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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 December 2013 (12.12.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/184093 Al

(51) International Patent Classification:
G06F 17/30 (2006.01) *G06F 15/16* (2006.01)

(21) International Application Number:
PCT/US20 12/040764

(22) International Filing Date:
4 June 2012 (04.06.2012)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US):* **HEW¬LETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 11445 Compaq Center Drive W., Houston, Texas 77070 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only):* **SEERING, Adam** [US/US]; 150 Cambridge Park Drive, 10th Floor, Cambridge, Massachusetts 02140 (US). **VENKATESH, Rajat** [IN/US]; 150 Cambridge Park Drive, 10th Floor, Cambridge, Massachusetts 02140 (US). **BEAR, Charles, Ed¬ward** [US/US]; 150 Cambridge Park Drive, 10th Floor, Cambridge, Massachusetts 02140 (US). **LAWANDE, Shilpa** [US/US]; 150 Cambridge Park Drive, 10th Floor, Cambridge, Massachusetts 02140 (US). **LAMB, Andrew, Allinson** [US/US]; 150 Cambridge Park Drive, 10th Floor, Cambridge, Massachusetts 02140 (US).

(74) Agents: **JAKOB SEN, Kraig A.** et al; Hewlett-Packard Company, Intellectual Property Administration, 3404 E. Harmony Road, Mail Stop 35, Fort Collins, Colorado 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, 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, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD,

[Continued on next page]

(54) Title: USER-DEFINED LOADING OF DATA ONTO A DATABASE

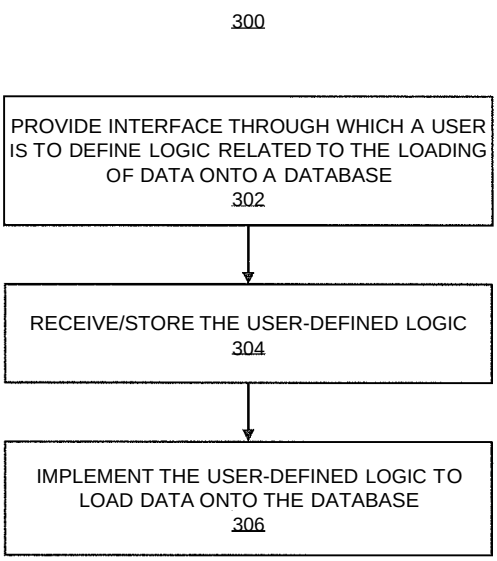

FIG. 3

(57) Abstract: As part of managing the loading of data from a source onto a database, according to an example, an interface through which a user is to define logic related to the loading of the data onto the database is provided. The user-defined logic pertains to at least one of a user-defined location identification of the source, a user-defined filter to be applied on the data, and a user-defined parsing operation to be performed on the data to convert the data into an appropriate format for the database. In addition, the user-defined logic is received and the user-defined logic is implemented to load the data onto the database.

WO 2013/184093 A1

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, 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, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(if)*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

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

USER-DEFINED LOADING OF DATA ONTO A DATABASE

BACKGROUND

[0001] Data is commonly loaded onto databases from a number of different types of sources. Examples of various types of sources include a local storage location, a file transfer protocol (FTP) site, and Hadoop™ (e.g., stored within the Hadoop™ File system, which is also known as HDFS) In addition, the data that is loaded onto the databases often comprise various types of formats. Examples of various formats include Javascript object notation (JSON) files, extensible markup language (XML) files, custom binary formats, tarballs, and 7-Zip files.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Features of the present disclosure are illustrated by way of example and not limited in the following figure(s), in which like numerals indicate like elements, in which:

[0003] FIG. 1 shows a block diagram of a network environment, according to an example of the present disclosure;

[0004] FIG. 2 illustrates a block diagram of a database machine, according to an example of the present disclosure;

[0005] FIGS. 3 and 4, respectively, depict flow diagrams of methods for managing loading of data from a source to a database, according to examples of the present disclosure; and

[0006] FIG. 5 illustrates a schematic representation of a computing device, which may be employed to perform various functions of the database machines depicted in FIGS. 1 and 2, according to an example of the present disclosure.

DETAILED DESCRIPTION

[0007] For simplicity and illustrative purposes, the present disclosure is described by referring mainly to an example thereof. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present disclosure. It will be readily apparent however, that the present disclosure may be practiced without limitation to these specific details. In other instances, some methods and structures have not been described in detail so as not to unnecessarily obscure the present disclosure. In the present disclosure, the term "includes" means includes but not limited thereto, the term "including" means including but not limited thereto. The term "based on" means based at least in part on. In addition, the terms "a" and "an" are intended to denote at least one of a particular element.

[0008] As used throughout the present disclosure, the term "data" is generally intended to encompass electronic data, such as electronic mail (e-mail), word processing documents, spreadsheet documents, webpages, computer aided drawing documents, electronic file folders, database records, logs, sales information, patient information, etc.

[0009] Disclosed herein is a method for managing loading of data from a source onto a database. Also disclosed herein are an apparatus for implementing the method and a non-transitory computer readable medium on which is stored machine readable instructions that implement the method. As discussed herein, the method for managing loading of data from a source onto a database comprises providing an interface through which a user is to define logic related to the loading of the data onto the database, in which the user-defined logic pertains to a user-defined location identification of the source and retrieval of data from the source, a user-defined filter to be applied on the data, and a parsing operation to be performed on the data to convert the data into an appropriate format for the database. The method also includes receiving the user-defined logic and implementing the user-defined logic to load the data onto the database.

[0010] According to an example, the user-defined logic also pertains to a user-defined policy that defines a condition to be met prior to the data being retrieved from the source. In this example, the user-defined logic may be registered with the database meta-data, such that the loading of the data may be deferred to query time, which is also known as "late binding." In one regard, late binding enables federated use of the user-defined data sources, for instance, the ability to run SQL queries directly over data stored in HDFS. In addition, the user may define the user-defined policy to dynamically transition between immediate loading of the data onto the database and late binding. By way of example, the user may define the user-defined policy to automatically keep the last loaded content in the database for up to a week if the data set is accessed more than a predetermined number times in a predetermined time period via late binding. As another example, the user may define the user-defined policy to move the data set from the database to another storage, such as HDFS, if the data set is rarely used (e.g., used less than a predetermined number of times within a predetermined time period), and to access the data from the another storage via late binding.

[0011] In one regard, the late binding of the data may or may not actually need to persist to disk, because the data is only used for the duration of the query. Instead the data load operation may be streamed directly into the query processing pipeline as the data is being extracted by the user-defined load command disclosed herein.

[0012] Data to be loaded onto a database often has one of a variety of different types of formats and sizes. As such, it is typically impractical for any single piece of software to fully support every possible data format and encoding that may be of practical value to any number of users. In one regard, the method disclosed herein enables users to define customized logic or functions, such as programming, code, etc., to manage the loading of the data onto a database, and in certain instances, to manage data that has already been loaded onto the database. As such, a user may implement the method disclosed herein to load

data onto a database regardless of the format and size of the data as well as the format of the source from which the data is to be loaded. Particularly, for instance, the method disclosed herein enables data that is at least one of encoded using a custom library, data that is in a custom format, data that is available via a custom source, etc., to be loaded onto the database in a relatively simple manner.

[0013] In addition, the user-defined logic of at least one user may be stored for later retrieval by the user or by other users. In this regard, users may mix and match the stored user-defined logic to manage loading of various types of data from various types of sources on to various types of databases, without having to define the logic each time the data is to be loaded. Furthermore, users may access the stored user-defined logic to manage data that has already been loaded onto the database.

[0014] According to an example, the method disclosed herein is implemented by an apparatus that also manages the database. As such, a uniform interface for loading the data, which includes a relatively simple user interface, may be employed to manage the loading of the data. In addition, management tools may relatively easily track and support the loading of the data. Moreover, the loading of the data may substantially be optimized, for instance, to substantially minimize bottlenecks, to substantially maximize efficiency, etc.

[0015] Through implementation of various examples of the present disclosure, users may be provided with an interface and options that afford a relatively high level of flexibility in managing the loading of data onto a database. In addition, the users may customize various aspects of the loading of the data, as well as the management of data that has already been loaded onto the database, without having to understand and cope with a full-fledged distribution computation system. As such, the various examples of the present disclosure enable users to load the data onto a database, as well as management of the data on the database, in a relatively simple and efficient manner.

[0016] With reference first to FIG. 1, there is shown a block diagram of a network environment 100, according to an example of the present disclosure. The network environment 100 is depicted as including a distributed database system 110, a network device 114, a network 120, and a source 130. The distributed database system 110 is also depicted as having a plurality of database machines 112a-112n, in which "n" represents an integer greater than one. It should be understood that the network environment 100 may include additional components and that one or more of the components described herein may be removed and/or modified without departing from a scope of the network environment 100. For instance, the network environment 100 may include any number of database machines 112a-112n and sources 130.

[0017] The database machines 112a-112n comprise servers or other apparatuses on which data is stored and that are able to receive data transferred from the source 130. According to another example, a database machine 112a comprises a machine that manages the transfer of the data between other database machines 112b-112n, without having the data stored on the database machine 112a. The distributed database system 110 is a database in which the database machines 112a-112n are not all attached to a common central processing unit (CPU). In one regard, the distributed database system 110 enables data to be stored in multiple ones of the database machines 112a-112n, which may be located in one physical location or dispersed in geographically disparate locations. According to an example, the database provided or hosted by the distributed database system 110 is a relational database.

[0018] The network device 114 comprises any of, for instance, a network switch, router, hub, etc., through which data may be communicated to and from the database machines 112a-112n. In addition, or alternatively, at least two of the database machines 112a-112n may be directly connected to each other, i.e., without going through the network device 114. The network 120 represents an infrastructure through which the database machines 112a-112n may communicate

with the source 130. In this regard, the network 120 comprises any of, a local area network, a wide area network, the Internet, etc., and may include additional network devices 114, which are not shown. Although not shown, any of the database machines 112a-112n may be connected to the network 120 through a plurality of network devices 114, for instance, a series of interconnected switches.

[0019] Various manners in which data is loaded from the source 130 onto a database hosted on the distributed database system 110, for instance, on at least one of the database machines 112a-112n, are described in detail herein below. Particularly, the loading of data from the source 130 to the database may be managed in a manner that enables users, such as computer system administrators, information technology personnel, end users, etc., to substantially customize any of extraction, transformation, and loading of the data onto the database, as well as management of previously loaded data, in a relatively simple and efficient manner. In addition, the data may be loaded from the source 130 to the database in a substantially optimized manner, as also discussed in greater detail herein below.

[0020] The loading of the data may be considered as being substantially optimized when at least one of the amount of time required to load the data is substantially minimized, the number of copies of the data being loaded is substantially minimized, the consumption of network resources is substantially minimized during the loading of the data, the use of the most efficient network resources is substantially maximized during the loading of the data, etc. In addition, the loading of the data may be considered as being substantially optimized when the loading of the data is performed in a manner that is less than an optimized data loading manner in instances where the optimized data loading manner is infeasible, violates a policy, etc. Thus, for instance, the loading of the data may be considered as being substantially optimized when the loading of the data is performed in a manner that is optimized with respect to any of a variety of constraints applicable on the loading of the data.

[0021] Turning now to FIG. 2, there is shown a block diagram of a database machine 200, according to an example of the present disclosure. The database machine 200 generally represents any of the database machines 112a-112n depicted in FIG. **1.** It should be understood that the database machine 200 may include additional components and that one or more of the components described herein may be removed and/or modified without departing from a scope of the database machine 200.

[0022] The database machine 200 is depicted as including a processor 202, a data store 204, an input/output interface 206, and a data loading manager 210. The database machine 200 comprises any of, for instance, a server, a computer, a laptop computer, a tablet computer, a personal digital assistant, a cellular telephone, or other electronic apparatus that is to perform a method for managing loading of data from a source onto a database disclosed herein. The database machine 200 may therefore store the data and/or may manage the storage of data in other database machines 112a-112n.

[0023] The data loading manager 210 is depicted as including an input/output module 212, an interface providing module 214, a user-defined logic receiving module 216, a user-defined logic implementation module 218, a predefined load stack implementation module 220, a data loading optimization module 222, and a data loading module 224. The processor 204, which may comprise a microprocessor, a micro-controller, an application specific integrated circuit (ASIC), or the like, is to perform various processing functions in the database machine 200. One of the processing functions includes invoking or implementing the modules 212-224 contained in the data loading manager 210 as discussed in greater detail herein below.

[0024] According to an example, the data loading manager 210 comprises a hardware device, such as a circuit or multiple circuits arranged on a board. In this example, the modules 212-224 comprise circuit components or individual circuits. According to another example, the data loading manager 210 comprises a volatile

or non-volatile memory, such as dynamic random access memory (DRAM), electrically erasable programmable read-only memory (EEPROM), magnetoresistive random access memory (MRAM), Memristor, flash memory, floppy disk, a compact disc read only memory (CD-ROM), a digital video disc read only memory (DVD-ROM), or other optical or magnetic media, and the like. In this example, the modules 212-224 comprise software modules stored in the data loading manager 210. According to a further example, the modules 212-224 comprise a combination of hardware and software modules.

[0025] The input/output interface 206 comprises a hardware and/or a software interface. In any regard, the input/output interface 206 may be connected to a network, such as the Internet, an intranet, etc., through the network device 114, over which the data loading manager 210 may receive and communicate information, for instance, information relevant to the user-defined logic and the loading of the data. The processor 202 may store information received through the input/output interface 206 in the data store 204 and may use the information in implementing the modules 212-224. The data store 204 comprises volatile and/or non-volatile memory, such as DRAM, EEPROM, MRAM, phase change RAM (PCRAM), Memristor, flash memory, and the like. In addition, or alternatively, the data store 204 comprises a device that is to read from and write to a removable media, such as a floppy disk, a CD-ROM, a DVD-ROM, or other optical or magnetic media.

[0026] Various manners in which the modules 212-224 may be implemented are discussed in greater detail with respect to the methods 300 and 400 depicted in FIGS. 3 and 4. FIGS. 3 and 4, respectively depict flow diagrams of methods 300, 400 for managing loading of data from a source 130 to a database that is hosted on at least one of the database machines 112a-112n, 200, according to an example of the present disclosure. It should be apparent to those of ordinary skill in the art that the methods 300 and 400 represent generalized illustrations and that other operations may be added or existing operations may be removed, modified or

rearranged without departing from the scopes of the methods 300 and 400. Although particular reference is made to the data loading manager 210 depicted in FIG. 2 as comprising an apparatus and/or a set of machine readable instructions that may perform the operations described in the methods 300 and 400, it should be understood that differently configured apparatuses and/or machine readable instructions may perform the methods 300 and 400 without departing from the scopes of the methods 300 and 400. The method 400 is related to the method 300 in that the method 400 includes operations in addition to those included in the method 300. Thus, the method 400 is to be construed as including all of the features discussed below with respect to the method 300.

[0027] Generally speaking, the methods 300 and 400 may separately be implemented to manage loading of data from a source 130 to a database hosted on at least one of the database machines 112a-112n. In one regard, the methods 300 and 400 may be implemented to manage the loading of the data, such that a user may relatively easily customize various stages associated with loading of the data. In another regard, the methods 300 and 400 may be implemented to manage the loading of the data, such that the data is transferred in a substantially optimized manner, as discussed above. In a further regard, the method 400 may be implemented to manage data that is loaded onto the database, for instance, to manage the conditions under which the loaded data is transferred to another storage.

[0028] The source 130 may comprise a single entity or a plurality of entities on which data is stored. Likewise, the database onto which the data from the source 130 is to be loaded may comprise a single database machine 112a or a plurality of database machines 112a-112n. In addition, the source 130 may comprise the same or a different number of entities than the database machine(s) 112a-112n, 200 onto which the data is to be loaded. According to an example, the methods 300, 400 are implemented in a database machine 112a-112n, 200. In a

further example, the methods 300, 400 are implemented in a plurality of database machines 112a-1 12n.

[0029] With reference first to FIG. 3, at block 302, an interface through which a user is to define logic related to the loading of the data onto the database is provided, for instance, by the interface providing module 214. The interface may comprise, for instance, a portal, such as a web portal, a desktop portal, etc., that provides various options and features to a user regarding the loading of data onto a database from a source 130. According to an example, the interface is displayed on a monitor that is local to the database machine 200, i.e., connected to the database machine 200, or remote from the database machine 200, i.e., connected to a client machine (not shown) of the database machine 200.

[0030] The interface generally provides a mechanism through which a user may direct the data loading manager 210 to load data from the source 130. Particularly, the interface provides a mechanism through which the user may define logic pertaining to a user-defined location identification of the source 130, a user-defined filter to be applied on the data, and a parsing operation to be performed on the data to convert the data into an appropriate format for the database. The interface may also provide a mechanism through which the user may define logic pertaining to a user-defined policy that defines a condition to be met prior to the data being retrieved from the source. The user-defined policy may in addition, or alternatively, define how data loaded onto the database is to be managed.

[0031] According to an example, the interface provides the user with access to a library that includes previously defined logic. In addition, or alternatively, the interface provides the user with the ability to program the logic, for instance, through any suitable programming language, such as C++, structured query language (SQL), etc. Thus, for instance, the user may define the logic related to the loading of the data from the source 130 to the database by directly programming the logic or by selecting a previously defined, e.g., programmed, logic. The previously defined logic may have been created by the user, another

user, a programmer of the data loading manager, a third-party vendor, etc., and stored in the library.

[0032] Generally speaking, the user-defined logic provides mechanisms by which users may write and install custom libraries to handle various stages of the data loading process. In one regard, by providing users with an interface to define logic for performing various stages of the data loading process, users are able to selectively replace arbitrary portions of a data-load stack with custom implementations. In addition, users, such as developers, may write fully-functional and interchangeable logic for the various stages of the data loading process. Users are therefore provided with a relatively high level of flexibility in loading various types of data from various types of sources 130 onto a database.

[0033] In one example, the user defines a location identification of the source 130. The location identification of the source 130 may comprise, for instance, an Internet Protocol (IP) address of the source 130, or other location identifier of the source 130 that may be used to access the source 130. The user may also define parameters associated with accessing the appropriate data from the source 130. The parameters may include, for instance, the particular folder in which the data is stored on the source 130 as well as the file name of the data. In addition, the parameters may include appropriate passwords, security keys, and other information that may be required to access the data on the source 130.

[0034] In another example, the user defines at least one filter to be applied to the data retrieved from the source 130. The user may define the at least one filter based upon various factors associated with the data, such as whether the data is compressed, what type of compression scheme was used to compress the data, whether the data is to be compressed, what type of compression scheme is to be used to compress the data, whether the data is to be converted from one format to another, the type of format to which the data is to be converted, whether the data is to be decrypted, what type of decryption scheme is to be employed, etc. As such, the filter(s) may comprise at least one of a filter to decompress the data, a

filter to compress the data, a filter to convert the data from one format to another, a filter to decrypt the data, a filter to render the data (for instance, convert a JPEG image to a bitmap form), etc.

[0035] In a further example, the user defines a parsing operation to be performed on the data to convert the data into an appropriate format for loading onto the database. Alternatively, the suitable parsing operation may be predefined because the format required for the database may already be known. In any regard, the parsing operation may comprise formatting of the data into tuples for use by the database system 110, which may comprise a relational database system.

[0036] In a yet further example, the user defines a policy that defines a condition to be met prior to the data being retrieved from the source. In this regard, the user-defined policy may cause the loading of the data onto the database to be deferred, for instance, until query time, which is also known as "late binding." In one regard, late binding enables federated use of the user-defined data sources, for instance, the ability to run SQL queries directly over data stored in HDFS. In addition, the user may define the user-defined policy to dynamically transition between immediate loading of the data onto the database and late binding.

[0037] According to an example, the late binding capability is implemented through use of External Tables, in which an EXTERNAL keyword is added to SQL CREATE TABLE, which indicates to the database that the data is not stored within the database. Whereas typical use of External tables involves providing a list of data files external to the database, in case of the user-defined logic disclosed herein, the COPY command is associated with the CREATE EXTERNAL TABLE statement. As such, whenever the External table is used in a query, the user-defined logic command is run to first load the data into the database and the query is run on top of the user-defined logic command. In another example, the late binding capability is implemented through use of the meta-data on the source system, for instance, in the case of Hadoop™, HCatalog is used as the meta-data

source.

[0038] In a yet further example, the user defines a policy that defines how data loaded onto the database is to be managed. In this regard, the user-defined policy may cause the data loaded onto the database to be managed in particular manners. By way of example, the user may define the user-defined policy to automatically keep the last loaded content in the database for up to a week if the data set is accessed more than a predetermined number times in a predetermined time period via late binding. As another example, the user may define the user-defined policy to move the data set from the database to another storage, such as HDFS, if the data set is rarely used (e.g., used less than a predetermined number of times within a predetermined time period), and to access the data from the another storage via late binding.

[0039] At block 304, the user-defined logic is received, for instance, by the user-defined logic receiving module 216. According to an example, a user may define the logic related to the loading of the data from the source 130 onto the database through the interface and may submit the defined logic through the interface. In this regard, for instance, the interface may provide a user-defined logic submission feature, which the user may select to submit the user-defined logic to the data loading manager 210.

[0040] In addition, at block 304, the user-defined logic may be stored in a library, for instance, to enable the user-defined logic to be retrievable for later use by the user or by another user. In this example, the user may provide a title for the logic that the user defined, for instance, as the logic relates to the source 130, the format of the data, and/or the types of filters that have been defined. The user-defined logic may therefore be stored using a descriptive title to enable users to find and implement the user-defined logic.

[0041] Once the user-defined logic has been written, compiled, and received, the database engine, for instance, the processor 202, is to be informed of the existence of the user-defined logic. This is performed, for instance, through the

addition of new SQL statements to the database machine 112a that allow registration of a user-defined logic based on the path of the shared library representing the compiled user-defined logic, and the name of the user-defined logic object inside that shared library that is to be used.

[0042] At block 306, the user-defined logic is implemented to load the data onto the database from the source 130, for instance, by the user-defined logic implementation module 218. Although not shown, the user-defined logic may be implemented as part of a pre-configured data-load stack, for instance, by the predefined load stack implementation module 220. Particularly, for instance, the predefined load stack implementation module 220 may perform predefined functions in addition to the functions corresponding to the user-defined logic. The predefined functions may include, for instance, a parsing operation that transforms the data into a format suitable for loading the data onto the database, a converting operation to convert data from a particular format to a format that is compatible for processing in the parsing operation, etc. The predefined functions may also include, for instance, invocation of various calls associated with implementation of the user-defined filters.

[0043] According to an example, to invoke the user-defined logic, an extension to the existing syntax for loading files into a database is generated, for instance, the COPY statement. The COPY statements add a WITH clause. By way of particular example, a user may write "WITH SOURCE MyExampleSourceQ" or "WITH PARSER MyExampleParser()". The calls may also take keyword arguments; for example, "WITH FILTER MyExampleFilter(argument="value", integer_argument= 12345)". Arguments are passed to the user-defined logic, which may use them to inform how the user-defined logic performs the computation that the user-defined logic is performing.

[0044] In any regard, and according to an example, because the user-defined logic is implemented by the data loading manager 210, which may also manage the database, the loading operation may substantially be optimized, for

instance, by the data loading optimization module 222. More particularly, for instance, the data loading optimization module 222 may obtain information relevant to the loading of the data from the source 130, such as the number of sources 130 from which the data is to be loaded, the number of database machines 112a-112n available for receiving the data, the locations of the sources 130, the locations of the database machines 112a-112n, the type of data to be transferred, the size of the data to be transferred, the content of the data, the types of filters to be applied to the data, etc. The data loading optimization module 222 may obtain the information relevant to the loading of the data as part of the receipt of the user-defined logic.

[0045] In addition, the data loading optimization module 222 may determine a data loading operation that substantially optimizes the loading of the data based upon the information relevant to the loading of the data. The substantially optimized data loading operation may be determined through implementation of any of a variety of different types of optimizers on the information relevant to the loading of the data. For instance, a rule-based optimizer may be implemented that determines the substantially optimized data loading operation based upon whether the information relevant to the data loading indicates that predetermined conditions have been met. In this example, the rule-based optimizer may be programmed with instructions that indicate that certain data loading operations are to be implemented under certain conditions.

[0046] As another example, a cost-based optimizer may be implemented that determines the data loading operation based upon a determination as to which data loading implementation results in the lowest cost. For instance, the cost-based optimizer may run predictive models on various data loading operation scenarios, determine costs associated with each of the data loading operation scenarios, and may identify the data loading operation scenario that results in the lowest cost to transfer the data. The cost in this example may comprise any of, for instance, resource utilization, energy utilization, bandwidth consumption, etc.

[0047] As a further example, a machine-learning optimizer that employs historical information to determine the data loading operation may be implemented. In this example, the machine-learning optimizer may take as inputs various historical information pertaining to the loading of data onto the database machines 112a-112n and may use the historical information to determine, based upon the information relevant to the current loading of data, the data loading operation that substantially optimizes the current loading of the data. Any of a number of suitable machine-learning optimizers may be employed in this example.

[0048] Regardless of the type of optimizer implemented, any of a number of various types of data loading operations may be employed to substantially optimize the transfer of the data. In addition, various combinations of types of data loading operations may be employed to substantially optimize the loading of the data.

[0049] According to an example, the user-defined logic is implemented to stream the data from the source 130, to filter the data as the data is being streamed, to parse the streamed data, and to load the parsed data onto the database. In this example, a relatively large data file may be loaded onto the database without first storing the data file locally. Alternatively, however, the data may first be stored locally from the source 130 prior to performance of the filtering, parsing, and loading operations. Various operations that the data loading manager 210 may implement in loading the data onto the database from the source 130 are described in greater detail with respect to the method 400 depicted in FIG. 4.

[0050] Turning now to FIG. 4, at block 402, an interface through which a user is to define logic related to the loading of the data onto the database is provided, for instance, by the interface providing module 214. Block 402 is similar to block 302 discussed above with respect to FIG. 3. In addition, at block 404, the user-defined logic is received, for instance, by the user-defined logic receiving module 216, which is similar to block 304 discussed above with respect to FIG. 3.

[0051] At block 406, the user-defined logic is implemented to retrieve the data from the source 130, for instance, by the user-defined logic implementation

module 218. Particularly, the user-defined logic pertaining to a user-defined location identification of the source 130 is used to retrieve the data from the source 130. Thus, for instance, the user-defined logic implementation module 218 accesses the data based upon the user-defined location identification associated with the location of the source 130 and a particular file location on the source 130. As discussed above, the user-defined logic may include various information that enable the data loading manager 210 to access and retrieve the data stored on the source 130, such as passwords, security keys, etc. The user-defined logic implementation module 218 may also use the various information to access and retrieve the data from the source 130. As also discussed above, block 406 may be implemented at a deferred time from the time at which the user-defined logic is received at block 404, for instance, during a query time.

[0052] At block 408, the user-defined logic is implemented to filter the data retrieved from the source 130, for instance, by the user-defined logic implementation module 218. As discussed above, the user may define at least one filter to be applied onto the data, for instance, to process the data to be in a better or a more compatible form for loading onto the database. The filter may include, for instance, at least one of a filter to decompress the data, a filter to compress the data, a filter to convert the data from one format to another, etc.

[0053] In an example in which the data stored on the source 130 has been compressed using a particular compression scheme, the user-defined logic may define a filter that is to decompress the data. Thus, for instance, if the data has been compressed using a zip compression scheme (e.g., BZIP, GZIP, 7-ZIP, etc.), the user may define the logic to decompress the data using an appropriate decompression scheme.

[0054] As another example, in which the data stored on the source 130 is in a first format, the user-defined logic implementation module 218 may implement a filter that converts the data from the first format to a second format, as defined by the user. In this example, the user may define a filter that converts the data from

the first format, for instance, a native format, to the second format, for instance, a Native Varchar (NChar) format. In defining the filter, the user may specify that a particular converter program be executed to perform the conversion of the data. In one example, the particular converter program may comprise a previously defined program that the user may specify by an identification of the particular converter program. In this example, the user-defined logic implementation module 218 may be able to access the previously defined program, either from a local library or from a remote location. In any regard, the second format may comprise a format from which the data may appropriately be formatted for loading onto the database. By way of particular example, the filter may be defined to convert the data from a first format to a second format that is readily converted to a format suitable for being parsed into a format in which data is stored in the database.

[0055] At block 410, the data is parsed to convert the filtered data into an appropriate or compatible format for the database, for instance, by the user-defined logic implementation module 218. The parsing operation performed on the filtered data may be user-defined, i.e., the user may define the particular parsing operation to be performed on the filtered data. Alternatively, the parsing operation may be predefined in the user-defined logic implementation module 218 because the appropriate or compatible format for the database may already be known.

[0056] At block 412, the parsed data is loaded onto the database, for instance, by the data loading module 224. According to an example, the data loading module 224 loads the parsed data directly onto a particular database machine 112a. In another example, the data loading module 224 loads the parsed data onto multiple database machines 112a-1 12n according to a data loading optimization operation, for instance, as determined by the data loading optimization module 222. Various manners in which the data loading optimization module 222 may substantially optimize the loading of the data onto multiple ones of the database machines 112a-1 12n are described in greater detail herein above with respect to the method 300. The loading of the data may be substantially optimized

by at least one of segmenting and/or sorting the data and by storing the data in the segmented and/or sorted manner on multiple ones of the database machines 112a-1 12n.

[0057] According to an example, the loading of the data may also substantially be optimized through implementation of the method 400 on multiple ones of the database machines 112a-1 12n, to thereby distribute the workload associated with loading the data. Thus, for instance, a plurality of the database machines 112a-1 12n may retrieve different portions of the data and may concurrently filter, parse, and load the data. As another example, one of the database machines 112a may retrieve the data from the source 130, another one of the database machines 112b may filter the data, and another one of the database machines 112c may parse and load the data. The implementation of the database machines 112a-1 12n that substantially optimizes the loading of the data onto the database may be determined through performance of any suitable optimization technique.

[0058] According to an example, blocks 406-412 are implemented while the data is streamed from the source 130. In this example, therefore, relatively small portions of the data are processed at a given time, thereby, substantially minimizing data storage requirements of the database machines 112a-1 12n. In addition, an input buffer and an output buffer may be implemented, in which the input buffer contains some input from the stream of data and the output buffer is used to write process data from the input buffer.

[0059] In another example, at block 406, all of the data is copied from the source 130 and saved, for instance, in the data store 204. In this example, blocks 408-412 are implemented on the stored copy of the data, which therefore requires that the database machines 112a-1 12n have relatively larger data storage capabilities.

[0060] At block 414, the data loaded onto the database is managed, for instance, by the user-defined logic implementation module 218. As discussed

above, the data loaded onto the database may be managed, for instance, to control the length of time during which the data is stored in the database, conditions under which the data is to transferred to another storage, etc.

[0061] Some or all of the operations set forth in the methods 300 and 400 may be contained as a utility, program, or subprogram, in any desired computer accessible medium. In addition, the methods 300 and 400 may be embodied by computer programs, which may exist in a variety of forms both active and inactive. For example, they may exist as machine readable instructions, including source code, object code, executable code or other formats. Any of the above may be embodied on a non-transitory computer readable storage medium. Examples of non-transitory computer readable storage media include conventional computer system RAM, ROM, EPROM, EEPROM, and magnetic or optical disks or tapes. It is therefore to be understood that any electronic device capable of executing the above-described functions may perform those functions enumerated above.

[0062] Turning now to FIG. 5, there is shown a schematic representation of a computing device 500, which may be employed to perform various functions of the database machines 112a-112n, 200 respectively depicted in FIGS. 1 and 2, according to an example. The computing device 500 includes a processor 502, such as but not limited to a central processing unit; a display device 504, such as but not limited to a monitor; a network interface 508, such as but not limited to a Local Area Network LAN, a wireless 802.11 LAN, a 3G/4G mobile WAN or a WiMax WAN; and a computer-readable medium 510. Each of these components is operatively coupled to a bus 512. For example, the bus 512 may be an EISA, a PCI, a USB, a FireWire, a NuBus, or a PDS.

[0063] The computer readable medium 510 comprises any suitable medium that participates in providing instructions to the processor 502 for execution. For example, the computer readable medium 510 may be non-volatile media, such as memory. The computer-readable medium 510 may also store an operating system 514, such as but not limited to Mac OS, MS Windows, Unix, or

Linux; network applications 516; and a data loading management application 518. The operating system 514 may be multi-user, multiprocessing, multitasking, multithreading, real-time, and the like. The operating system 514 may also perform basic tasks such as but not limited to recognizing input from input devices, such as but not limited to a keyboard or a keypad; sending output to the display 504; keeping track of files and directories on medium 510; controlling peripheral devices, such as but not limited to disk drives, printers, image capture device; and managing traffic on the bus 512. The network applications 516 include various components for establishing and maintaining network connections, such as but not limited to machine readable instructions for implementing communication protocols including TCP/IP, HTTP, Ethernet, USB, and FireWire.

[0064] The data loading management application 518 provides various components for managing loading of data from a source to a database system as discussed above with respect to the methods 300 and 400 in FIGS. 3 and 4. The data loading management application 518 may thus comprise the input/output module 212, the interface providing module 214, the user-defined logic receiving module 216, the user-defined logic implementation module 218, the predefined load stack implementation module 220, the data loading optimization module 222, and the data loading module 224. In this regard, the data loading management application 518 may include modules for performing the methods 300 and/or 400.

[0065] In certain examples, some or all of the processes performed by the application 518 may be integrated into the operating system 514. In certain examples, the processes may be at least partially implemented in digital electronic circuitry, or in computer hardware, machine readable instructions (including firmware and software), or in any combination thereof, as also discussed above.

[0066] What has been described and illustrated herein are examples of the disclosure along with some variations. The terms, descriptions and figures used herein are set forth by way of illustration only and are not meant as limitations. Many variations are possible within the scope of the disclosure, which is intended

to be defined by the following claims - and their equivalents - in which all terms are meant in their broadest reasonable sense unless otherwise indicated.

CLAIMS

What is claimed is:

1. A method for managing loading of data from a source onto a database, said method comprising:

providing an interface through which a user is to define logic related to the loading of the data onto the database, wherein the user-defined logic pertains to a user-defined location identification of the source, a user-defined filter to be applied on the data, and a parsing operation to be performed on the data to convert the data into an appropriate format for the database;

receiving the user-defined logic; and

implementing, by a processor, the user-defined logic to load the data onto the database.

2. The method according to claim 1, wherein providing the interface further comprises providing the interface through which the user is to define the logic by programing the logic, said method further comprising:

storing the programmed logic in a library.

3. The method according to claim 1, wherein providing the interface further comprises providing access to a library on which is stored previously defined logic through the interface, and wherein receiving the user-defined logic further comprises receiving previously stored logic from the library.

4. The method according claim 1, wherein the user-defined filter to be applied on the data is to at least one of uncompress, compress, and convert the data.

5. The method according to claim 1, wherein implementing the user-defined logic further comprises implementing the user-defined logic as part of a pre-configured data-load stack.

6. The method according to claim 1, wherein implementing the user-defined logic further comprises:

retrieving the data from the source using the user-defined location identification of the source;

filtering the data using the user-defined filter;

parsing the data through performance of the parsing operation; and

loading the parsed data onto the database.

7. The method according to claim 6, wherein retrieving, filtering, parsing, and loading the parsed data further comprises retrieving, filtering, parsing, and loading the parsed data as the data is streamed from the source.

8. The method according to claim 1, wherein the user-defined logic further pertains to a user-defined policy that defines a condition to be met prior to the data being retrieved from the source.

9. The method according to claim 1, wherein the user-defined logic further pertains to a user-defined policy that defines how data loaded onto the database is to be managed.

10. The method according to claim 1, wherein implementing the user-defined logic further comprises implementing the user-defined logic in a system that manages the database, wherein the system that manages the database comprises a distributed database system and wherein implementing the user-defined logic further comprises implementing the user-defined logic on a plurality of the systems

in the distributed database system while implementing a database optimization operation.

11. An apparatus for managing loading of data from a source onto a database, said apparatus comprising:

at least one module to,

provide an interface through which a user is to define functions related to the loading of the data onto the database, wherein the user-defined functions pertain to a user-defined location identification of the source, a user-defined filter to be applied on the data, and a parsing operation to be performed on the data to convert the data into a format that is compatible with the database;

receive the user-defined functions; and

implement the user-defined functions to load the data onto the database;

a processor to implement the at least one module.

12. The apparatus according to claim 11, wherein the user-defined logic further pertains to a user-defined policy that defines at least one of a condition to be met prior to the data being retrieved from the source and management of the data loaded onto the database, and wherein the at least one module is further to:

retrieve the data from the source using the user-defined location identification of the source according to the user-defined policy;

filter the data using the user-defined filter;

parse the data through performance of the parsing operation; and

load the parsed data onto the database.

13. The apparatus according to claim 11, wherein the apparatus comprises a database management apparatus.

14. The apparatus according to claim 11, wherein the database is a relational database.

15. A non-transitory computer readable storage medium on which is stored machine readable instructions that when executed by a processor, implement a method for managing loading of data from a source onto a database, said machine readable instructions comprising code to:

provide an interface through which a user is to define code related to the loading of the data onto the database, wherein the user-defined code pertains to a user-defined location identification of the source, a user-defined filter to be applied on the data, a user-defined policy that defines at least one of a condition to be met prior to the data being retrieved from the source and management of the data loaded onto the database, and a parsing operation to be performed on the data to convert the data into format that is compatible with the database;

receive the user-defined code;

implement the user-defined code to transform the data into a format suitable for loading the data onto the database; and

implementing the user-defined code to load the data onto the database.

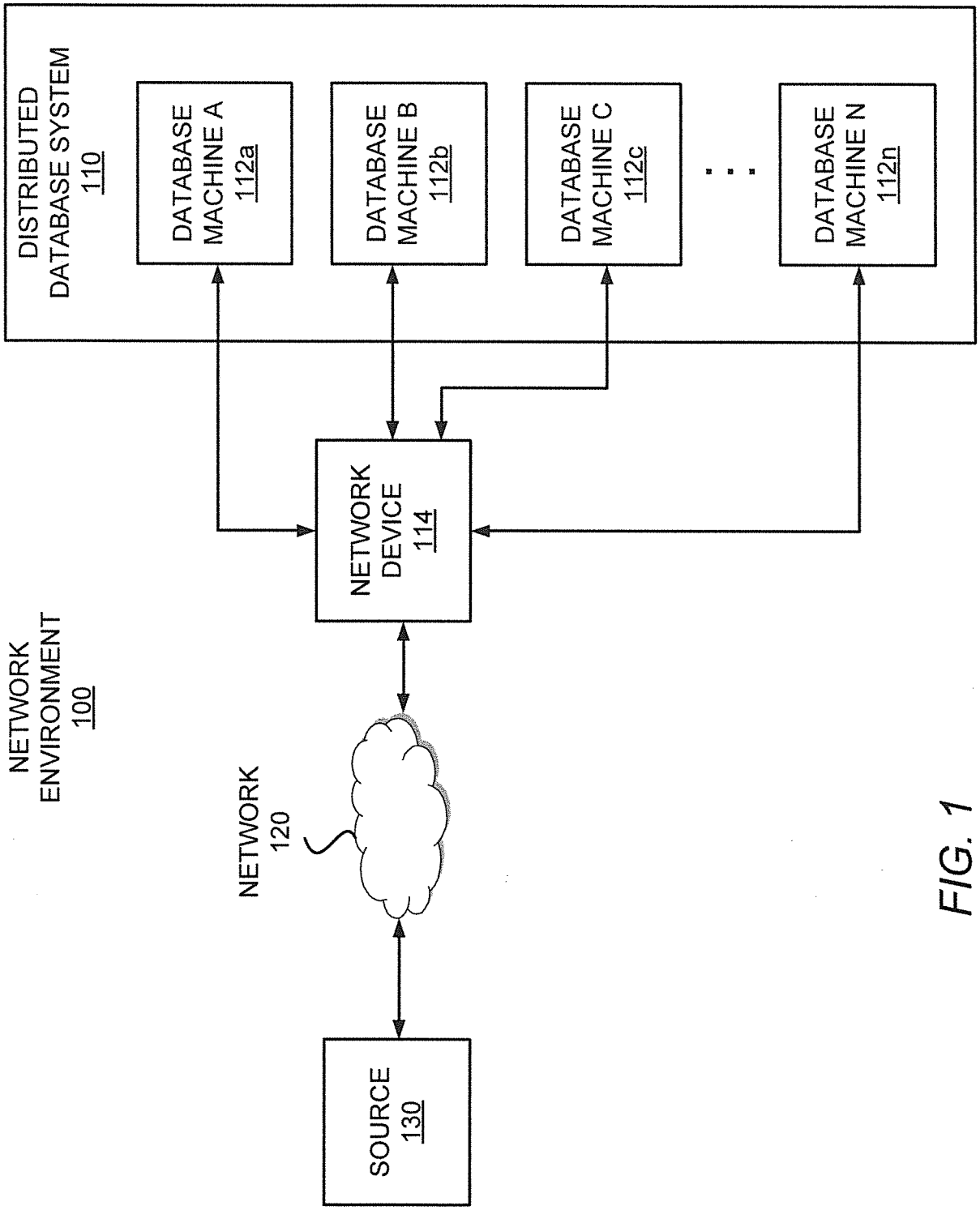
NETWORK ENVIRONMENT 100
DISTRIBUTED DATABASE SYSTEM 110
DATABASE MACHINE A 112a
DATABASE MACHINE B 112b
DATABASE MACHINE C 112c
DATABASE MACHINE N 112n
NETWORK DEVICE 114
NETWORK 120
SOURCE 130

FIG. 1

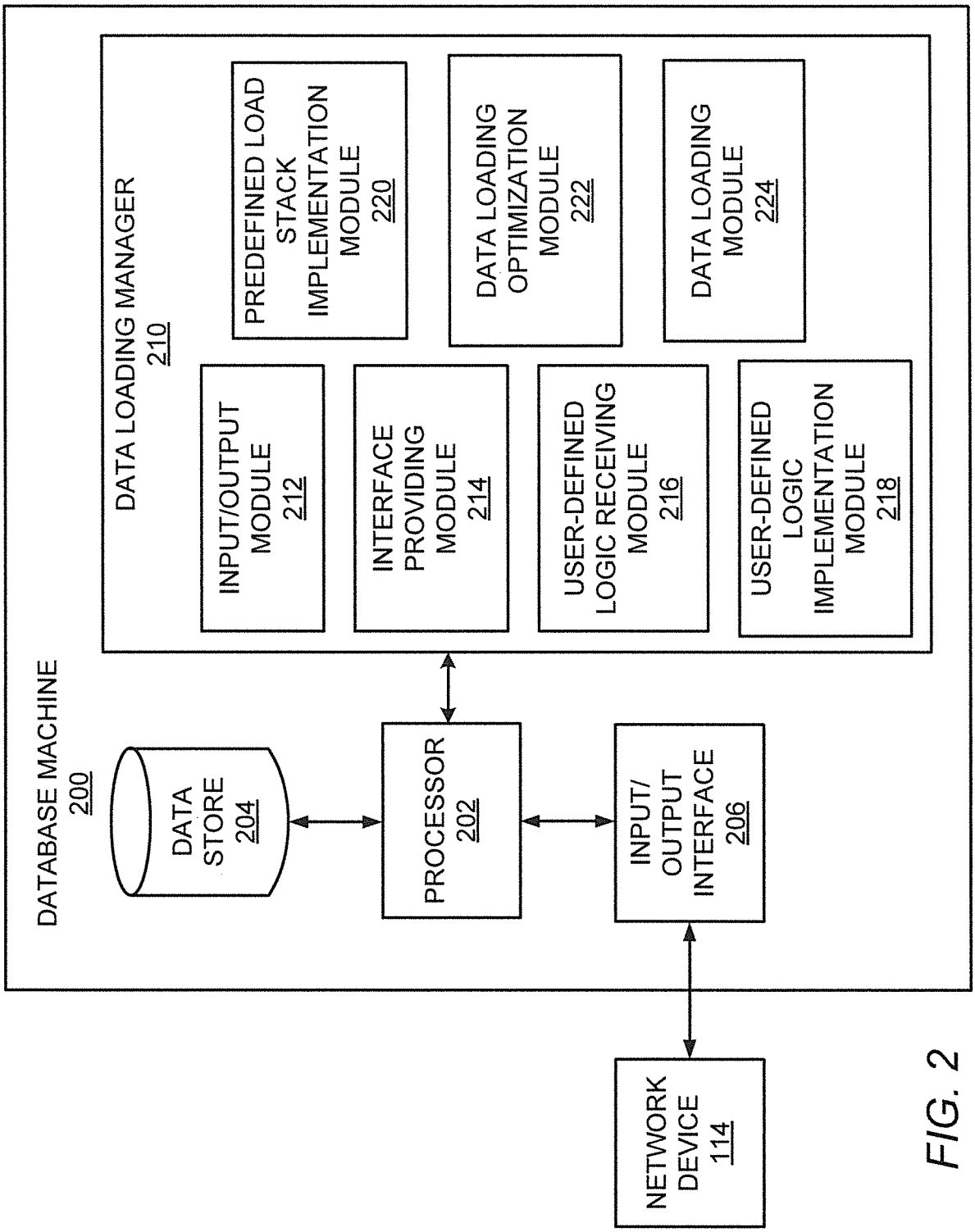
DATABASE MACHINE
200
DATA LOADING MANAGER
210
INPUT/OUTPUT MODULE
212
INTERFACE PROVIDING MODULE
214
USER-DEFINED LOGIC RECEIVING MODULE
216
USER-DEFINED LOGIC IMPLEMENTATION MODULE
218
PREDEFINED LOAD STACK IMPLEMENTATION MODULE
220
DATA LOADING OPTIMIZATION MODULE
222
DATA LOADING MODULE
224
DATA STORE
204
PROCESSOR
202
INPUT/ OUTPUT INTERFACE
206
NETWORK DEVICE
114

FIG. 2

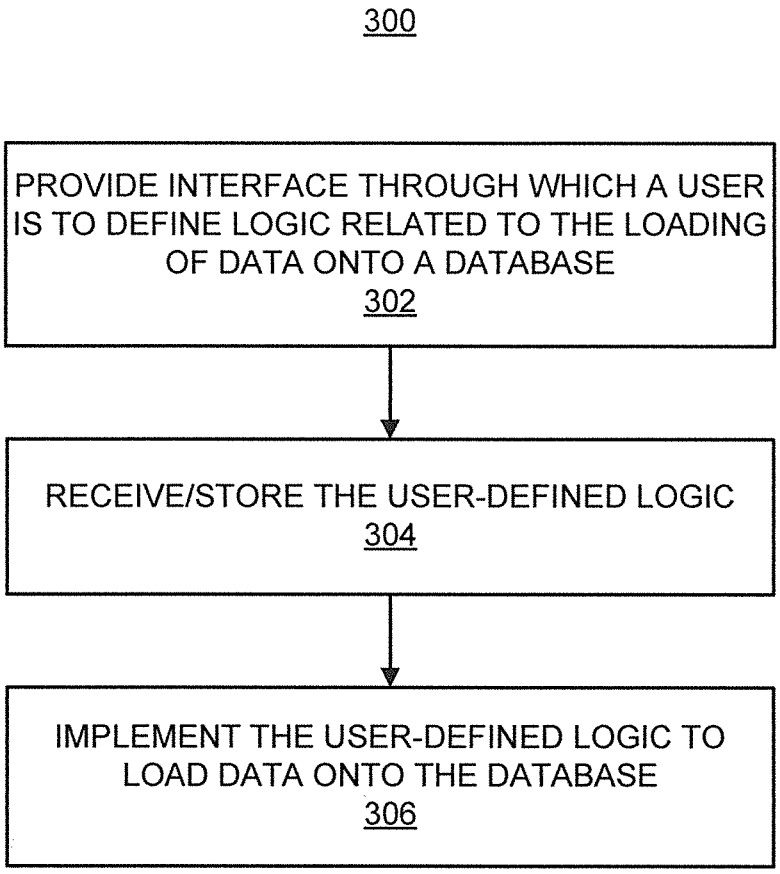
300
PROVIDE INTERFACE THROUGH WHICH A USER IS TO DEFINE LOGIC RELATED TO THE LOADING OF DATA ONTO A DATABASE
302
RECEIVE/STORE THE USER-DEFINED LOGIC
304
IMPLEMENT THE USER-DEFINED LOGIC TO LOAD DATA ONTO THE DATABASE
306

FIG. 3

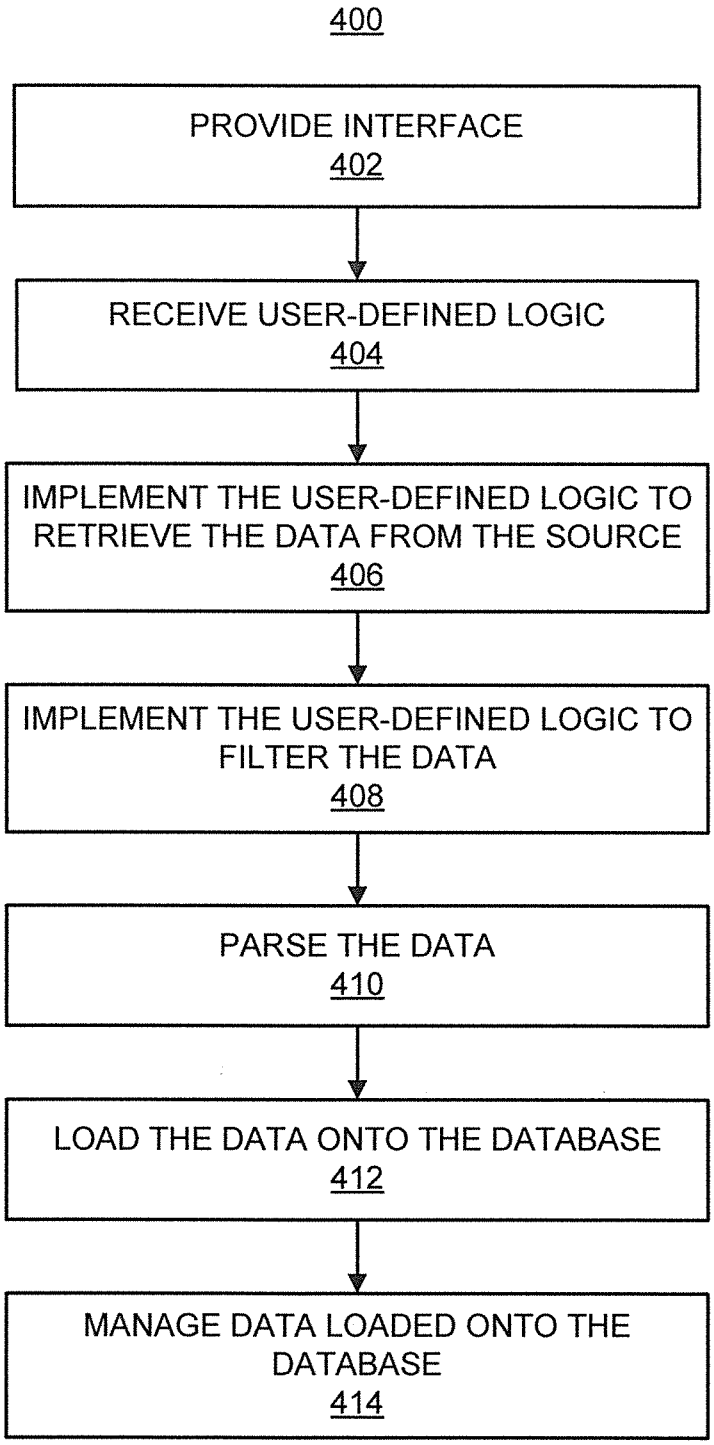
400
PROVIDE INTERFACE
402
RECEIVE USER-DEFINED LOGIC
404
IMPLEMENT THE USER-DEFINED LOGIC TO RETRIEVE THE DATA FROM THE SOURCE
406
IMPLEMENT THE USER-DEFINED LOGIC TO FILTER THE DATA
408
PARSE THE DATA
410
LOAD THE DATA ONTO THE DATABASE
412
MANAGE DATA LOADED ONTO THE DATABASE
414

FIG. 4

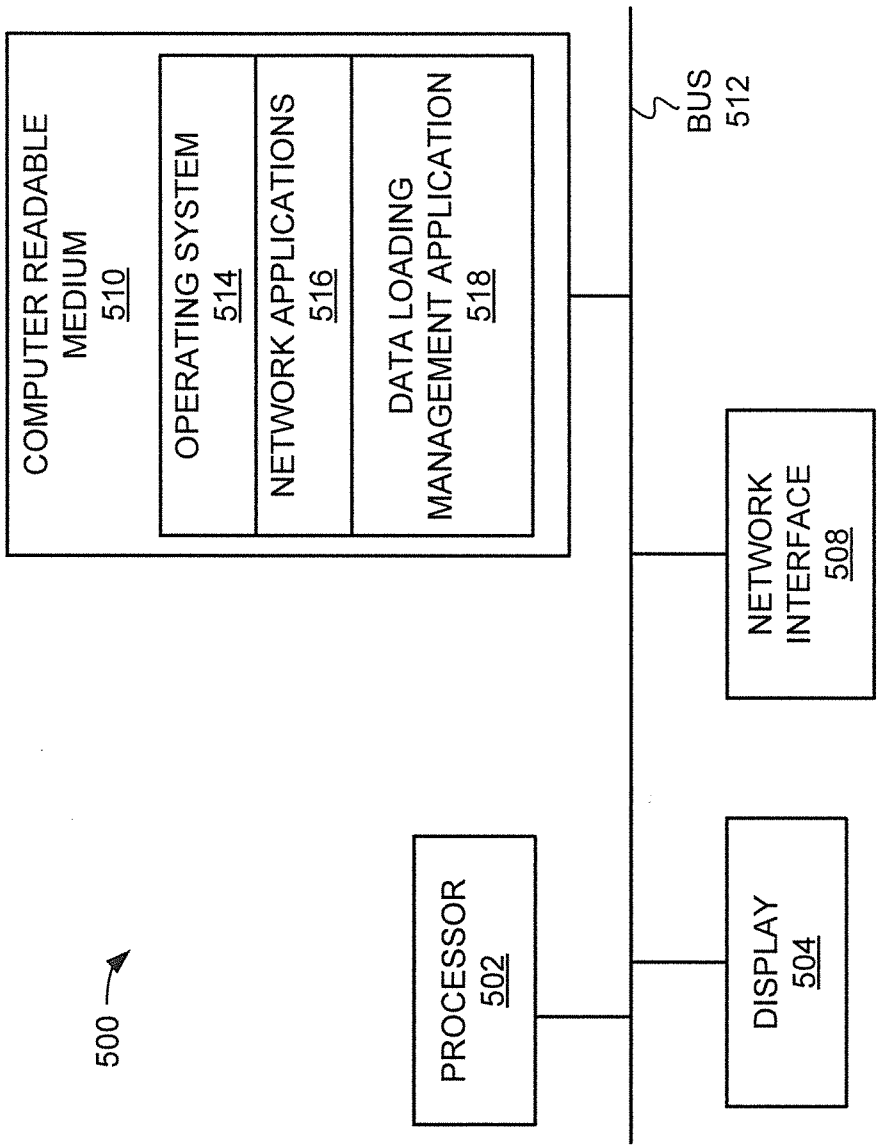
500
PROCESSOR
502
DISPLAY
504
NETWORK
INTERFACE
508
BUS
512
COMPUTER READABLE
MEDIUM
510
OPERATING SYSTEM
514
NETWORK APPLICATIONS
516
DATA LOADING
MANAGEMENT APPLICATION
518

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2012/040764

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/30(2006.01)i, G06F 15/16(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 7/00; G06Q 40/00; G06F 15/16

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: loading, data, source, database, location identification, IP address, filter, parse, convert, format, user-defined

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2010-0023546 Al (JOSEPH M. SILSBY) 28 January 2010 See paragraphs [0002] - [0010] , [0014]- [0015] , [0026H0040] , [0051H0052] ; and claims 1-2 , 7-8 , 13 .	1-15
A	US 2011-0302226 Al (DANIEL ABADI et al.) 08 December 2011 See paragraphs [0002] - [0020] , [0034] , [0040] , [0050H0051] , [0092] -[0118] ; and claims 1-3 .	1-15
A	US 2009-0319550 Al (JAMES SHAU et al.) 24 December 2009 See paragraphs [0002] - [0005] , [0043]- [0044] , [0057] -[0059] ; and claim 1.	1-15
A	US 2009-0222361 Al (GERALD f. SMITH et al.) 03 September 2009 See paragraphs [0001] - [0012] , [0029]- [0030] , [0046] ; and claim 1.	1-15
A	US 2009-0299987 Al (IAN ALEXANDER WILLSON) 03 December 2009 See abstract ; paragraphs [0002] -[0008] , [0026] - [0027] , [0035] - [0036] , [0107]- [0110] ; and claim 1.	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 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	Date of mailing of the international search report
08 August 2013 (08.08.2013)	**12 August 2013 (12.08.2013)**

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
BYUN Sung Cheal
Telephone No. +82-42-481-8262

Form PCT/ISA/210 (second sheet) (July 2009)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2012/040764

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2010-0023546 Al	28/01/2010	US 8401990 B2	19/03/2013
US 2011-0302226 Al	08/12/2011	EP 2577512 A2	10/04/2013
		WO 2011-153239 A2	08/12/2011
		Wo 2011-153239 A3	16/05/2013
US 2009-0319550 Al	24/12/2009	us 2011-0099155 Al	28/04/2011
		us 7895151 B2	22/02/2011
		us 8165988 B2	24/04/2012
US 2009-0222361 Al	03/09/2009	AU 2009-200674 Al	17/09/2009
		AU 2009-200674 B2	18/04/2013
		CA 2648611 Al	29/08/2009
		US 8473377 B2	25/06/2013
US 2009-0299987 Al	03/12/2009	GB 0909482 DO	15/07/2009
		GB 2460532 A	09/12/2009
		US 2012-0150791 Al	14/06/2012
		US 8271430 B2	18/09/2012
		wo 2012-138437 Al	11/10/2012