US 20180137667A1

(19) **United States**

(12) **Patent Application Publication**

Kindelsberger et al.

(10) **Pub. No.: US 2018/0137667 A1**

(43) **Pub. Date: May 17, 2018**

(54) **GRAPH VISUALIZATION TOOLS WITH SUMMARY VISUALIZATION FOR VERY LARGE LABELED GRAPHS**

(71) Applicant: **Oracle International Corporation**, Redwood Shores, CA (US)

(72) Inventors: **Julia Kindelsberger**, Munich (DE); **Daniel Langerenken**, Foster City, CA (US); **Korbinian Schmid**, San Mateo, CA (US); **Sungpack Hong**, Palo Alto, CA (US); **Hassan Chafi**, San Mateo, CA (US)

(21) Appl. No.: **15/469,674**

(22) Filed: **Mar. 27, 2017**

Related U.S. Application Data

(60) Provisional application No. 62/421,837, filed on Nov. 14, 2016.

Publication Classification

(51) **Int. Cl.**

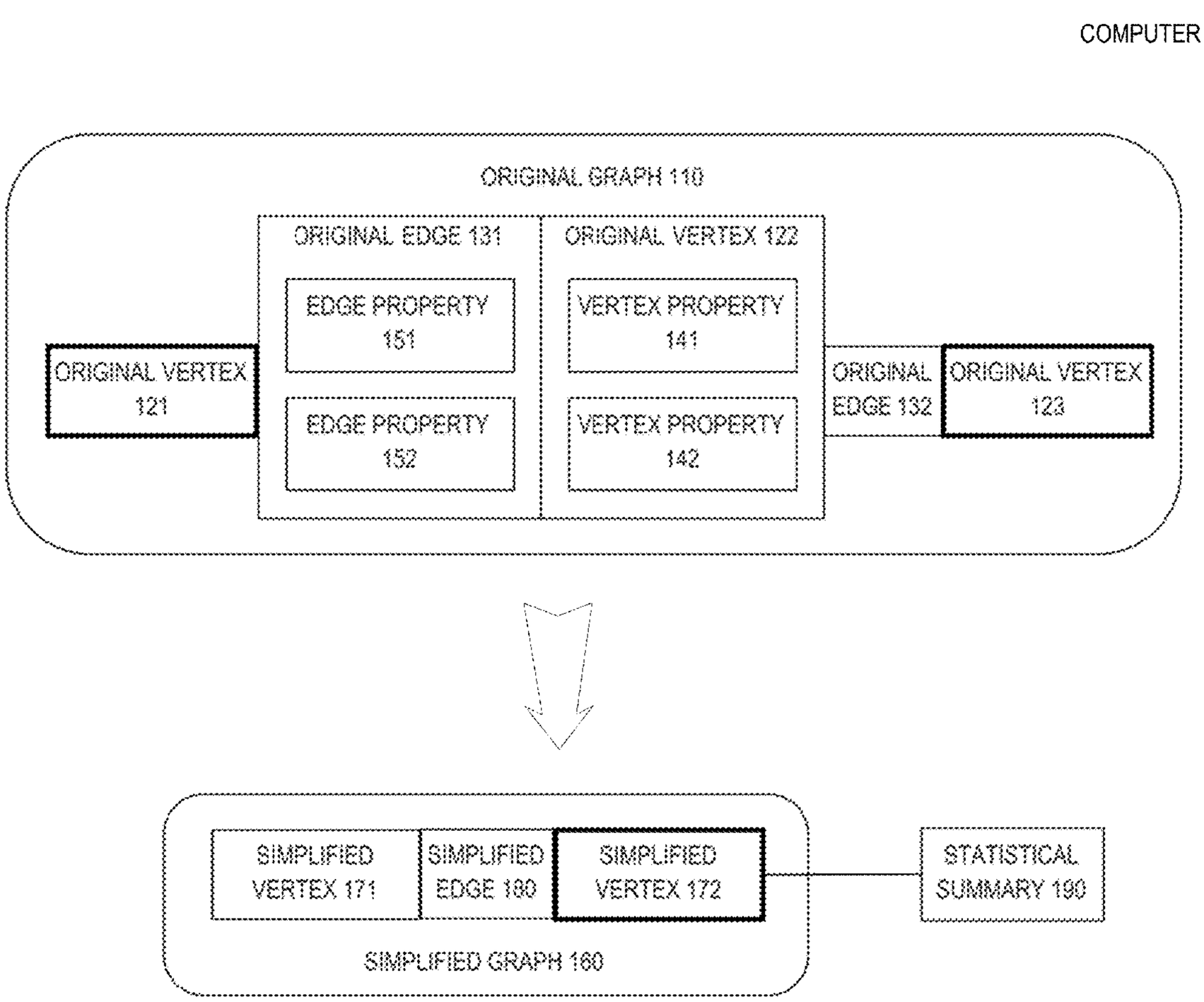
G06T 15/00	(2006.01)
G06T 11/20	(2006.01)
G06T 17/00	(2006.01)

(52) **U.S. Cl.**

CPC ***G06T 15/005*** (2013.01); ***G06T 17/005*** (2013.01); ***G06T 11/206*** (2013.01)

(57) **ABSTRACT**

Techniques herein visually simplify and summarize property graphs. In an embodiment, a computer loads an original graph that contains original vertices interconnected by original edges. Each original vertex contains vertex properties, each original edge contains edge properties. Based on the vertex properties of the original vertices and the edge properties of the original edges, the computer generates and displays a simplified graph that contains simplified vertices that each represents multiple original vertices, and simplified edges that each represents multiple original edges. Responsive to an interactive selection of a particular simplified vertex or edge of the simplified graph, the computer displays a statistical summary based on the multiple original vertices represented by the particular simplified vertex, or the multiple original edges represented by the particular simplified edge.



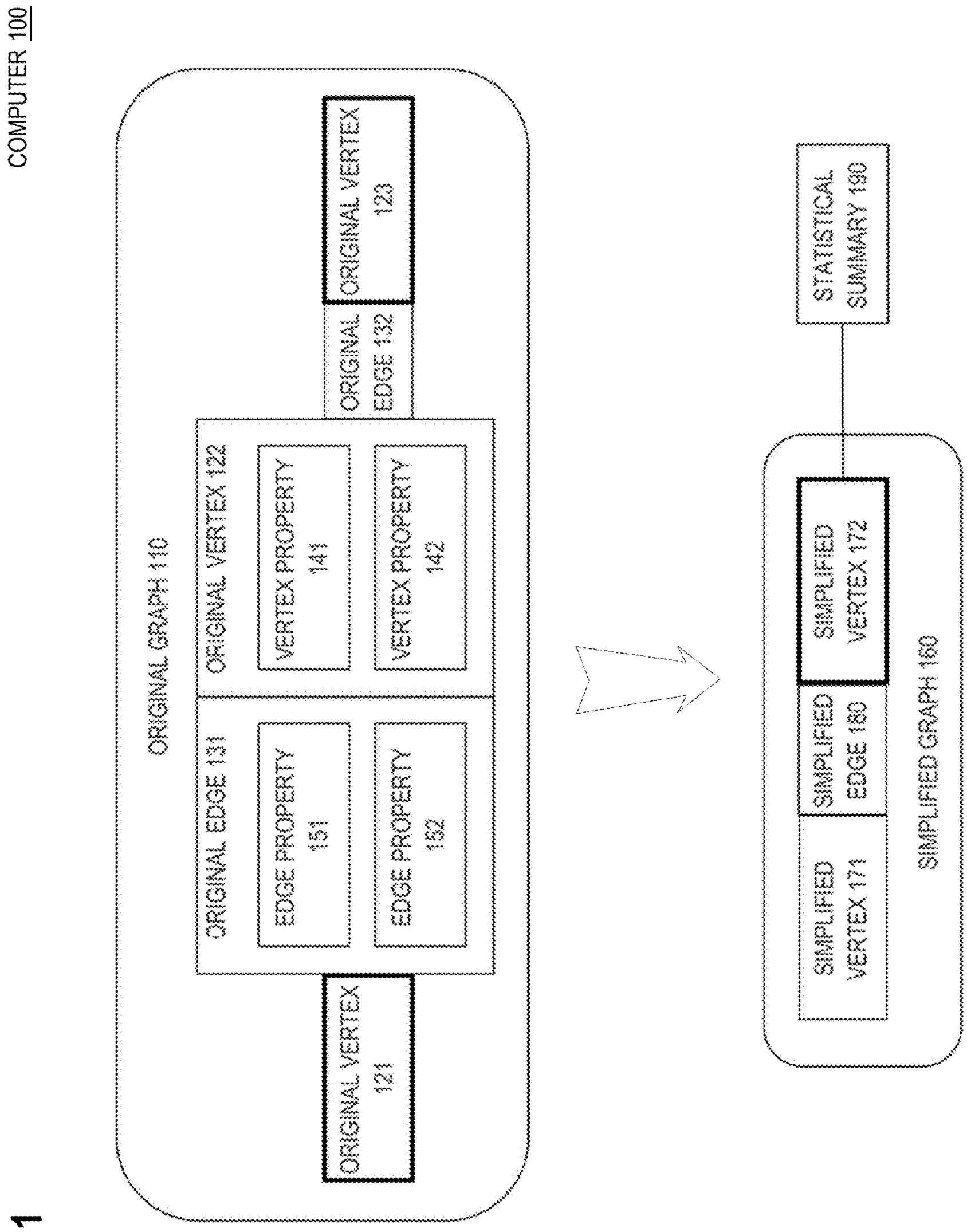

COMPUTER 100
ORIGINAL GRAPH 110
ORIGINAL VERTEX 123
ORIGINAL EDGE 132
ORIGINAL VERTEX 122
VERTEX PROPERTY 141
VERTEX PROPERTY 142
ORIGINAL EDGE 131
EDGE PROPERTY 151
EDGE PROPERTY 152
ORIGINAL VERTEX 121
STATISTICAL SUMMARY 190
SIMPLIFIED GRAPH 160
SIMPLIFIED VERTEX 172
SIMPLIFIED EDGE 180
SIMPLIFIED VERTEX 171

FIG. 1

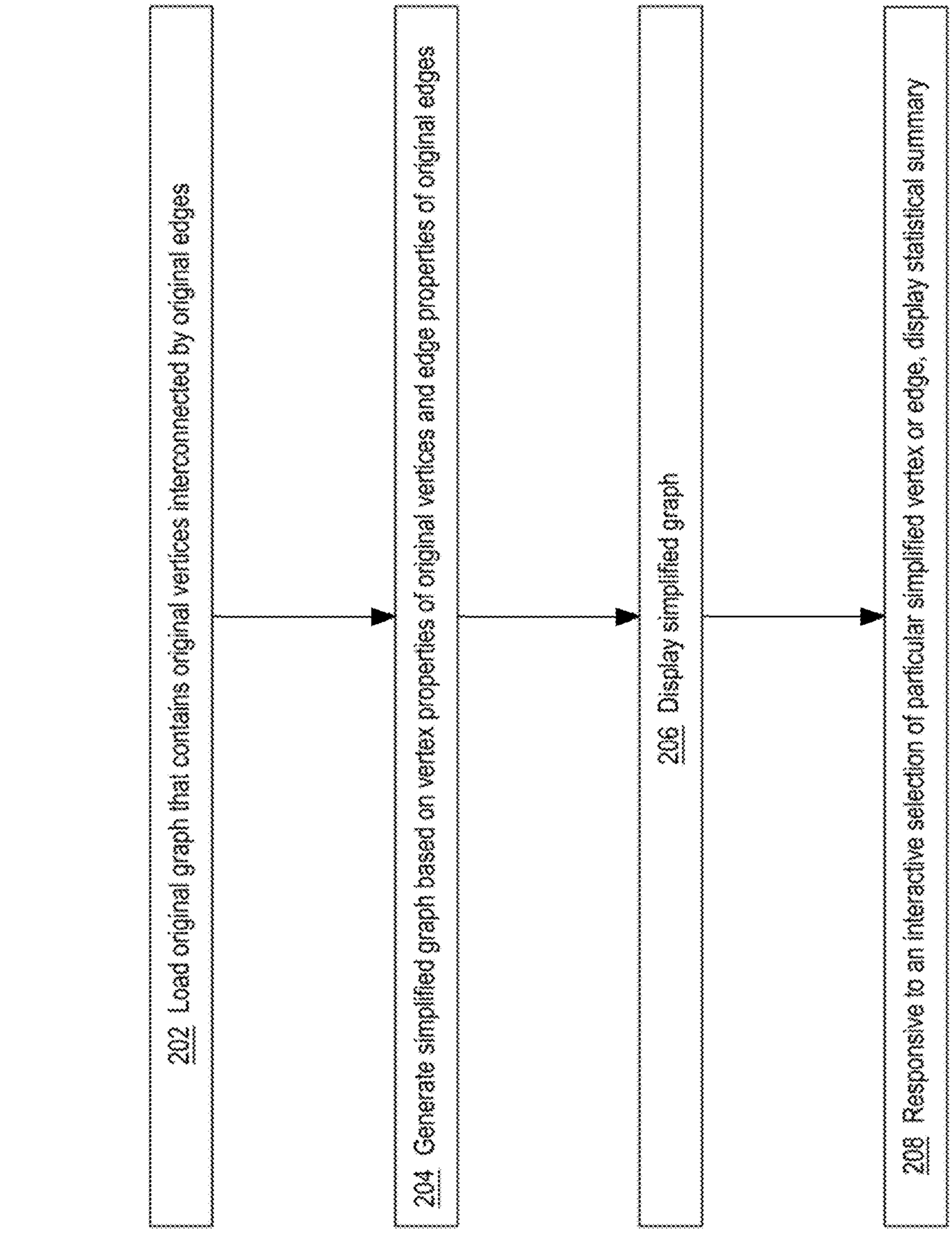

202 Load original graph that contains original vertices interconnected by original edges
204 Generate simplified graph based on vertex properties of original vertices and edge properties of original edges
206 Display simplified graph
208 Responsive to an interactive selection of particular simplified vertex or edge, display statistical summary

FIG. 2

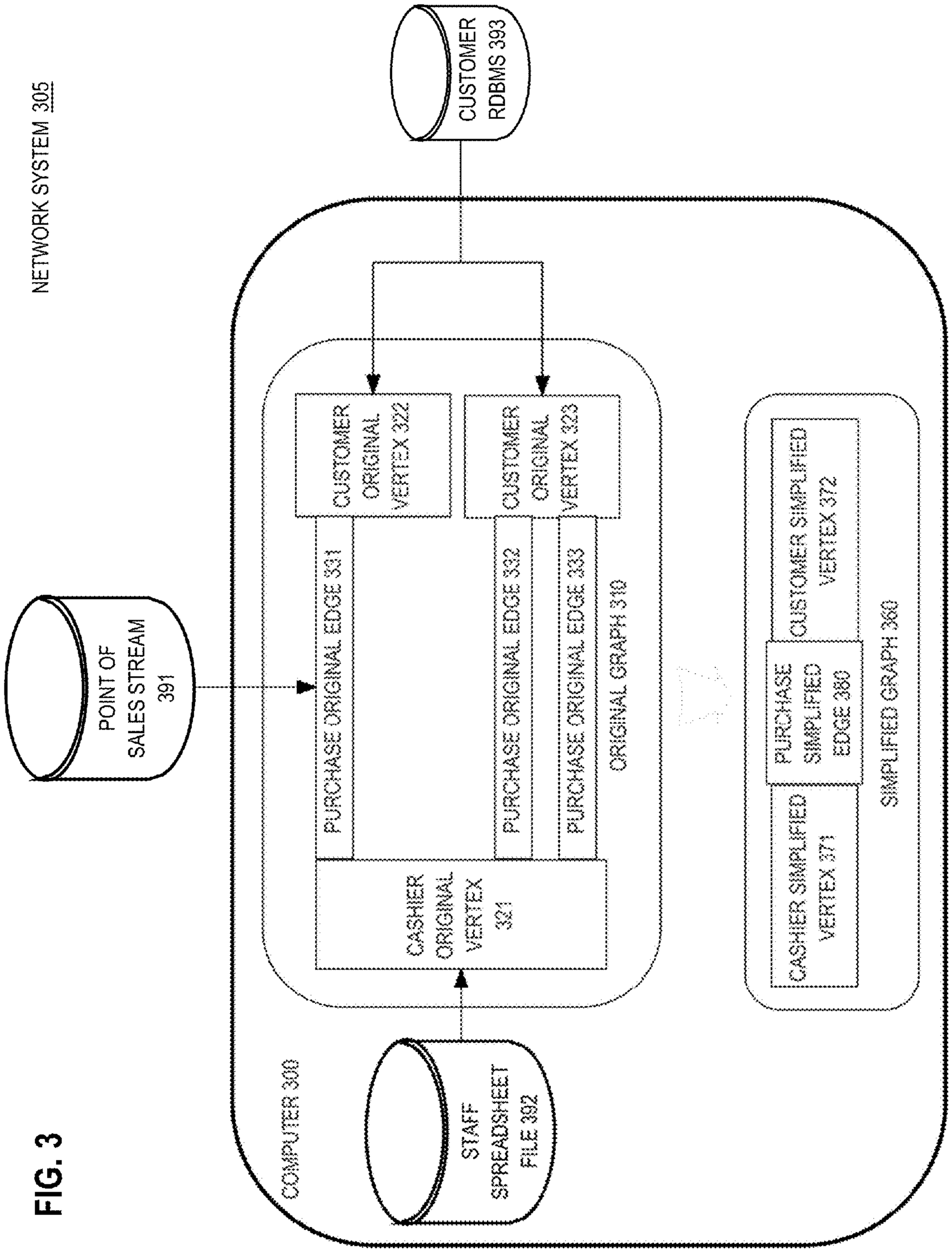
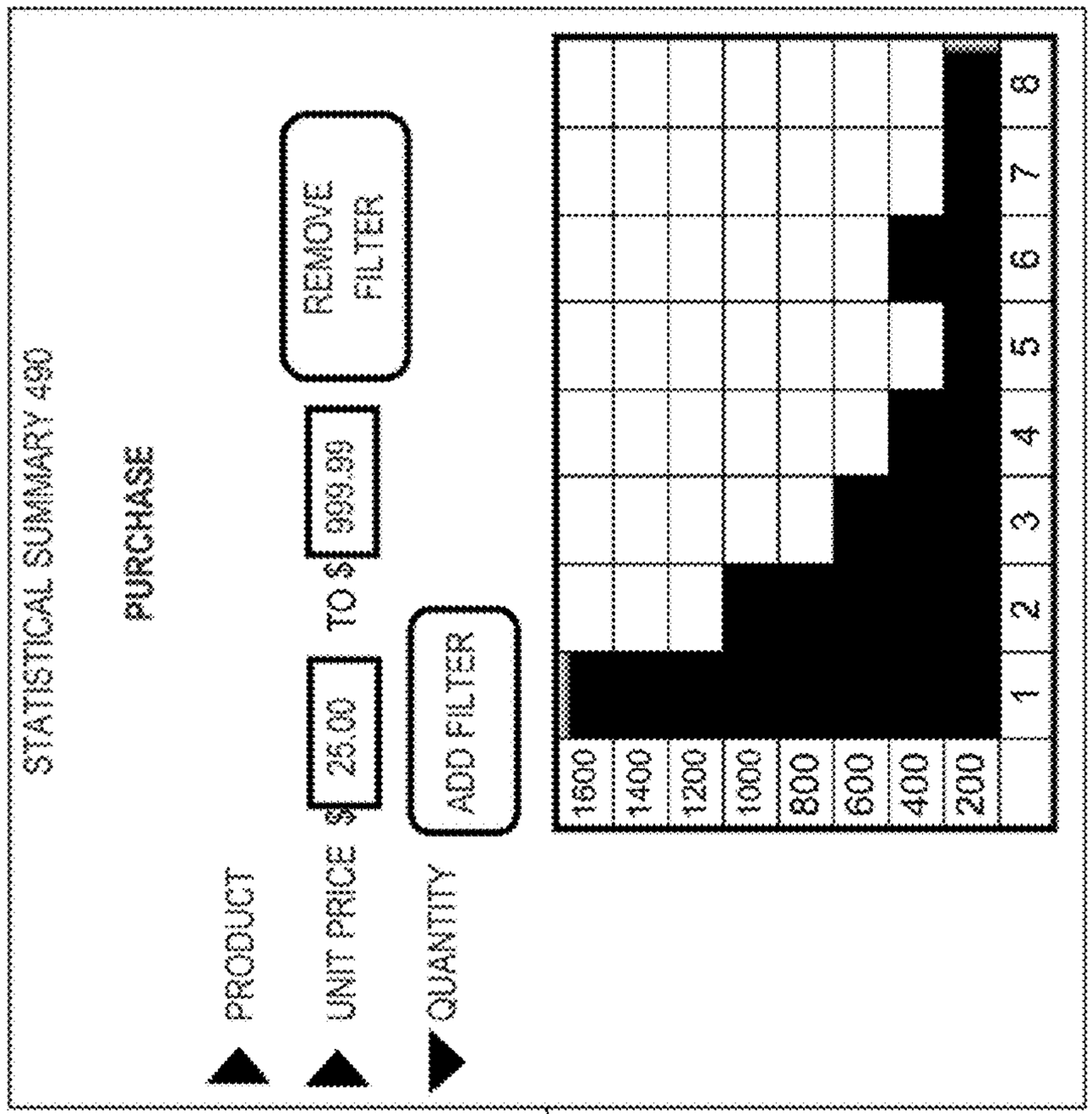

NETWORK SYSTEM 305
CUSTOMER RDBMS 393
POINT OF SALES STREAM 391
COMPUTER 300
ORIGINAL GRAPH 310
CUSTOMER ORIGINAL VERTEX 322
CUSTOMER ORIGINAL VERTEX 323
PURCHASE ORIGINAL EDGE 331
PURCHASE ORIGINAL EDGE 332
PURCHASE ORIGINAL EDGE 333
CASHIER ORIGINAL VERTEX 321
STAFF SPREADSHEET FILE 392
SIMPLIFIED GRAPH 360
CUSTOMER SIMPLIFIED VERTEX 372
PURCHASE SIMPLIFIED EDGE 380
CASHIER SIMPLIFIED VERTEX 371

FIG. 3

DISPLAY 400



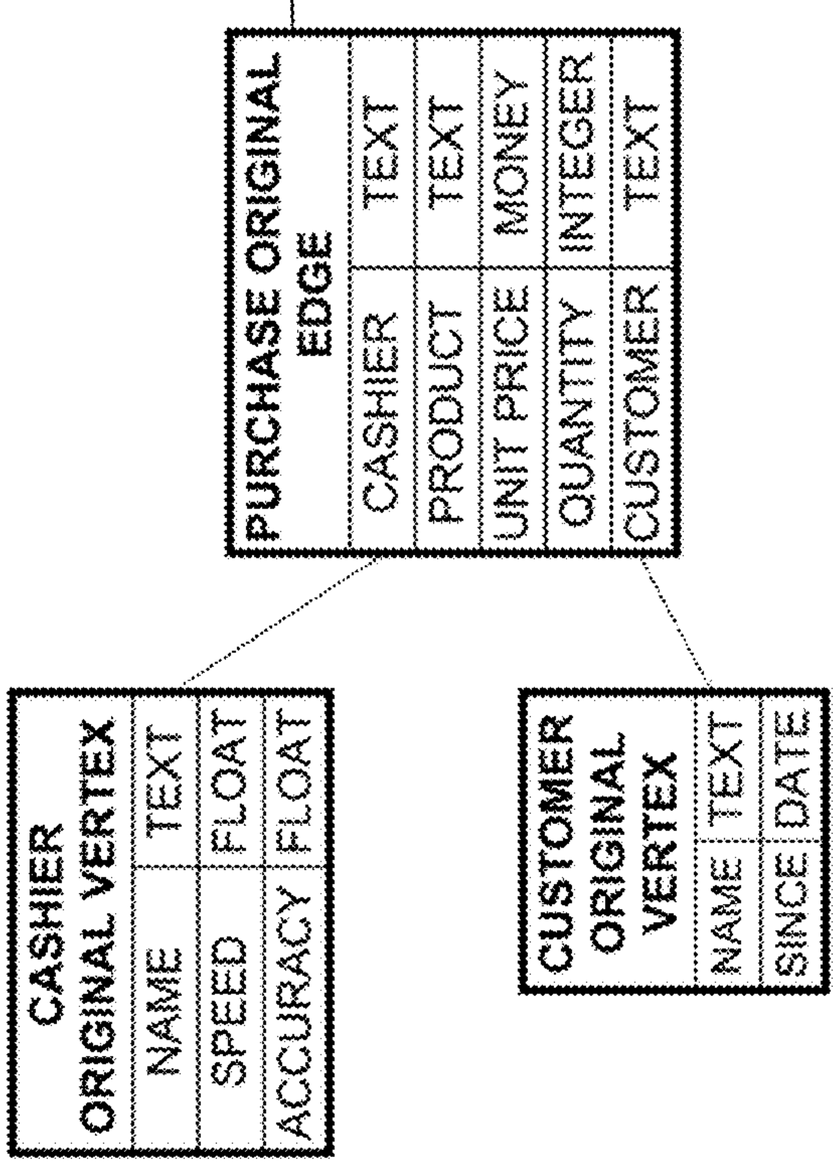


FIG. 4

DISPLAY 500

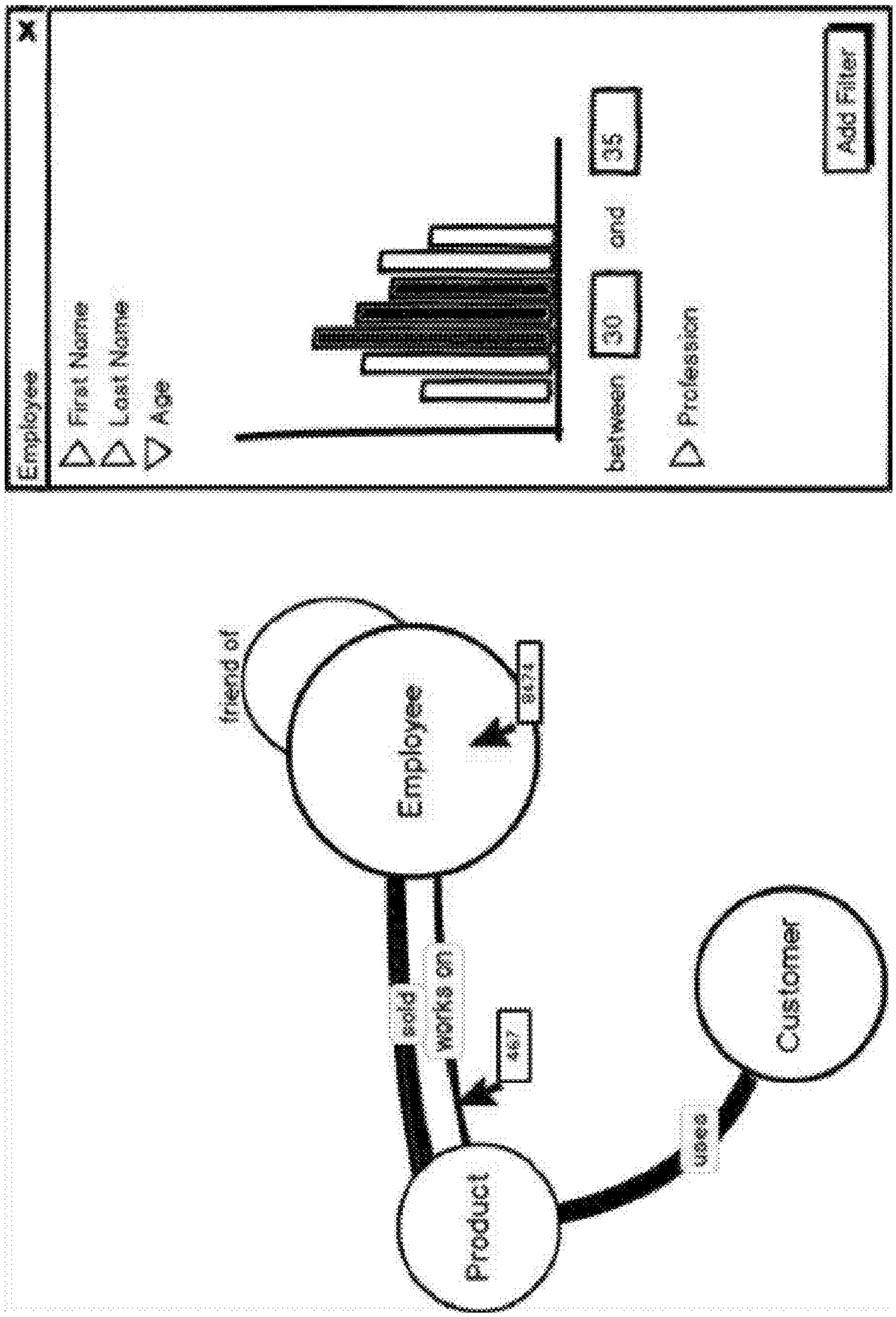
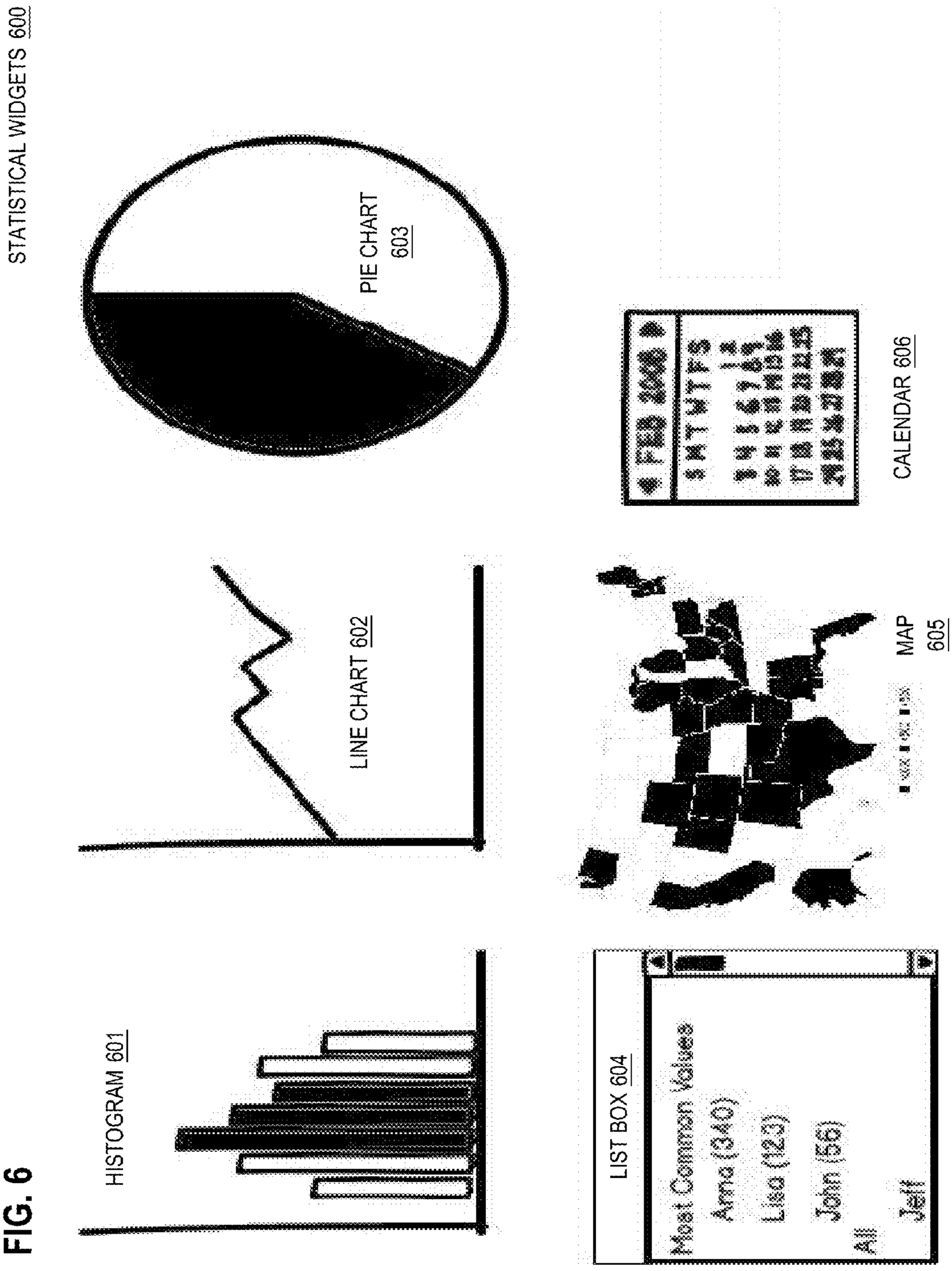


FIG. 5

FIG. 6


STATISTICAL WIDGETS 600
PIE CHART 603
LINE CHART 602
CALENDAR 606
MAP 605
HISTOGRAM 601
LIST BOX 604
Most Common Values
Arma (340)
Lisa (123)
John (56)
All
Jeff

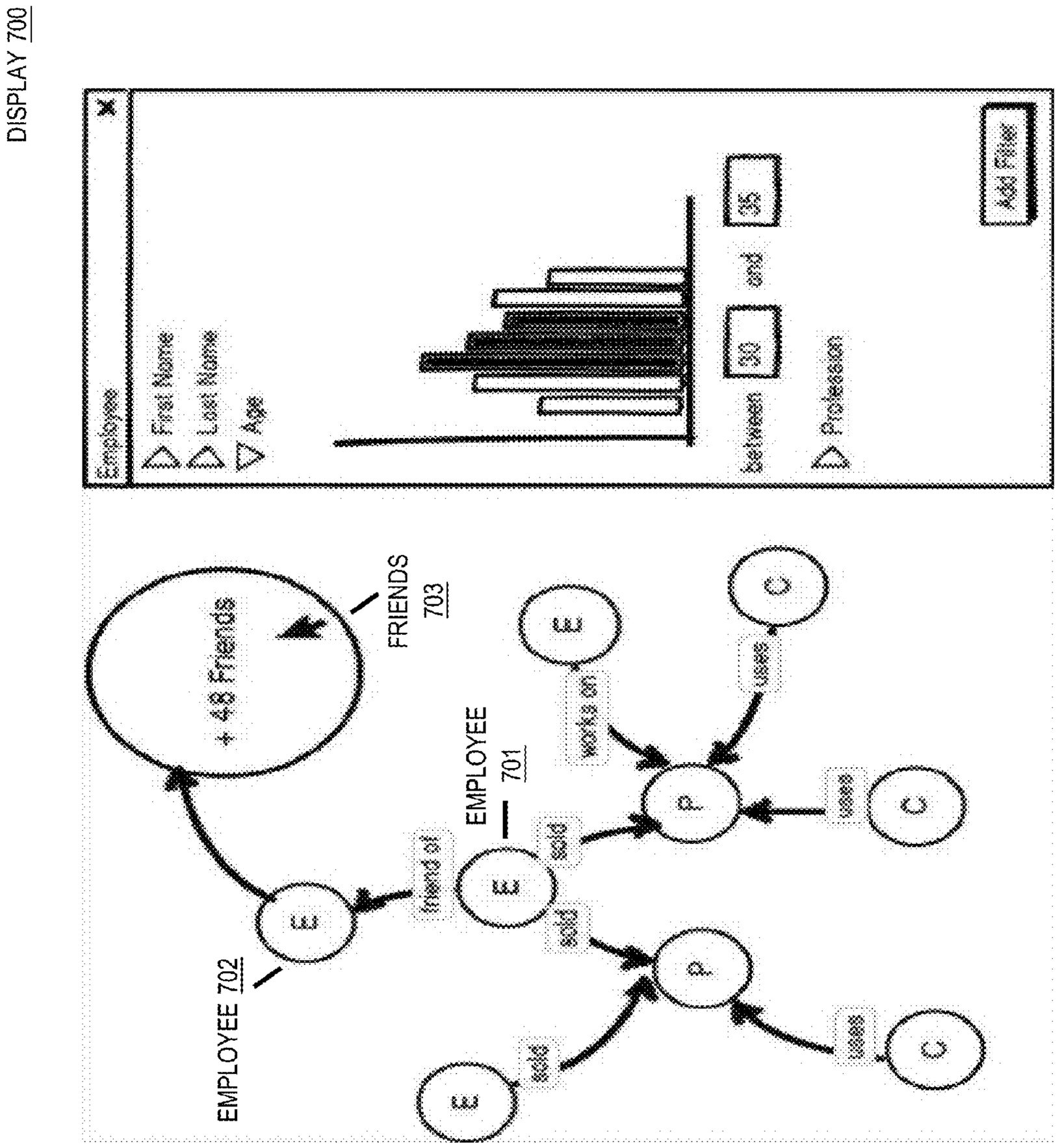

DISPLAY 700
Employee
First Name
Last Name
Age
Job
Add Filter
Between
and
Profession
EMPLOYEE 702
+ 48 Friends
FRIENDS 703
friend of
EMPLOYEE 701
sold
works on
uses
sold
uses
uses

FIG. 7

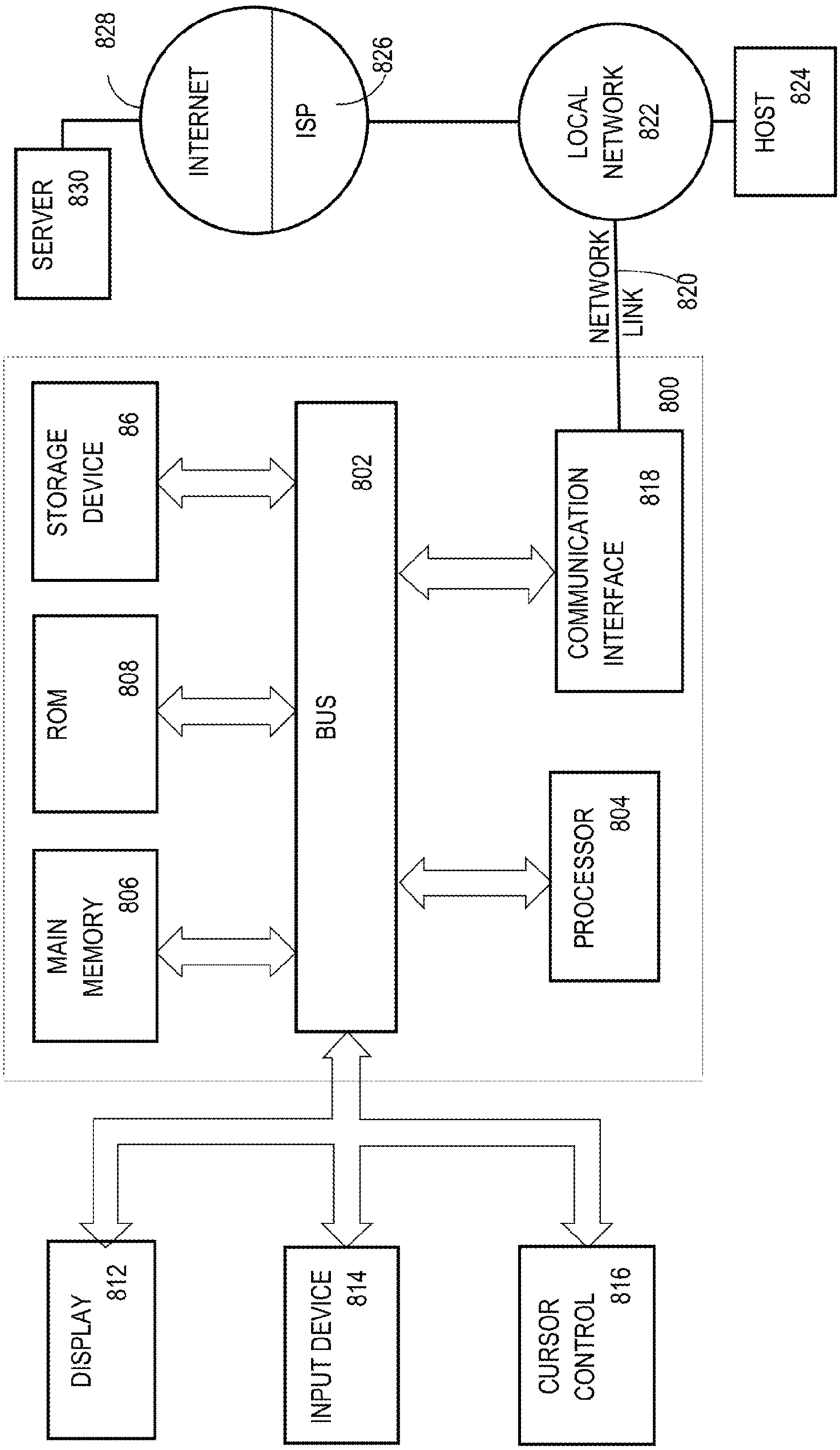

828
INTERNET
ISP
826
SERVER
830
LOCAL
NETWORK
822
HOST
824
NETWORK
LINK
820
800
STORAGE
DEVICE
86
ROM
808
MAIN
MEMORY
806
BUS
802
COMMUNICATION
INTERFACE
818
PROCESSOR
804
DISPLAY
812
INPUT DEVICE
814
CURSOR
CONTROL
816

FIG. 8

GRAPH VISUALIZATION TOOLS WITH SUMMARY VISUALIZATION FOR VERY LARGE LABELED GRAPHS

BENEFIT CLAIM

[0001] This application claims the benefit of Provisional Appln. 62/421,837 (Atty Docket No. 50277-5113), filed Nov. 14, 2016, the entire contents of which is hereby incorporated by reference as if fully set forth herein, under 35 U.S.C. § 119(e).

FIELD OF THE DISCLOSURE

[0002] This disclosure relates to graph drawing. Presented herein are techniques that simplify and summarize a property graph in a way that suits interactive exploration.

BACKGROUND

[0003] Graph drawing is a convenient and effective tool for visualizing relationship between data entities. By representing data entities as vertices and their fine-grained interconnections as edges, graph visualization provides users a quick and intuitive understanding of the data set. However, the effectiveness of graph visualization degrades as the size of a data set grows when there are too many vertices and edges to clearly visualize.

[0004] Once the amount of data becomes larger than some threshold, occlusion and data density become too high for an effective visualization. That is, the vertices and edges overlap extensively and therefore the user is overwhelmed by the volume of information, rather than being given a well-summarized understanding. Consequently, graph visualization has had limited success in visualizing very large data sets.

[0005] There have been some approaches to resolve the large graph visualization problem, most prominently pan-and-zoom, clustering/communities, and a hyperbolic fish eye. Pan and zoom is one way the user can explore large graphs, however the user sees just a part of the graph when zooming in, instead of getting an overview of all the information in the graph. Communities group vertices of a graph into sets, such that each set is densely connected internally. Each group/set can be visually distinguished (e.g. different color, size, shape), and the vertices that belong to a group can be visualized closer together. In the fish-eye view, the selected area is magnified and the surrounding areas are distorted (shrunk) to remain in view.

[0006] One approach for visualizing large and complex graphs uses variable scoping to impose a hierarchy on vertices in a graph. Hierarchy information is used in order to collapse vertices of a lower hierarchy into one vertex. However, that technique is based on topology, which presumes that the topology is more or less naturally layered and thus not generally applicable.

[0007] None of these conventional solutions focus on providing a summary of the graph data. Instead at best, they focus on hiding subgraphs that are explicitly indicated by the user, or readily recognizable based on topology.

[0008] Traditionally, large graph visualization methods focus on visualizing each vertex and edge and may offer techniques to hide parts of the graph or summarize communities based on topology. Those techniques do not perform summarization based on vertex or edge properties, as with a property graph. Essentially, those techniques ignore significant aspects of property summarization, such as property statistics.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] In the drawings:

[0010] FIG. **1** is a block diagram that depicts an example computer that displays a statistical summary of a selected vertex or edge of a simplified graph that is based on an original graph, in an embodiment;

[0011] FIG. **2** is a flow diagram that depicts an example process that displays a statistical summary of a selected vertex or edge of a simplified graph that is based on an original graph, in an embodiment;

[0012] FIG. **3** is a block diagram that depicts an example network system that receives information from multiple data sources to synthesize an original graph, in an embodiment;

[0013] FIG. **4** is a block diagram that depicts an example display that summarizes raw data as a preface to creating an original graph, in an embodiment;

[0014] FIG. **5** is a screen snapshot that depicts an example display that shows a simplified graph suitable for interactive exploration, in an embodiment;

[0015] FIG. **6** is a diagram that depicts example statistical widgets that present statistics in meaningful ways, in an embodiment;

[0016] FIG. **7** is a screen snapshot that depicts an example display that shows a simplified graph having mixed simplification, in an embodiment;

[0017] FIG. **8** is a block diagram that illustrates a computer system upon which an embodiment of the invention may be implemented.

DETAILED DESCRIPTION

[0018] In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

[0019] Embodiments are described herein according to the following outline:

[0020] 1.0 General Overview
[0021] 2.0 Example Computer
  [0022] 2.1 Original Graph
  [0023] 2.2 Simplified Graph
  [0024] 2.3 Properties
  [0025] 2.4 Simplification
  [0026] 2.5 Decoration
  [0027] 2.6 Rendition
  [0028] 2.7 Statistical Summary
  [0029] 2.8 Interactive Selection
  [0030] 2.9 Statistics
[0031] 3.0 Display Process
[0032] 4.0 Loading Raw Data
[0033] 5.0 Graph Schema
  [0034] 5.1 Interactive Schema Capture
  [0035] 5.2 Filtration
[0036] 6.0 Example Display
  [0037] 6.1 Reverse Source Vertex Array
  [0038] 6.2 Inbound Neighbor Array

[0039] 6.3 Backward Change Navigation
[0040] 6.4 Inbound Vertex Array
[0041] 6.5 Inbound Materialized Array
[0042] 7.0 Statistical Widgets
[0043] 8.0 Interactive Exploration
[0044] 8.1 Incomplete Simplification
[0045] 8.2 Drilling Down
[0046] 8.3 Rolling Up
[0047] 9.0 Hardware Overview

1.0 General Overview

[0048] Techniques are provided for visually simplifying and summarizing property graphs. In an embodiment, a computer loads an original graph that contains original vertices interconnected by original edges. Each original vertex contains vertex properties. Each original edge contains edge properties. Based on the vertex properties of the original vertices and the edge properties of the original edges, the computer generates and displays a simplified graph that contains simplified vertices that each represents multiple original vertices, and simplified edges that each represents multiple original edges. Responsive to an interactive selection of a particular simplified vertex or edge of the simplified graph, the computer displays a statistical summary based on the multiple original vertices represented by the particular simplified vertex, or the multiple original edges represented by the particular simplified edge.

[0049] Embodiments may load raw data from different data sources. The computer may show different kinds of vertices and edges that may load from the different data sources. Manually or automatically, a schema may be discovered, captured, or inferred that relates the different kinds of vertices and edges that will populate the original graph.

[0050] During schema capture or during display of the simplified graph, the computer may fill the statistical summary with statistics of properties of a selected vertex or edge or kind of vertex or edge. Provided herein are various statistical widgets that meaningfully present the property statistics.

[0051] A user may interactively explore a simplified graph by expanding a simplified vertex or edge to be replaced by individual vertices or edges, or by collapsing a selected group of vertices or edges into one consolidated vertex or edge. Thus, the user may zoom in or out of interesting parts of the simplified graph.

2.0 Example Computer

[0052] FIG. **1** is a block diagram that depicts an example computer **100**, in an embodiment. Computer **100** displays a statistical summary of a selected vertex or edge of a simplified graph that is based on an original graph.

[0053] Computer **100** may be a rack server such as a blade, a personal computer, a mainframe, a network appliance, a virtual machine, a smartphone, or other computing device. In embodiments, computer **100** accesses original graph **110** that is stored in memory, on disk, or over a network.

2.1 Original Graph

[0054] Original graph **110** is a logical graph that contains original vertices, such as **121-123**, that are interconnected by original edges, such as **131-132**. For example, original edge **131** connects original vertices **121-122**.

[0055] Original graph **110** (and original edges **131-132**) may be directed or undirected. Original graph **110** may be connected or may contain some disconnected original vertices (that lack original edges). Although not shown, an original edge may originate from and terminate at a same original vertex.

[0056] Original graph **110** may be too big to clearly display by computer **100**. For example, original graph **110** may contain billions of vertices and trillions of edges.

[0057] For example, original graph **110** may encode an artificial neural network. However in most cases, the subject matter of original graph **110** involves details of real-world (and perhaps familiar) objects. For example, original graph **110** may encode a social network.

[0058] For example, each of original vertices **121-123** may represent a person. Likewise, each of original edges **131-132** may represent a relationship, such as a friendship, between two people.

2.2 Simplified Graph

[0059] Original graph **110** may be too big to directly display in a meaningful way. Thus in operation, computer **100** suppresses or combines some of the logical details of original graph **110** to generate simplified graph **160**.

[0060] Thus, display of simplified graph **160** may be feasible and meaningful even though display of original graph **110** might not be feasible or meaningful. For example, a display device of computer **100** may have limited surface area or pixel resolution. For example, the display device may be a smartphone display.

[0061] Display of simplified graph **160** may occur on computer **100**, such as a laptop. Computer **100** may render simplified graph **160** to remote clients, such as a web browser or a dedicated and installed application such as a smartphone app.

[0062] In embodiments, computer **100** includes a compute cloud. In embodiments and by policy or engineering, original graph **110** never leaves the compute cloud, and the compute cloud sends simplified graph **160** to (possibly public, anonymous, or untrusted) remote clients.

[0063] When original graph **110** is big enough to need visual reduction (e.g. de-magnification or other shrinking) to entirely fit within the display device screen, but such de-magnifying may result in a graph rendition having details too small to be seen by a human eye. Likewise, densely packing visual details without shrinking them may result in details that compete for screen area by visually overlapping or otherwise occluding each other.

[0064] As such, computer **100** generates simplified graph **160** that has fewer details (e.g. fewer vertices and edges) than original graph **110** has. Thus, simplified graph **160** may appear lossy (missing details) compared to original graph **110**.

[0065] For example, a simplified vertex or edge may represent multiple original vertices or edges. For example, simplified edge **180** may represent original edges **131-132**. Likewise, simplified vertex **172** may represent original vertices **121** and **123**.

[0066] However, not all simplified vertices or edges need represent multiple original vertices or edges. For example, an original vertex or edge may be represented more or less identically by a simplified vertex or edge. For example, simplified vertex **171** may represent only one original vertex, such as **122**.

[0067] The simplifications performed by computer **100** to generate simplified graph **160** may be based on the semantics of original graph **110**. For example, original vertex **122** may be a teacher, and original vertices **121** and **123** may be students in an academic class of the teacher.

[0068] Thus, student original vertices **121** and **123** may be combined into student simplified vertex **172**. Whereas, teacher original vertex **122** should not be combined into student simplified vertex **172** because the teacher is not a student.

2.3 Properties

[0069] The simplifications performed by computer **100** may be encoded as rules or criteria that computer **100** may apply to original graph **110** to generate simplified graph **160**. Such criteria may refer to properties of original vertices or edges.

[0070] For example, original vertex **122** contains vertex properties **141-142**. Likewise, original edge **131** contains edge properties **151-152**.

[0071] Properties, such as **141-142** and **151-152**, may be encoded as name-value pairs. For example, vertex property **141** may be named "role" and may have a value of "teacher". Whereas, original vertex **121** may have a vertex property (not shown) with the same name but a different value, such as "student".

[0072] For example, original graph **110** may be a property graph, in which each vertex or edge is more or less merely a collection of name-value pairs. Indeed, original graph **110** may be created exclusively from sets of properties or name-value pairs.

[0073] However, original vertices or edges do not have to have similar properties. For example, student original vertex **121** may have a vertex property (not shown) named "assigned seat". Whereas, teacher original vertex **122** need not have such a named vertex property.

[0074] Thus, original vertices **121-123** may have some, but not all, property names in common. Thus, original vertices **121-123** may each have a different amount of vertex properties. Likewise, edge properties may have common names that are different from those of vertex properties.

[0075] The original vertices and edges of original graph **110** may be originally provided by different data sources. For example, teachers may be loaded from a staff database, and students may be loaded from a customer database to compose original graph **110**. Thus, the vertex property names and/or values of original vertices **121-123** may depend on which original vertex originates from which database.

2.4 Simplification

[0076] The property-based rules or criteria that computer **100** uses to perform simplification may be automatically or manually specified and may be manually adjusted. For example, computer **100** may automatically recognize that original vertices loaded from the teacher database have one value (teacher) for the role property.

[0077] Whereas, original vertices loaded from the customer database have another value (student) for the role property. Thus, computer **100** may automatically synthesize a rule that student original vertices are combined into one simplified vertex and another rule that teacher original vertices are combined into another simplified vertex.

[0078] Thus, simplified vertex **171** may represent all teachers, and simplified vertex **172** may represent all students. Likewise, a user may manually enter rules by which original vertices or edges or combined.

[0079] Simplified graph **160** may assist multidimensional analysis and visualization. For an example based on vertex properties, computer **100** may separate (combine into separate simplified vertices) teachers from students in simplified graph **160**.

[0080] Computer **100** may further separate (not shown) students based on academic performance. For example (not shown), a first simplified vertex may represent teachers; a second simplified vertex may represent passing students; and a third simplified vertex may represent failing students.

[0081] Thus, computer **100** may have a simplification rule such as "vertex.role=teacher", another such as "vertex.role=student and vertex.performance=pass", and yet another such as "vertex.role=student and vertex.performance=fail".

[0082] In embodiments, simplification rules are encoded using a formal grammar for human readable text. In embodiments, the formal grammar is based on an industry standard, such as structured query language (SQL), a graph language such as object graph navigation language (OGNL) or Sparql, a constraint language, an object language such as JavaScript or JavaScript object notation (JSON), or a selector language such as XML, path language (XPath) or cascading style sheet (CSS).

2.5 Decoration

[0083] As explained, simplification may be lossy. Thus, the richness of original graph **110** may be visually absent from simplified graph **160**.

[0084] As such, a user may sometimes desire more information than simplified graph **160** shows. To some extent, computer **100** may render simplified graph **160** with visual details that suggest aspects of original graph **110**.

[0085] For example because there are more students than teachers, student simplified vertex **172** may appear larger (or darker) than teacher simplified vertex **171**. Likewise, simplified edge **180** may have a width or brightness that suggests how many original edges are represented by simplified edge **180**.

[0086] Likewise, a simplified vertex or edge may bear a visual label that indicates a count of original vertices or edges that are represented by the simplified vertex or edge. Likewise, a color, label, shape, or other visual detail may distinguish teacher simplified vertex **171** from student simplified vertex **172**.

2.6 Rendition

[0087] In embodiments, simplified graph **160** is declared and/or rendered in an XML rendering dialect such as scalable vector graphics (SVG) or extensible hypertext markup language (XHTML) and instrumented with browser scripts such as JavaScript. For example, computer **100** may generate simplified graph **160** by emitting XHTML and dynamic HTML (DHTML).

[0088] In embodiments, simplified graph **160** is displayed in a zero administration client (ZAC) such as a web browser. In embodiments, simplified graph **160** is displayed using lightweight or native graphical widgets.

2.7 Statistical Summary

[0089] Meaningful decoration of simplified graph **160** may still not provide enough information to a user. Thus, computer **100** provides statistical summary **190** to reveal still more (not necessarily all) details of original graph **110** that are absent from simplified graph **160**.

[0090] In embodiments, statistical summary **190** is a graphical user interface element such as a window, a popup, a balloon, a tool tip, or a modal or modeless dialog. In declarative embodiments, statistical summary **190** may be a rendered document element such as a div, a frame, or an iframe in hypertext markup language (HTML), or other document element that is typical of two or three dimensional rendering dialects of extensible markup language (XML).

2.8 Interactive Selection

[0091] Statistical summary **190** textually and/or graphically displays statistics that computer **100** derives from the original vertices or edges that are represented by an interactively selected subset of simplified vertices or edges.

[0092] In embodiments, the selected subset should have exactly one simplified vertex or edge. In embodiments, interactive selection involves multi-selection that occurs by a gesture such as lasso selection of multiple simplified vertices or a mouse click or finger touch on each of multiple simplified edges. In embodiments, a button press after interactive selection causes computer **100** to display statistical summary **190** based on the selected simplified vertices or edges.

[0093] In embodiments, the selection of simplified vertices may change even though statistical summary **190** already shows statistics for a previous selection in simplified graph **160**. For example, a user may initially click on simplified vertex **172** and cause statistical summary **190** to appear filled with statistics of simplified vertex **172**, and then subsequently click on simplified vertex **171**. In embodiments, computer **100** dynamically repaints or otherwise visually updates statistical summary **190** to reflect such changes to the selected subset.

[0094] In embodiments, statistical summary **190** shows a population count of how many original graph vertices are represented by a selected simplified graph vertex. For example when simplified vertex **172** is selected, statistical summary **190** may report that two original vertices are represented by the selection.

[0095] In embodiments, statistical summary **190** may show vertex property names that are shared by all or most original vertices represented by the selection. For example, selecting student simplified vertex **172** may cause statistical summary **190** to list property names such as role and assigned seat. Whereas for teacher simplified vertex **171**, statistical summary **190** may list only role as a property name, because teachers do not have an assigned seat property.

2.9 Statistics

[0096] In embodiments, statistical summary **190** may show one or more statistics for some named properties. For example student original vertices **121** and **123** may have a letter grade property. The value of that vertex property may be A− for original vertex **121** and C+ for original vertex **123**. Statistical summary **190** may show that the average value of the letter grade property is B.

[0097] Statistics of a property may include values such as a population count, an arithmetic mean, an arithmetic sum, a statistical mode, a statistical variance, a standard deviation, a minimum, or a maximum.

[0098] A statistic of a property may have multiple values. For example, a multi-valued statistic may list distinct values, such as all manufacturers of cars represented by a simplified vertex.

[0099] A multi-valued statistic may list frequent values, such as five most popular values or all values occurring at least 1,000 times or occurring at least 20% of the time. A multi-valued statistic may report a range, without distinct values within the range, such as 1987-2010 for model year of represented cars.

[0100] A multi-valued statistic may show a histogram or a pie chart. Example statistics of single or multiple values include a histogram, a geographic map, a pie chart, a line chart, a minimum, a maximum, an average, a median, a list of frequent values, or a list of distinct values.

3.0 Display Process

[0101] FIG. **2** is a flow diagram that depicts an example process that displays a statistical summary of a selected vertex or edge of a simplified graph that is based on an original graph. FIG. **2** is discussed with reference to FIG. **1**.

[0102] Step **202** loads an original graph that contains original vertices interconnected by original edges. For example, computer **100** may load teacher data from a staff database, and student data from a customer database.

[0103] For example, computer **100** may submit SQL queries to the staff and customer databases to retrieve teachers and students as records or to individually retrieve properties of teachers and students as discrete name-value pairs. Computer **100** may use the loaded data to create original vertices and edges that populate original graph **110**.

[0104] When step **202** finishes, original graph **110** is fully loaded and ready for processing. Step **204** generates a simplified graph based on vertex properties of original vertices and edge properties of original edges of the original graph.

[0105] For example, simplified graph **160** is created and populated with simplified vertices and edges that each represents one or more original vertices or edges. For example, simplified vertex **172** is created to represent original vertices **121** and **123**.

[0106] Step **206** displays the simplified graph. For example, computer **100** may render simplified graph **160** as HTML that is received and displayed by a remote web browser. Graph drawing and layout algorithms based on principles such as force, layering, or rings may avoid overlap of vertices and minimize crossing of edges.

[0107] Step **208** responds to interactive selection of a particular simplified vertex or edge by displaying a statistical summary. For example, the user may click on simplified vertex **172** to cause statistical summary **190** to appear filled with statistics about the original vertices represented by simplified vertex **172** or their properties. For example, statistical summary **190** may show a count of the represented original vertices, a list of property names, and/or statistics about one or more properties of the original vertices.

4.0 Loading Raw Data

[0108] FIG. **3** is a block diagram that depicts example network system **305** that receives information from multiple data sources to synthesize an original graph. For example, different types of original vertices or edges may be loaded from different databases.

[0109] Network system **305** contains computer **300**, point of sales stream **391**, and customer relational database management system (RDBMS) **393**. Customer RDBMS **393** may be a hosted on a computer or a compute cloud.

[0110] Customer RDBMS **393** may have a customer table that records information about people who have made purchases at a given retail chain. Computer **300** may read and write customer records in the customer table.

[0111] Customer RDBMS **393**, computer **300**, and point of sales stream **391** may share a communication network such as a local area network (LAN) or the global Internet. Point of sales (POS) stream **391** contains records of itemized purchases.

[0112] POS stream **391** may be a stream such as a continuous feed from cash registers, a file such as an hourly batch of POS data, or a database of historic purchases. Examples of data sources include a file, a database, a relational table, or a resource description framework (RDF) triplestore.

[0113] Computer **300** may be an implementation of computer **100**. In operation, computer **300** processes each purchase record from POS stream **391** to create a purchase original edge, such as **331-333**, within original graph **310**.

[0114] Each purchase record contains the names of the customer, the cashier, and the product purchased. In embodiments, these names may be foreign keys into other data sources. For example, computer **300** may use a customer name from a purchase record to retrieve a customer record from customer RDBMS **393**.

[0115] Likewise, a cashier name from a purchase record may be used to retrieve a cashier record from staff spreadsheet file **392**. For example, each row of spreadsheet **392** may contain a cashier record.

[0116] In embodiments and based on foreign keys such as names, computer **300** creates purchase original edges that connect to a cashier original vertex, such as **321**, and a customer original vertex, such as **322-323**. Computer **300** may eagerly create a cashier original vertex for every cashier record in spreadsheet **392** and a customer original vertex for every customer record in customer RDBMS **393**.

[0117] Alternatively, computer **300** may lazily (just in time) create original vertices only when involved in a purchase (i.e. referenced by a purchase record). From original graph **310**, computer **300** may create simplified graph **360** based on simplification criteria.

5.0 Graph Schema

[0118] FIG. **4** is a block diagram that depicts example display **400** that summarizes raw data as a preface to creating an original graph. For example, FIG. **3** shows original graph **310** created from data sources **391-393**, whereas FIG. **4** shows a graphical user interface (GUI) to establish a schema with which to create original graph **310**.

[0119] For example, computer **100** may render display **400** to a computer screen. In operation, a user may select data sources **391-393** to supply information to populate original vertices and edges of original graph **310**.

[0120] For example, the user may enter environmental details into the GUI to declare data sources **391-393**. For example, the user may enter a file system path that identifies staff spreadsheet file **392**.

[0121] Likewise, the user may enter an open database connectivity (ODBC) uniform resource locator (URL) connection string that identifies customer RDBMS **393**. Likewise, the user may specify a socket port number that identifies POS stream **391**.

[0122] With each data source that the user enters, the user may also declare which kind of original vertex or edge type to load. For example, the user may declare that purchase original edges should be loaded from POS stream **391**. The GUI may use these entered environmental details to connect to data sources **391-393** and discover schema details that describe the format of the cashier records, the customer records, and the purchase records.

[0123] By automatic discovery, such as schema inspection or schema inference from sample records, or by manual entry, the GUI obtains enough metadata to show (FIG. **4**) that there are two kinds of vertices and one kind of edge to create an original graph with. The GUI also shows that each kind of vertex or edge has named properties of particular data types. For example, a cashier has a name, which is text.

5.1 Interactive Schema Capture

[0124] If a relational schema is available for a data source, the GUI may automatically discover primary keys, foreign keys, and relations. If such relational metadata is unavailable, the user can click and drag to draw a line between a foreign key of one kind of vertex to a primary key of another kind of vertex to interactively declare a type of edge between those kinds of vertices.

[0125] If a kind of edge is already available (e.g. the purchase original edge), then the user can use the same gestures to connect the cashier and customer vertices to the purchase edge. Such schematic connectivity is shown (FIG. **4**) as dotted diagonal lines that touch the cashier and customer fields of the purchase original edge.

[0126] The user may interactively select any kind of vertex or edge to cause statistical summary **490** to populate. As shown, each property name appears with a triangle beside it.

[0127] If the triangle points rightwards, that property is collapsed, and the statistics of the property are hidden. For example, the product property is collapsed.

[0128] If the triangle points downwards, that property is expanded, and statistics of the property are shown. For example, the quantity property is expanded.

[0129] The quantity statistic shows as a histogram that summarizes reoccurring actual values. For example, 400 (y-axis of histogram) of the purchased items were bought by the half dozen (quantity x-axis). Property statistics are discussed later herein.

5.2 Filtration

[0130] The raw data from data sources **391-393** may contain thousands or millions of uninteresting (noisy) records. Extraneous records can distract the user or slow analytics.

[0131] Thus, the user may click the Add Filter button to enter filtration criteria that eliminate noisy records. For

example, the user might only be interested in bulk purchases having a quantity of at least four.

[0132] For example as shown, the unit price property is filtered to include only expensive items. The computer may use such filtration criteria to skip noisy records and create fewer instances of an original vertex or edge.

[0133] In embodiments, the user can also use statistical summary **490** to separate records into groups. For example, the user could use the purchase records to create two kinds (not shown) of purchase original edges.

[0134] For example, the user could create an expensive purchase original edge with one unit price criteria. Likewise, the user could create an inexpensive purchase original edge with another unit price criteria.

[0135] Thus, display **400** entails the GUI operating in a design mode through which a schema for an original graph may be created. When graph schema creation is complete, the user may switch the GUI from design mode to live mode to view a simplified graph. For example in design mode, the GUI need not load all or any records and need not create an original graph nor a simplified graph.

6.0 Example Display

[0136] FIG. **5** is a screen snapshot that depicts example display **500** that shows a simplified graph suitable for interactive exploration. For example after graph schema metadata is entered, the GUI may load records from the data sources, create an original graph, synthesize a simplified graph, and then display the simplified graph.

[0137] On the left side of display **500** is a simplified graph with vertices drawn as circles and edges shown as arcs. The black arrow may be a mouse cursor or a finger touch.

[0138] For example, the mouse may hover over the employee simplified vertex to cause a tool tip to appear to reveal that there are 8,474 employee original vertices represented by the employee simplified vertex. Visual magnification (rendering size) may indicate relative populations of simplified vertices and edges.

[0139] For example, there are more employees than customers because the employee simplified vertex has a larger diameter than the customer simplified vertex. Likewise, more products were sold than were worked on because the sold simplified edge is thicker than the works on simplified edge.

[0140] The simplified (and/or original) graph may can reflexive (self directed) edges. For example as shown, one employee can be a friend of another employee.

[0141] On the right side of display **500** is a statistical summary. For example as shown, clicking on the employee simplified vertex causes the statistical summary to show employee statistics.

[0142] As shown, employee age is filtered on a five year range. A vertical bar of the histogram is drawn black if it satisfies the age filter, and drawn white if it does not satisfy the filter.

[0143] For example, the user could interactively replace (reenter) the upper age bound of 35 with a new value of 100. In embodiments, such interactive filtration adjustment may automatically cause more histogram vertical bars to turn black, and the employee simplified vertex to expand its diameter. In turn, that may cause the sold simplified edge to get thicker to represent more employees selling more products.

7.0 Statistical Widgets

[0144] How a particular statistic is displayed within a statistical summary depends on the nature of the statistic, the nature of the property domain, and the distribution of the property values. FIG. **6** is a diagram that depicts example statistical widgets **600** that present statistics in meaningful ways.

[0145] Simple statistics having only one or two values may be displayed as text. For example, a mean or the extremes of a range (minimum and maximum) can be shown as a number or pair of numbers.

[0146] Other statistics may benefit from dedicated visualization, either because of the nature of the property domain or the amount of statistical values involved. For example, histogram **601** is a bar chart that may show cars sold by manufacture. For example, distinct values may be shown on the horizontal axis, and frequency counts may increase upwards along the vertical axis.

[0147] As shown, some vertical bars of histogram **601** are black to indicate satisfaction of a filter. Whereas, other vertical bars are white to indicate suppression by the filter. For example, foreign auto makers may be suppressed (filtered out).

[0148] Line chart **602** may show time series data, with time as the horizontal axis. Pie chart **603** may provide an alternate rendition of a histogram. Calendar **606** may instead be used to show temporal data.

[0149] Pie chart **603** emphasizes relative fractions of a whole, such as market share of a whole market. Whereas a bar chart emphasizes absolute quantities, such as counts.

[0150] List box **604** may enumerate actual values that occur for a property. As shown, most frequent values can be shown at the top with population counts.

[0151] Distinct values are shown below the frequent values. The list may scroll, such as with a scroll bar.

[0152] A statistic may be decomposed by geographic regions, such as map **605**. For example, a histogram whose bins are geographic regions may be rendered as map **605**.

[0153] An aspect that is common to many of widgets **601-606** is the grouping of multiple data points. For example, multiple data points are represented by each wedge of pie chart **603**, each bar of histogram **601**, and each geographic state of map **605**. Hovering, clicking, or other interaction with a visual detail, such as a pie wedge, may cause a tool tip that reveals details such as a count. Likewise, such interactions may facilitate entry of a filter.

8.0 Interactive Exploration

[0154] FIG. **7** is a screen snapshot that depicts example display **700** that shows a simplified graph having mixed simplification. On the left side of display **700** is a simplified graph. On the right side of display **700** is a statistical summary for a selected simplified vertex or edge.

8.1 Incomplete Simplification

[0155] The simplified graph is incompletely simplified, such that some vertices may be separately drawn even though they are of a same kind of simplified vertex. Here, employee simplified vertex **701** and the simplified vertices shown beneath it each represent one original vertex.

[0156] For example, the user may have specified employee simplified vertex **701** as a focus of the analysis or otherwise have indicated to the GUI that employee simpli-

fied vertex **701** and vertices directly connected to **701** should be expanded to have a separate simplified vertex for each original vertex.

[0157] Whereas, there are 48 employee friends that are not directly connected to employee **701** and are instead connected to employee **702**. So many friends may either exceed a threshold and/or be too distant by virtue of not being connected to employee **701**, and thus are consolidated into one friends simplified vertex **703** that represents **48** employee original vertices.

8.2 Drilling Down

[0158] The user may interact with friends simplified vertex **703** to cause the computer to replace vertex **703** with new employee simplified vertices (not shown) that each represent one employee original vertex. For example, the user may double click on friends simplified vertex **703** to cause it to expand into 48 separate employee vertices.

8.3 Rolling Up

[0159] In embodiments, the user may select multiple employee simplified nodes and instruct the computer to combine them into one employee simplified vertex. For example, the user may lasso select employee vertices **701-702** by mouse dragging that temporarily renders an elastic (rubber band) box that encompasses employee vertices **701-702** and then press a toolbar button (not shown) to collapse/combine employee vertices **701-702** into a new employee simplified vertex that replaces both employee vertices **701-702**.

[0160] Likewise, the user may simultaneously select vertices of different kinds, and the computer may combine those of one kind into one new simplified vertex and those of another kind into a different new simplified vertex. Thus by expanding or collapsing vertices, the user can interactively explore (drill down into, zoom into, zoom out of) interesting portions of the original graph that are oversimplified in the simplified graph.

9.0 Hardware Overview

[0161] According to one embodiment, the techniques described herein are implemented by one or more special-purpose computing devices. The special-purpose computing devices may be hard-wired to perform the techniques, or may include digital electronic devices such as one or more application-specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs) that are persistently programmed to perform the techniques, or may include one or more general purpose hardware processors programmed to perform the techniques pursuant to program instructions in firmware, memory, other storage, or a combination. Such special-purpose computing devices may also combine custom hard-wired logic, ASICs, or FPGAs with custom programming to accomplish the techniques. The special-purpose computing devices may be desktop computer systems, portable computer systems, handheld devices, networking devices or any other device that incorporates hard-wired and/or program logic to implement the techniques.

[0162] For example, FIG. **8** is a block diagram that illustrates a computer system **800** upon which an embodiment of the invention may be implemented. Computer system **800** includes a bus **802** or other communication mechanism for communicating information, and a hardware processor **804** coupled with bus **802** for processing information. Hardware processor **804** may be, for example, a general purpose microprocessor.

[0163] Computer system **800** also includes a main memory **806**, such as a random access memory (RAM) or other dynamic storage device, coupled to bus **802** for storing information and instructions to be executed by processor **804**. Main memory **806** also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor **804**. Such instructions, when stored in non-transitory storage media accessible to processor **804**, render computer system **800** into a special-purpose machine that is customized to perform the operations specified in the instructions.

[0164] Computer system **800** further includes a read only memory (ROM) **808** or other static storage device coupled to bus **802** for storing static information and instructions for processor **804**. A storage device **86**, such as a magnetic disk or optical disk, is provided and coupled to bus **802** for storing information and instructions.

[0165] Computer system **800** may be coupled via bus **802** to a display **812**, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device **814**, including alphanumeric and other keys, is coupled to bus **802** for communicating information and command selections to processor **804**. Another type of user input device is cursor control **816**, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor **804** and for controlling cursor movement on display **812**. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

[0166] Computer system **800** may implement the techniques described herein using customized hard-wired logic, one or more ASICs or FPGAs, firmware and/or program logic which in combination with the computer system causes or programs computer system **800** to be a special-purpose machine. According to one embodiment, the techniques herein are performed by computer system **800** in response to processor **804** executing one or more sequences of one or more instructions contained in main memory **806**. Such instructions may be read into main memory **806** from another storage medium, such as storage device **86**. Execution of the sequences of instructions contained in main memory **806** causes processor **804** to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions.

[0167] The term "storage media" as used herein refers to any non-transitory media that store data and/or instructions that cause a machine to operation in a specific fashion. Such storage media may comprise non-volatile media and/or volatile media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device **86**. Volatile media includes dynamic memory, such as main memory **806**. Common forms of storage media include, for example, a floppy disk, a flexible disk, hard disk, solid state drive, magnetic tape, or any other magnetic data storage medium, a CD-ROM, any other optical data storage medium, any physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, NVRAM, any other memory chip or cartridge.

[0168] Storage media is distinct from but may be used in conjunction with transmission media. Transmission media participates in transferring information between storage media. For example, transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus **802**. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

[0169] Various forms of media may be involved in carrying one or more sequences of one or more instructions to processor **804** for execution. For example, the instructions may initially be carried on a magnetic disk or solid state drive of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system **800** can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus **802**. Bus **802** carries the data to main memory **806**, from which processor **804** retrieves and executes the instructions. The instructions received by main memory **806** may optionally be stored on storage device **86** either before or after execution by processor **804**.

[0170] Computer system **800** also includes a communication interface **818** coupled to bus **802**. Communication interface **818** provides a two-way data communication coupling to a network link **820** that is connected to a local network **822**. For example, communication interface **818** may be an integrated services digital network (ISDN) card, cable modem, satellite modem, or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface **818** may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface **818** sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

[0171] Network link **820** typically provides data communication through one or more networks to other data devices. For example, network link **820** may provide a connection through local network **822** to a host computer **824** or to data equipment operated by an Internet Service Provider (ISP) **826**. ISP **826** in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" **828**. Local network **822** and Internet **828** both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link **820** and through communication interface **818**, which carry the digital data to and from computer system **800**, are example forms of transmission media.

[0172] Computer system **800** can send messages and receive data, including program code, through the network(s), network link **820** and communication interface **818**. In the Internet example, a server **830** might transmit a requested code for an application program through Internet **828**, ISP **826**, local network **822** and communication interface **818**.

[0173] The received code may be executed by processor **804** as it is received, and/or stored in storage device **86**, or other non-volatile storage for later execution.

[0174] In the foregoing specification, embodiments of the invention have been described with reference to numerous specific details that may vary from implementation to implementation. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense. The sole and exclusive indicator of the scope of the invention, and what is intended by the applicants to be the scope of the invention, is the literal and equivalent scope of the set of claims that issue from this application, in the specific form in which such claims issue, including any subsequent correction.

What is claimed is:

1. A method comprising:

loading an original graph that contains original vertices interconnected by original edges, wherein:

each original vertex of the original vertices contains one or more vertex properties, and

each original edge of the original edges contains one or more edge properties;

generating, based on the vertex properties of the original vertices and the edge properties of the original edges, a simplified graph that contains:

at least one simplified vertex that represents multiple original vertices, and

at least one simplified edge that represents multiple original edges;

displaying the simplified graph;

displaying, responsive to an interactive selection of a particular simplified vertex of the simplified graph or a particular simplified edge of the simplified graph, a first statistical summary based on one of:

the multiple original vertices represented by the particular simplified vertex, or

the multiple original edges represented by the particular simplified edge;

wherein the method is performed by one or more computers.

2. The method of claim **1** wherein:

loading the original graph comprises:

receiving first data from first data source, and

receiving second data from second data source;

the at least one simplified vertex comprises:

a first simplified vertex that represents the first data, and

a second simplified vertex that represents the second data.

3. The method of claim **2** wherein generating the simplified graph comprises:

displaying a first representation of the first data that shows attribute names and attribute types that are common to data items within the first data;

displaying a second representation of the second data that shows attribute names and attribute types that are common to data items within the second data;

receiving at least one of:

a first selection in the first representation of an attribute name that cross-references the second data, or

a second selection of the first representation and the second representation as related.

4. The method of claim **3** wherein generating the simplified graph further comprises displaying, responsive to receiving a third selection of the first representation, a second statistical summary of the first data.

5. The method of claim **4** wherein displaying the second statistical summary comprises displaying at least one of: a

histogram, a geographic map, a pie chart, a line chart, a minimum, a maximum, an average, a median, a list of frequent values, a list of distinct values.

6. The method of claim **2** wherein the first data source is a file, a database, a relational table, or a resource description framework (RDF) triplestore, a simple API for XML (SAX) parser, or a stream parser.

7. The method of claim **2** wherein:

the first data source is a first database, and

the second data source is a second database.

8. The method of claim **1** wherein displaying the first statistical summary comprises displaying at least one of: a histogram, a geographic map, a pie chart, a line chart, a minimum, a maximum, an average, a median, a list of frequent values, a list of distinct values.

9. The method of claim **1** wherein displaying the simplified graph comprises visually magnifying at least one of:

a simplified vertex of the at least one simplified vertex based on a count of the multiple original vertices represented by the simplified vertex, or

a simplified edge of the at least one simplified edges based on a count of the multiple original edges represented by the simplified edge.

10. The method of claim **1** further comprising receiving a request to perform at least one of:

split a particular simplified vertex into multiple simplified vertices based on at least one attribute name and at least one predicate, or

split a particular simplified edge into multiple simplified edges based on at least one attribute name and at least one predicate.

11. The method of claim **1** further comprising receiving a request to perform at least one of:

merge a plurality of simple vertices into a new simple vertex, or

merge a plurality of simple edges into a new simple edge.

12. One or more non-transient computer-readable media storing instructions that, when executed by one or more processors, cause:

loading an original graph that contains original vertices interconnected by original edges, wherein:

each original vertex of the original vertices contains one or more vertex properties, and

each original edge of the original edges contains one or more edge properties;

generating, based on the vertex properties of the original vertices and the edge properties of the original edges, a simplified graph that contains:

at least one simplified vertex that represents multiple original vertices, and

at least one simplified edge that represents multiple original edges;

displaying the simplified graph;

displaying, responsive to an interactive selection of a particular simplified vertex of the simplified graph or a particular simplified edge of the simplified graph, a first statistical summary based on one of:

the multiple original vertices represented by the particular simplified vertex, or

the multiple original edges represented by the particular simplified edge.

13. The one or more non-transient computer-readable media of claim **12** wherein:

loading the original graph comprises:

receiving first data from first data source, and

receiving second data from second data source;

the at least one simplified vertex comprises:

a first simplified vertex that represents the first data, and

a second simplified vertex that represents the second data.

14. The one or more non-transient computer-readable media of claim **13** wherein generating the simplified graph comprises:

displaying a first representation of the first data that shows attribute names and attribute types that are common to data items within the first data;

displaying a second representation of the second data that shows attribute names and attribute types that are common to data items within the second data;

receiving at least one of:

a first selection in the first representation of an attribute name that cross-references the second data, or

a second selection of the first representation and the second representation as related.

15. The one or more non-transient computer-readable media of claim **14** wherein generating the simplified graph further comprises displaying, responsive to receiving a third selection of the first representation, a second statistical summary of the first data.

16. The one or more non-transient computer-readable media of claim **15** wherein displaying the second statistical summary comprises displaying at least one of: a histogram, a geographic map, a pie chart, a line chart, a minimum, a maximum, an average, a median, a list of frequent values, a list of distinct values.

17. The one or more non-transient computer-readable media of claim **12** wherein displaying the first statistical summary comprises displaying at least one of: a histogram, a geographic map, a pie chart, a line chart, a minimum, a maximum, an average, a median, a list of frequent values, a list of distinct values.

18. The one or more non-transient computer-readable media of claim **12** wherein displaying the simplified graph comprises visually magnifying at least one of:

a simplified vertex of the at least one simplified vertex based on a count of the multiple original vertices represented by the simplified vertex, or

a simplified edge of the at least one simplified edges based on a count of the multiple original edges represented by the simplified edge.

19. The one or more non-transient computer-readable media of claim **12** wherein the instructions, when executed by the one or more processors, further cause receiving a request to perform at least one of:

split a particular simplified vertex into multiple simplified vertices based on at least one attribute name and at least one predicate, or

split a particular simplified edge into multiple simplified edges based on at least one attribute name and at least one predicate.

20. The one or more non-transient computer-readable media of claim **12** wherein the instructions, when executed by the one or more processors, further cause receiving a request to perform at least one of:

merge a plurality of simple vertices into a new simple vertex, or

merge a plurality of simple edges into a new simple edge.

* * * * *