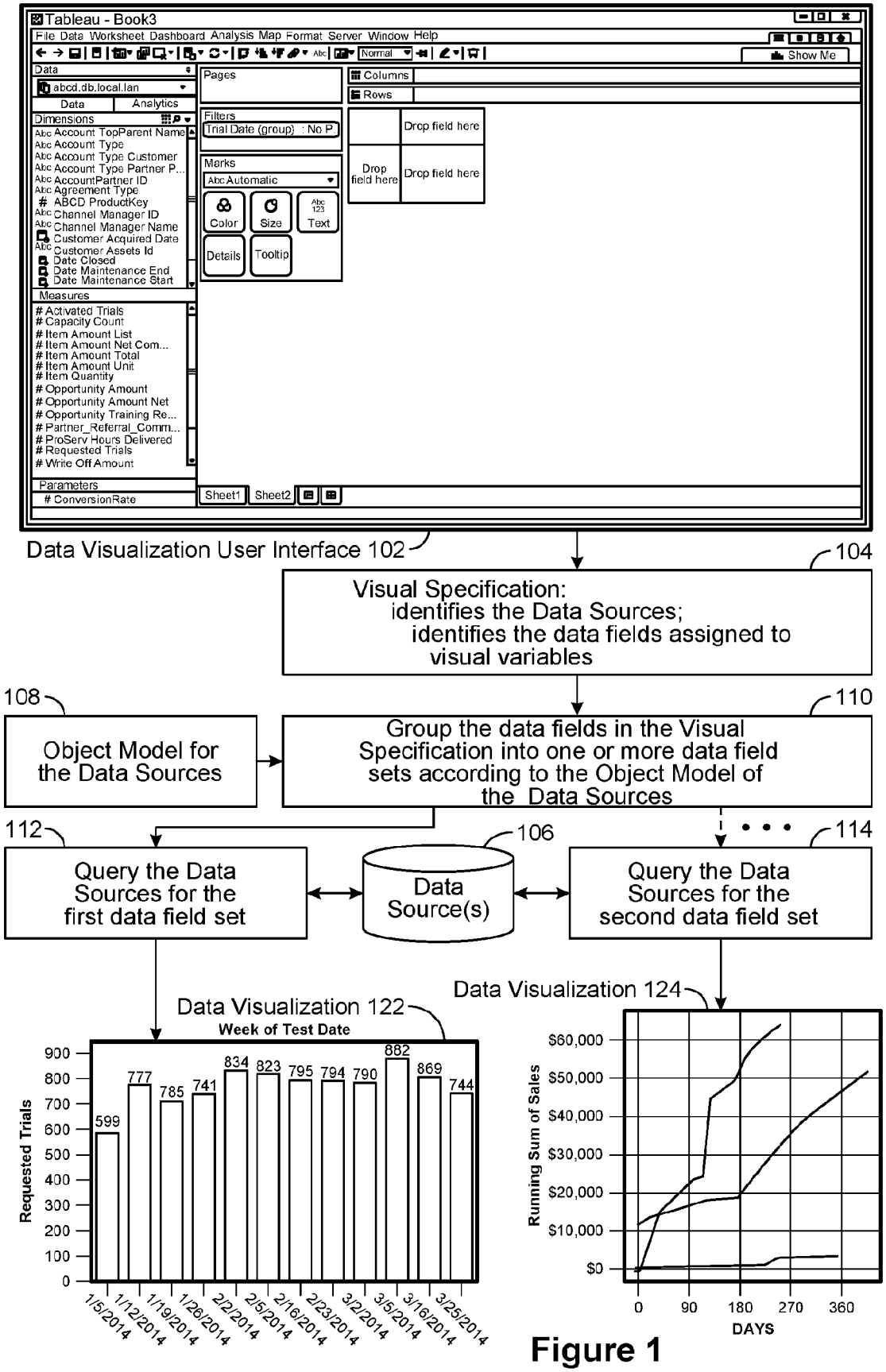


Tableau - Book3
File Data Worksheet Dashboard Analysis Map Format Server Window Help
Show Me
Data
abcd.db.local.lan
Data
Analytics
Dimensions
Abc Account TopParent Name
Abc Account Type
Abc Account Type Customer
Abc Account Type Partner P...
Abc AccountPartner ID
Abc Agreement Type
ABCD ProductKey
Abc Channel Manager ID
Abc Channel Manager Name
Customer Acquired Date
Abc Customer Assets 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 Trials
Write Off Amount
Parameters
ConversionRate
Pages
Filters
Trial Date (group) : No P...
Marks
Abc Automatic
Color
Size
Text
Details
Tooltip
Columns
Rows
Drop field here
Drop field here
Drop field here
Sheet1
Sheet2
Data Visualization User Interface 102
104
Visual Specification:
identifies the Data Sources;
identifies the data fields assigned to visual variables
108
Object Model for the Data Sources
110
Group the data fields in the Visual Specification into one or more data field sets according to the Object Model of the Data Sources
112
Query the Data Sources for the first data field set
106
Data Source(s)
114
Query the Data Sources for the second data field set
Data Visualization 122
Week of Test Date
Requested Trials
900
800
700
600
500
400
300
200
100
0
599
777
785
741
834
823
795
794
790
882
869
744
1/5/2014
1/12/2014
1/19/2014
1/26/2014
2/2/2014
2/9/2014
2/16/2014
2/23/2014
3/2/2014
3/9/2014
3/16/2014
3/25/2014
Data Visualization 124
Running Sum of Sales
$60,000
$50,000
$40,000
$30,000
$20,000
$10,000
$0
0
90
180
270
360
DAYS

Figure 1

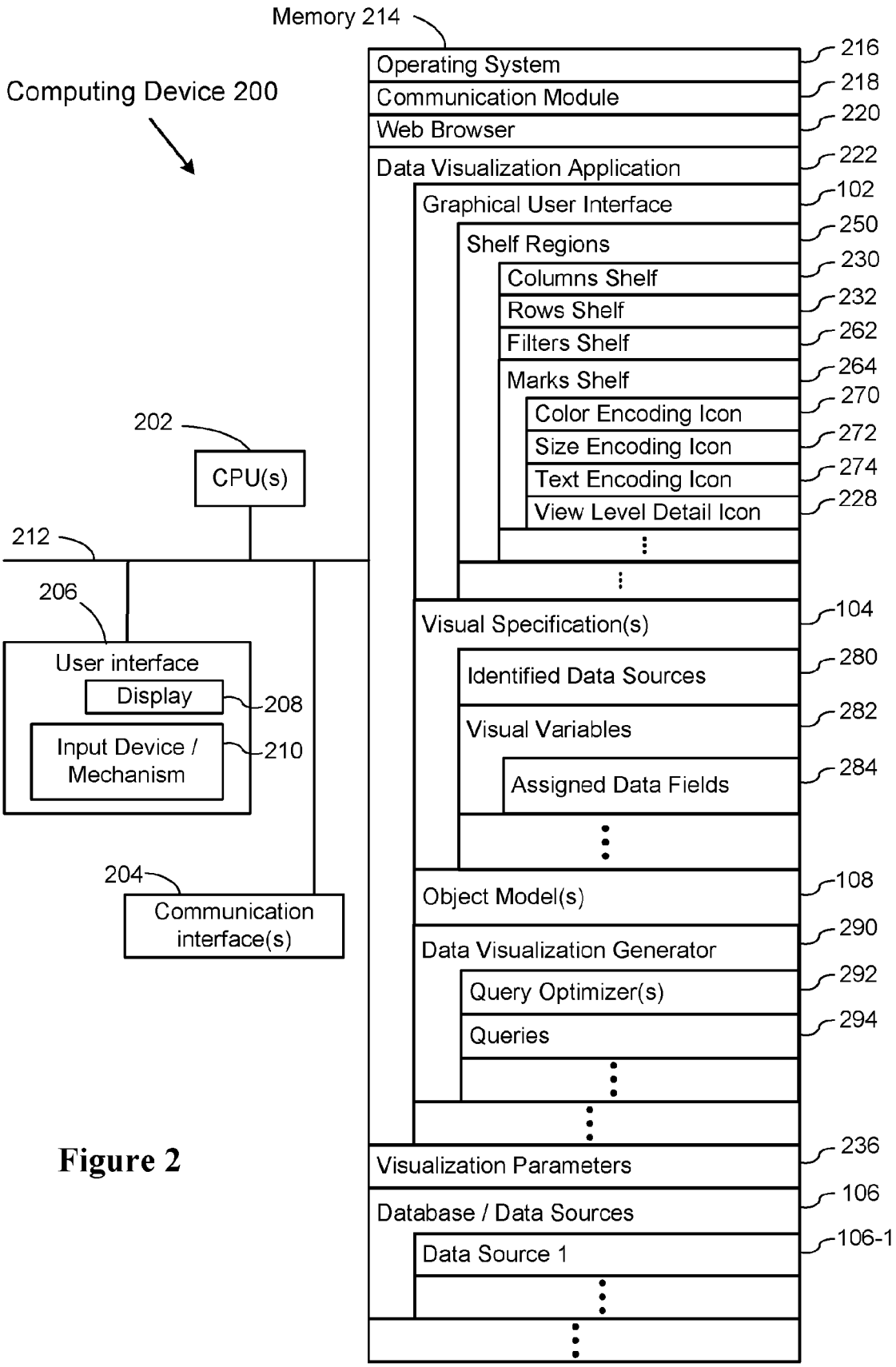

Computing Device 200
Memory 214
Operating System
216
Communication Module
218
Web Browser
220
Data Visualization Application
222
Graphical User Interface
102
Shelf Regions
250
Columns Shelf
230
Rows Shelf
232
Filters Shelf
262
Marks Shelf
264
Color Encoding Icon
270
Size Encoding Icon
272
Text Encoding Icon
274
View Level Detail Icon
228
202
CPU(s)
212
206
User interface
Display
208
Input Device / Mechanism
210
204
Communication interface(s)
Visual Specification(s)
104
Identified Data Sources
280
Visual Variables
282
Assigned Data Fields
284
Object Model(s)
108
Data Visualization Generator
290
Query Optimizer(s)
292
Queries
294
Visualization Parameters
236
Database / Data Sources
106
Data Source 1
106-1

Figure 2

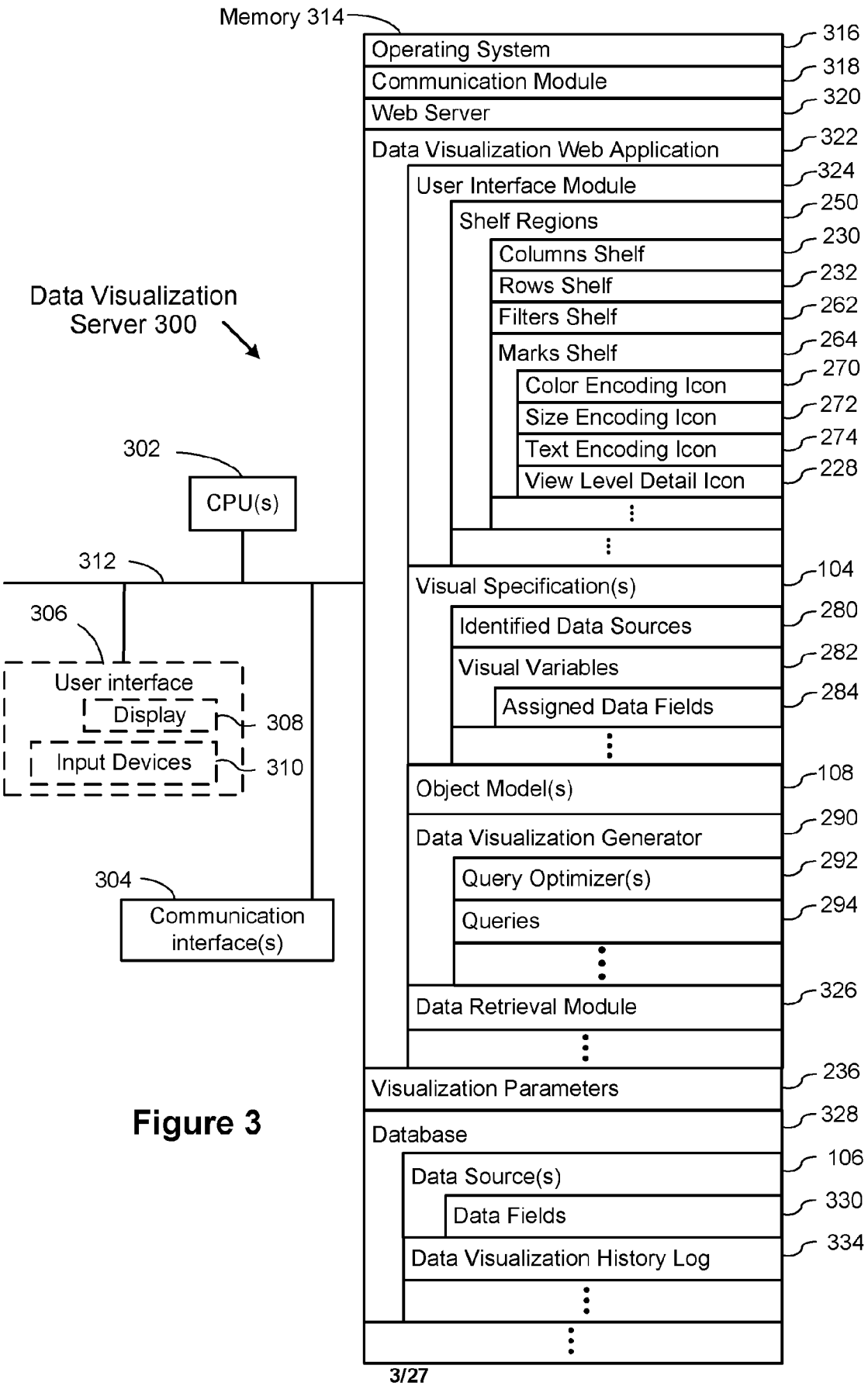

Memory 314
Operating System
316
Communication Module
318
Web Server
320
Data Visualization Web Application
322
User Interface Module
324
Shelf Regions
250
Columns Shelf
230
Rows Shelf
232
Filters Shelf
262
Marks Shelf
264
Color Encoding Icon
270
Size Encoding Icon
272
Text Encoding Icon
274
View Level Detail Icon
228
Visual Specification(s)
104
Identified Data Sources
280
Visual Variables
282
Assigned Data Fields
284
Object Model(s)
108
Data Visualization Generator
290
Query Optimizer(s)
292
Queries
294
Data Retrieval Module
326
Visualization Parameters
236
Database
328
Data Source(s)
106
Data Fields
330
Data Visualization History Log
334
Data Visualization Server 300
302
CPU(s)
312
306
User interface
Display
308
Input Devices
310
304
Communication interface(s)

Figure 3

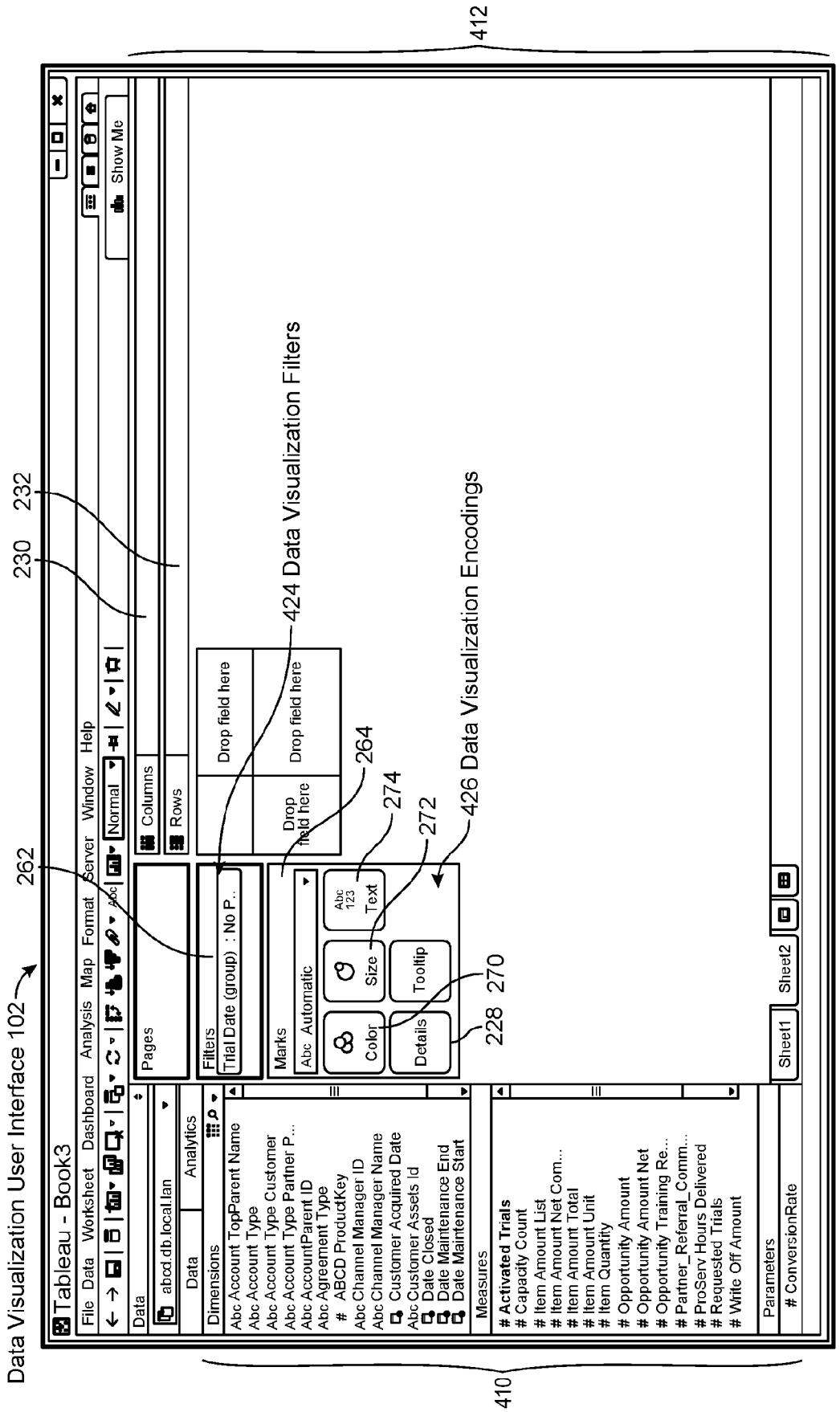

Data Visualization User Interface 102
262
230
232
412
410
424 Data Visualization Filters
426 Data Visualization Encodings
264
274
272
270
228
Tableau - Book3
File Data Worksheet Dashboard Analysis Map Format Server Window Help
Normal
Show Me
Data
abcd.db.local.lan
Data
Analytics
Dimensions
Abc Account TopParent Name
Abc Account Type
Abc Account Type Customer
Abc Account Type Partner P...
Abc AccountParent ID
Abc Agreement Type
ABCD ProductKey
Abc Channel Manager ID
Abc Channel Manager Name
Customer Acquired Date
Abc Customer Assets 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 Trials
Write Off Amount
Parameters
ConversionRate
Pages
Columns
Rows
Filters
Trial Date (group) : No P...
Drop field here
Drop field here
Drop field here
Marks
Abc Automatic
Color
Size
Abc 123 Text
Details
Tooltip
Sheet1
Sheet2

Figure 4

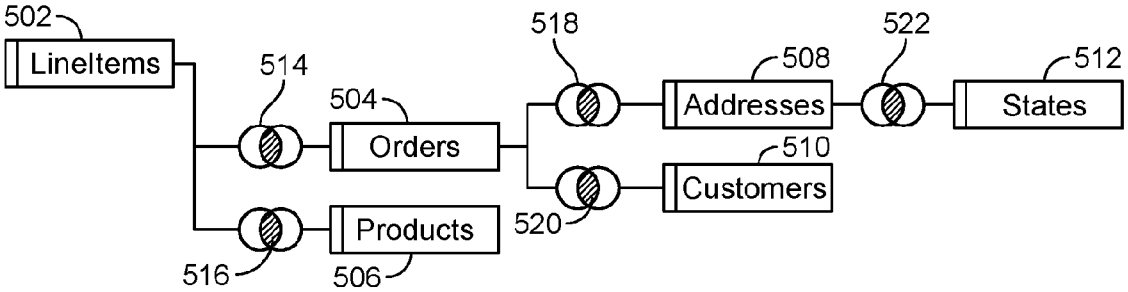


Figure 5A

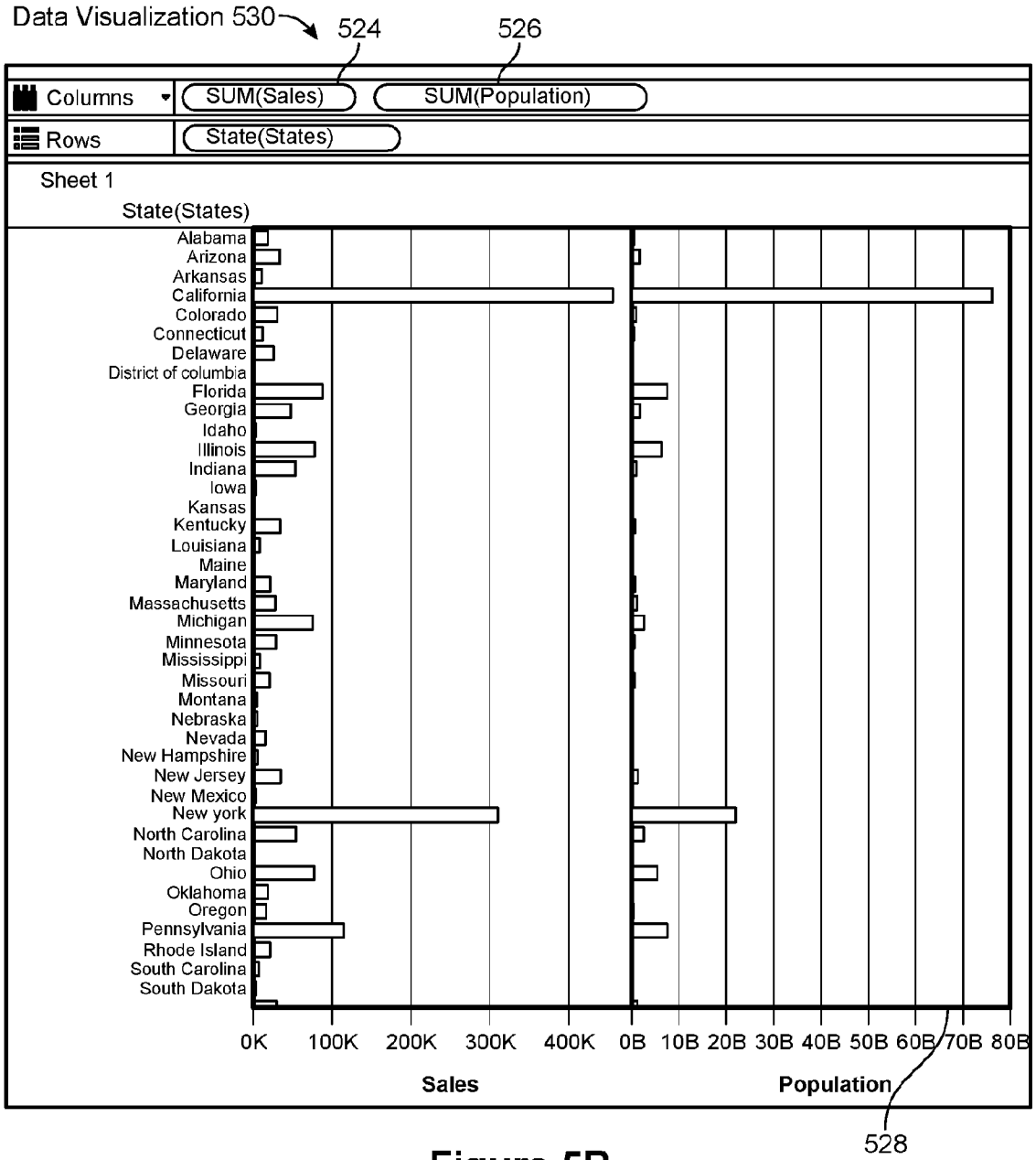


Figure 5B

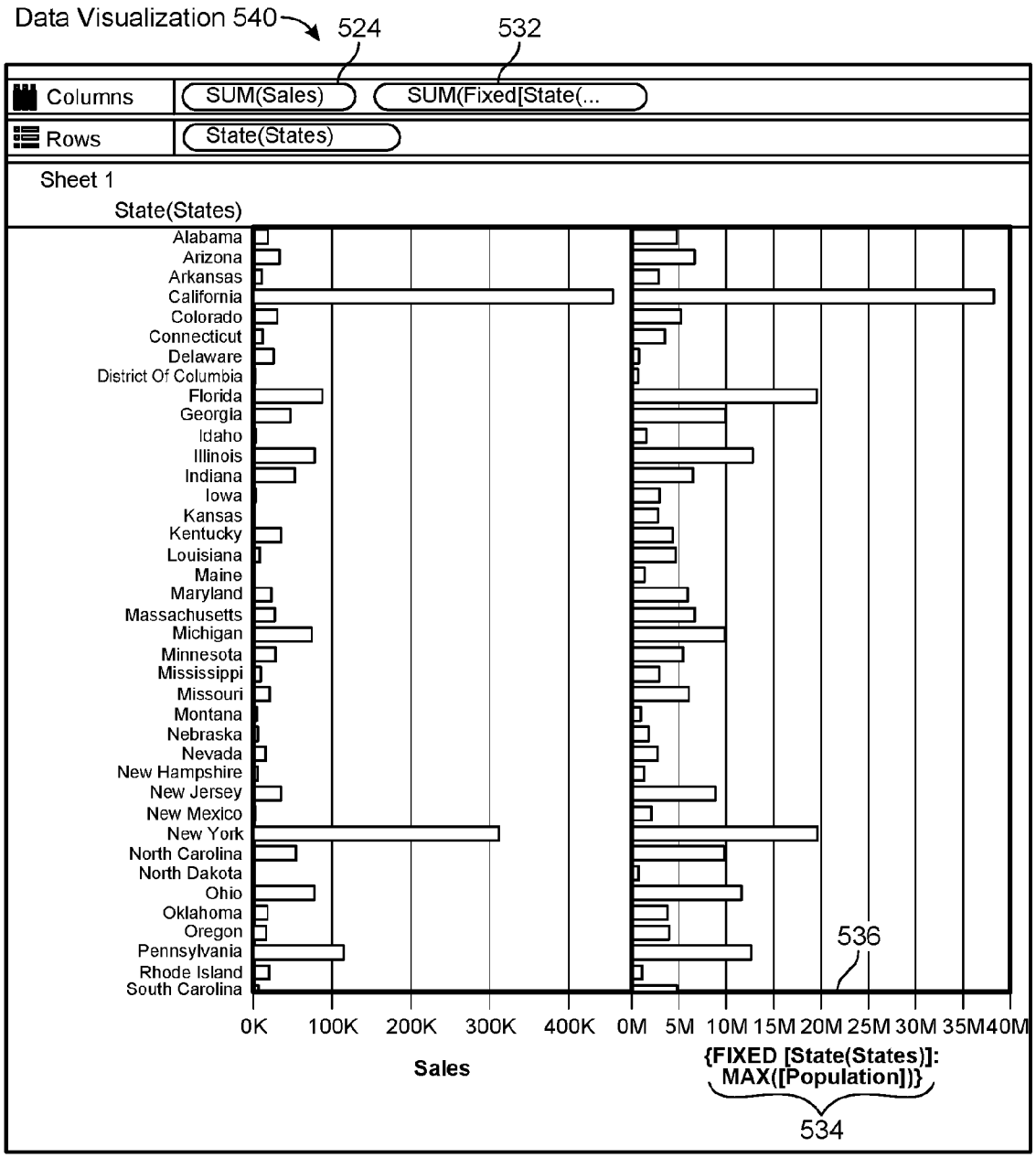

Data Visualization 540
524
532
Columns
SUM(Sales)
SUM(Fixed[State(...
Rows
State(States)
Sheet 1
State(States)
Alabama
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District Of Columbia
Florida
Georgia
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
536
0K 100K 200K 300K 400K 0M 5M 10M 15M 20M 25M 30M 35M 40M
Sales
{FIXED [State(States)]:
MAX([Population])}
534

Figure 5C

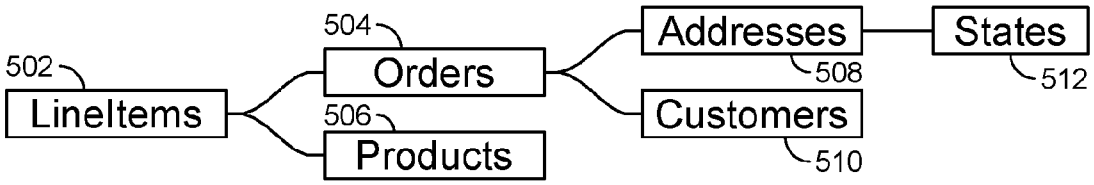


Figure 6A

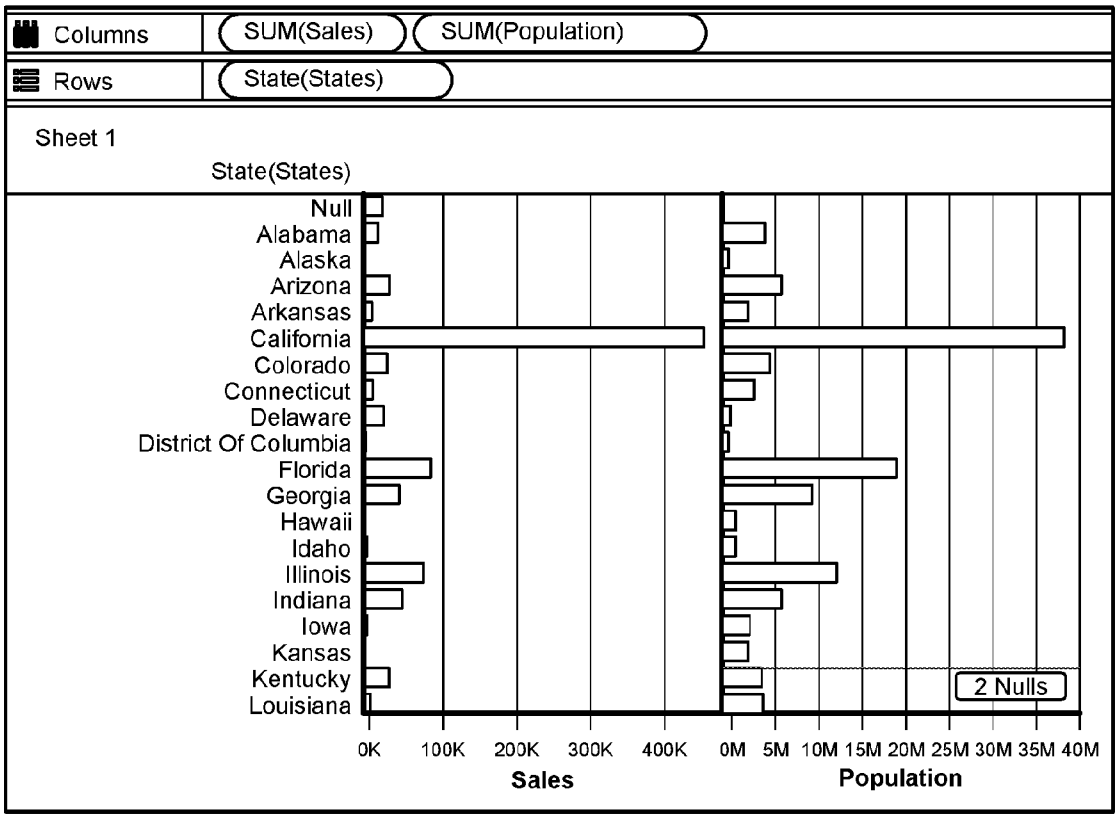


Figure 6B

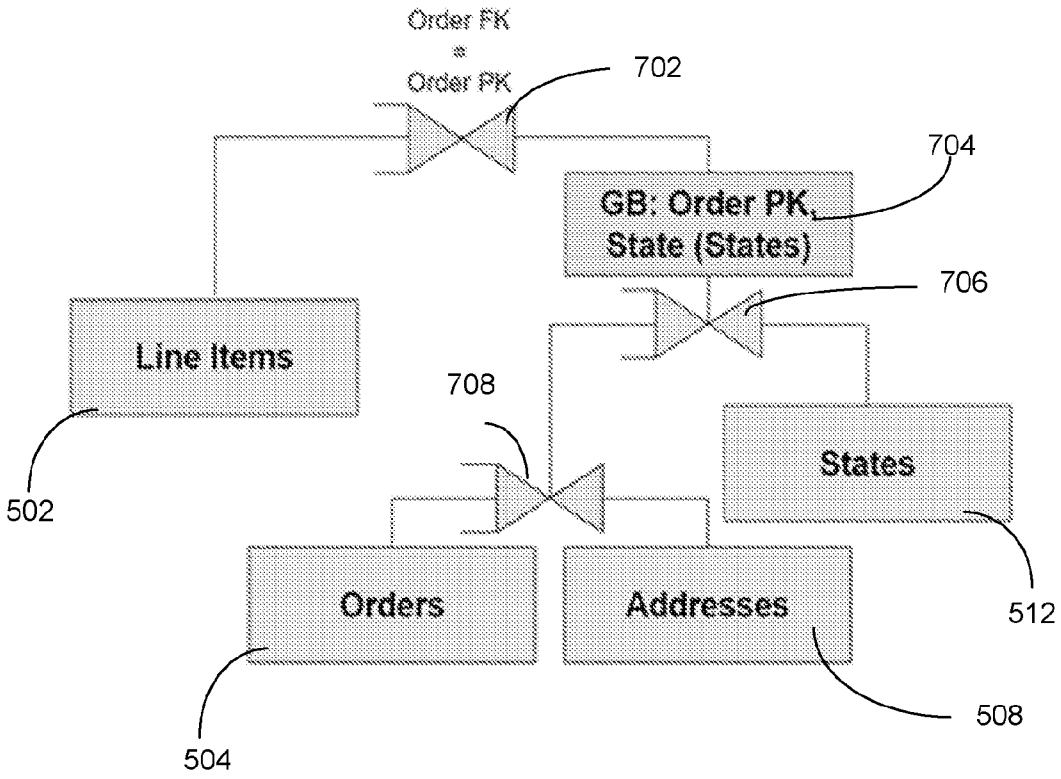

Order FK
=
Order PK
702
704
GB: Order PK, State (States)
706
Line Items
708
States
502
Orders
Addresses
512
508
504

Figure 7

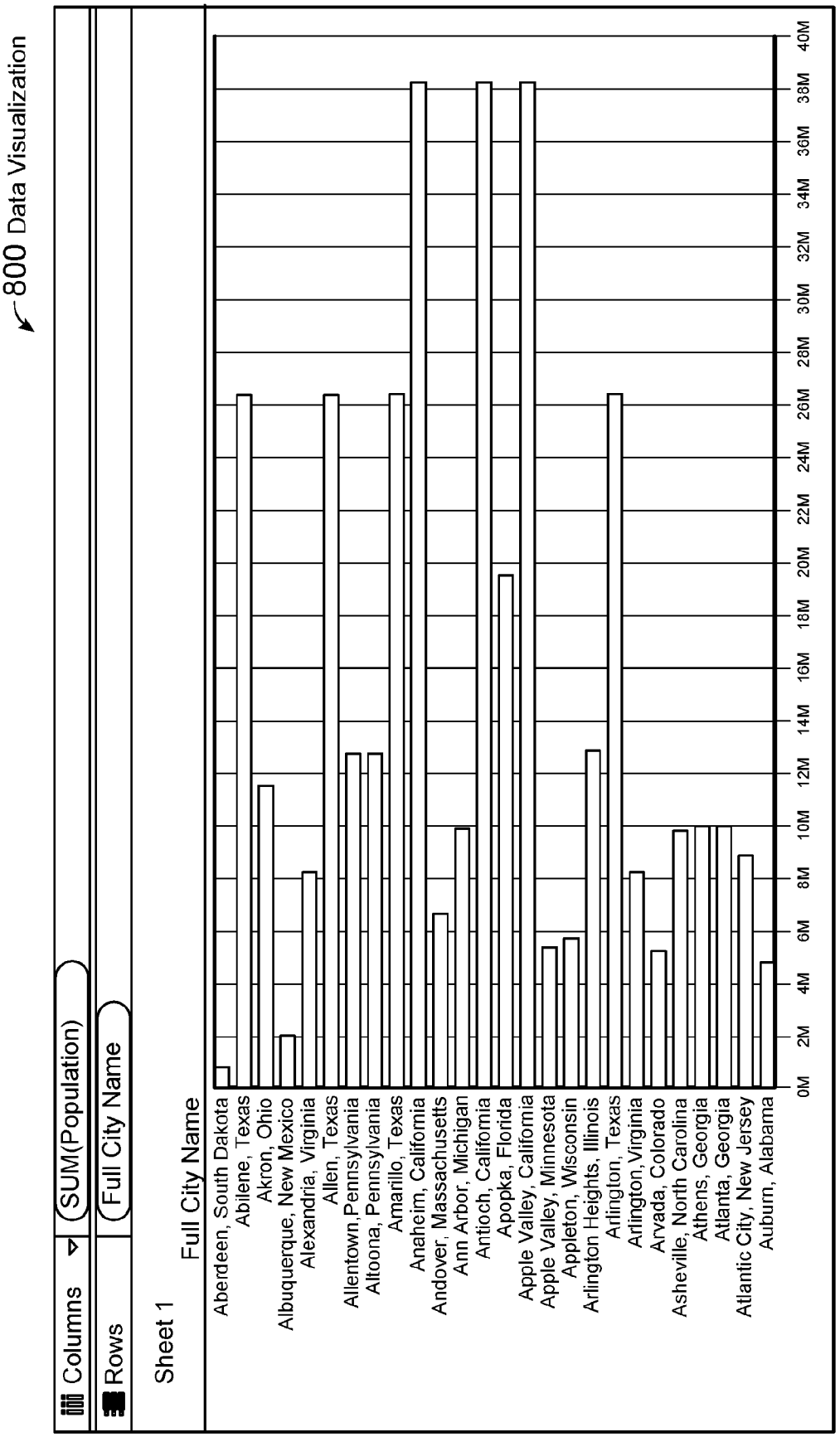

800 Data Visualization
Columns
SUM(Population)
Rows
Full City Name
Sheet 1
Full City Name
Aberdeen, South Dakota
Abilene, Texas
Akron, Ohio
Albuquerque, New Mexico
Alexandria, Virginia
Allen, Texas
Allentown,Pennsylvania
Altoona, Pennsylvania
Amarillo, Texas
Anaheim, California
Andover, Massachusetts
Ann Arbor, Michigan
Antioch, California
Apopka, Florida
Apple Valley, California
Apple Valley, Minnesota
Appleton, Wisconsin
Arlington Heights, Illinois
Arlington, Texas
Arlington,Virginia
Arvada, Colorado
Asheville, North Carolina
Athens, Georgia
Atlanta, Georgia
Atlantic City, New Jersey
Auburn, Alabama
0M
2M
4M
6M
8M
10M
12M
14M
16M
18M
20M
22M
24M
26M
28M
30M
32M
34M
36M
38M
40M

Figure 8A

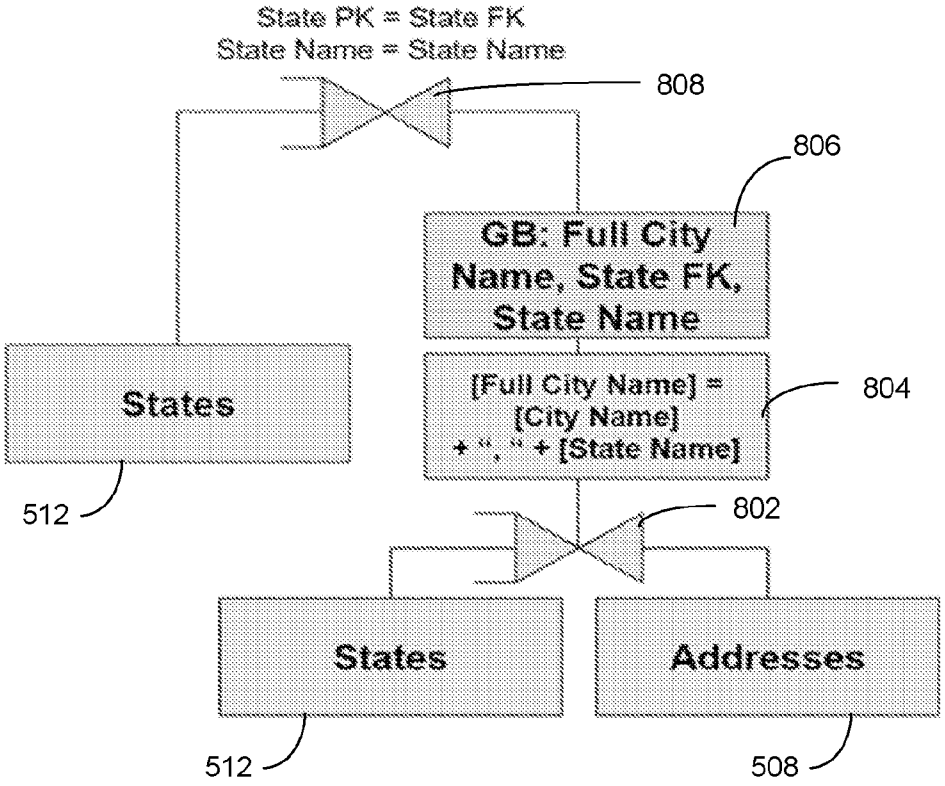

State PK = State FK
State Name = State Name
808
806
GB: Full City Name, State FK, State Name
States
[Full City Name] = [City Name] + ", " + [State Name]
804
512
802
States
Addresses
512
508

Figure 8B

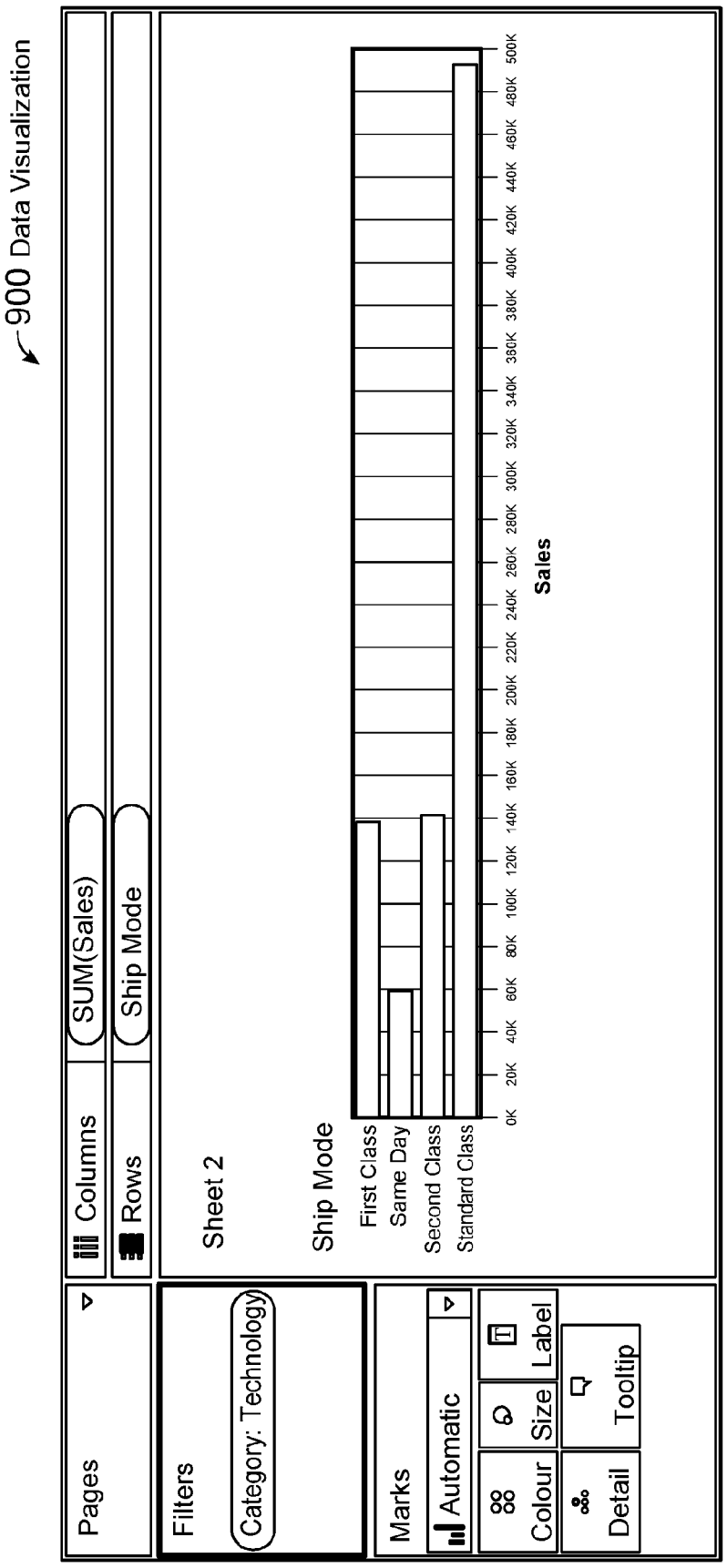

900 Data Visualization
Pages
Columns
SUM(Sales)
Rows
Ship Mode
Filters
Category: Technology
Sheet 2
Ship Mode
First Class
Same Day
Second Class
Standard Class
0K 20K 40K 60K 80K 100K 120K 140K 160K 180K 200K 220K 240K 260K 280K 300K 320K 340K 360K 380K 400K 420K 440K 460K 480K 500K
Sales
Marks
Automatic
Colour
Size
Label
Detail
Tooltip

Figure 9A

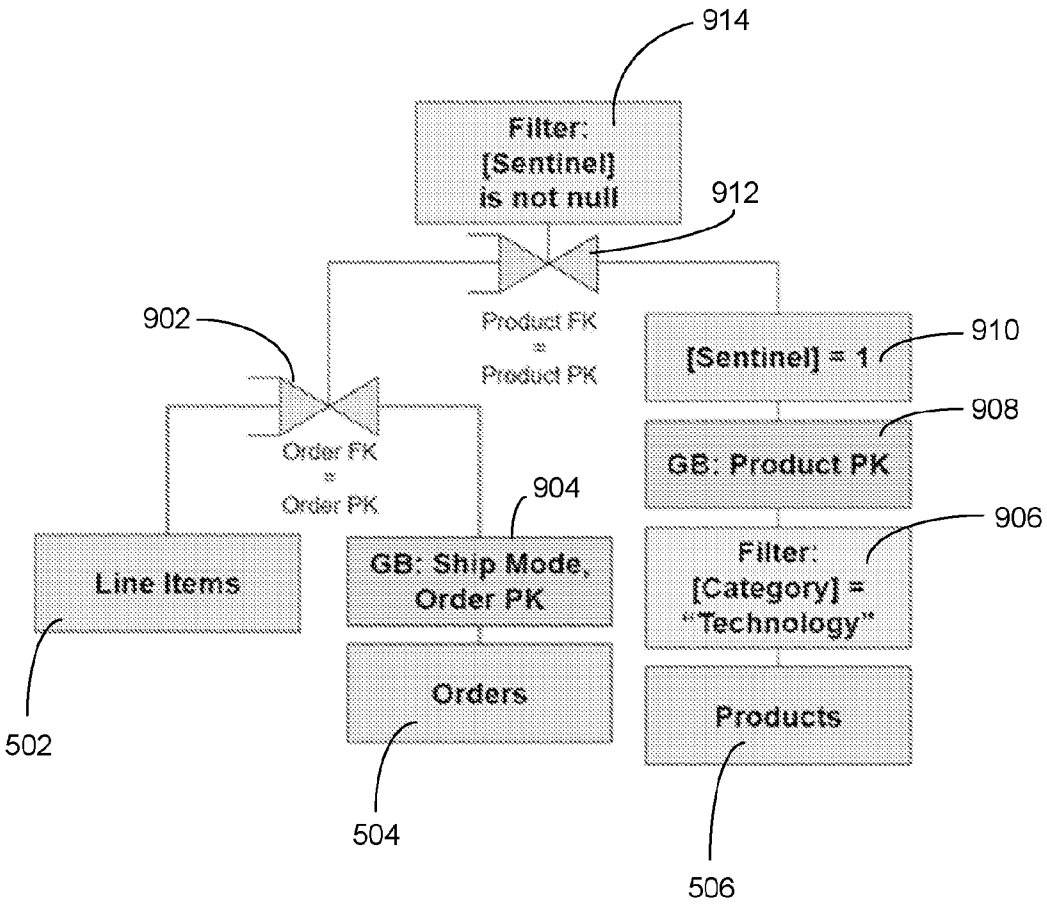

914
Filter:
[Sentinel]
is not null
912
Product FK
=
Product PK
902
Order FK
=
Order PK
910
[Sentinel] = 1
908
GB: Product PK
904
906
Line Items
GB: Ship Mode,
Order PK
Filter:
[Category] =
"Technology"
Orders
Products
502
504
506

Figure 9B

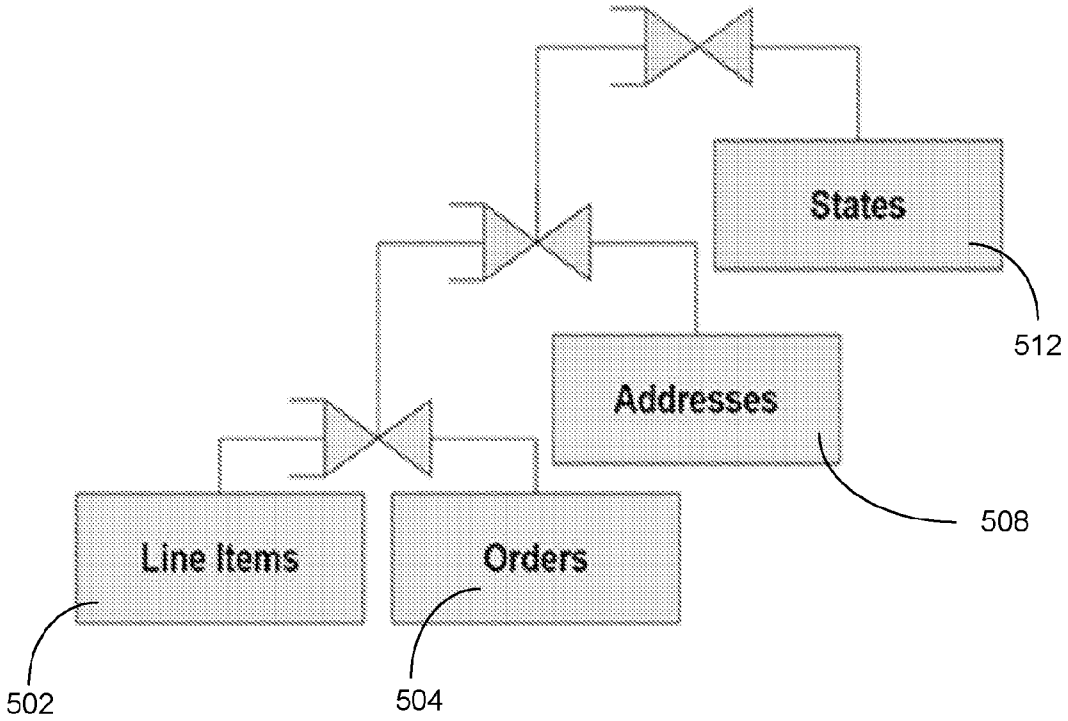


Figure 10A

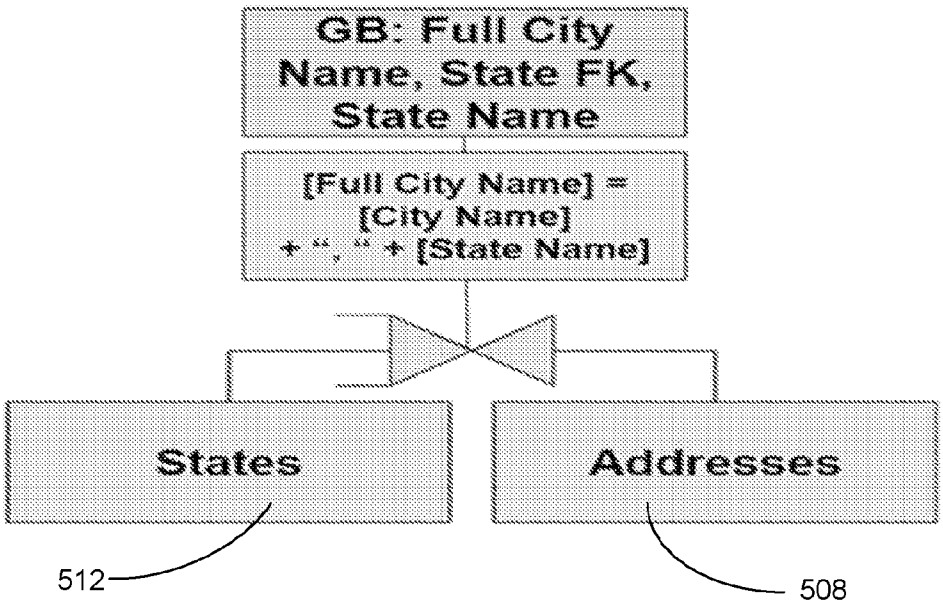


Figure 10B

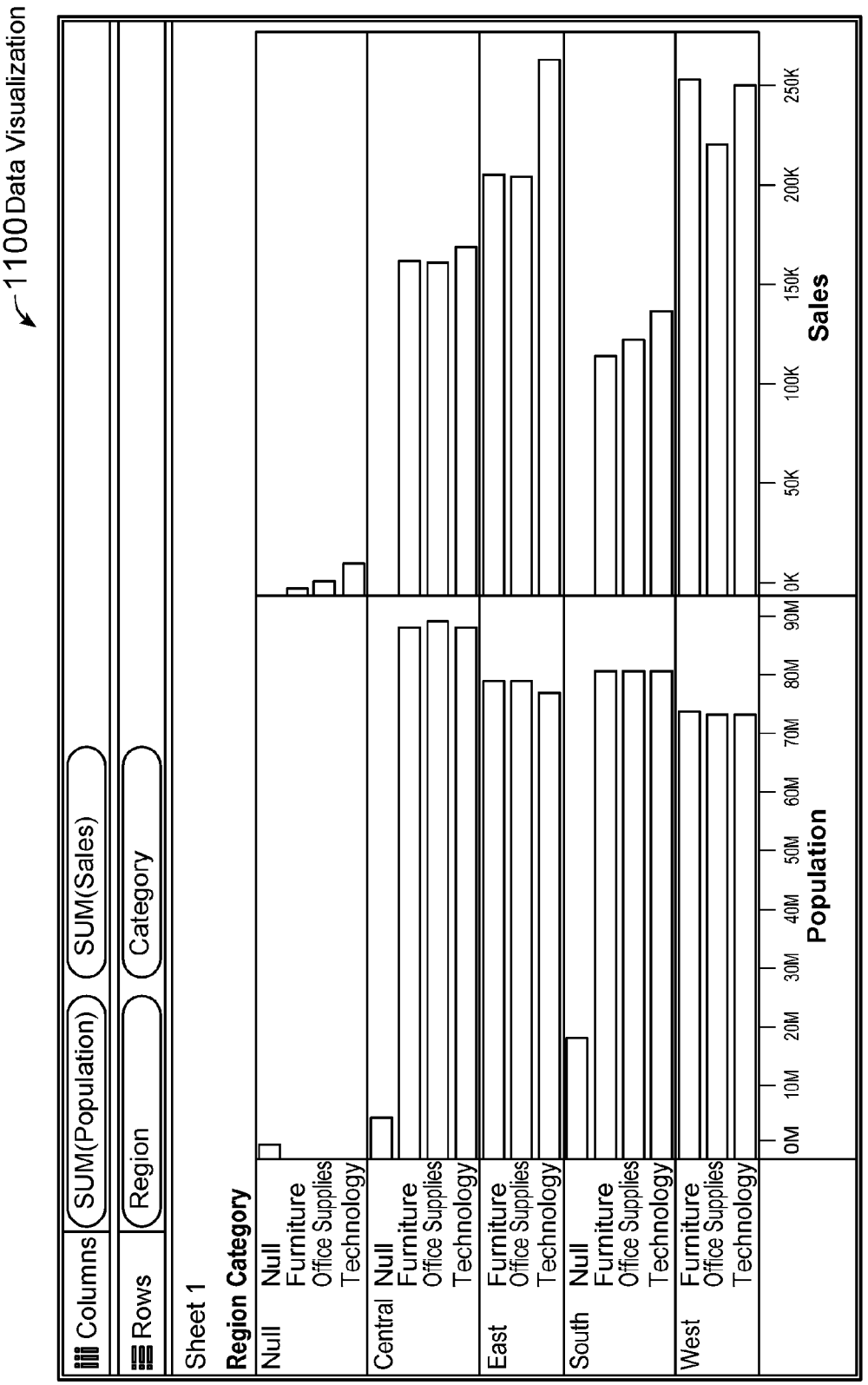

1100 Data Visualization
Columns
SUM(Population)
SUM(Sales)
Rows
Region
Category
Sheet 1
Region Category
Null
Null
Furniture
Office Supplies
Technology
Central
Null
Furniture
Office Supplies
Technology
East
Furniture
Office Supplies
Technology
South
Null
Furniture
Office Supplies
Technology
West
Furniture
Office Supplies
Technology
0M 10M 20M 30M 40M 50M 60M 70M 80M 90M
Population
0K 50K 100K 150K 200K 250K
Sales

Figure 11A

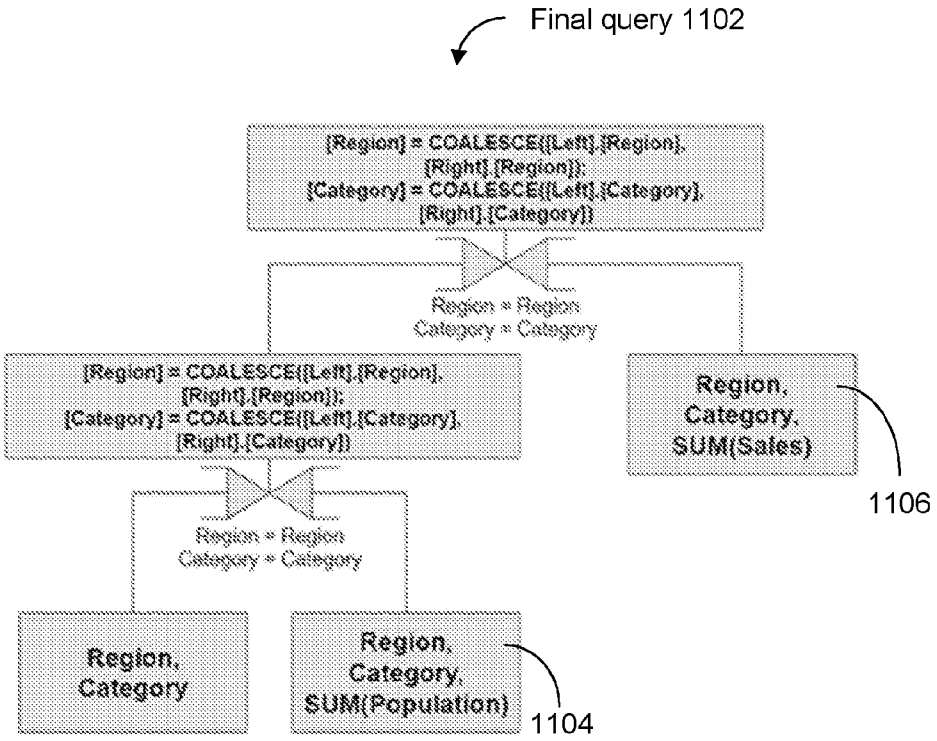

Final query 1102
[Region] = COALESCE([Left].[Region], [Right].[Region]);
[Category] = COALESCE([Left].[Category], [Right].[Category])
Region = Region
Category = Category
[Region] = COALESCE([Left].[Region], [Right].[Region]);
[Category] = COALESCE([Left].[Category], [Right].[Category])
Region = Region
Category = Category
Region, Category, SUM(Sales)
1106
Region, Category
Region, Category, SUM(Population)
1104

Figure 11B

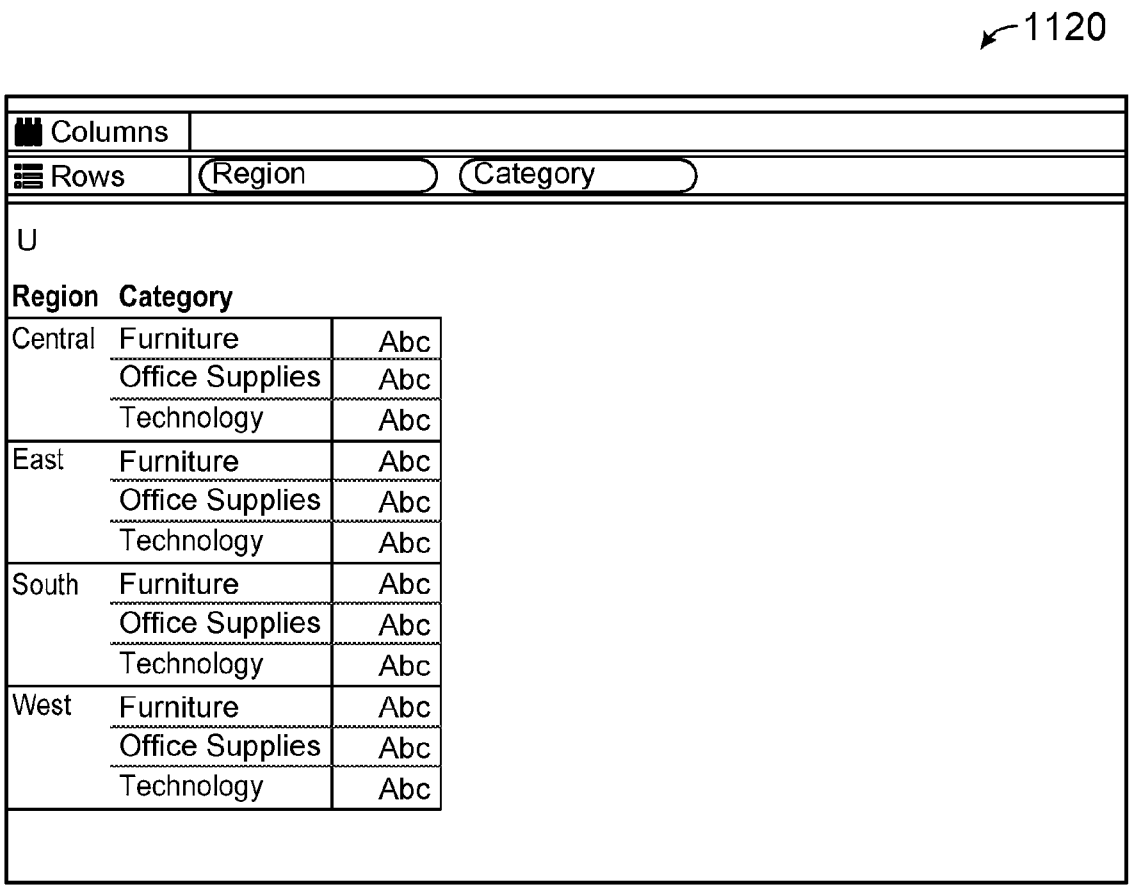

1120
Columns
Rows
Region
Category
U
Region
Category
Central
Furniture
Abc
Office Supplies
Abc
Technology
Abc
East
Furniture
Abc
Office Supplies
Abc
Technology
Abc
South
Furniture
Abc
Office Supplies
Abc
Technology
Abc
West
Furniture
Abc
Office Supplies
Abc
Technology
Abc

Figure 11C

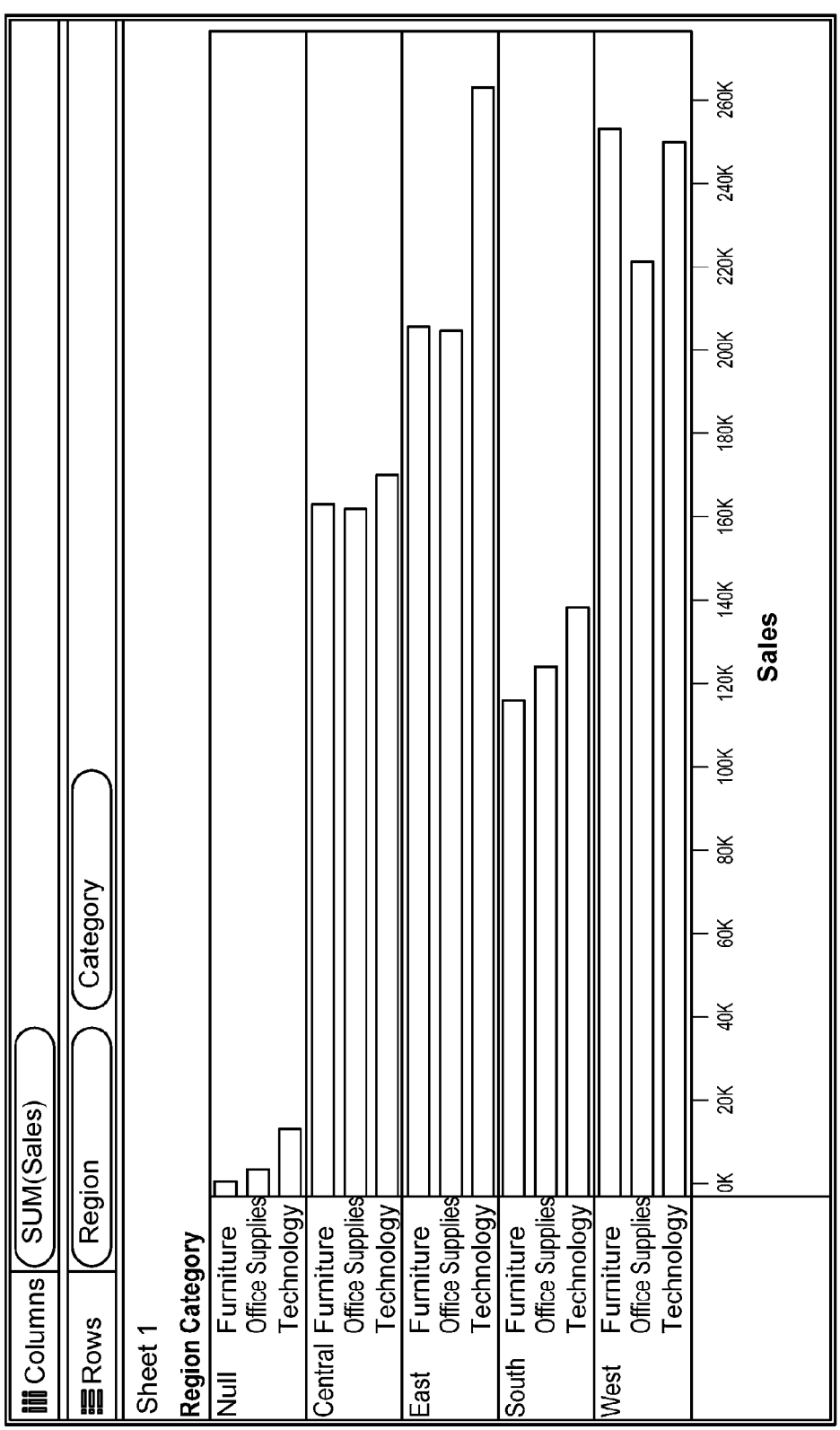

1130 Data Visualization
Columns
SUM(Sales)
Rows
Region
Category
Sheet 1
Region Category
Null Furniture
Office Supplies
Technology
Central Furniture
Office Supplies
Technology
East Furniture
Office Supplies
Technology
South Furniture
Office Supplies
Technology
West Furniture
Office Supplies
Technology
0K
20K
40K
60K
80K
100K
120K
140K
160K
180K
200K
220K
240K
260K
Sales

Figure 11D

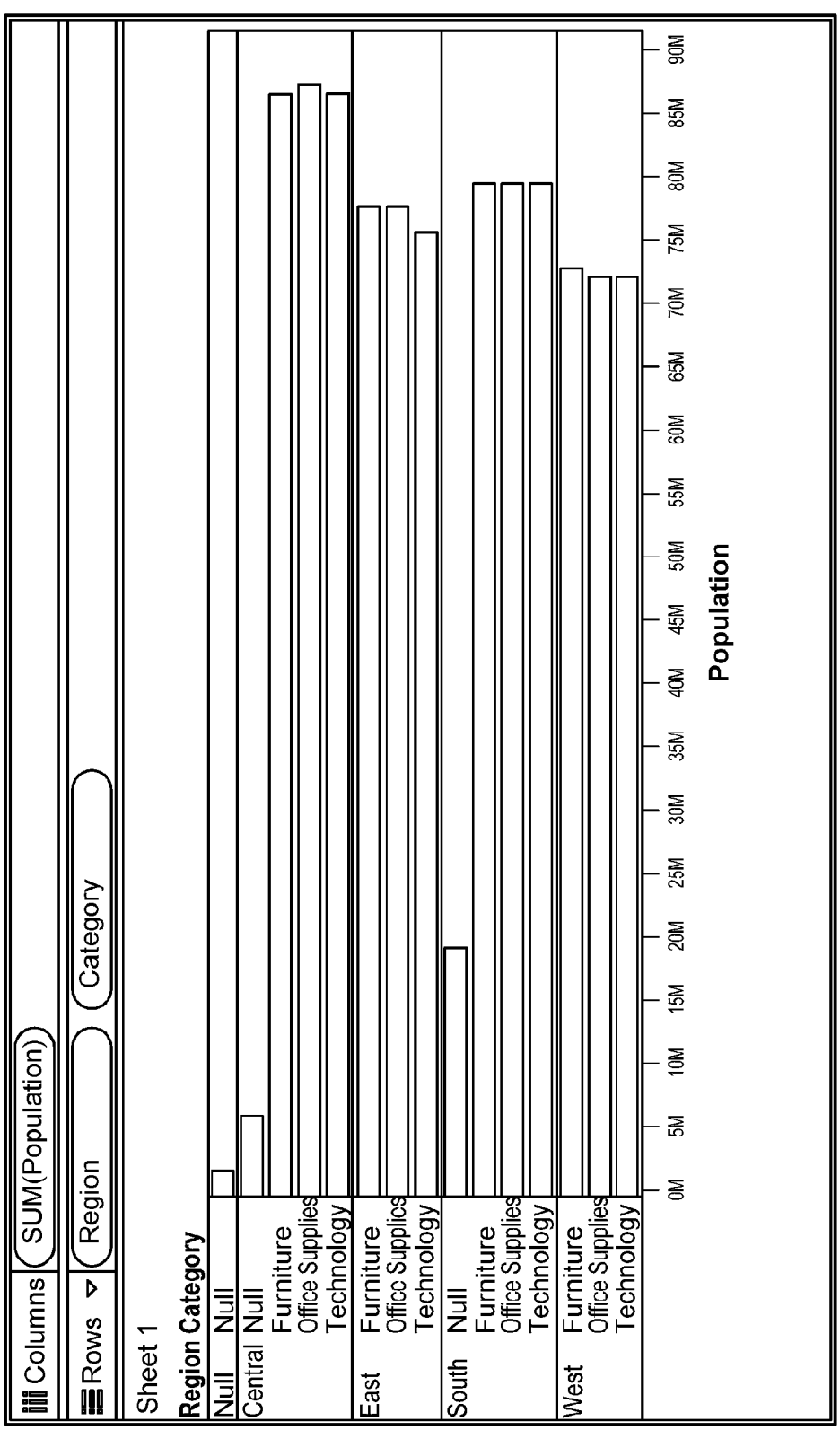

1140 Data Visualization
Columns
SUM(Population)
Rows
Region
Category
Sheet 1
Region Category
Null Null
Central Null
Furniture
Office Supplies
Technology
East Furniture
Office Supplies
Technology
South Null
Furniture
Office Supplies
Technology
West Furniture
Office Supplies
Technology
0M 5M 10M 15M 20M 25M 30M 35M 40M 45M 50M 55M 60M 65M 70M 75M 80M 85M 90M
Population

Figure 11E

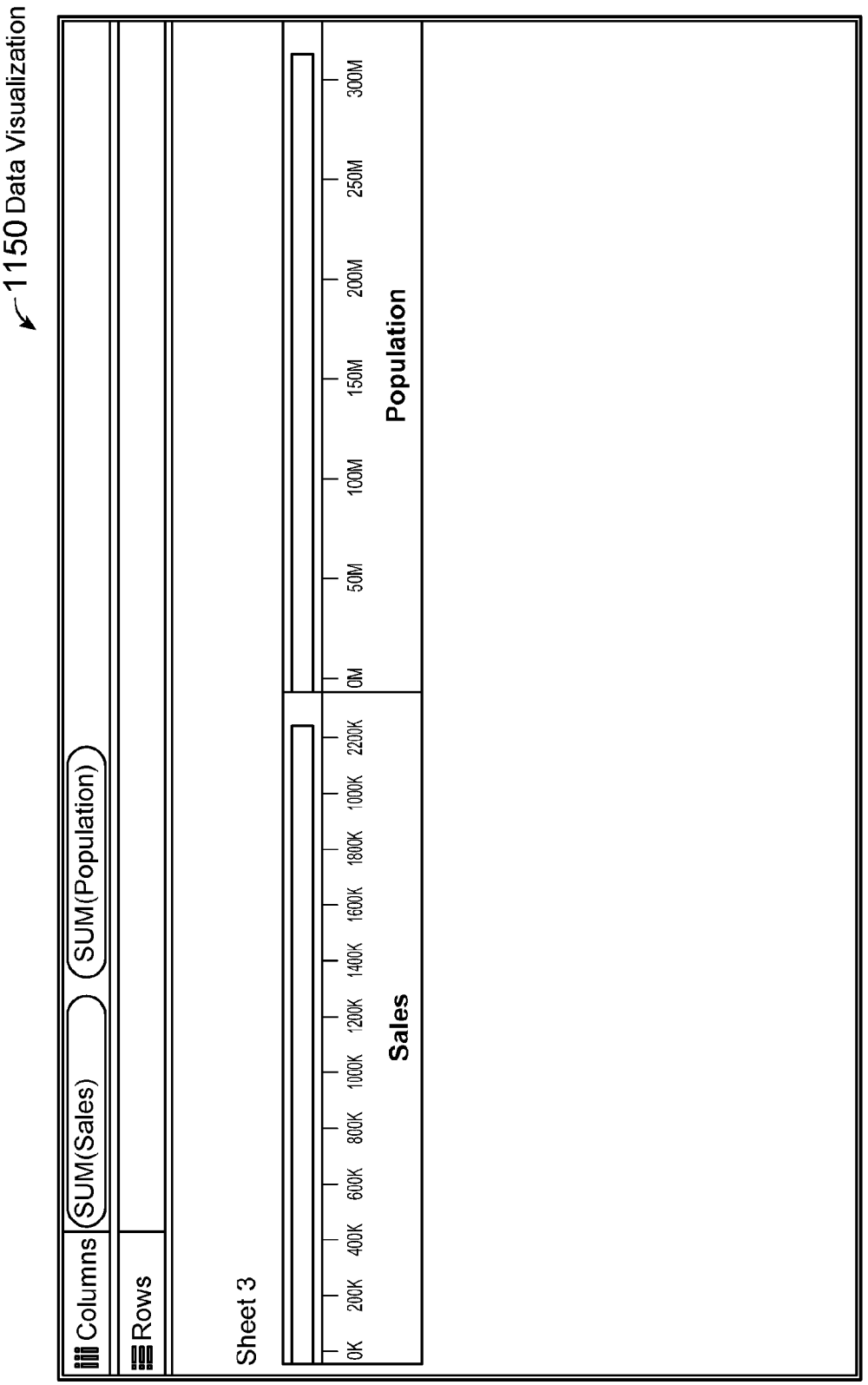

1150 Data Visualization
Columns
SUM(Sales)
SUM(Population)
Rows
Sheet 3
0K 200K 400K 600K 800K 1000K 1200K 1400K 1600K 1800K 1000K 2200K
Sales
0M 50M 100M 150M 200M 250M 300M
Population

Figure 11F

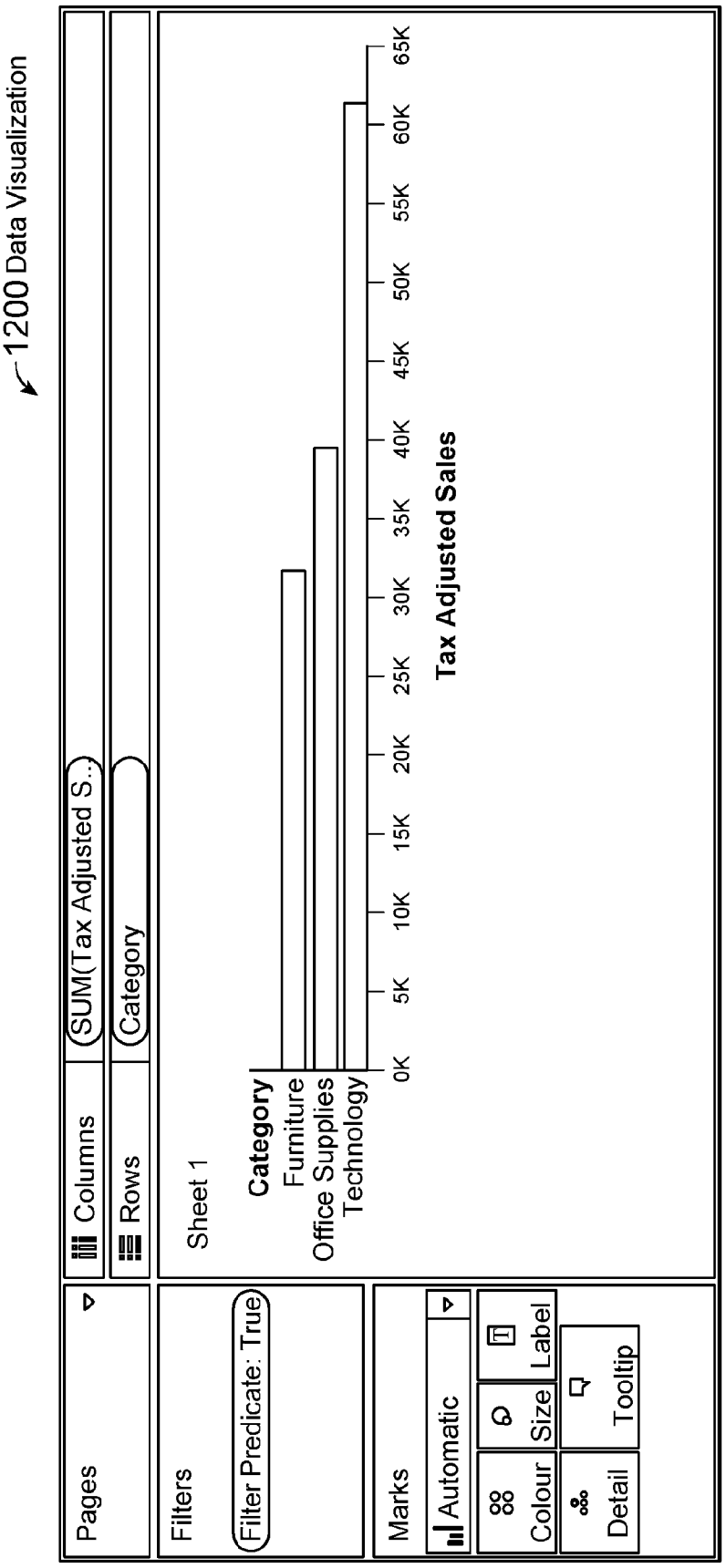

1200 Data Visualization
Pages
Columns
SUM(Tax Adjusted S...)
Rows
Category
Filters
Filter Predicate: True
Sheet 1
Category
Furniture
Office Supplies
Technology
0K
5K
10K
15K
20K
25K
30K
35K
40K
45K
50K
55K
60K
65K
Tax Adjusted Sales
Marks
Automatic
Colour
Size
Label
Detail
Tooltip

Figure 12A

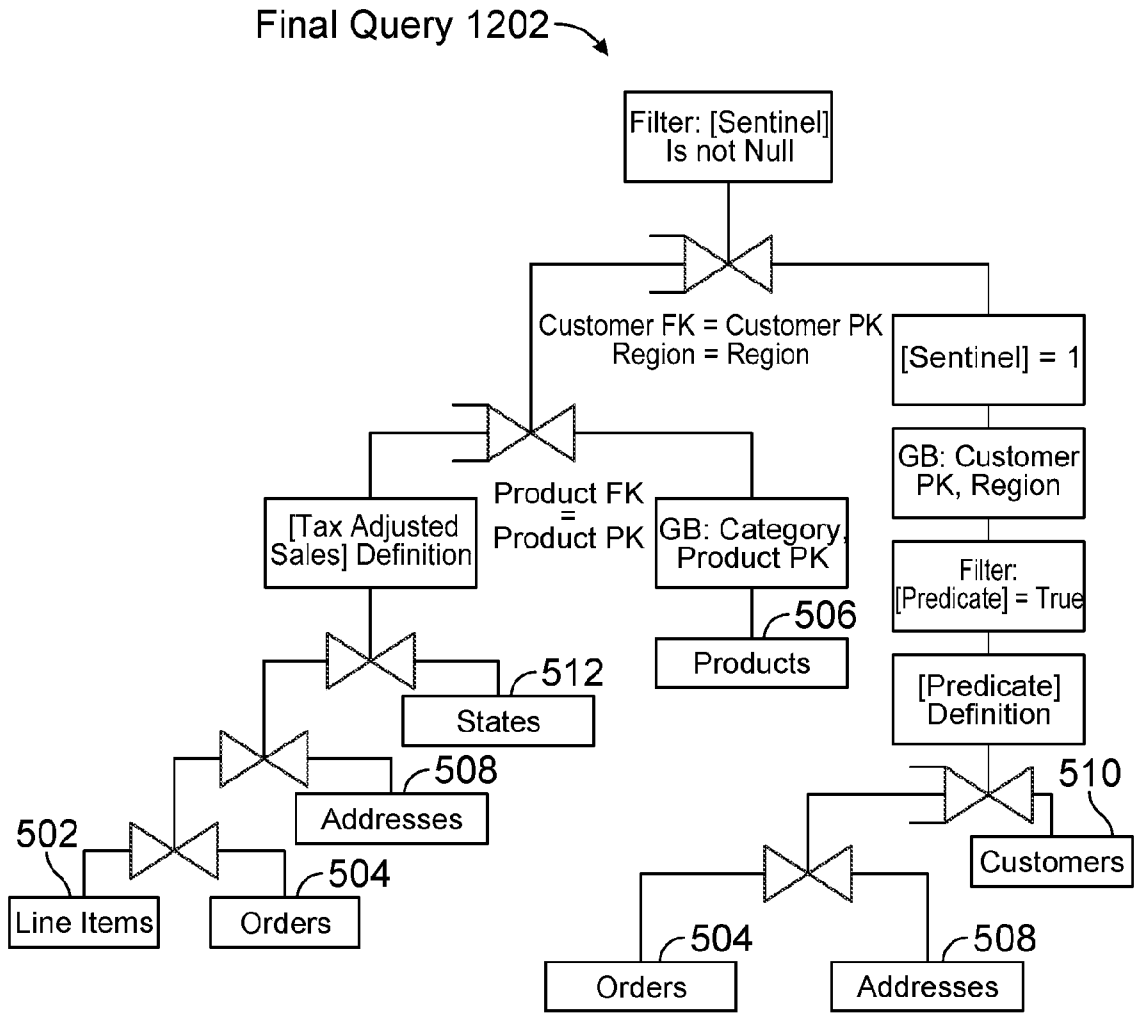

Final Query 1202
Filter: [Sentinel] Is not Null
Customer FK = Customer PK
Region = Region
[Sentinel] = 1
GB: Customer PK, Region
Filter: [Predicate] = True
[Predicate] Definition
Product FK = Product PK
[Tax Adjusted Sales] Definition
GB: Category, Product PK
506
Products
512
States
508
Addresses
502
504
Line Items
Orders
510
Customers
504
Orders
508
Addresses

Figure 12B

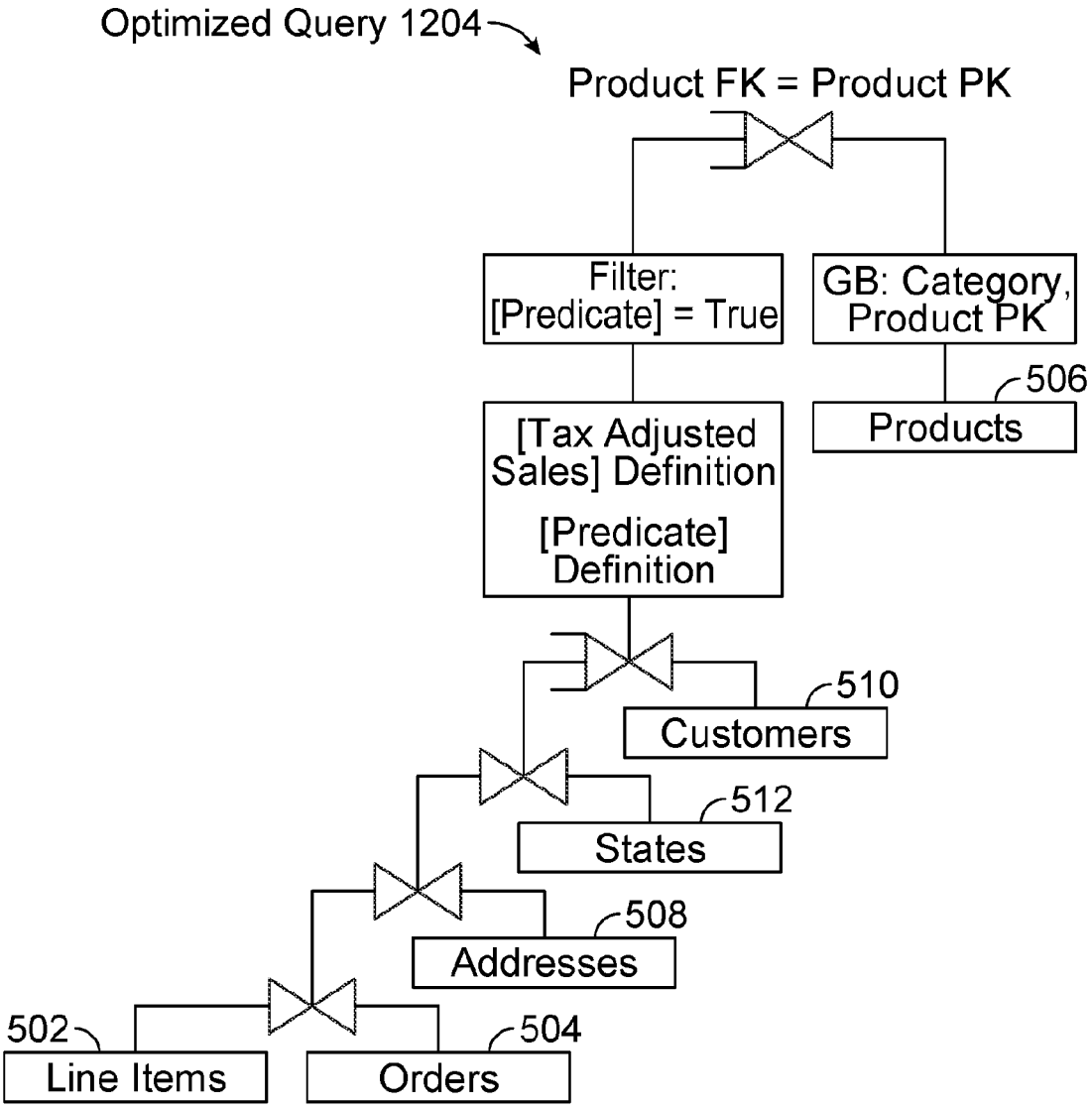

Optimized Query 1204
Product FK = Product PK
Filter: [Predicate] = True
GB: Category, Product PK
506
Products
[Tax Adjusted Sales] Definition
[Predicate] Definition
510
Customers
512
States
508
Addresses
502
Line Items
504
Orders

Figure 12C

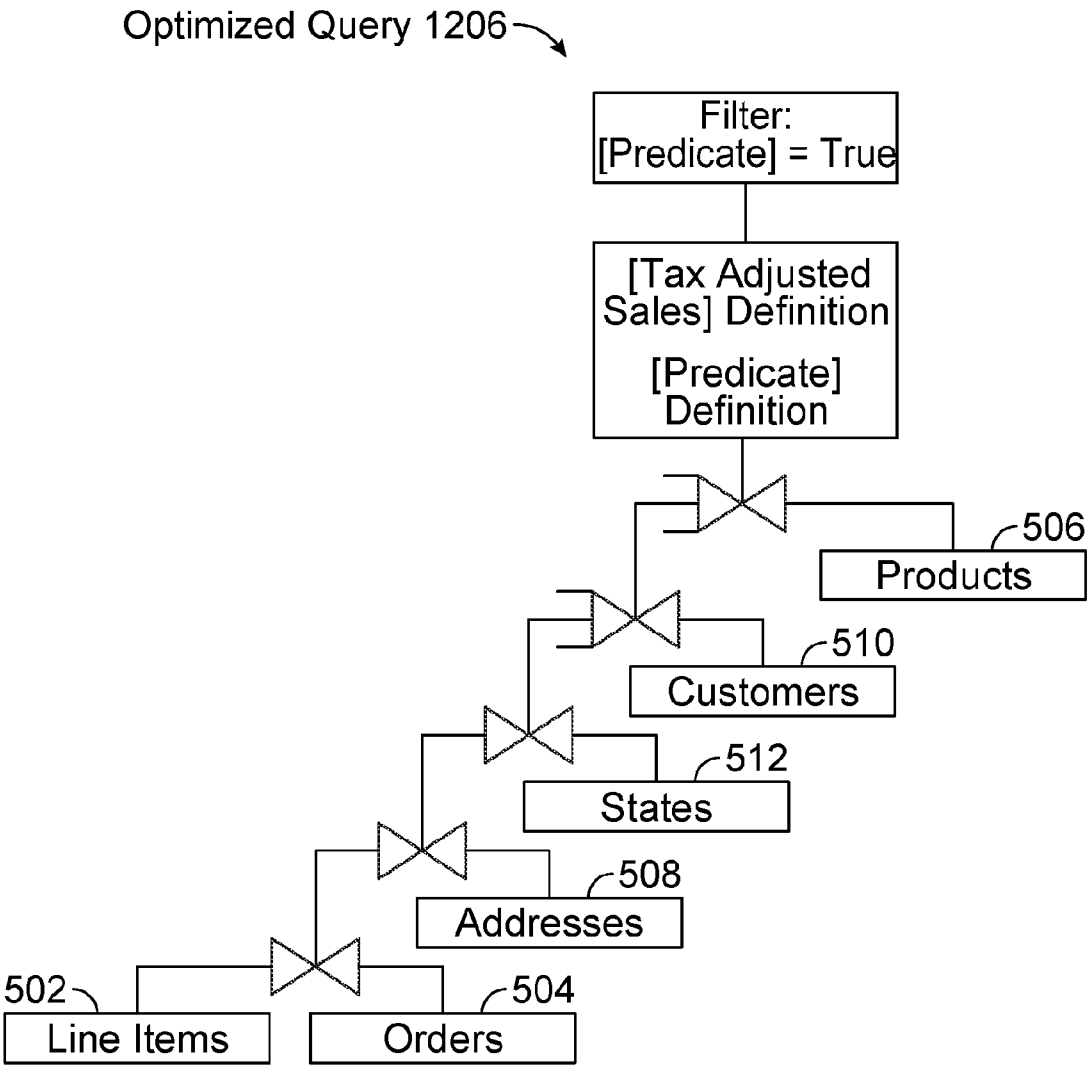

Optimized Query 1206
Filter:
[Predicate] = True
[Tax Adjusted Sales] Definition
[Predicate] Definition
506
Products
510
Customers
512
States
508
Addresses
502
Line Items
504
Orders

Figure 12D

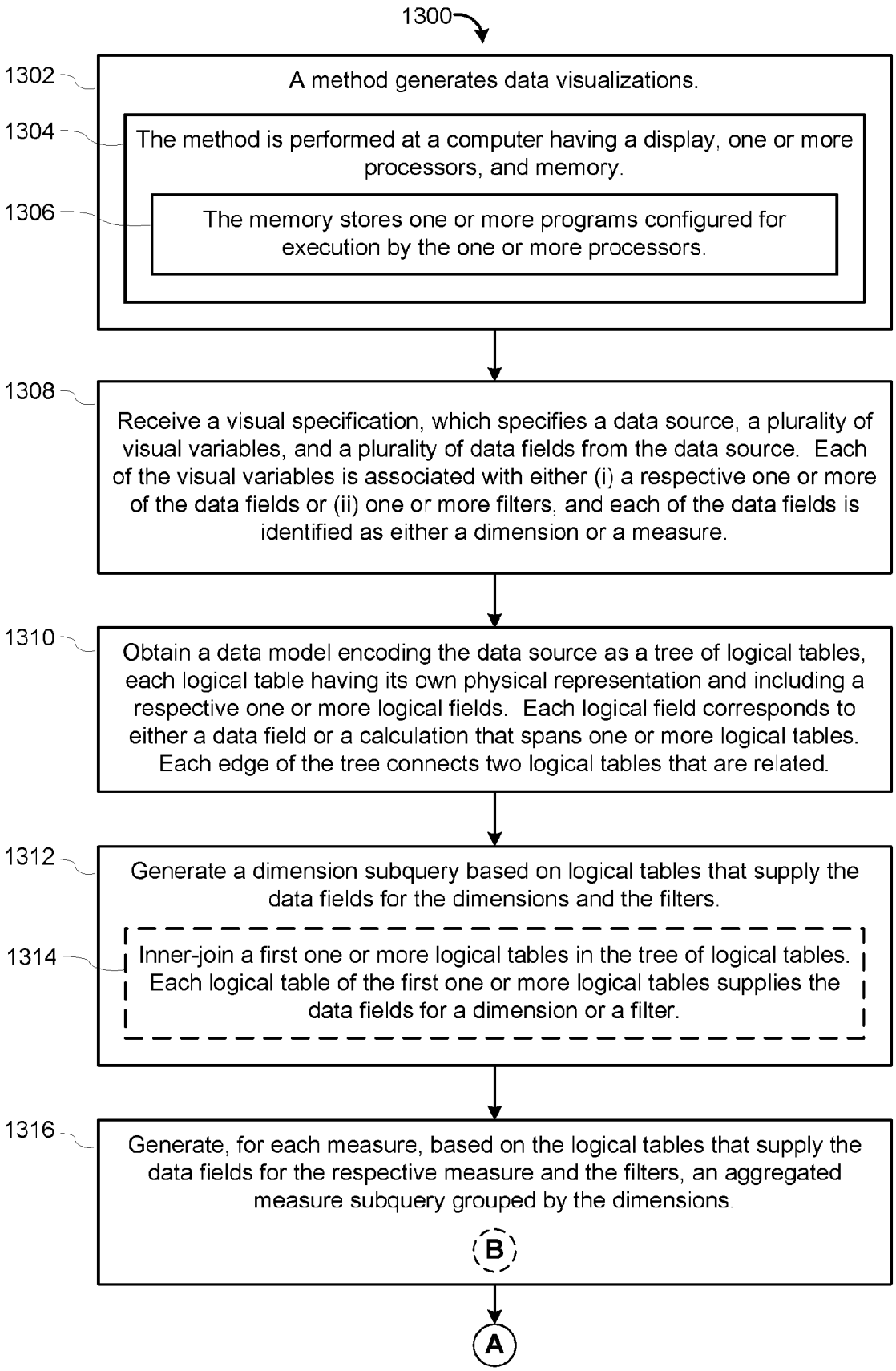

1300
1302
A method generates data visualizations.
1304
The method is performed at a computer having a display, one or more processors, and memory.
1306
The memory stores one or more programs configured for execution by the one or more processors.
1308
Receive a visual specification, which specifies a data source, a plurality of visual variables, and a plurality of data fields from the data source. Each of the visual variables is associated with either (i) a respective one or more of the data fields or (ii) one or more filters, and each of the data fields is identified as either a dimension or a measure.
1310
Obtain a data model encoding the data source as a tree of logical tables, each logical table having its own physical representation and including a respective one or more logical fields. Each logical field corresponds to either a data field or a calculation that spans one or more logical tables. Each edge of the tree connects two logical tables that are related.
1312
Generate a dimension subquery based on logical tables that supply the data fields for the dimensions and the filters.
1314
Inner-join a first one or more logical tables in the tree of logical tables. Each logical table of the first one or more logical tables supplies the data fields for a dimension or a filter.
1316
Generate, for each measure, based on the logical tables that supply the data fields for the respective measure and the filters, an aggregated measure subquery grouped by the dimensions.
B
A

Figure 13A

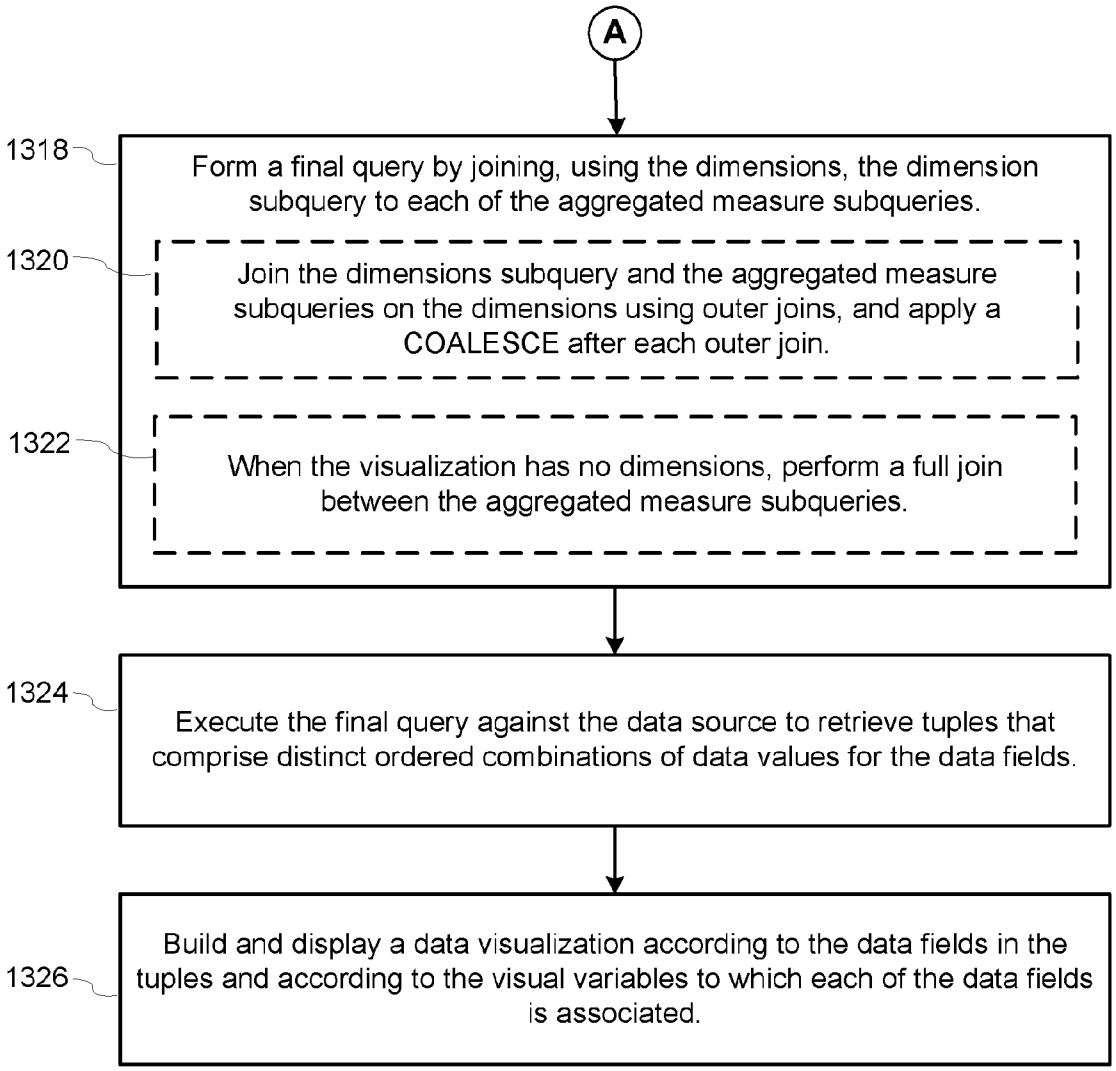

A
1318
Form a final query by joining, using the dimensions, the dimension subquery to each of the aggregated measure subqueries.
1320
Join the dimensions subquery and the aggregated measure subqueries on the dimensions using outer joins, and apply a COALESCE after each outer join.
1322
When the visualization has no dimensions, perform a full join between the aggregated measure subqueries.
1324
Execute the final query against the data source to retrieve tuples that comprise distinct ordered combinations of data values for the data fields.
1326
Build and display a data visualization according to the data fields in the tuples and according to the visual variables to which each of the data fields is associated.

Figure 13B

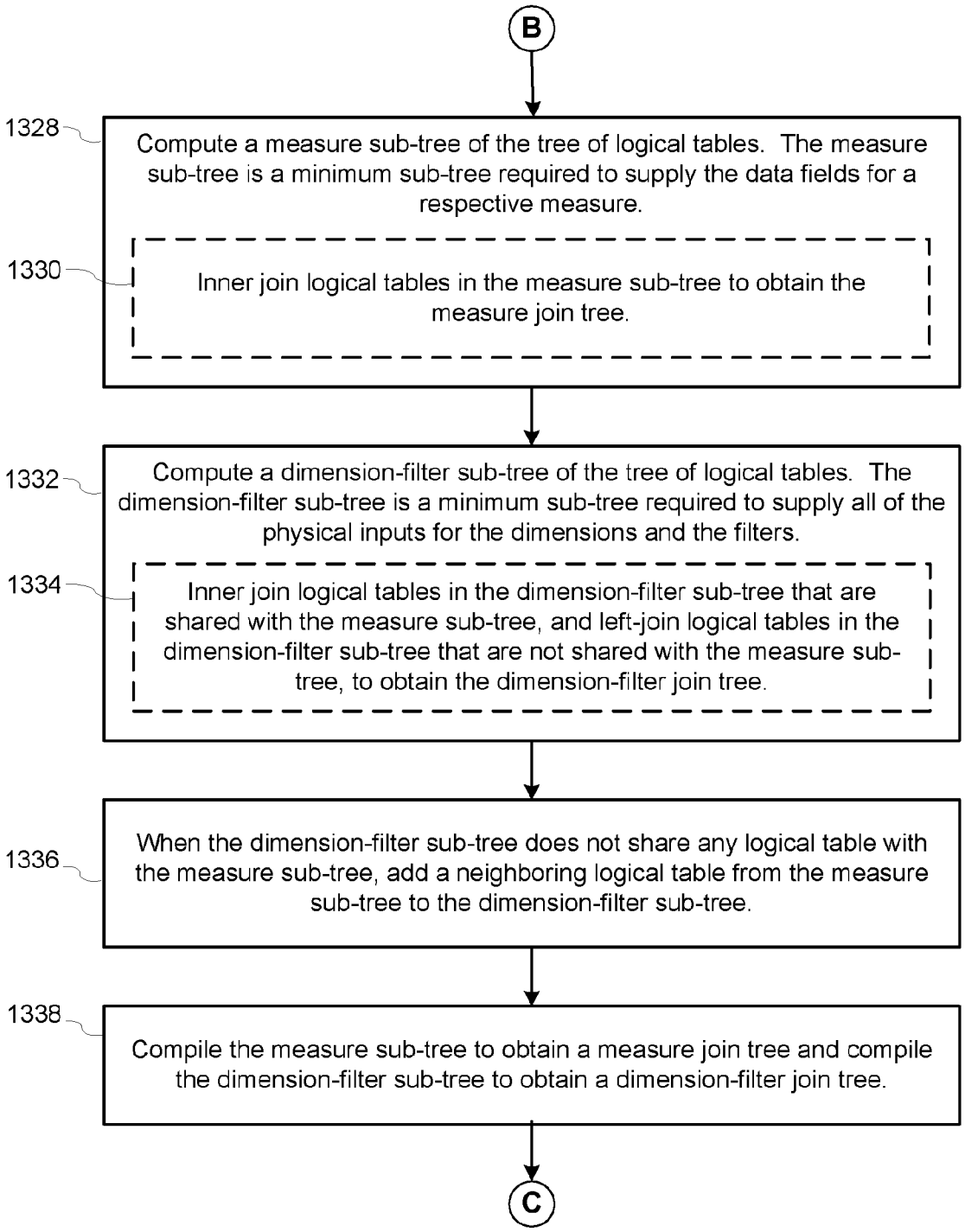

B
1328
Compute a measure sub-tree of the tree of logical tables. The measure sub-tree is a minimum sub-tree required to supply the data fields for a respective measure.
1330
Inner join logical tables in the measure sub-tree to obtain the measure join tree.
1332
Compute a dimension-filter sub-tree of the tree of logical tables. The dimension-filter sub-tree is a minimum sub-tree required to supply all of the physical inputs for the dimensions and the filters.
1334
Inner join logical tables in the dimension-filter sub-tree that are shared with the measure sub-tree, and left-join logical tables in the dimension-filter sub-tree that are not shared with the measure sub-tree, to obtain the dimension-filter join tree.
1336
When the dimension-filter sub-tree does not share any logical table with the measure sub-tree, add a neighboring logical table from the measure sub-tree to the dimension-filter sub-tree.
1338
Compile the measure sub-tree to obtain a measure join tree and compile the dimension-filter sub-tree to obtain a dimension-filter join tree.
C

Figure 13C

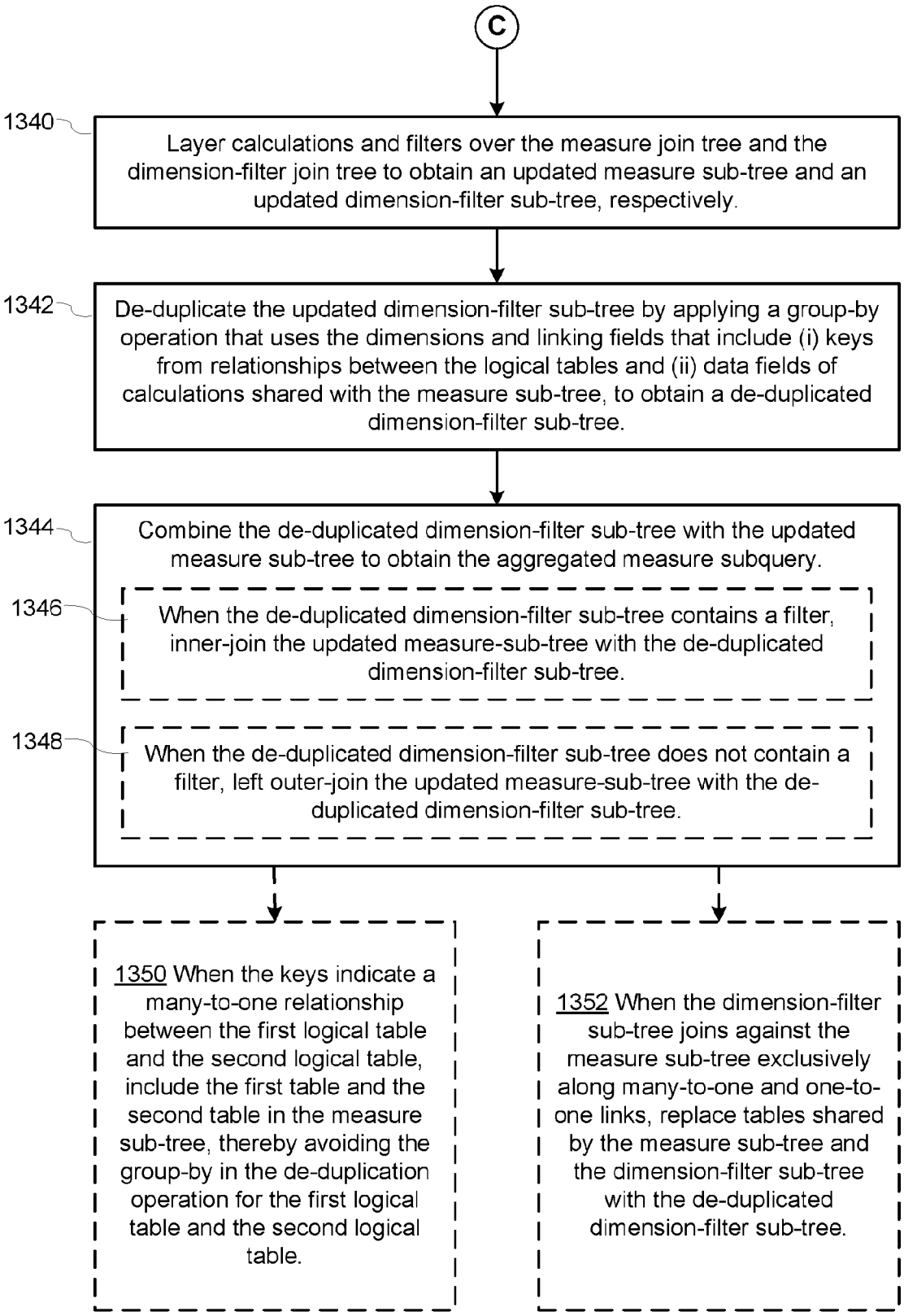

C
1340
Layer calculations and filters over the measure join tree and the dimension-filter join tree to obtain an updated measure sub-tree and an updated dimension-filter sub-tree, respectively.
1342
De-duplicate the updated dimension-filter sub-tree by applying a group-by operation that uses the dimensions and linking fields that include (i) keys from relationships between the logical tables and (ii) data fields of calculations shared with the measure sub-tree, to obtain a de-duplicated dimension-filter sub-tree.
1344
Combine the de-duplicated dimension-filter sub-tree with the updated measure sub-tree to obtain the aggregated measure subquery.
1346
When the de-duplicated dimension-filter sub-tree contains a filter, inner-join the updated measure-sub-tree with the de-duplicated dimension-filter sub-tree.
1348
When the de-duplicated dimension-filter sub-tree does not contain a filter, left outer-join the updated measure-sub-tree with the de-duplicated dimension-filter sub-tree.
1350 When the keys indicate a many-to-one relationship between the first logical table and the second logical table, include the first table and the second table in the measure sub-tree, thereby avoiding the group-by in the de-duplication operation for the first logical table and the second logical table.
1352 When the dimension-filter sub-tree joins against the measure sub-tree exclusively along many-to-one and one-to-one links, replace tables shared by the measure sub-tree and the dimension-filter sub-tree with the de-duplicated dimension-filter sub-tree.

Figure 13D

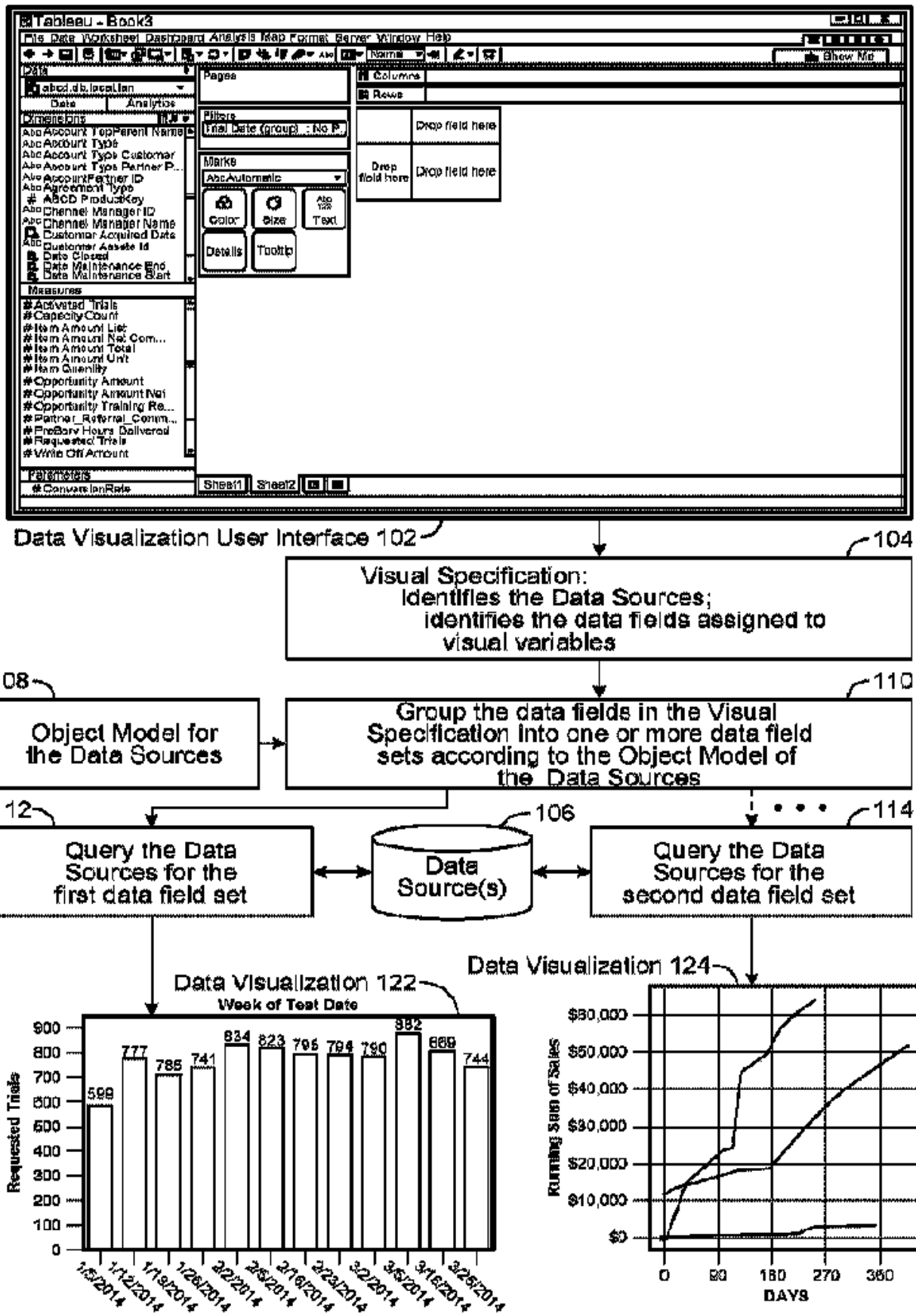


Tableau - Book3
File Data Worksheet Dashboard Analysis Map Format Server Window Help
Data
Pages
Columns
Rows
Filters
Marks
Automatic
Color
Size
Text
Details
Tooltip
Drop field here
Measures
Parameters
Sheet1
Sheet2
Data Visualization User Interface 102
104
Visual Specification:
Identifies the Data Sources;
identifies the data fields assigned to visual variables
108
Object Model for the Data Sources
110
Group the data fields in the Visual Specification into one or more data field sets according to the Object Model of the Data Sources
112
Query the Data Sources for the first data field set
106
Data Source(s)
114
Query the Data Sources for the second data field set
Data Visualization 122
Week of Test Date
Requested Trials
599
777
785
741
834
823
795
794
790
882
809
744
1/5/2014
1/12/2014
1/19/2014
1/26/2014
2/2/2014
2/9/2014
2/16/2014
2/23/2014
3/2/2014
3/9/2014
3/16/2014
3/25/2014
Data Visualization 124
Running Sum of Sales
$60,000
$50,000
$40,000
$30,000
$20,000
$10,000
$0
0
90
180
270
360
DAYS