



- (51) **International Patent Classification:**
G06F 17/30 (2006.01) *G06F 17/00* (2006.01)
- (21) **International Application Number:**
PCT/US2015/024641
- (22) **International Filing Date:**
7 April 2015 (07.04.2015)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (71) **Applicant:** HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).
- (72) **Inventors:** VAQUERO GONZALEZ, Luis Miguel; Long Down Avenue, Stoke Gifford, Bristol South Gloucestershire BS34 8QZ (GB). SAE LOR, Suksant; Long Down Avenue, Stoke Gifford, Bristol South Gloucestershire BS34 8QZ (GB). HAWKES, Rycharde; Long Down Avenue, Stoke Gifford, Bristol South Gloucestershire BS34 8QZ (GB).
- (74) **Agents:** FOUGERE, Jeffrey R. et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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).

Declarations under Rule 4.17:

— as to the identity of the inventor (Rule 4.17(i))

Published:

— with international search report (Art. 21(3))

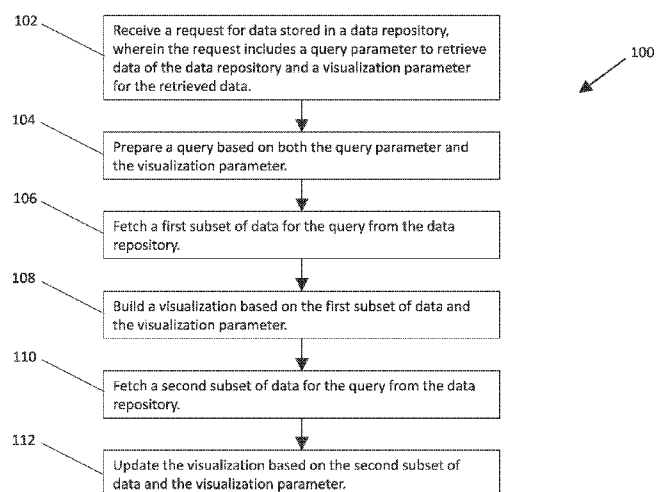
(54) **Title:** BUILDING A DATA QUERY VISUALIZATION

FIG. 1

(57) **Abstract:** In some examples, a method includes a step of receiving a request for data stored in a data repository. The request can, for example, include a query parameter to retrieve data of the data repository and a visualization parameter for the retrieved data. The method can further include a step of preparing a query based on both the query parameter and the visualization parameter. The method can further include a step of fetching a first subset of data for the query from the data repository. The method can further include a step of building a visualization based on the first subset of data and the visualization parameter. The method can further include a step of fetching a second subset of data for the query from the data repository. The method can further include a step of updating the visualization based on the second subset of data and the visualization parameter.

BUILDING A DATA QUERY VISUALIZATION

BACKGROUND

[0001] Collections of data are commonly organized in computer databases. Such databases can interface with one or more database management systems to allow analysis and retrieval of the data. For example, certain database management systems can be in the form of computer software applications that interact with a user, other applications, and the database itself to allow retrieval of data and other administrative functions of the database. Data retrieved from such databases can be presented in many forms. For example, some database management systems are designed to present retrieved data in the form of tables. Other database management systems can allow retrieved data to be displayed in the form of multi-dimensional graphs or other advanced visualizations.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] For a detailed description of various examples, reference will now be made to the accompanying drawings in which:

[0003] FIG. 1 is a flowchart for a method, according to an example.

[0004] FIG. 2 is a flowchart for a method, according to another example.

[0005] FIG. 3 is a data visualization, according to an example.

[0006] FIG. 4 is a data visualization, according to an example.

[0007] FIG. 5 is a diagram of a system, according to an example.

[0008] FIG. 6 is a diagram of a machine-readable storage medium, according to an example.

DETAILED DESCRIPTION

[0009] The following discussion is directed to various examples of the disclosure. Although one or more of these examples may be preferred, the examples disclosed herein should not be interpreted, or otherwise used, as limiting the scope of the disclosure, including the claims. In addition, the following description has broad application, and the discussion of any example is meant only to be descriptive of that example, and not intended to intimate that the scope of the disclosure, including the claims, is limited to that example. Throughout the present disclosure, the terms "a" and "an" are intended to denote at

least one of a particular element. In addition, as used herein, the term "includes" means includes but not limited to, the term "including" means including but not limited to. The term "based on" means based at least in part on.

[0010] As provided above, certain database management systems can allow retrieved data to be displayed in the form of multi-dimensional graphs or other advanced visualizations. In such situations, a data analyst or other individual may be tasked to manually choose from visualizations based on properties of the results, such as size or quality, or other factors. For example, for a database query that renders 10 results that are of poor quality (e.g., the results contain many null values), a data analyst may choose to represent the results as a single pixel on a graph because the results contain relatively little information. Later, that same query may offer millions of results of reasonable quality and the data analyst may choose to now represent the results as large icons each summarizing collections of many results. It is appreciated that such manual preparation of visualizations can be a time-consuming and inconsistent process.

[0011] Certain implementations of the present disclosure are directed to improved systems, methods, mediums, and the like that allow for preparation of data visualizations that are designed to address the above issues as well as other issues. For example, certain implementations of the present disclosure can be used to automate the data visualization process and link the way data is visualized as part of a query specification. For example, one implementation of the present disclosure is in the form of a method that includes: (a) receiving a request for data stored in a data repository, wherein the request includes a query parameter to retrieve data of the data repository and a visualization parameter for the retrieved data, (b) preparing a query based on both the query parameter and the visualization parameter, (c) fetching a first subset of data for the query from the data repository, (d) building a visualization based on the first subset of data and the visualization parameter, (e) fetching a second subset of data for the query from the data repository, and (f) updating the visualization based on the second subset of data and the visualization parameter.

[0012] Certain implementations of the present disclosure can be used to allow access to different records in a data repository and perform, at query time, one or more processing actions, such as for example data cleaning, filtering, transformation, or other

processing for use in building or updating a visualization. A query engine incorporating such functionality can thereby perform additional processing aimed at preparing data for visualization at query time, which can save time by allowing an individual to avoid running a separate visualization script after obtaining the query result. The above and other advantages of implementations presented herein will be apparent upon review of the description and figures.

[0013] FIG. 1 is a flowchart for an example method 100 that can provide automatic preparation of data visualizations. It is appreciated that method 100 as well as the methods described herein can, for example, be implemented in the form of machine readable instructions stored on memory of a computing system (see, e.g., the implementation of FIG. 5), executable instructions stored on a non-transitory machine readable storage medium (see, e.g., the implementation of FIG. 6), in the form of electronic circuitry, or another suitable form.

[0014] Method 100 includes a step 102 of receiving a request for data stored in a data repository. The data repository can, for example, be located on a computer storage medium in a database server (or other computing device). With reference to a database server implementation, the server can, for example, be in the form of a dedicated computer that stores the actual database and runs limited software such as a database management system and related software. Such a database server can, for example, be a multiprocessor computer, with RAID disk arrays used for stable storage.

[0015] Suitable data repositories for use with method 100 can, for example, be in the form of suitable relational databases, hierarchical databases, network databases, object databases, or any other suitable form of database that allows interaction with a database management system. The database management system can, for example, interact with the data repository to allow for various functionality, such as: (1) data definition (e.g., the creation, modification and removal of definitions that define the organization of the data), (2) data updating (e.g., the insertion, modification, and deletion of data), (3) data retrieval (e.g., the providing of information in a form directly usable or for further processing by other applications, and (4) administration (e.g., the registration and monitoring of users, enforcement of data security, monitoring of performance, maintenance of data integrity, dealing with concurrency control, and

information recovery). It is appreciated that the database management system can provide other suitable functionality.

[0016] In some implementations, the request of step 102 can include a query parameter to retrieve data of the data repository and a visualization parameter for the retrieved data.

[0017] The query parameter can, for example, be a programming language, such as for example, a Structured Query Language (SQL) command. It is appreciated that SQL commands, functionality, and syntax are provided herein as an example programming language that can interface with a data repository through a database management system, however, other suitable programming languages and syntax can be used.

[0018] In some implementations where SQL commands are used, the query parameter can be in the form of a SELECT statement (e.g., "SELECT * FROM T") to return a result set of records from one or more tables in a database. It is appreciated that more advanced query parameters can be included in the request, such as for example query parameters that: (1) specify certain rows of the database to retrieve, (2) group rows sharing a property so that an aggregate function can be applied to each group, (3) select among pre-defined groups, (4) specify an order in which to return the rows, (5) provide an alias which can be used to temporarily rename tables or columns. In some implementations, the query parameter is not in the form of query language syntax for a specific programming language and method 100 includes a step of converting the query parameter into suitable query language syntax.

[0019] The visualization parameter can, for example, provide rules for graphing fetched data based on values of the fetched data. In some implementations, the visualization parameter is in the form of a function that considers one or more metadata attributes of fetched data and provides specific instructions for building visualizations based on the attributes. For example, in some implementations, the visualization parameter can indicate that fetched data should be represented using a first type of graph if the number of fetched results is above a certain threshold and should be represented using a second type of graph if the number of fetched results is below or at the threshold. Further examples of visualization parameter functions and structure are described below.

[0020] Method 100 includes a step 104 of preparing a query based on both the query parameter and the visualization parameter. The prepared query can, for example, be in the form of an extended or standard SQL query employing SQL syntax or in the form of another type of query language. In some implementations, the step 104 of preparing a query based on both the query parameter and the visualization parameter includes modifying a query parameter based on rules of the visualization parameter. For example, in certain implementations, the visualization parameter may include a set of instructions that includes a query filter for filtering results of the query parameter. For example, the visualization parameter can include rules that only the first 500 results should be displayed on the visualization (whereas the query parameter might indicate that the first 1000 results should be retrieved). As another example, the visualization parameter can include rules that only results matching certain criteria (e.g., only voters having a certain area code) should be retrieved (whereas the query parameter does not include such a matching criteria). In such implementations, preparing a query based on both the query parameter and the visualization parameter includes applying the query filter to the query parameter. This can, for example, provide for a reduced search space which may lead to faster query results that are more fit for the specific visualization. It is appreciated that alternative and/or more advanced query filtering functionality can be provided by the visualization parameter.

[0021] In some implementations, preparing a query based on both the query parameter and the visualization parameter can include modifying the query parameter to only retrieve data to be used in the visualization. For example, a visualization parameter may indicate that only 3 fields of each result are to be used in building the visualization, whereas the query parameter may indicate that 5 fields of each result. As a result, the prepared query can modify the query parameter to only fetch the 3 fields of each results to be used in building the visualization.

[0022] Pseudocode for an example query parameter and visualization parameter is provided below:

PLOT STUDENTS BY TYPE (SQL DB EXAMPLE)

```

SELECT * FROM PEOPLE RENDERED AS RANKED_GROUPS_VP;

RANKED_GROUP_VP:

    groups = filter(age < 18).LOOM.rank(LOOM.similarTo(STUDENT_TYPE))

    if( groups.size == 1)

        R.plot(group.getFirst.x, group.getFirst.y) // this could be a call to an
        external language that supports the function ("R" in this case) or a nested
        call to a visualization script else

        R.barchar(groups,type)

```

[0023] In this example, "RANKED_GROUP_VP" in the above pseudocode is a visualization parameter in the form of a set of instructions that inspect the result set and prepare the results for visualization. The "rank" operation in the above pseudocode is a process of bucketing results. Rather than letting the visualization script run their instructions, a query module would know that the SQL query needs to be modified to make this bucketisation in place. This way, the buckets are created as the query module passes through the data, preventing further processing by an external visualization module. The "LOOM.similarTo" operation in the above pseudocode is an example operation that may be calling an external machine learning algorithm to rate similarity based on a prior classification of people in different types. The "filter" operation in the above pseudocode can be considered to be a result set analysis lambda (although it is evaluated as the query module loops through the data, not afterwards). The "plot" operation in the above pseudocode is a result set analysis lambda whereas the "rank" operation is an accumulating lambda.

[0024] Method 100 includes a step 106 of fetching a first subset of data for the query from the data repository. The first subset of data for the query is less than the entire result set so as to allow "on the fly" (i.e., "streaming") building of a visualization. For example, certain implementations of the present disclosure can be used to aggregate data for visualization as it goes (e.g. a moving average). This can, in some situations, save processing time by not relying on a separate loop across data once the query results are completely fetched. Moreover, and as described in further detail herein, such dynamic

building of data visualizations can be used to process query results (e.g., cleaning nulls, trimming streams, fixing typos by checking string similarity against a dictionary, etc.) as results are fetched and can further modify query terms based on already fetched results.

[0025] In some implementations, the first subset of data is a single result for the query from the data repository, whereas in other implementations, the first subset of data can be several results for the query. The first subset of data can be defined as a first number of results of a query. For example, the first subset of data can be defined as the first 10, 100, 1000, or another suitable number of results. In some implementations, the first subset of data can be determined based on a time period of the fetching step. For example, the first subset of data can be defined as all the data retrieved in 0.001 seconds, 1 seconds, 10 seconds, or another suitable time period. It is appreciated that the first subset of data can be determined based on multiple factors (e.g., either the first 1000 results or 1 second of elapsed time) and may be determined based on additional and/or alternative factors (e.g., the number of results that meet certain criteria defined by the query parameter or visualization parameter).

[0026] Method 100 includes a step 108 of building a visualization based on the first subset of data and the visualization parameter. In some implementations, the visualization can be in the form of a graph. For example, the first subset of data can be used to create a visualization in the form of a pie graph, with certain elements of such a pie graph (e.g., color, three-dimensional (3D) offset, legend location, etc.) defined by the visualization parameter. It is appreciated that other two-dimensional (2D) or 3D graphs (e.g., column chart, bar graph, radial chart, etc.) or other forms of visualizations can be used. For example, one form of visualization that can be prepared by method 100 is illustrated as visualization 111 in FIG. 2. For context, this graph is in the form of a radial graph that identifies CPU utilization of instances of virtual machines in a virtual network. Although this visualization illustrates one example of the type of data that can be visualized using the present disclosure, this visualization is merely provided as an example and other more suitable techniques, templates, bindings may be applicable.

[0027] In some implementations, the visualization can be in the form of a multi-dimensional mathematical function. For example, in some implementations a two-dimensional

canvas with Cartesian (x, y) coordinates can be provided and users can specify their visualization scripts as mathematical two-dimensional functions where the specific function depends on the extracted properties of the result set. For example, if the user specifies the function " $\exp(-((x-4)^2+(y-4)^2)/1000)$ " and provides a visualization parameter identifying the ranges as " $x=[-10,10]$, $y=[-10,10]$ and $z=[0,0.75]$ ", then the resulting visualization can, for example, be illustrated as shown by visualization 113 of FIG. 3. It is appreciated that in such an implementation, the visualization parameter can provide other visualization rules for producing such a visualization, such as color, 3D offset, size, etc.

[0028] In some implementations, building a visualization based on the first subset of data and the visualization parameter includes comparing one or more metadata attribute values for the first subset of data to one or more reference attribute values. For example, in some implementations, the metadata attribute value is a value corresponding to a number of null results in the first subset of data and the reference attribute value can be a value previously defined by a user, generated by a computer, or provided by another source for comparison purposes. In such an implementation, if the number of null results in the first subset of data is greater than a reference attribute value, then a first type of visualization and/or visualization rule is applied, whereas if the number of null results in the first subset of data is less than or equal to a reference attribute value, then a second type of visualization and/or visualization is applied. As another example, the visualization parameter can be defined such that a pie chart is created for retrieved data if the results identify between 3 and 10 categories and with each category constituting at least 3% of the total results. In this example, the visualization parameter can be defined such that a bar graph is created for retrieved data if the results identify less than 3 or greater than 10 categories or if any category constitutes less than 3% of the total results. It is appreciated that the above examples are for illustration only and that any number of modifications to the visualization parameter can be envisioned.

[0029] It is appreciated building a visualization based on the first subset of data and the visualization parameter include any suitable data operation and does not need to rely on simple reference value comparisons. For example, such an operation can include running a Fast-Fourier Transform (FFT) to compare noise level of query results on a

database of music files and decide to represent that in frequency (with no further transformation) or in time (inverse FFT) after applying some noise cancelation filter that entirely depends on the results of the FFT itself. In such an implementation and in other implementations, data can be processed during query time and can be aggregate on the fly. As an example, if a user is to perform a FFT during query time, some FFT samples (e.g., previous entries already inspected and processed) can be accumulated in order to give the FFT value for the currently processed database entry.

[0030] In the examples above, the visualization is generally data centric and designed to aggregate and display large amounts of data. In some implementations, method 100 can be used to prepare visualizations in the form of diagrams or other non data-centric visualization. For example, method 100 can be used to build a visualization in the form of a 3D arrangement of hardware in a data center. The visualization parameter can include instructions that define geographic constraints on various items of hardware (e.g., a number of servers in a given rack, a number of racks in a room based on thermal constraints, power constraints, height constraints, or other constraints). In such an implementations, method 100 can be employed to prepare a diagram for a data center based on retrieved query results for available data center hardware.

[0031] In some implementations, method includes a step of further modifying the modified query as results are fetched from the data repository. For example, referring to the above pie chart/bar graph implementations, the visualization parameter can indicate that if fetched results indicate that more than 10 categories have already been fetched and that a bar graph visualization will be used, the query may be modified to be optimized based on bar graph visualizations. For example, if the initial query retrieved 5 fields for each result of the data query, but a bar graph only relied on 3 fields for each result, then subsequent queries to satisfy the query may be modified to only return the 3 relevant fields for each result. It is appreciated that the above example is only used for illustration of a suitable query modification step and that other examples are envisioned.

[0032] Method 100 includes a step 110 of fetching a second subset of data for the query from the data repository. In some implementations, the second subset of data for the query is less than the entire result set so as to allow additional "on the fly" building of a

visualization. Step 110 can incorporate one or more aspects of step 106 described above with respect to fetching the first subset of data. For example, and as described above with respect to step 106 of fetching the first subset of data, the second subset of data may be a single result or several results for the query. Additionally, and as described above with respect to step 106 of fetching the first subset of data, the second subset of data can be defined as a next number of results of a query, can be determined based on a time period of the fetching step, may be determined based on multiple factors, etc.

[0033] Method 100 includes a step 112 of updating the visualization based on the second subset of data and the visualization parameter (and in some implementations based on both the first subset of data and the second subset of data and the visualization parameter). Step 112 can incorporate one or more aspects of step 108 described above with respect to building the visualization based on the first subset of data and the visualization parameter. For example, updating the visualization based on the second subset of data and the visualization parameter can include comparing one or more metadata attribute values for the second subset of data (or the combined first and second subset of data) to one or more reference attribute values. Additional details regarding the process of building a visualization based on subsets of data and the visualization parameter is provided above with respect to step 108.

[0034] Although the flowchart of FIG. 1 and description of method 100 identifies one order of performance, it is appreciated that this order may be rearranged into another suitable order, may be executed concurrently or with partial concurrence, include additional or comparable steps to achieve the same or comparable functionality, or a combination thereof.

[0035] FIG. 4 illustrates another example of method 100 in accordance with the present disclosure. Several steps of the example method 100 of FIG. 1 are provided for illustration in the example method 100 of FIG. 4 and the same reference numbers are used between figures. However, it is appreciated that additional and/or alternative steps or aspects of steps may be applied to the example of method 100 of FIG. 4.

[0036] Method 100 includes a step 114 of fetching a complete data set for the query. As described above, steps 106 and 110 are used to fetch subsets of a complete result set for the query so as to allow “on the fly” building of a visualization. In some implementations, step 114 completes the process to fetch the remainder of the data set (e.g., following steps 106 and 110 or additional fetching steps). In other implementations, step 114 is a separate fetching operation that fetches the complete data set for the query from the beginning and does not rely on the first and second subsets of data fetched in steps 106 and 108.

[0037] Method 100 includes a step 116 of updating the visualization based on the complete data set and the visualization parameter. For example, in some implementations, updating the visualization based on the complete data set and the visualization parameter includes comparing a metadata attribute value for the complete data set to a first reference attribute value. Step 116 can incorporate one or more aspects of steps 108 and 112 described above with respect to building the visualization based on the first subset of data and updating the visualization based on the second subset of data. For example, updating the visualization based on the complete data set and the visualization parameter can include comparing one or more metadata attribute values for the complete data set of data to one or more reference attribute values. Additional details regarding the process of building a visualization based on subsets of data and the visualization parameter is provided above with respect to steps 108 and 112.

[0038] In some implementations, updating the visualization based on the complete data set and the visualization parameter includes comparing a first metadata attribute value for the complete data set to a first reference attribute value and comparing a second metadata attribute value for the complete data set to a second reference attribute value. The first metadata attribute value can, for example, correspond to a number of results in the complete data set. The second metadata attribute value can, for example, correspond to a number of null results in the complete data set. It is appreciated that such multiple comparisons can be applied to other steps in method 100, such as the visualization building step 108 based on the first subset of data or the visualization updating step 112 based on the second subset of data.

[0039] FIG. 5 illustrates a diagram of an example system 118 in accordance with the present disclosure. Although system 118 makes reference to method 100 as well as other implementations of the disclosure, it is appreciated that system 118 may include additional, alternative, or fewer steps, features, or other aspects compared to method 100. As described in further detail below, system 118 includes a processor 120 and a memory 122 that stores machine-readable instructions that when executed by processor 120 are to modify a query for a data repository based on visualization rules (instructions 124), and to build a visualization based on the visualization rules for results of the modified query as results are fetched from the data repository (instructions 126). The various aspects of system 118 including processor 120, memory 122, and instructions 124 and 126 will be described in further detail below.

[0040] Instructions 124 stored on memory 122 are, when executed by processor 120, to cause processor 120 to modify a query for a data repository based on visualization rules. Instructions 124 can incorporate one or more aspects of step 104 relating to preparing a query or another suitable aspect of method 100 (and vice versa). As but one example, in some implementations, modifying a query for a data repository based on visualization rules can include modifying the query parameter to only retrieve data to be used for a given visualization, as described above with respect to method 100.

[0041] Instructions 126 stored on memory 122 are, when executed by processor 120, to cause processor 120 to build a visualization based on the visualization rules for results of the modified query as results are fetched from the data repository. Instructions 126 can incorporate one or more aspects of step 106 relating to fetching a first subset of data, step 108 relating to building a visualization based on the first subset of data, step 110 relating to fetching a second subset of data, and step 112 relating to updating the visualization based on the second subset of data (and in some implementations based on both the first subset of data and the second subset of data). It is appreciated that instructions 126 can incorporate other suitable aspects of method 100 (and vice versa).

[0042] Processor 120 of system 118 can, for example, be in the form of a central processing unit (CPU), a semiconductor-based microprocessor, a digital signal processor (DSP) such as a digital image processing unit, other hardware devices or processing elements suitable to retrieve and execute instructions stored in memory 122, or suitable

combinations thereof. Processor 120 can, for example, include single or multiple cores on a chip, multiple cores across multiple chips, multiple cores across multiple devices, or suitable combinations thereof. Processor 120 can be functional to fetch, decode, and execute instructions as described herein. As an alternative or in addition to retrieving and executing instructions, processor 120 can, for example, include at least one integrated circuit (IC), other control logic, other electronic circuits, or suitable combination thereof that include a number of electronic components for performing the functionality of instructions stored on memory 122. Processor 120 can, for example, be implemented across multiple processing units and instructions may be implemented by different processing units in different areas of system 118.

[0043] Memory 122 of system 118 can, for example, be in the form of a non-transitory machine-readable storage medium, such as a suitable electronic, magnetic, optical, or other physical storage apparatus to contain or store information such as machine-readable instructions 124 and 126. Such instructions can be operative to perform one or more functions described herein, such as those described herein with respect to the method of FIG. 1 or other methods described herein. Memory 122 can, for example, be housed within the same housing as processor 120 for system 118, such as within a computing tower case for system 118. In some implementations, memory 122 and processor 120 are housed in different housings. As used herein, the term “machine-readable storage medium” can, for example, include Random Access Memory (RAM), flash memory, a storage drive (e.g., a hard disk), any type of storage disc (e.g., a Compact Disc Read Only Memory (CD-ROM), any other type of compact disc, a DVD, etc.), and the like, or a combination thereof. In some implementations, memory 122 can correspond to a memory including a main memory, such as a Random Access Memory (RAM), where software may reside during runtime, and a secondary memory. The secondary memory can, for example, include a nonvolatile memory where a copy of machine-readable instructions are stored. It is appreciated that both machine-readable instructions as well as related data can be stored on memory mediums and that multiple mediums can be treated as a single medium for purposes of description.

[0044] Memory 122 can be in communication with processor 120 via a communication link 134. Communication link 134 can be local or remote to a machine (e.g., a computing

device) associated with processor 120. Examples of a local communication link 134 can include an electronic bus internal to a machine (e.g., a computing device) where memory 122 is one of volatile, non-volatile, fixed, and/or removable storage medium in communication with processor 120 via the electronic bus.

[0045] In some implementations, one or more aspects of system 118 can be in the form of functional modules that can, for example, be operative to execute one or more processes of instructions 124 and 126 or other functions described herein relating to other implementations of the disclosure (e.g., the method of FIG. 1, the medium of FIG. 6). As used herein, the term “module” refers to a combination of hardware (e.g., a processor such as an integrated circuit or other circuitry) and software (e.g., machine- or processor-executable instructions, commands, or code such as firmware, programming, or object code). A combination of hardware and software can include hardware only (i.e., a hardware element with no software elements), software hosted at hardware (e.g., software that is stored at a memory and executed or interpreted at a processor), or hardware and software hosted at hardware. It is further appreciated that the term “module” is additionally intended to refer to one or more modules or a combination of modules. Each module of a system 118 can, for example, include one or more machine-readable storage mediums and one or more computer processors.

[0046] In view of the above, it is appreciated that the various instructions of system 118 described above can correspond to separate and/or combined functional modules. For example, instructions 124 can correspond to a “query modification module” to modify a query for a data repository based on visualization rules and instructions 126 can correspond to a “visualization preparation module” to build a visualization based on the visualization rules for results of the modified query as results are fetched from the data repository. It is further appreciated that a given module can be used for multiple related functions. As but one example, in some implementations, a single module can be used to both modify the query (e.g., corresponding to the process of instructions 124) as well as to prepare the visualization (corresponding to the process of instructions 126).

[0047] FIG. 6 illustrates an example machine-readable storage medium 136 including various instructions that can be executed by a processor to prepare data visualizations. For illustration, the description of machine-readable storage medium 136 provided herein

makes reference to various aspects of system 118 (e.g., processor 120) and other implementations of the disclosure. Although one or more aspects of system 118 (as well as its corresponding instructions 124 and 126) and/or method 100 can be applied or otherwise incorporated with medium 136, it is appreciated that in some implementations, medium 136 may be stored or housed separately from such a system. For example, in some implementations, medium 136 can be in the form of Random Access Memory (RAM), flash memory, a storage drive (e.g., a hard disk), any type of storage disc (e.g., a Compact Disc Read Only Memory (CD-ROM), any other type of compact disc, a DVD, etc.), and the like, or a combination thereof.

[0048] Medium 136 includes machine-readable instructions 128 stored thereon to cause processor 120 to receive a request for data stored in a data repository. Instructions 128 of medium 136 can incorporate one or more aspects of instructions 124 described above with respect to system 118, one or more aspects of step 102 described above with respect to method 100, and vice versa. As an example and as described in detail above with respect to method 100, the request can include a query parameter in the form of a SELECT statement and a visualization parameter in the form of a function. It is appreciated that other forms of the query parameter and visualization parameter can be provided.

[0049] Medium 136 includes machine-readable instructions 130 stored thereon to cause processor 120 to prepare a query based on both the query parameter and the visualization parameter. Instructions 130 of medium 136 can incorporate one or more aspects of instructions 124 described above with respect to system 118, one or more aspects of step 104 of method 100, and vice versa. As an example and as described in detail above with respect to method 100, preparing a query based on both the query parameter and the visualization parameter can include modifying a query parameter based on rules of the visualization parameter.

[0050] Medium 136 includes machine-readable instructions 132 stored thereon to cause processor 120 to dynamically build and update a visualization for the query based on the visualization parameter as data for the query is fetched from the data repository. Instructions 132 of medium 136 can incorporate one or more aspects of instructions 126 described above with respect to system 118, one or more aspects of steps 106, 108,

110, and 112 of method 100 and vice versa. As an example, in some implementations, instructions 132 are to allow “on the fly” building of a visualization based on incremental subsets of data fetched from the data repository.

[0051] Several example implementations will now be described to illustrate aspects of the present disclosure and to provide additional implementation details for specific implementations of the present disclosure. In some implementation a query module may be provided. The query module can, for example, be in the form of a backend of a relational database management system (RDBMS), such as Loom. The backend of an existing RDBMS can, for example, be modified to include (or work with) a binding module, which is described in detail below. Such a modification can include extending a query language to enable users to specify visualisation options in the form of a Visualisation Parameters (VP), which are described in further detail below. For instance, “SELECT * FROM CAR_TABLE RENDERED AS VP1.” In some implementations, the query module can simply ignore the VPs (i.e., the part after “RENDERED AS”) and can process the query as usual. In some implementations, and as described below, VP lambdas can be provided to allow the query module to accumulate results at query runtime in order to avoid looping multiple times over the same data, or for other reasons. This approach may require changes to the backend of a RDBMS such that the result of a query is not just the resulting data set, but also a set of properties of the data set that were computed at query runtime.

[0052] After executing the query, the query module can pass the result set and the VP to the binding engine. In this implementation, the binding engine can be considered “a dumb component” that merely executes the scripted instructions as specified in the VPs. For example, the binding engine, can begin its processing by examining specified properties of the result set. Some of these may come directly from the query module (such as the number of resulting elements), while others requires further inspection of the result set (e.g. data quality). The query module can then take code (e.g. in the form of a lambda) and close on the result set to calculate the desired properties that will define the type of visualisation to be performed. As mentioned above, in some implementations, the binding engine may loop on the data again to find accumulating properties (e.g. moving average of a value on time-based sliding windows of different sizes). Additional

functionality can be provided by enabling query runtime lambdas in VPs and adapting the query module appropriately.

[0053] In this specific example, VPs can include various aspects, such as: (1) result-set analysis lambdas, which can be in the form of executable code that takes the result set as input and extracts a series of attributes that can only be computed after execution, (2) accumulating analysis lambdas, which can be in the form of executable code to be executed by the query module at runtime to gather accumulative properties of the result set and prevent multiple loops on the data, and (3) data visualization scripting. Such data visualization scripting can be in the form of code that takes the result set and the extracted attributes and writes a static visualisation parameter to be used by a visualization module to plot and layout the data. Such code can, for example, contain multiple branches that evaluate the values of the extracted attributes (e.g. "if (numOfElements < 100) then...") and decide how to plot/layout (e.g. "then createSpiralLayout(resultSet)"). The set of sentences after the "then" part of this branched conditional code can, for example, be dependent on the visualization module being used to render the data, but they can also be implemented in a configurable manner that detects the visualization module being used and automatically converts these high level primitives (such as createSpiralLayout()) to visualization module-specific instructions. Additional and/or alternative details of VPs are described above, for example with respect to method 100.

[0054] One specific implementation of the present disclosure is to add a set of rendering operations in a data flow query language (such as Loom). Adding such rendering operations at the end of a flow can, for example, help to extract the result set-dependent properties used for rendering. Loom has built-in capabilities that let users define their own data flow operations in the form of lambdas that close on the data resulting from previous phases of the dataflow. One result of adding such visualisation operations in the dataflow is not just providing a raw dataset resulting from executing the query, but providing a transformed version that lets a virtualization module that Loom relies on (e.g. Loom's Weaver) know how to actually render that information.

[0055] Another specific implementation of the present disclosure can be undertaken for a RDBMS that processes a normal query and extracts the results. The RDBMS can, for

example, be operate as a front-end with a wrapper that strips off the part of the “visQL” that deals with VPs. The term “visQL” as used herein can, for example, refer to a SQL enhanced to contain VPs like, for instance, “SELECT * FROM CAR_TABLE RENDERED AS VP1.” After this operation, the results can be written to a file in a predefined location so that the binding engine can then take them and execute VP-specified lambdas when instructed to do so by the RDMBS wrapper, which acts as an orchestrator to coordinate all the components. When the binding engine has extracted the properties of the result set, it can then invoke one or more data visualisation scripts that create static templates. These templates can then be returned to the client (e.g., written in HTML5/CSS/Javascript) together with the result set to render them appropriately.

[0056] While certain implementations have been shown and described above, various changes in form and details may be made. For example, some features that have been described in relation to one implementation and/or process can be related to other implementations. In other words, processes, features, components, and/or properties described in relation to one implementation can be useful in other implementations. Furthermore, it should be appreciated that the systems and methods described herein can include various combinations and/or sub-combinations of the components and/or features of the different implementations described. Thus, features described with reference to one or more implementations can be combined with other implementations described herein.

[0057] As used herein, “logic” is an alternative or additional processing resource to perform a particular action and/or function, etc., described herein, which includes hardware, e.g., various forms of transistor logic, application specific integrated circuits (ASICs), etc., as opposed to machine executable instructions, e.g., software firmware, etc., stored in memory and executable by a processor. Further, as used herein, “a” or “a number of” something can refer to one or more such things. For example, “a number of widgets” can refer to one or more widgets. Also, as used herein, “a plurality of” something can refer to more than one of such things.

CLAIMS

What is claimed is:

1. A method comprising:

receiving a request for data stored in a data repository, wherein the request includes a query parameter to retrieve data of the data repository and a visualization parameter for the retrieved data;

preparing a query based on both the query parameter and the visualization parameter;

fetching a first subset of data for the query from the data repository;

building a visualization based on the first subset of data and the visualization parameter;

fetching a second subset of data for the query from the data repository; and

updating the visualization based on the second subset of data and the visualization parameter.

2. The method of claim 1, wherein the visualization is in the form of a graph.

3. The method of claim 1, wherein the visualization parameter provides rules for graphing fetched data based on values of the fetched data.

4. The method of claim 1, wherein the first subset of data is a single result for the query from the data repository.

5. The method of claim 1, wherein the visualization parameter is a set of instructions that includes a query filter, and

wherein preparing a query based on both the query parameter and the visualization parameter includes applying the query filter to the query parameter.

6. The method of claim 1, wherein building a visualization based on the first subset of data and the visualization parameter includes comparing a metadata attribute value for the first subset of data to a reference attribute value.
7. The method of claim 6, wherein the metadata attribute value is a value corresponding to a number of null results in the first subset of data.
8. The method of claim 1, further comprising:
 - fetching a complete data set for the query; and
 - updating the visualization based on the complete data set and the visualization parameter.
9. The method of claim 8, wherein updating the visualization based on the complete data set and the visualization parameter includes comparing a metadata attribute value for the complete data set to a reference attribute value.
10. The method of claim 8,
 - wherein updating the visualization based on the complete data set and the visualization parameter includes comparing a first metadata attribute value for the complete data set to a first reference attribute value and comparing a second metadata attribute value for the complete data set to a second reference attribute value,
 - wherein the first metadata attribute value corresponds to a number of results in the complete data set, and
 - wherein the second metadata attribute value corresponds to a number of null results in the complete data set.

11. A non-transitory machine readable storage medium having stored thereon machine readable instructions to cause a computer processor to:

receive a request for data stored in a data repository, wherein the request includes a query parameter and a visualization parameter;

prepare a query based on both the query parameter and the visualization parameter; and

dynamically build and update a visualization for the query based on the visualization parameter as data for the query is fetched from the data repository.

12. The medium of claim 11, wherein the visualization is based on a multi-dimensional mathematical function.

13. A system comprising:

a processor; and

a memory storing machine readable instructions to cause the processor to:

modify a query for a data repository based on visualization rules; and

build a visualization based on the visualization rules for results of the modified query as results are fetched from the data repository.

14. The system of claim 13, wherein the memory further stores machine readable instructions to cause the processor to:

further modify the modified query as results are fetched from the data repository.

15. The system of claim 13, wherein the query and the modified query are Structured Query Language (SQL) queries.

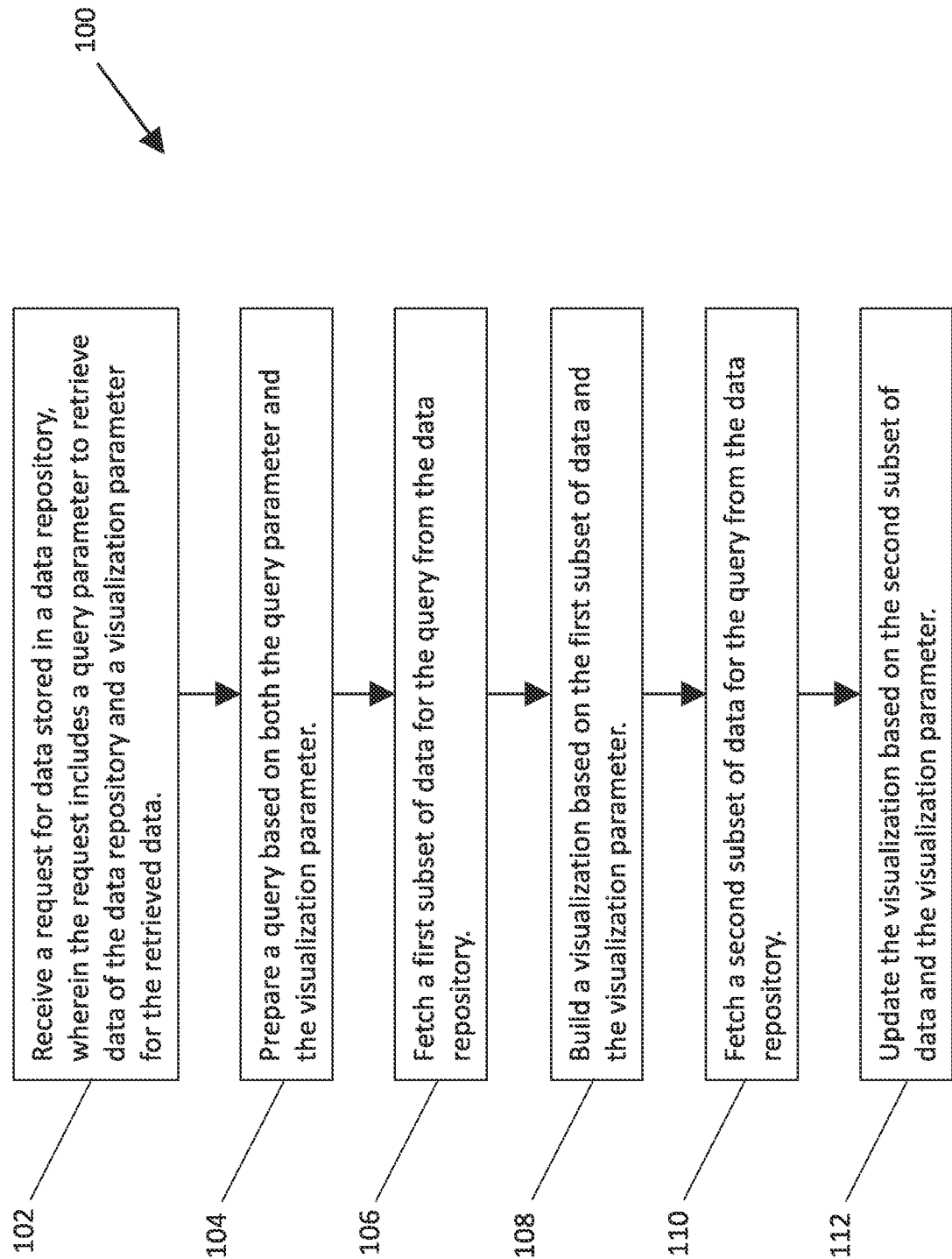


FIG. 1

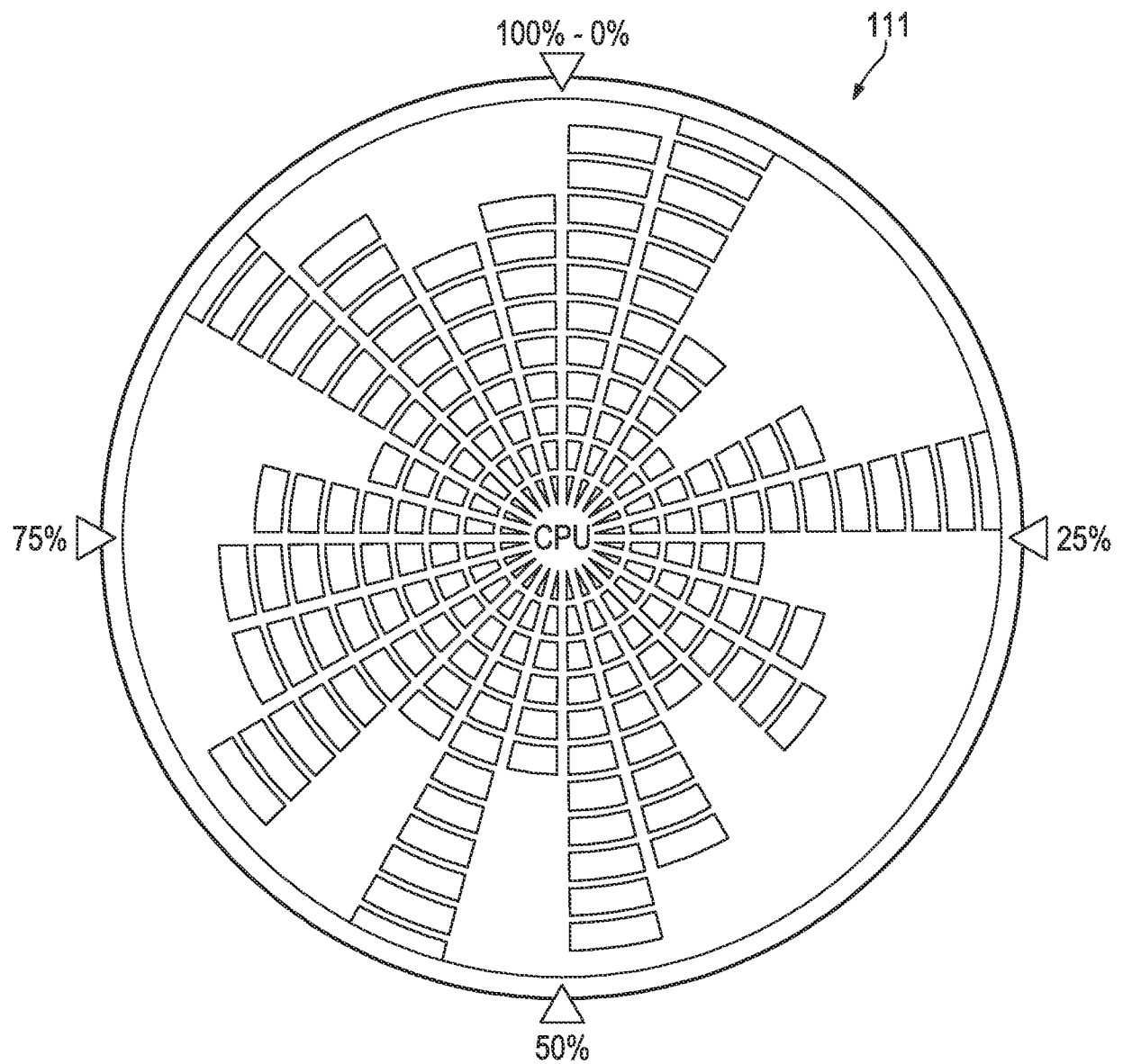


FIG. 2

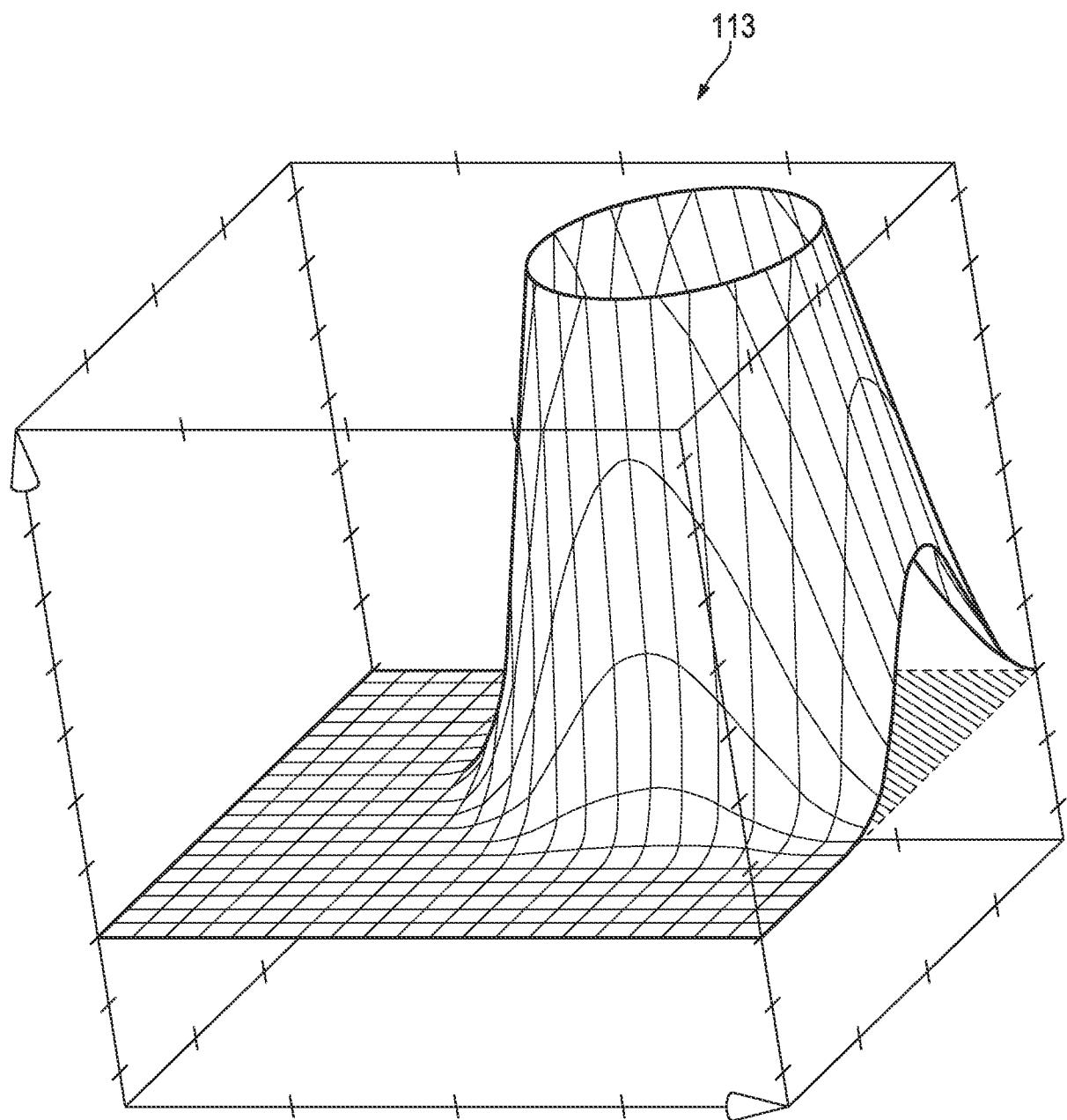


FIG. 3

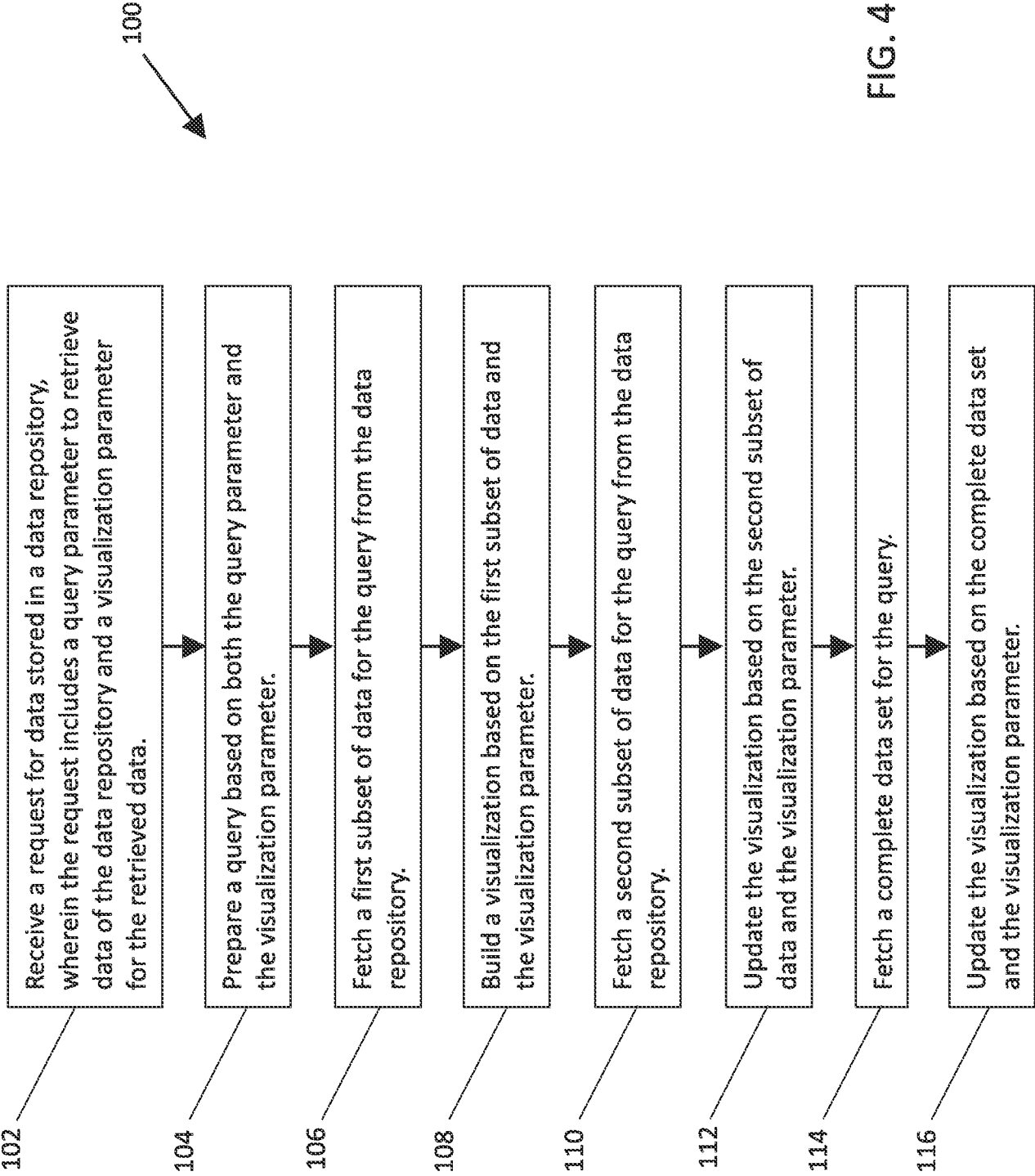


FIG. 4

118 →

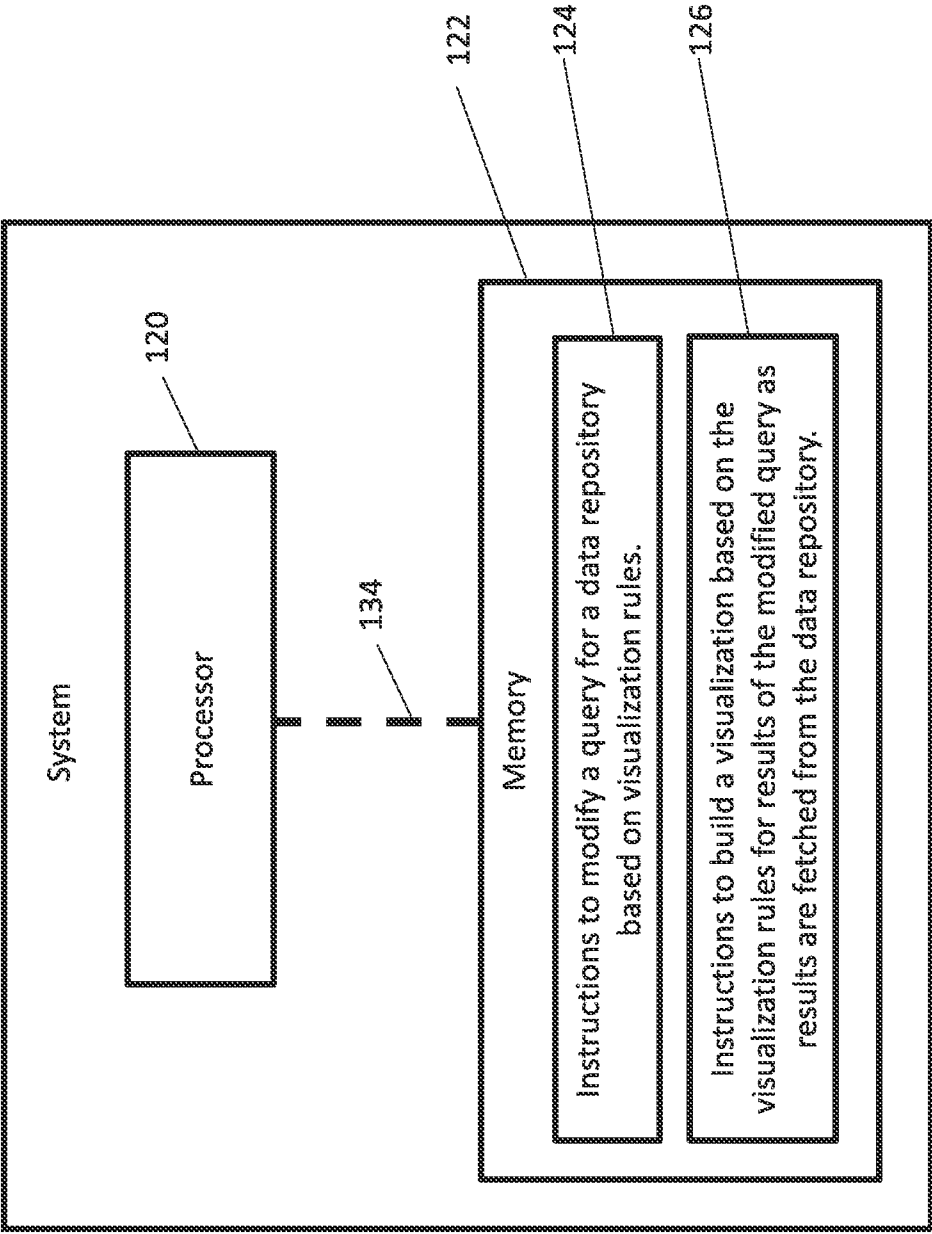


FIG. 5

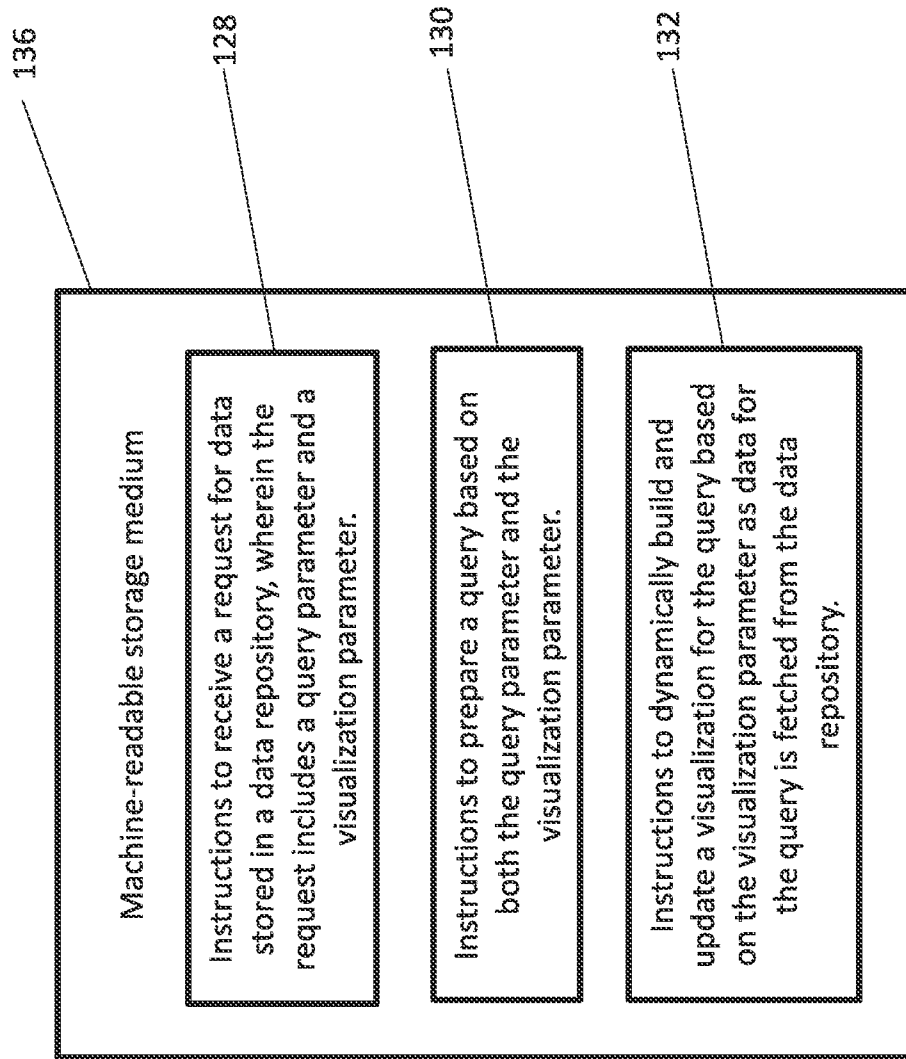


FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/024641**A. CLASSIFICATION OF SUBJECT MATTER****G06F 17/30(2006.01)i, G06F 17/00(2006.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 17/30; G06F 17/24; G06F 15/18; G06F 17/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: query, visualization, subset, parameter, data repository, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2014-0280158 A1 (THE PYXIS INNOVATION INC.) 18 September 2014 See paragraphs [0034]-[0036], [0072]-[0075], [0089], [0104], [0132]-[0133], and [0138]; claims 1, 4-5, and 8; and figures 8-11.	13-15
A		1-12
Y	EVGENIY YUR'EVICH GORODOV et al., "Analytical Review of Data Visualization Methods in Application to Big Data," Journal of Electrical and Computer Engineering, Volume 2013, Article ID 969458, pages 1-7, January 2013 See left column of page 4, lines 25-30; left column of page 6, lines 43-50; and figure 3.	13-15
A	US 2013-0268520 A1 (DANYEL A. FISHER et al.) 10 October 2013 See paragraph [0034] and figure 1.	1-15
A	US 2014-0304581 A1 (THE BOARD OF TRUSTEES OF THE LELAND STANFORD, JR. UNIVERSITY) 09 October 2014 See paragraph [0115] and figure 10.	1-15
A	US 7,216,116 B1 (MARTIN B. NILSSON et al.) 08 May 2007 See column 16, lines 1-19; and figure 6.	1-15



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

27 January 2016 (27.01.2016)

Date of mailing of the international search report

28 January 2016 (28.01.2016)

Name and mailing address of the ISA/KR

International Application Division
Korean Intellectual Property Office
189 Cheongsu-ro, Seo-gu, Daejeon Metropolitan City, 35208,
Republic of Korea

Facsimile No. +82-42-472-7140

Authorized officer

NHO, Ji Myong

Telephone No. +82-42-481-8528



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/024641

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014-0280158 A1	18/09/2014	CA 2846717 A1	15/09/2014
US 2013-0268520 A1	10/10/2013	US 8983936 B2	17/03/2015
US 2014-0304581 A1	09/10/2014	AU 2004-248607 A1	23/12/2004
		CA 2526825 A1	23/12/2004
		EP 1634142 A2	15/03/2006
		EP 1634142 A4	16/01/2008
		JP 2006-526840 A	24/11/2006
		US 2004-243593 A1	02/12/2004
		US 2006-259509 A1	16/11/2006
		US 2011-131250 A1	02/06/2011
		US 2012-179713 A1	12/07/2012
		US 7089266 B2	08/08/2006
		US 7716173 B2	11/05/2010
		US 7882144 B1	01/02/2011
		US 8140586 B2	20/03/2012
		US 8713072 B2	29/04/2014
		US 8972457 B2	03/03/2015
		WO 2004-111786 A2	23/12/2004
		WO 2004-111786 A3	23/03/2006
US 7216116 B1	08/05/2007	US 6014661 A	11/01/2000
		US 6405195 B1	11/06/2002