

(19) **DANMARK**

(10) **DK/EP 2921975 T3**



(12)

Oversættelse af europæisk patentskrift

Patent- og
Varemærkestyrelsen

-
- (51) Int.Cl.: **G 06 F 16/178 (2019.01)**
- (45) Oversættelsen bekendtgjort den: **2019-10-28**
- (80) Dato for Den Europæiske Patentmyndigheds
bekendtgørelse om meddelelse af patentet: **2019-08-21**
- (86) Europæisk ansøgning nr.: **15159520.4**
- (86) Europæisk indleveringsdag: **2015-03-17**
- (87) Den europæiske ansøgnings publiceringsdag: **2015-09-23**
- (30) Prioritet: **2014-03-18 US 201461955054 P** **2014-04-16 US 201414254773**
2014-04-16 US 201414254757
- (84) Designerede stater: **AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV
MC MK MT NL NO PL PT RO RS SE SI SK SM TR**
- (73) Patenthaver: **Palantir Technologies Inc., 100 Hamilton Avenue, Palo Alto, CA 94301, USA**
- (72) Opfinder: **Fisher, William, 100 Hamilton Avenue, Palo Alto, CA California 94301, USA**
Maag, Peter, 100 Hamilton Avenue, Palo Alto, CA California 94301, USA
- (74) Fuldmægtig i Danmark: **Zacco Denmark A/S, Arne Jacobsens Allé 15, 2300 København S, Danmark**
- (54) Benævnelse: **Bestemmelse og ekstrahering af ændrede data fra en datakilde**
- (56) Fremdragne publikationer:
WO-A1-2013/030595

DESCRIPTION

PRIORITY APPLICATIONS

[0001] This application claims the benefit of U.S. Application No. 14/254,757, filed April 16, 2014, U.S. Application No. 14/254,773, filed April 16, 2014, and U.S. Provisional Application No. 61/955054, filed March 18, 2014.

TECHNICAL FIELD

[0002] The present disclosure relates to systems and techniques for data integration and analysis. More specifically, the present disclosure relates to identifying changes in the data of a data source.

BACKGROUND

[0003] Organizations and/or companies are producing increasingly large amounts of data. Such data may be stored in different data sources. Data sources may be updated, e.g., periodically.

[0004] WO 2013/030595 A1 discloses a system for identifying data items for the purpose of synchronizing the data items between a plurality of devices based on an allocation of an update token that is superseded by a successive update token.

SUMMARY

[0005] The invention is defined by the independent claims, taking due account of any element which is equivalent to an element specified in the claims. The dependent claims concern optional features of some embodiments of the invention.

[0006] The systems, methods, and devices described herein each have several aspects, no single one of which is solely responsible for its desirable attributes. Without limiting the scope of this disclosure, several non-limiting features will now be discussed briefly.

[0007] In one embodiment, a computer system configured to obtain changed data from a data source comprises: one or more hardware computer processors configured to execute code in order to cause the system to: obtain information indicating a plurality of groupings of data stored in one or more files or databases in a data source, the information indicating a number

of data items included in each of the plurality of groupings; determine a first grouping of the plurality of groupings including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical first number of data items included in a corresponding local version of the first grouping, wherein the corresponding local version of the first grouping is created based on data items included in the first grouping at a first time prior to said obtaining the information indicating the plurality of groupings of the data; access data items included in the first grouping from the data source; compare the data items included in the first grouping to data items of the corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed; extract the changed data items of the first grouping; and forward the extracted changed data items to a destination system.

[0008] In another embodiment, a method of obtaining changed data from a data source comprises: obtaining, by one or more hardware computer processors, information indicating a plurality of groupings of data stored in one or more files or databases in a data source, the information indicating a number of data items included in each of the plurality of groupings; determining, by the one or more hardware computer processors, a first grouping of the plurality of groupings including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical first number of data items included in a corresponding local version of the first grouping, wherein the corresponding local version of the first grouping is created based on data items included in the first grouping at a first time prior to said obtaining the information indicating the plurality of groupings of the data; accessing, by the one or more hardware computer processors, data items included in the first grouping from the data source; comparing, by the one or more hardware computer processors, the data items included in the first grouping to data items of the corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed; extracting, by the one or more hardware computer processors, the changed data items of the first grouping; and forwarding, by the one or more hardware computer processors, the extracted changed data items to a destination system.

[0009] In yet another embodiment, a transitory or non-transitory computer readable medium comprises instructions for obtaining changed data from a data source that cause a computer processor to: obtain information indicating a plurality of groupings of data stored in one or more files or databases in a data source, the information indicating a number of data items included in each of the plurality of groupings; determine a first grouping of the plurality of groupings including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical first number of data items included in a corresponding local version of the first grouping, wherein the corresponding local version of the first grouping is created based on data items included in the first grouping at a first time prior to said obtaining the information indicating the plurality of groupings of the data; access data items included in the first grouping from the data source; compare the data items included in the first grouping to data items of the corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed; extract the changed data items of the first grouping; and forward the extracted changed data items to

a destination system.

[0010] In some embodiments, a computer system configured to obtain changed data from a data source comprises: one or more hardware computer processors configured to execute code in order to cause the system to: obtain information indicating a plurality of groupings of data of a data source, the information indicating a number of data items included in each of the plurality of groupings; determine a first grouping of the plurality of groupings including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical number of data items included in each of the plurality of groupings; access data items included in the first grouping from the data source; compare the data items included in the first grouping to data items of a corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed, wherein the corresponding local version of the first grouping of data items is a compressed version of the first grouping of data items; extract the changed data items of the first grouping; and forward the extracted changed data items to a destination system.

[0011] In certain embodiments, a method of obtaining changed data from a data source comprises: obtaining, by one or more hardware computer processors, information indicating a plurality of groupings of data of a data source, the information indicating a number of data items included in each of the plurality of groupings; determining, by the one or more hardware computer processors, a first grouping of the plurality of groupings including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical number of data items included in each of the plurality of groupings; accessing, by the one or more hardware computer processors, data items included in the first grouping from the data source; comparing, by the one or more hardware computer processors, the data items included in the first grouping to data items of a corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed, wherein the corresponding local version of the first grouping of data items is a compressed version of the first grouping of data items; extracting, by the one or more hardware computer processors, the changed data items of the first grouping; and forwarding, by the one or more hardware computer processors, the extracted changed data items to a destination system.

[0012] In other embodiments, a transitory or non-transitory computer readable medium comprises instructions for obtaining changed data from a data source that cause a computer processor to: obtain information indicating a plurality of groupings of data of a data source, the information indicating a number of data items included in each of the plurality of groupings; determine a first grouping of the plurality of groupings including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical number of data items included in each of the plurality of groupings; access data items included in the first grouping from the data source; compare the data items included in the first grouping to data items of a corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed, wherein the corresponding local version of the first grouping of data items is a compressed version of the first grouping of

data items; extract the changed data items of the first grouping; and forward the extracted changed data items to a destination system.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013]

Figure 1 is a block diagram illustrating one embodiment of a change determination system configured to determine and obtain changes in data of a plurality of data sources.

Figure 2A is a data flow diagram illustrative of the interaction between the various components of a change determination system configured to determine and obtain changes in data of a plurality of data sources, according to one embodiment.

Figure 2B is a data flow diagram illustrative of the interaction between the various components of a change determination system configured to determine and obtain changes in data of a plurality of data sources, according to another embodiment.

Figure 3 is an example of information obtained from a data source and/or information processed by the change determination system.

Figure 4A is a flowchart illustrating one embodiment of a process for determining and obtaining changes in data of a plurality of data sources.

Figure 4B is a flowchart illustrating another embodiment of a process for determining and obtaining changes in data of a plurality of data sources.

Figure 5 is a block diagram illustrating a computer system with which certain methods discussed herein may be implemented.

DETAILED DESCRIPTION

Overview

[0014] Organizations may need to obtain data from one or more data sources. Often, data for a particular timeframe is downloaded from a data source. For example, a data source may contain log files, and log files for the past two days may be downloaded. However, some of the data may already have been obtained at a previous time, and the system that is requesting the data may not be able to distinguish between data that it already has and new or changed data that it has not yet been obtained. For instance, the requesting system may simply store the data it downloads each time without considering whether some data is duplicated. Some data

may be downloaded again even though it already exists in the system. Accordingly, there is a need for identifying and extracting changed data from a data source in an efficient manner.

[0015] As disclosed herein, a change determination system may be configured to identify and obtain changes in data from one or more data sources. For example, the system can determine that there are changes to the data of a data source (e.g., the data for a particular timeframe, such as a day) based on some summary information for a current set of data from the data source (e.g., lines of data associated with a particular day in a previously received data set) compared to a current set of data from the data source (e.g., lines of data associated with the particular day in a current data set). Once pieces of data with changes are identified (e.g., days with different amounts of lines of data), the changed data may be obtained and compared to a local version of the data (or some representation of the data) in order to identify the particular data items (e.g., particular lines of data) that have changed, such that only those particular data items need be provided by the data source.

[0016] A data source can be one or more databases and/or one or more files. The actual changes can be forwarded to a destination system for storage. The change determination system can act as an intermediary between data sources and one or more destination systems to identify changed data and forward only the changed data to the destination systems.

[0017] It may take a lot of time to download from a data source (e.g., due to slow speed, amount of data, etc.), and re-downloading data that already exists in the destination system can lead to spending unnecessary time and resources. Moreover, saving duplicate data can take up unnecessary storage space in the destination system. By identifying and forwarding only the changed data, the change determination system can provide a way to obtain data from a data source in an intelligent manner and can save time and/or resources for the destination system. This can be very helpful especially when a data source contains large amounts of data, and only a small portion of the data has changed. The change determination system can also identify the changes quickly, for example, by performing a grouping operation on the data explained in detail below.

Change Determination System

[0018] Figure 1 is a block diagram illustrating one embodiment of a change determination system 100 configured to determine and obtain changes in data of a plurality of data sources 110. A data source 110 can include one or more databases 110a, one or more files 110b (e.g., flat file or file system), any other type of data structure, or a combination of multiple data structures. Data in the database 110a may be organized into one or more tables, which include rows and columns. Data in the file 110b may be organized as lines with various fields. For example, a file 110b may be in CSV format. Data in files, databases, or other data structure, may be referred to in terms of "lines," where a line is a subset of the file. For example, lines of a file may be groups of data between newline markers or divisions of text of the file into predetermined size groups (e.g., each line includes 255 characters) and lines of a database

may be a row or some other subset of information in the database. Data in a database 110a and a file 110b may be handled or processed in a similar manner by the change determination system ("CDS") 100. In certain embodiments, the CDS 100 obtains changes from a single data source 110, instead of multiple data sources 110.

[0019] The CDS 100 may include one or more components (not shown) that perform functions relating to determining and obtaining changes in the data of data sources 110. The CDS 100 may also include local storage 150, which can store any local version of the data in a data source 110, such as a summarized and/or compressed version of the data. The local version of the data can be used to identify the actual changes in the data of the data source 210 and/or portions of the data (e.g., lines of the data) that include changes. The local version may include some or all of the data of a data source, depending on the embodiment.

[0020] One or more destination systems 270 may request changed data from the CDS 100. The destination systems 270 may send a request on a periodic basis (e.g., scheduled), on demand, etc. The CDS 100 may also periodically check a data source(s) 210 and forward any changes without receiving a request from a destination system 270. For example, the CDS 100 may be scheduled to check the data sources 210 every 2 hours. Such a schedule may be defined as one or more policies. In one embodiment, a destination system 270 includes the CDS 100, such that the functionality described herein with reference to the CDS 100 may be performed by the destination system 270 itself.

[0021] Figure 2A is a data flow diagram illustrative of the interaction between the various components of a change determination system 200a configured to determine and obtain changes in data of a plurality of data sources, according to one embodiment. The CDS 200a and corresponding components of Figure 2A may be similar to or the same as the CDS 100 and similarly named components of Figure 1.

[0022] At data flow action 1, the CDS 200a performs a query on the data in a data source 210 to group by a particular attribute, such as a column of information in a table. For purposes of discussion herein, many examples are discussed with reference to grouping based on one or more "columns," where each column is associated with a particular attribute. In other embodiments, attributes may be associated with different display features of a data structure (e.g., besides columns). For example, if the data source 210 is a database 210a, the CDS 200a can perform an SQL query and group by a particular column in a table, e.g., by using the GROUP BY clause, which is used in SQL to group rows having common values into a smaller set of rows. The smaller sets of rows may be referred to as partitions or groups. Each partition includes rows that have the same value for the designated column. GROUP BY is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set.

[0023] The column that is designated as the column for GROUP BY should be able to provide some indication of which rows are new or changed from the previous time the CDS 200a obtained data from the data source 210. In one example, a table in the database 210a includes a last updated column, which includes a timestamp for when the data in the row was

last updated, and the data in the database 210a can be grouped by the last updated column. Since the timestamp can include the hour, minute, second, etc. in addition to the date, only the date of the timestamp might be used for GROUP BY. In such case, a partition would be based on a day, and each partition would contain the rows for each day. An aggregate function such as COUNT can be applied to the results of GROUP BY in order to obtain the number of rows for each partition. The results of the query from the data source 210 can include one or more partitions from the GROUP BY and the number of rows included in each partition. The CDS 200a may store the results or keep track of the results locally so that the results can be compared the next time the CDS 200a requests this type of information from the data source 210.

[0024] Data from a file data source 210b may also be queried in a similar manner. The CDS 200a may use different adapters to access data residing in one or more databases 210a and data residing in one or more files 210b, but once the data is obtained, it can be handled in the same manner by the CDS 200a, regardless of whether the data source is in the form of a database 210a or a file 210b. The details discussed with respect to a database data source 210a can be generalized to other types of data sources 210, including a file data source 210b. For example, a partition can refer to a grouping used on a text (or other field type) resulting from an operation that is similar to SQL GROUP BY. A partition may also be referred to as a "grouping." A grouping may include data from a database 210a or a file 210b. A unit of data included in a grouping may be referred to as a "data item." A data item may be a row in case of a database 210a or a line in case of a file 210b. The column or field to group by can be any type that can provide partitions of appropriate size for comparison (e.g., provide uneven or non-uniform distribution). Some details relating to the group by column/field are explained further below.

[0025] At data flow action 2, the CDS 200a determines which partition(s) have changed. The number of rows for the partitions obtained at data flow action 1 may be compared to the number of rows for corresponding partitions obtained at a previous time. The current number of rows in partitions may be referred to as "current grouping data," and the number of rows for various partitions from a previous time may be referred to as "historical grouping data." The CDS 200a can compare the current grouping data against the historical grouping data to determine whether the number of rows for a particular partition changed. For example, if the number of rows for Day 1 is 1,000 in the historical grouping data, but the number of rows for Day 1 is 1,050 in the current grouping data, the partition for Day 1 is a candidate for checking whether the actual data changed. It is likely that 50 new rows were added for Day 1, and the CDS 200a can determine which rows of 1,050 are new and extract them to forward to a destination system 270. In this manner, the CDS 200a may identify one or more partitions that have changed.

[0026] At data flow action 3, the CDS 200a obtains data for any identified changed partition(s). In particular, once the CDS 200a identifies a changed partition, the CDS 200a obtains the data for the particular partition from the data source 210. In the example above, the CDS 200a requests the 1,050 rows for Day 1. The CDS 200a can store the 1,050 rows locally, e.g., in

local storage 250a. The 1,050 rows can then be used for comparison the next time the data for this partition is changed.

[0027] At data flow action 4, the CDS 200a compares the obtained data with the local version of the changed partition. The CDS 200a can compare the downloaded data against a corresponding local version of the data. For example, the CDS 200a may have stored the 1,000 rows for Day 1 from a previous time in local storage 250a. The data for the partition from a previous time may be referred to as "historical partition data." Similarly, the current data for the partition may be referred to as "current partition data." The local version of the data may include data for one partition or a number of partitions. By comparing the 1,050 current rows for Day 1 in the current partition data to the 1,000 previous rows for Day 1 in the historical partition data, the CDS 200a may identify which 50 rows have changed. These 50 rows can be flagged or put in a file so that they can be sent to any relevant destination system 270.

[0028] At data flow action 5, the CDS 200a forwards the changed data to the destination system 270. As explained above, the changed data may be extracted from the current partition data, saved to a file, and sent to the destination system 270. The destination system 270 may then store the received data in its storage device(s) without having to check whether any duplicate exists in the received data. Because the CDS 210a can send the exact changed data, the destination system 270 can simply store what it receives from the CDS 210a and does not need to implement much functionality at its end.

[0029] In Figure 2A, in order to compare the current partition data to the historical partition data, the CDS 200a may locally maintain all or a subset of the data from a data source 210. However, in some cases, the local storage 250a may have limited storage space, and the CDS 200a may not be able to store all of the data used in the comparison. Accordingly, in such cases, the CDS 200a may maintain a compressed version of the data from a data source 210 for comparison. In some embodiments, the compressed version of the data can be one or more Bloom filters. Such embodiments are described in further detail in connection with Figure 2B.

[0030] Figure 2B is a data flow diagram illustrative of the interaction between the various components of a change determination system 200b configured to determine and obtain changes in data of a plurality of data sources, according to another embodiment. The CDS 200b and corresponding components of Figure 2B may be similar to or the same as the CDS 100, 200a and similarly named components of Figures and 2A.

[0031] Data flow actions 1-5 can be similar to data flow actions 1-5 in Figure 2A. Certain details relating to the CDS 200b are explained above in more detail in connection with Figure 2A. In general, though, at data flow action 1, the CDS 200b performs a query on the data in a data source 210 to group by a particular column. At data flow action 2, the CDS 200b determines which partition has changed. At data flow action 3, the CDS 200b obtains data for the changed partition.

[0032] As explained above, a Bloom filter(s) can be used in comparison of the current partition data and the historical partition data. A Bloom filter may refer to a space-efficient probabilistic data structure that is used to test whether an element is a member of a set (e.g., data items are part of a partition). False positive matches are possible, but false negatives are not. For example, a query can return either "possibly in set" or "definitely not in set." Elements can be added to the set, but generally cannot be removed. As more elements are added to the set, the probability of false positives becomes larger.

[0033] In one embodiment, a Bloom filter used by the CDS 200b is a bit array of m bits and has k different hash functions that are used to add an element to the Bloom filter. In order to add an element to the Bloom filter, an element is fed to each of the k hash functions to get k array positions. The bits at these k positions are set to 1. In order to query for an element to determine whether it is in the set, the element is fed to each of the k hash functions to get the k array positions. If any of the bits at these positions is 0, the element is definitely not in the set. If all of the bits at these positions are 1, the element is either in the set, or the bits were set to 1 by chance when adding other elements. If the bits were set to 1 by chance, this can lead to a false positive. The Bloom filter is not required to store the elements themselves.

[0034] Although a risk of false positives exists, Bloom filters can provide a strong space advantage over other data structures for representing sets, such as self-balancing binary search trees, hash tables, simple arrays, linked lists, etc. Other data structures may require storing at least the data items themselves, which can require anywhere from a small number of bits (e.g., for small integers) to an arbitrary number of bits (e.g., for strings). On the other hand, a Bloom filter with 1% error and an optimal value of k may require only about 9.6 bits per element (e.g., data item), regardless of the size of the elements. The space advantage can be partly due to the compactness of the Bloom filter, inherited from arrays, and partly due to the probabilistic nature of the Bloom filter. The 1% false-positive rate can be reduced by a factor of ten by adding only about 4.8 bits per element.

[0035] At data flow action 4, the CDS 200b compares the current partition data to historical partition data using one or more Bloom filters 255. One or more Bloom filters 255 may be stored in local storage 250b. A Bloom filter can be created for the local version of the data. For example, at the time of a previous download, the CDS 200b may have added the data items from the data source 210 to a Bloom filter. Although the Bloom filter does not store the actual data items (e.g., the actual data items can be deleted after corresponding Bloom filters are generated), it can determine with high probability whether a data item is included in the previous version of the data or not. The Bloom filter can take up much less space than storing the historical or current partition data and can serve as a compressed version of the data. For each data item included in the current partition data, the CDS 200b can query the Bloom filter that includes the corresponding historical partition data to check whether the data item was included in the historical partition data or if it is new. In one embodiment, if n number of hash functions are defined for the Bloom filter, the Bloom filter applies the n hash functions to the data item to return n number of array positions. If any of the array positions is 0, the data item was not included in the historical partition data. If all array positions are 1, the data item was

likely included in the historical partition data, although a small probability of false positive exists.

[0036] Partition data obtained from a data source 210 may include a number of individual data items (e.g., rows, lines, etc.), and the actual number of data items included in the data of a partition can vary; some partitions may include a small number of data items, and other partitions can include a large number of data items. In one embodiment, the size of a Bloom filter is predetermined, and it may not be optimal to use the same Bloom filter for a small amount of data and a large amount of data. The probability of the Bloom filter returning false positives increases with the number of elements added to the Bloom filter. Therefore, if too many elements are added, the Bloom filter may become saturated, and the accuracy of the Bloom filter can deteriorate, e.g., to a point of returning almost 100% false positives. Accordingly, Bloom filters of different sizes can be used to accommodate data of varying size. For example, a Bloom filter has a predetermined size of m bits when it is created and may not be able to accommodate data that includes more than a specific number of elements (e.g., x number of elements) without deterioration of accuracy. For data that includes more than x elements, a Bloom filter having a size larger than m bits can be used. Because the size of data from different data sources can vary, the CDS 200b may use a series of Bloom filters of increasing size in order to accommodate different data size. For instance, the CDS 200b may have a number of Bloom filters of varying sizes available for use, or may create one as needed. In one example, the CDS 200b may begin with a Bloom filter having a size of m bits, and if this Bloom filter is too small for the data, the CDS 200b may select or create a Bloom filter having a size of $m + y$ bits and so on until the CDS 200b finds a Bloom filter having the right size for the data. The data from various data sources 210 may share the same set of Bloom filters. Or in certain embodiments, the CDS 200b may keep Bloom filters for different data sources 210 separate from each other.

[0037] In one embodiment, the CDS 200b may store Bloom filters 255 on storage that provides high accessibility. For example, the local storage 250b can include storage that is a type which is more accessible than storage used by a destination system 270. For example, the local storage 250b can use Network Attached Storage (NAS) since it is very accessible to attached devices. A more accessible storage type may be more expensive than less accessible storage type, and since Bloom filters can save space, the CDS 200b can reduce costs associated with the local storage 250b.

[0038] At data flow action 5, the CDS 200b extracts and forwards the changed data to the destination system 270. This step can be similar to data flow action 5 of Figure 2A. The CDS 200b can forward the changed data to one or more destination systems 270.

[0039] Figure 3 is an example of information obtained from a data source and/or information processed by the change determination system. A specific, illustrative example will be explained with respect to Figure 3. Various aspects will be explained with reference to the CDS 200a in Figure 2A, but the example can also apply to the CDS 100, 200b of Figures 1 and 2B. The example will refer to data of a data source 210 at time T_0 and data of the data source 210 at time T_1 , where T_0 is earlier than T_1 .

[0040] At time T1, the CDS 200a performs a query on the data of the data source 210 to group the data by the last changed or updated column or field. The data can be grouped into one or more groupings based on the date. The data source 210 can return a result that includes groupings 310 organized by date. The result can be referred to as "current grouping data." The current grouping data 310 may list the date for a grouping and the number of data items included in that grouping. The current grouping data 310 shows that Grouping 1 is for 2/10/14, and the number of data items in Grouping 1 is 400; Grouping 2 is for 2/11/14, and the number of data items in Grouping 2 is 310; and Grouping 3 is for 2/12/14, and the number of data items in Grouping 3 is 175.

[0041] The CDS 200a compares the current grouping data 310 to historical grouping data 315. Historical grouping data 315 can include the grouping data obtained from the data source 210 at various times in the past. Historical grouping data 315 can include grouping data for one or more days. In Figure 3, the historical grouping data 315 shows the grouping data at TO. The historical grouping data 315 shows that Grouping 1 is for 2/10/14, and the number of data items in Grouping 1 is 380; Grouping 2 is for 2/11/14, and the number of data items in Grouping 2 is 310; and Grouping 3 is for 2/12/14, and the number of data items in Grouping 3 is 165.

[0042] By comparing the number of data items in the same groupings at different points in time, the CDS 200 can identify that certain groupings have changed or are potential candidates having changed data items. The number of data items for Grouping 1 at T1 is 400, and the number of data items for Grouping 1 at TO is 380. The number of data items for Grouping 2 at T1 is 310, and the number of data items for Grouping 2 at TO is 310. The number of data items for Grouping 3 at T1 is 175, and the number of data items for Grouping 3 at TO is 165. The CDS 200a can see that the number of data items in Groupings 1 and 3 changed from TO to T1, while the number of data items in Grouping 2 remained the same from TO to T1. From this comparison, the CDS 200a can determine that data for Grouping 1 and Grouping 3 may have changed and should be obtained from the data source 210. The CDS 200a may keep track of the changed groupings 320, e.g., to request data from these groupings from the data source 210. For example, the changed groupings 320 information can list Groupings 1 and 3.

[0043] The CDS 200a obtains the data for Grouping 1 from the data source 210, and also obtains the data for Grouping 3 from the data source 210 (or some summary of the groupings, such as Bloom filters, in other embodiments). The example will be further explained with the obtained Grouping 3 data 330. Grouping 3 data 330 includes all 175 data items included in the grouping. The data source 210 can be a database 210a, and Grouping 3 data 330 may include rows as data items. Each row in Grouping 3 data 330 can include the date and time for the row (e.g., the timestamp of the last updated column) and the data of that row.

[0044] The CDS 200a compares Grouping 3 data 330 against Grouping 3 historical data 335. Grouping 3 data 330 can be associated with T1, and Grouping 3 historical data 335 can be associated with TO. For example, Grouping 3 historical data 335 can be Grouping 3 data that

was obtained at TO. Grouping 3 historical data may also include rows as data items. Each row in Grouping 3 historical data 335 can also include the date and time for the row and the data of that row. By comparing Grouping 3 data 330 and Grouping 3 historical data 335, the CDS 200a can determine that Row 3 changed, for example, Row 3 may have been inserted after TO. The CDS 200a flags Row 3 as a data item to send to a destination system 270. By going through the rest of Grouping 3 data 330 and Grouping 3 historical data 335, the CDS 200a identifies 10 rows in this example that were added. The CDS 200a can keep track of the changed data items in a list, such as Grouping 3 changed items list 340. In some embodiments, instead of comparing the data items to the previous version of the data items, the CDS 200a uses a Bloom filter to which the data items in the previous version have been added. The CDS 200a queries the Bloom filter to determine if a data item is in the set.

[0045] The CDS 200a may obtain grouping information for all dates for which data is available in the data source 210. For example, a data source 210 contains data for 1,000 days, the CDS 200a can get the grouping information for all 1,000 days. Or the CDS 200a may specify a timeframe for which it wants to obtain grouping information, such as 60 days. Grouping information can be easily obtained from a data source 210 without placing a burden on the resources of the data source 210. By comparing to historical grouping information, the CDS 200a can easily identify which groupings may have changed data.

[0046] Because comparison of grouping information can make it easy to spot changed data over a long period of time, the CDS 200a can capture all of the changes in the data. For example, in a system that downloads data for last 5 days may miss any data items whose last updated timestamp has changed to fall outside this 5-day window. However, the CDS 200a can detect that a data item has been removed or added to a particular grouping in any time window. For example, a user accidentally changes the last updated timestamp for Row 1 to Day 1 of Day 1,000. The system that only downloads last 5 days of data will miss Row 1, but the CDS 200a will recognize Row 1 as a change because it will be reflected in the number of data items for Day 1 in the grouping information.

[0047] The grouping unit or size and the grouping column can be selected such that most of the new data added to the data source 210 falls into one of the grouping units. In one embodiment, the grouping unit or size can relate to the desired latency of the pipeline, and the grouping column can relate to the distribution. The grouping unit or size may be selected at different levels of granularity. For example, a grouping may be based on a unit of multiple days, a day, multiple hours, an hour, etc. The unit or size of a grouping can be selected as appropriate, e.g., based on the requirements of the data source 210, CDS 200a, and/or the destination system 270. The unit or size of a grouping can be specified at a level that provides a meaningful comparison of groupings. In some embodiments, the grouping unit that leads to an even distribution of data items into groupings may not be very helpful since each grouping will have a change to the number of data items, and the CDS 200a has to check almost all groupings. For example, if GROUP BY was by an hour, instead of a day, the partition for each hour will probably include a few rows, and almost all partitions would have to be checked, which can lead to obtaining data for most of the partitions. On the other hand, GROUP BY by a

day will probably lead to recently added data falling into the more recent partitions. Under similar reasoning, the column or field used for grouping by can have a characteristic that leads to more "skewed" distribution than even distribution. In one example, if the data items were grouped by first letter of a person's last name, the grouping for each alphabet letter will likely contain new data items, and groupings for all alphabet letters will have to be checked. In other embodiments, even distribution of data items may be desired, and accordingly, the grouping unit or size and the grouping column can be selected to provide an even distribution of data items. For example, this may be done such that most of data items to be processed are not placed into one grouping.

[0048] In the example where the data is grouped by the last updated column, the CDS 200a may not distinguish between a data item that has been added and an existing data item that has been updated. In certain embodiments, the CDS 200a may implement a way to distinguish between the two types of change. For example, each data item may be assigned a unique identifier, e.g., when the data item is stored locally in local storage 250a. The unique identifier can be used to track whether a data item has been updated. In this case, the CDS 200a may not be able to use Bloom filters since actual data is not stored in Bloom filters.

[0049] In some embodiments, the CDS 200a may recognize that some data items have been deleted. For instance, the number of data items for a grouping may have decreased in comparison the previous number of data items for that grouping. The CDS 200a can identify the deleted data by comparing the data for the grouping to the local version of the data for the grouping. The CDS 200a may send information to the destination system 270 that the identified data items have been deleted, and the destination system 270 can delete the data items from its storage based on the information sent by the CDS 200a.

[0050] As described above, the CDS 200a can offer many advantages. The CDS 200a can identify a changed subset of data in a data source 210 without downloading all of the data. The CDS 200a can do so for large amounts, which can be very efficient. Only a portion of the data that may include changes is downloaded to extract the actual change. The CDS 200a can also identify changes in a generic way and can work with various data sources 210. Often, the CDS 200a may not have any information about the data of a data source 210. For example, the CDS 200a may not know how the data is structured (e.g., database schema, file format, etc.), how frequently the data is updated, ways in which the data is updated, or how the data is updated. The CDS 200a may identify a column such as the last updated column that can indicate whether a data item might be new and proceed to identify changes by performing a group by on the selected column. The CDS 200a may also handle data in different formats, such as databases and files, in the same or a similar manner. Because the CDS 200a only obtains or grabs changed data from a data source 210 to send to a destination system 270, the CDS 200a may also be referred to as a "grabber."

[0051] Figure 4A is a flowchart illustrating one embodiment of a process 400a for determining and obtaining changes in data of a plurality of data sources. The process 400a may be implemented by one or more systems described with respect to Figures 1-2 and 5. For

illustrative purposes, the process 400a is explained below in connection with the CDS 200a in Figure 2A. Certain details relating to the process 400a are explained in more detail with respect to Figures 1-5. Depending on the embodiment, the process 400a may include fewer or additional blocks, and the blocks may be performed in an order that is different than illustrated.

[0052] At block 401a, the CDS 200a obtains information indicating groupings of data of a data source 210. The data may be stored in one or more files or databases in the data source 210. The information can indicate a number of data items included in each of the groupings. The groupings can be based on timestamps of respective data items. The timestamps can indicate respective times at which data items were last updated. In certain embodiments, the timestamps of the respective data items include the date and the time at which the respective data items were last updated, and the groupings are based on only the date of the timestamps of the respective data items. For example, a grouping operation is performed based on only the date of the timestamps associated with the data items. In such case, each grouping is associated with a specific date. In some embodiments, the groupings are based on field of respective data items that can provide an uneven distribution of data items included in each grouping.

[0053] The CDS 200a may obtain the information indicating the groupings of the data at an interval. The CDS 200a may also obtain the information indicating the groupings of the data in response to receiving a request from a destination system 270. The CDS 200a may obtain information indicating the groupings of the data stored in one or more files in the data source 210 using a first adapter. The CDS 200a may obtain information indicating the groupings of the data stored in one or more databases in the data source 210 using a second adapter. The first adapter and the second adapter may be different.

[0054] At block 402a, the CDS 200a determines a grouping whose data items have changed. The CDS 200a can determine whether data items of a grouping have changed by comparing a number of data items included in the grouping and a historical number of data items included in a corresponding local version of that grouping. The corresponding local version of the first grouping may be created based on data items included in the grouping at a time prior to obtaining the information indicating the groupings of the data. This time may be referred to as time TO.

[0055] At block 403a, the CDS 200a obtains data items in the changed grouping from the data source 210. A data item included in the grouping can be a row in the one or more databases of the data source 210 or a line in the one or more files in the data source 210. In certain embodiments, if the data source 210 includes one or more files, the CDS 200a can check the timestamp of a file and compare it to the timestamp of the previous version of the file in order to determine whether the file may include new data. By comparing the timestamps of the current file and the previous version of the file, the CDS 200a does not need to parse through the data in the file to determine whether new data has been added. In such embodiments, the CDS 200a may not obtain grouping information. In addition, with respect to blocks 402a and 403a, the CDS 200a can directly compare the data items in the current file and the data items

in the previous version of the file, instead of determining changed grouping(s) and/or obtaining the data items in the changed grouping from the data source 210.

[0056] At block 404a, the CDS 200a compares data items in the grouping with the data items of the corresponding local version of the grouping. By comparing the data items, the CDS 200 can determine which data items of the grouping from the data source 210 have changed. The corresponding local version of the grouping can include a copy of the data items included in the grouping at TO. In certain embodiments, where the data source 210 includes one or more files, the CDS 200a can treat each newline in the file as a data item and compare the data items in the current file and the data items in the previous version of the file to identify the changed data items.

[0057] At block 405a, the CDS 200a extracts the changed data items of the grouping. The CDS 200a can forward the extracted changed data items to one or more destination systems 270.

[0058] If the number of data items included in the grouping is higher than the historical number of data items included in the corresponding local version of the grouping, the CDS 200a can identify the changed data items as added or updated data items, and forward the changed data items to the destination system 270 to be stored. If the number of data items included in the grouping is lower than the historical number of data items included in the corresponding local version of the grouping, the CDS 200a can identify the changed data items as deleted data items, and forward the changed data items to the destination system 270 to be removed.

[0059] In some embodiments, the CDS 200a assigns a unique identifier to each of the data item included in the grouping. The CDS 200a can determine whether a changed item is a new data item or an updated data item based on the unique identifier associated with the changed data item.

[0060] Figure 4B is a flowchart illustrating another embodiment of a process 400b for determining and obtaining changes in data of a plurality of data sources. The process 400b may be implemented by one or more systems described with respect to Figures 1-2 and 5. For illustrative purposes, the process 400b is explained below in connection with the CDS 200b in Figure 2B. Certain details relating to the process 400b are explained in more detail with respect to Figures 1-5. Depending on the embodiment, the process 400b may include fewer or additional blocks, and the blocks may be performed in an order that is different than illustrated.

[0061] At block 401b, the CDS 200b obtains information indicating groupings of data of a data source 210. The data source 210 may be a database or a file. The information can indicate a number of data items included in each of the groupings. The groupings can be based on timestamps of respective data items. The timestamps can indicate respective times at which data items were last updated.

[0062] At block 402b, the CDS 200b determines a grouping whose data items have changed.

The CDS 200b can determine whether data items of a grouping have changed by comparing a number of data items included in the grouping and a historical number of data items included in each of the groupings. For example, the historical number of data items included in each of the groupings may be stored in historical grouping data 315 discussed with respect to Figure 3.

[0063] At block 403b, the CDS 200b obtains data items in the changed grouping from the data source 210. If the data source 210 is a database 210a, a data item included in the grouping can be a row, and if the data source 210 is a file 210b, a data item included in the grouping can be a line.

[0064] At block 404b, the CDS 200b compares data items in the grouping with a compressed version of the data. The CDS 200b can compare the data items in the grouping with a corresponding local version of the grouping, which can be a compressed version of the data. Based on the comparison, the CDS 200b can determine which data items of the grouping from the data source 210 have changed. The compressed version of the data may be a space-efficient probabilistic data structure, such as a Bloom filter. The space-efficient probabilistic data structure may include information about data items included in the grouping at a time prior to obtaining the information indicating the groupings of the data of the data source 210. This time may be referred to as time TO. The space-efficient probabilistic data structure can identify whether a data item included in the grouping was included in the grouping at the prior time (e.g., time TO). In some embodiments, the space-efficient probabilistic data structure is a Bloom filter. The Bloom filter may be selected from multiple Bloom filters each having a different size. The compressed version of the data may not comprise a copy of the grouping.

[0065] The compressed version of the data is stored on local storage 250b, and the extracted changed data items forwarded to the destination system 270 are stored on storage in the destination system 270. In certain embodiments, the local storage 250b has a smaller storage capacity than the destination system 270 storage. In one embodiment, the local storage 250b includes NAS.

[0066] At block 405b, the CDS 200b extracts the changed data items of the grouping. At block 406b, the CDS 200b forwards the changed data items to the destination system 270. Blocks 405b and 406b can be similar to blocks 405a in Figure 4A.

Implementation Mechanisms

[0067] According to one embodiment, the techniques described herein are implemented by one or more special-purpose computing devices. The special-purpose computing devices may be hard-wired to perform the techniques, or may include circuitry or digital electronic devices such as one or more application-specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs) that are persistently programmed to perform the techniques, or may include one or more hardware processors programmed to perform the techniques pursuant to program instructions in firmware, memory, other storage, or a combination. Such special-

purpose computing devices may also combine custom hard-wired logic, ASICs, or FPGAs with custom programming to accomplish the techniques. The special-purpose computing devices may be desktop computer systems, server computer systems, portable computer systems, handheld devices, networking devices or any other device or combination of devices that incorporate hard-wired and/or program logic to implement the techniques.

[0068] Computing device(s) are generally controlled and coordinated by operating system software, such as iOS, Android, Chrome OS, Windows XP, Windows Vista, Windows 7, Windows 8, Windows Server, Windows CE, Unix, Linux, SunOS, Solaris, iOS, Blackberry OS, VxWorks, or other compatible operating systems. In other embodiments, the computing device may be controlled by a proprietary operating system. Conventional operating systems control and schedule computer processes for execution, perform memory management, provide file system, networking, I/O services, and provide a user interface functionality, such as a graphical user interface ("GUI"), among other things.

[0069] For example, Figure 8 is a block diagram that illustrates a computer system 500 upon which an embodiment may be implemented. For example, the computing system 500 may comprises a server system that accesses law enforcement data and provides user interface data to one or more users (e.g., executives) that allows those users to view their desired executive dashboards and interface with the data. Other computing systems discussed herein, such as the user (e.g., executive), may include any portion of the circuitry and/or functionality discussed with reference to system 500.

[0070] Computer system 500 includes a bus 502 or other communication mechanism for communicating information, and a hardware processor, or multiple processors, 504 coupled with bus 502 for processing information. Hardware processor(s) 504 may be, for example, one or more general purpose microprocessors.

[0071] Computer system 500 also includes a main memory 506, such as a random access memory (RAM), cache and/or other dynamic storage devices, coupled to bus 502 for storing information and instructions to be executed by processor 504. Main memory 506 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 504. Such instructions, when stored in storage media accessible to processor 504, render computer system 500 into a special-purpose machine that is customized to perform the operations specified in the instructions.

[0072] Computer system 500 further includes a read only memory (ROM) 808 or other static storage device coupled to bus 502 for storing static information and instructions for processor 504. A storage device 510, such as a magnetic disk, optical disk, or USB thumb drive (Flash drive), etc., is provided and coupled to bus 502 for storing information and instructions.

[0073] Computer system 500 may be coupled via bus 502 to a display 512, such as a cathode ray tube (CRT) or LCD display (or touch screen), for displaying information to a computer user. An input device 514, including alphanumeric and other keys, is coupled to bus 502 for

communicating information and command selections to processor 504. Another type of user input device is cursor control 516, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 504 and for controlling cursor movement on display 512. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane. In some embodiments, the same direction information and command selections as cursor control may be implemented via receiving touches on a touch screen without a cursor.

[0074] Computing system 500 may include a user interface module to implement a GUI that may be stored in a mass storage device as executable software codes that are executed by the computing device(s). This and other modules may include, by way of example, components, such as software components, object-oriented software components, class components and task components, processes, functions, attributes, procedures, subroutines, segments of program code, drivers, firmware, microcode, circuitry, data, databases, data structures, tables, arrays, and variables.

[0075] In general, the word "module," as used herein, refers to logic embodied in hardware or firmware, or to a collection of software instructions, possibly having entry and exit points, written in a programming language, such as, for example, Java, Lua, C or C++. A software module may be compiled and linked into an executable program, installed in a dynamic link library, or may be written in an interpreted programming language such as, for example, BASIC, Perl, or Python. It will be appreciated that software modules may be callable from other modules or from themselves, and/or may be invoked in response to detected events or interrupts. Software modules configured for execution on computing devices may be provided on a computer readable medium, such as a compact disc, digital video disc, flash drive, magnetic disc, or any other tangible medium, or as a digital download (and may be originally stored in a compressed or installable format that requires installation, decompression or decryption prior to execution). Such software code may be stored, partially or fully, on a memory device of the executing computing device, for execution by the computing device. Software instructions may be embedded in firmware, such as an EPROM. It will be further appreciated that hardware modules may be comprised of connected logic units, such as gates and flip-flops, and/or may be comprised of programmable units, such as programmable gate arrays or processors. The modules or computing device functionality described herein are preferably implemented as software modules, but may be represented in hardware or firmware. Generally, the modules described herein refer to logical modules that may be combined with other modules or divided into sub-modules despite their physical organization or storage

[0076] Computer system 500 may implement the techniques described herein using customized hard-wired logic, one or more ASICs or FPGAs, firmware and/or program logic which in combination with the computer system causes or programs computer system 500 to be a special-purpose machine. According to one embodiment, the techniques herein are performed by computer system 500 in response to processor(s) 504 executing one or more

sequences of one or more instructions contained in main memory 506. Such instructions may be read into main memory 506 from another storage medium, such as storage device 510. Execution of the sequences of instructions contained in main memory 506 causes processor(s) 504 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions.

[0077] The term "non-transitory media," and similar terms, as used herein refers to any media that store data and/or instructions that cause a machine to operate in a specific fashion. Such non-transitory media may comprise non-volatile media and/or volatile media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 510. Volatile media includes dynamic memory, such as main memory 506. Common forms of non-transitory media include, for example, a floppy disk, a flexible disk, hard disk, solid state drive, magnetic tape, or any other magnetic data storage medium, a CD-ROM, any other optical data storage medium, any physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, NVRAM, any other memory chip or cartridge, and networked versions of the same.

[0078] Non-transitory media is distinct from but may be used in conjunction with transmission media. Transmission media participates in transferring information between nontransitory media. For example, transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 502. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

[0079] Various forms of media may be involved in carrying one or more sequences of one or more instructions to processor 504 for execution. For example, the instructions may initially be carried on a magnetic disk or solid state drive of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 500 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 502. Bus 502 carries the data to main memory 506, from which processor 504 retrieves and executes the instructions. The instructions received by main memory 506 may retrieve and execute the instructions. The instructions received by main memory 506 may optionally be stored on storage device 510 either before or after execution by processor 504.

[0080] Computer system 500 also includes a communication interface 518 coupled to bus 502. Communication interface 518 provides a two-way data communication coupling to a network link 520 that is connected to a local network 522. For example, communication interface 518 may be an integrated services digital network (ISDN) card, cable modem, satellite modem, or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 518 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN (or WAN component to communicated with a WAN). Wireless links may also be implemented. In any such implementation, communication interface 518 sends and receives electrical, electromagnetic or

optical signals that carry digital data streams representing various types of information.

[0081] Network link 520 typically provides data communication through one or more networks to other data devices. For example, network link 520 may provide a connection through local network 522 to a host computer 524 or to data equipment operated by an Internet Service Provider (ISP) 526. ISP 526 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 525. Local network 522 and Internet 525 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 520 and through communication interface 518, which carry the digital data to and from computer system 500, are example forms of transmission media.

[0082] Computer system 500 can send messages and receive data, including program code, through the network(s), network link 520 and communication interface 518. In the Internet example, a server 530 might transmit a requested code for an application program through Internet 525, ISP 526, local network 522 and communication interface 518.

[0083] The received code may be executed by processor 504 as it is received, and/or stored in storage device 510, or other non-volatile storage for later execution.

[0084] Each of the processes, methods, and algorithms described in the preceding sections may be embodied in, and fully or partially automated by, code modules executed by one or more computer systems or computer processors comprising computer hardware. The processes and algorithms may be implemented partially or wholly in application-specific circuitry.

[0085] The various features and processes described above may be used independently of one another, or may be combined in various ways. All possible combinations and subcombinations are intended to fall within the scope of this disclosure. In addition, certain method or process blocks may be omitted in some implementations. The methods and processes described herein are also not limited to any particular sequence, and the blocks or states relating thereto can be performed in other sequences that are appropriate. For example, described blocks or states may be performed in an order other than that specifically disclosed, or multiple blocks or states may be combined in a single block or state. The example blocks or states may be performed in serial, in parallel, or in some other manner. Blocks or states may be added to or removed from the disclosed example embodiments. The example systems and components described herein may be configured differently than described. For example, elements may be added to, removed from, or rearranged compared to the disclosed example embodiments.

[0086] Conditional language, such as, among others, "can," "could," "might," or "may," unless specifically stated otherwise, or otherwise understood within the context as used, is generally intended to convey that certain embodiments include, while other embodiments do not include, certain features, elements and/or steps. Thus, such conditional language is not generally

intended to imply that features, elements and/or steps are in any way required for one or more embodiments or that one or more embodiments necessarily include logic for deciding, with or without user input or prompting, whether these features, elements and/or steps are included or are to be performed in any particular embodiment.

[0087] Any process descriptions, elements, or blocks in the flow diagrams described herein and/or depicted in the attached figures should be understood as potentially representing modules, segments, or portions of code which include one or more executable instructions for implementing specific logical functions or steps in the process. Alternate implementations are included within the scope of the embodiments described herein in which elements or functions may be deleted, executed out of order from that shown or discussed, including substantially concurrently or in reverse order, depending on the functionality involved, as would be understood by those skilled in the art.

[0088] Various aspects of the present invention may relate to one or more of the Enumerated Example Embodiments (EEEs) below, each of which are examples. As with any other related discussion provided in the present document, the EEEs should not be construed as limiting any claim.

EEE1: A computer system configured to obtain changed data from a data source, the computer system comprising: one or more hardware computer processors configured to execute code in order to cause the system to: obtain information indicating a plurality of groupings of data stored in one or more files or databases in a data source, the information indicating a number of data items included in each of the plurality of groupings; determine a first grouping of the plurality of groupings including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical first number of data items included in a corresponding local version of the first grouping, wherein the corresponding local version of the first grouping is created based on data items included in the first grouping at a first time prior to said obtaining the information indicating the plurality of groupings of the data; access data items included in the first grouping from the data source; compare the data items included in the first grouping to data items of the corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed; extract the changed data items of the first grouping; and forward the extracted changed data items to a destination system.

EEE 2: The system of EEE 1, wherein the plurality of groupings is based on timestamps of respective data items, wherein the timestamps indicate respective times at which data items were last updated.

EEE 3: The system of EEE 1 or EEE 2, wherein the timestamps of the respective data items include a date and a time at which the respective data items were last updated, and wherein the plurality of groupings is based on only the date of the timestamps of the respective data items.

EEE 4: The system of EEE 3, wherein each of the plurality of groupings is associated with a

different date.

EEE 5: The system of EEE 3 or EEE 4, wherein the dates associated with each of the plurality of groupings are within a range of dates.

EEE 6: The system of any of EEEs 1-5, wherein the plurality of groupings is based on a first field of respective data items, the first field configured to provide an uneven distribution of data items included in each of the plurality of groupings.

EEE 7: The system of any of EEEs 1-6, wherein the corresponding local version of the first grouping comprises a copy of the data items included in the first grouping at the first time.

EEE 8: The system of any of EEEs 1-7, wherein, in response to determining that the first number of data items is higher than the historical first number of data items, the code is further configured to cause the system to: identify the changed data items as added or updated data items; and forward the changed data items to the destination system to be stored.

EEE 9: The system of any of EEEs 1-8, wherein, in response to determining that the first number of data items is lower than the historical first number of data items, the code is further configured to cause the system to: identify the changed data items as deleted data items; and forward the changed data items to the destination system to be removed.

EEE 10: The system of any of EEEs 1-9, wherein the code is further configured to: assign a unique identifier to each of the data items included in the first grouping; and determine whether a first changed data item of the changed data items is a new data item or an updated data item based on the unique identifier associated with the first changed data item.

EEE 11: The system of any of EEEs 1-10, wherein a data item included in the first grouping is a row in the one or more databases or a line in the one or more files.

EEE 12: The system of any of EEEs 1-11, wherein the code is further configured to cause the system to: obtain the information indicating the plurality of groupings of the data stored in the one or more files in the data source using a first adapter; and obtain the information indicating the plurality of groupings of the data stored in the one or more databases in the data source using a second adapter.

EEE 13: The system of any of EEEs 1-12, wherein the code is further configured to cause the system to obtain the information indicating the plurality of groupings of the data at an interval.

EEE 14: The system of any of EEEs 1-13, wherein the code is further configured to cause the system to obtain the information indicating the plurality of groupings of the data in response to receiving a request from the destination system.

EEE 15: A method of obtaining changed data from a data source, the method comprising: obtaining, by one or more hardware computer processors, information indicating a plurality of groupings of data stored in one or more files or databases in a data source, the information indicating a number of data items included in each of the plurality of groupings; determining, by the one or more hardware computer processors, a first grouping of the plurality of groupings

including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical first number of data items included in a corresponding local version of the first grouping, wherein the corresponding local version of the first grouping is created based on data items included in the first grouping at a first time prior to said obtaining the information indicating the plurality of groupings of the data; accessing, by the one or more hardware computer processors, data items included in the first grouping from the data source; comparing, by the one or more hardware computer processors, the data items included in the first grouping to data items of the corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed; extracting, by the one or more hardware computer processors, the changed data items of the first grouping; and forwarding, by the one or more hardware computer processors, the extracted changed data items to a destination system.

EEE 16: The method of EEE 15, wherein the plurality of groupings is based on timestamps of respective data items, wherein the timestamps indicate respective times at which data items were last updated.

EEE 17: The method of EEE 15 or EEE 16, wherein the plurality of groupings is based on a first field of respective data items, the first field configured to provide an uneven distribution of data items included in each of the plurality of groupings.

EEE 18: The method of any of EEEs 15-17, wherein the corresponding local version of the first grouping comprises a copy of the data items included in the first grouping at the first time.

EEE 19: The method of any of EEEs 15-18, wherein a data item included in the first grouping is a row in the one or more databases or a line in the one or more files.

EEE 20: A transitory or non-transitory computer readable medium comprising instructions for obtaining changed data from a data source that cause a computer processor to: obtain information indicating a plurality of groupings of data stored in one or more files or databases in a data source, the information indicating a number of data items included in each of the plurality of groupings; determine a first grouping of the plurality of groupings including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical first number of data items included in a corresponding local version of the first grouping, wherein the corresponding local version of the first grouping is created based on data items included in the first grouping at a first time prior to said obtaining the information indicating the plurality of groupings of the data; access data items included in the first grouping from the data source; compare the data items included in the first grouping to data items of the corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed; extract the changed data items of the first grouping; and forward the extracted changed data items to a destination system.

EEE 21: A computer system configured to obtain changed data from a data source, the computer system comprising: one or more hardware computer processors configured to execute code in order to cause the system to: obtain information indicating a plurality of groupings of data of a data source, the information indicating a number of data items included

in each of the plurality of groupings; determine a first grouping of the plurality of groupings including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical number of data items included in each of the plurality of groupings; access data items included in the first grouping from the data source; compare the data items included in the first grouping to data items of a corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed, wherein the corresponding local version of the first grouping of data items is a compressed version of the first grouping of data items; extract the changed data items of the first grouping; and forward the extracted changed data items to a destination system.

EEE 22: The system of EEE 21, wherein the compressed version of the first grouping of data items is a space-efficient probabilistic data structure including information about data items included in the first grouping at a first time prior to said obtaining the information indicating the plurality of groupings of the data of the data source.

EEE 23: The system of EEE 22, wherein the space-efficient probabilistic data structure is configured to determine whether a particular data item included in the first grouping was included in the first grouping at the first time.

EEE 24: The system of EEE 22 or EEE 23, wherein the space-efficient probabilistic data structure is a Bloom filter.

EEE 25: The system of EEE 23 or EEE 24, wherein the Bloom filter is selected from a plurality of Bloom filters that can each include a different number of data items.

EEE 26: The system of any of EEES 21-25, wherein the compressed version of the data does not comprise a copy of the data items of the first grouping.

EEE 27: The system of any of EEES 21-26, wherein the corresponding local version of the data is stored on first storage and the extracted changed data items forwarded to the destination system are stored on second storage, wherein the first storage has a smaller storage capacity than the second storage.

EEE 28: The system of EEE 27, wherein the first storage is a Network Attached Storage (NAS).

EEE 29: The system of any of EEES 21-28, wherein the data source is a database or a file.

EEE 30: The system of any of EEES 21-29, wherein the plurality of groupings is based on timestamps of respective data items, wherein the timestamps indicate respective times at which data items were last updated.

EEE 31: A method of obtaining changed data from a data source, the method comprising: obtaining, by one or more hardware computer processors, information indicating a plurality of groupings of data of a data source, the information indicating a number of data items included in each of the plurality of groupings; determining, by the one or more hardware computer processors, a first grouping of the plurality of groupings including one or more data items that

have changed by comparing a first number of data items included in the first grouping and a historical number of data items included in each of the plurality of groupings; accessing, by the one or more hardware computer processors, data items included in the first grouping from the data source; comparing, by the one or more hardware computer processors, the data items included in the first grouping to data items of a corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed, wherein the corresponding local version of the first grouping of data items is a compressed version of the first grouping of data items; extracting, by the one or more hardware computer processors, the changed data items of the first grouping; and forwarding, by the one or more hardware computer processors, the extracted changed data items to a destination system.

EEE 32: The method of EEE 31, wherein the compressed version of the first grouping of data items is a space-efficient probabilistic data structure including information about data items included in the first grouping at a first time prior to said obtaining the information indicating the plurality of groupings of the data of the data source.

EEE 33: The method of EEE 32, wherein the space-efficient probabilistic data structure determines whether a particular data item included in the first grouping was included in the first grouping at the first time.

EEE 34: The method of EEE 31 or EEE 32, wherein the space-efficient probabilistic data structure is a Bloom filter.

EEE 35: The method of any of EEES 31-34, wherein the compressed version of the data does not comprise a copy of the first grouping.

EEE 36: The method of any of EEES 31-35, wherein the corresponding local version of the data is stored on first storage and the extracted changed data items forwarded to the destination system are stored on second storage, wherein the first storage has a smaller storage capacity than the second storage.

EEE 37: The method of EEE 36, wherein the first storage is a Network Attached Storage (NAS).

EEE 38: The method of any of EEES 31-37, wherein the data source is a database or a file.

EEE 39: The method of any of EEES 31-38, wherein the plurality of groupings is based on timestamps of respective data items, wherein the timestamps indicate respective times at which data items were last updated.

EEE 40: A transitory or non-transitory computer readable medium comprising instructions for obtaining changed data from a data source that cause a computer processor to: obtain information indicating a plurality of groupings of data of a data source, the information indicating a number of data items included in each of the plurality of groupings; determine a first grouping of the plurality of groupings including one or more data items that have changed by comparing a first number of data items included in the first grouping and a historical number of data items included in each of the plurality of groupings; access data items included in the

first grouping from the data source; compare the data items included in the first grouping to data items of a corresponding local version of the first grouping to determine which data items of the first grouping from the data source have changed, wherein the corresponding local version of the first grouping of data items is a compressed version of the first grouping of data items; extract the changed data items of the first grouping; and forward the extracted changed data items to a destination system.

[0089] It should be emphasized that many variations and modifications may be made to the above-described embodiments, the elements of which are to be understood as being among other acceptable examples. All such modifications and variations are intended to be included herein within the scope of this disclosure. The foregoing description details certain embodiments of the invention. It will be appreciated, however, that no matter how detailed the foregoing appears in text, the invention can be practiced in many ways. As is also stated above, it should be noted that the use of particular terminology when describing certain features or aspects of the invention should not be taken to imply that the terminology is being re-defined herein to be restricted to including any specific characteristics of the features or aspects of the invention with which that terminology is associated. The scope of the invention should therefore be construed in accordance with the appended claims and any equivalents thereof.

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- [US25475714A \[0001\]](#)
- [US25477314A \[0001\]](#)
- [US61955054A \[0001\]](#)
- [WO2013030595A1 \[0004\]](#)

P a t e n t k r a v

1. Fremgangsmåde (400a, 400b) til opnåelse af ændrede data (340) fra en datakilde (110, 210) og videresendelse af de opnåede ændrede data til et destinationssystem (170, 270), hvilken fremgangsmåde (400a) omfatter:

5 opnåelse (401a, 401b), gennem en eller flere hardware-computerprocessorer (504), af information (310), der angiver en flerhed af grupperinger af data, der er lagret i en eller flere filer (110b, 210b) eller databaser (110a, 210a) i en datakilde (110, 210), hvor flerheden af grupperinger er baseret på dato og tidspunkt af tidsstempler af respektive dataelementer, således at dataelementerne er uensartet fordelt over flerheden af grupperinger, hvor hver af grupperingerne er forbundet med et forskelligt dataelementattribut, således at respektive grupperinger hver omfatter dataelementer med det tilsvarende dataelementattribut, hvor informationen (310) angiver en aktuel mængde af dataelementer,

10 der er inkluderet i hver af flerheden af grupperinger på et aktuelt tidspunkt, hvor informationen (310) opnås under anvendelse af en forespørgsel om en tidsstempeldataelementattribut;

at få adgang, gennem en eller flere hardware-computerprocessorer (504), til historisk grupperingsinformation (315), der angiver en historisk mængde af dataelementer, der er indeholdt i mindst nogle af flerheden af grupperinger på et tidligere tidspunkt, hvor, for en første delmængde (320) af grupperingerne, en

20 aktuel mængde af dataelementer på det aktuelle tidspunkt og en historisk mængde af dataelementer ikke er lige store, og for en anden delmængde af grupperingerne, en aktuel mængde af dataelementer og en historisk mængde af dataelementer er lige store;

at sammenligne, gennem den ene eller de flere hardware-computerprocessorer (504), mængden af dataelementer i hver af de tilsvarende grupperinger på det tidligere tidspunkt og det aktuelle tidspunkt;

at bestemme (402a, 402b), gennem den ene eller de flere hardwarecomputerprocessorer (504) for hver gruppering (330) i den første delmængde (320) baseret på sammenligningen, at en aktuel mængde af dataelementer, der er inkluderet i en tilsvarende gruppering (330), og en historisk mængde af dataelementer, der er inkluderet i en tilsvarende historisk gruppering (335) af dataelementer, ikke er lige store, hvilket indikerer, at et eller flere dataelementer af

30 den tilsvarende gruppering (330) har ændret sig; og

35

som reaktion på bestemmelsen af, at den aktuelle mængde af dataelementer er højere eller lavere end den historiske mængde af dataelementer, udføres følgende fremgangsmådetrin for hver gruppering (330) i den første delmængde (320):

5 at få adgang (403a, 403b), gennem den ene eller de flere hardware-computerprocessorer (504), til dataelementer, der er inkluderet i en gruppering (330) i den første delmængde (320) fra datakilden (110, 210);

10 at sammenligne (404a, 404b), gennem den ene eller de flere hardware-computerprocessorer (504), de dataelementer, der er inkluderet i grupperingen (330) i den første delmængde (320), med dataelementer af en tilsvarende lokal version af grupperingen (335) til bestemmelse af, hvilke dataelementer af grupperingen (330) i den første delmængde (320) fra datakilden (110, 210) der har ændret sig;

15 at ekstrahere (405a, 405b), gennem den ene eller de flere hardware-computerprocessorer (504), de ændrede dataelementer (340) af grupperingen (330) i den første delmængde (320); og

at videresende, gennem den ene eller de flere hardware-computerprocessorer (504), de ekstraherede ændrede dataelementer (340) til destinationssystemet (170, 270).

20

2. Fremgangsmåde (400a, 400b) ifølge krav 1, hvor flerheden af grupperinger er baseret på tidsstemplerne af de respektive dataelementer, hvor tidsstemplerne angiver respektive tidspunkter, ved hvilke dataelementer sidst blev opdateret.

25

3. Fremgangsmåde (400a, 400b) ifølge krav 2, hvor tidsstemplerne af de respektive dataelementer indbefatter en dato og et tidspunkt, ved hvilket de respektive dataelementer sidst blev opdateret, og hvor flerheden af grupperinger kun er baseret på datoen af tidsstemplerne af de respektive dataelementer.

30

4. Fremgangsmåde (400a, 400b) ifølge krav 3, hvor hver af flerheden af grupperinger er forbundet med en forskellig dato, og/eller datoerne, der er forbundet med hver af flerheden af grupperinger, er inden for et interval af datoer.

35

5. Fremgangsmåde (400a, 400b) ifølge et hvilket som helst af kravene 1-4, hvor fremgangsmåden, som reaktion på bestemmelse af, at den aktuelle mængde af dataelementer er højere end den tilsvarende historiske mængde af dataelementer, yderligere omfatter:

- 5 at identificere de ændrede dataelementer (340) som tilføjede eller opdaterede dataelementer; og
at videresende de ændrede dataelementer (340) til destinationssystemet (170, 270), således at de kan lagres,
og/eller
- 10 hvor fremgangsmåden, som reaktion på bestemmelse af, at den aktuelle mængde af dataelementer er lavere end den tilsvarende historiske mængde af dataelementer, yderligere omfatter:
at identificere de ændrede dataelementer (340) som slettede dataelementer; og
- 15 at videresende de ændrede dataelementer (340) til destinationssystemet (170, 270), således at de kan fjernes.

6. Fremgangsmåde (400a, 400b) ifølge et hvilket som helst af kravene 1-5, hvor fremgangsmåden endvidere omfatter:

- 20 at tildele en unik identifikator til hver af de dataelementer, der er inkluderet i hver gruppering (330) i den første delmængde, og
at bestemme, om et første ændret dataelement af de ændrede dataelementer (340) er et nyt dataelement eller et opdateret dataelement baseret på den unikke identifikator, der er forbundet med det første ændrede dataelement,
- 25 og/eller
hvor et dataelement, der er inkluderet i en gruppering (330) i den første delmængde, er en række i den ene eller de flere databaser (110a, 210a) eller en linje i den ene eller de flere filer (110b, 210b).

7. Fremgangsmåde (400a, 400b) ifølge et hvilket som helst af kravene 1-6, hvor fremgangsmåden endvidere omfatter:

- 30 at opnå (401a, 401b) informationen (310), der angiver flerheden af grupperinger af de data, der er lagret i den ene eller de flere filer (110b, 210b) i datakilden (110, 210), under anvendelse af en første adapter, og

at opnå (401a, 401b) informationen (310), der angiver flerheden af grupperinger af de data, der er lagret i den ene eller de flere databaser (110a, 210a) i datakilden (110, 210), under anvendelse af en anden adapter, og/eller

5 at opnå (401a, 401b) informationen (310), der angiver flerheden af grupperinger af dataene i et interval, og/eller

at opnå (401a, 401b) informationen (310), der angiver flerheden af grupperinger af dataene som reaktion på at modtage en anmodning fra destinationssystemet (170, 270).
10

8. Fremgangsmåde (400a, 400b) ifølge et hvilket som helst af kravene 1-7, hvor den tilsvarende lokale version af grupperingen (335) omfatter en kopi af dataelementer, der er inkluderet i grupperingen (335) på et første tidspunkt før opnåelsen (401a) af informationen (310), der angiver flerheden af grupperinger af dataene, og/eller

15 hvor et dataelement, der er inkluderet i grupperingen (330) i den første delmængde (320), er en række i den ene eller de flere databaser (110a, 210a) eller en linje i den ene eller de flere filer (110b, 210b).
20

9. Fremgangsmåde (400b) ifølge et hvilket som helst af kravene 1-8, hvor den tilsvarende lokale version af grupperingen (330) af dataelementer er en komprimeret version af grupperingen (330) af dataelementer.
25

10. Fremgangsmåde (400b) ifølge krav 9, hvor den komprimerede version af grupperingen (330) af dataelementer er en pladsbesparende probabilistisk datastruktur, der indbefatter information (315) om dataelementer, der er inkluderet i grupperingen (335) på et første tidspunkt før opnåelsen (401b) af informationen (310), der angiver flerheden af grupperinger af dataene i datakilden (110, 210).
30

11. Fremgangsmåde (400b) ifølge krav 10, hvor den pladsbesparende probabilistiske datastruktur bestemmer, om et bestemt dataelement, der er inkluderet i grupperingen (330), var inkluderet i grupperingen (335) på det første tidspunkt,
35

og/eller

hvor den pladsbesparende probabilistiske datastruktur er et Bloom-filter (255), hvor Bloom-filteret (255), i nogle udførelsesformer, er udvalgt blandt en flerhed af Bloom-filtre, der hver kan indeholde et forskelligt antal dataelementer.

5

12. Fremgangsmåde (400b) ifølge et hvilket som helst af kravene 9-11, hvor den komprimerede version af dataene ikke omfatter en kopi af grupperingen (330),

og/eller

10

hvor den tilsvarende lokale version af dataene lagres på første datalager, og de ekstraherede ændrede dataelementer (340), der videresendes til destinationssystemet (170, 270), lagres på andet datalager, hvor det første datalager har en mindre lagringskapacitet end det andet datalager, hvor det første datalager i nogle udførelsesformer er et Network Attached Storage (NAS).

15

13. Fremgangsmåde (400b) ifølge et hvilket som helst af kravene 9-12, hvor datakilden (110, 210) er en database (110a, 210a) eller en fil (110b, 210b),

og/eller

20

hvor flerheden af grupperinger er baseret på tidsstempler af respektive dataelementer, hvor tidsstemplerne angiver respektive tidspunkter, ved hvilke dataelementer sidst blev opdateret.

25

14. Computersystem (100, 200a, 200b), der er konfigureret til at opnå ændrede data (340) fra en datakilde (110, 210) og til at videresende de opnåede ændrede data til et destinationssystem (170, 270), hvilket computersystem (100, 200a, 200b) omfatter:

en eller flere hardware-computerprocessorer (504), der er konfigureret til at eksekvere en kode for at få systemet (100, 200a, 200b) til at udføre operationer, herunder operationerne ifølge et hvilket som helst af kravene 1-13.

30

15. Computerlæsbart medium omfattende instruktioner til opnåelse af ændrede data (340) fra en datakilde (110, 210), der får en computerprocessor (504) til at udføre operationer, herunder operationerne ifølge et hvilket som helst af kravene 1-13.

35

DRAWINGS

DATA SOURCES

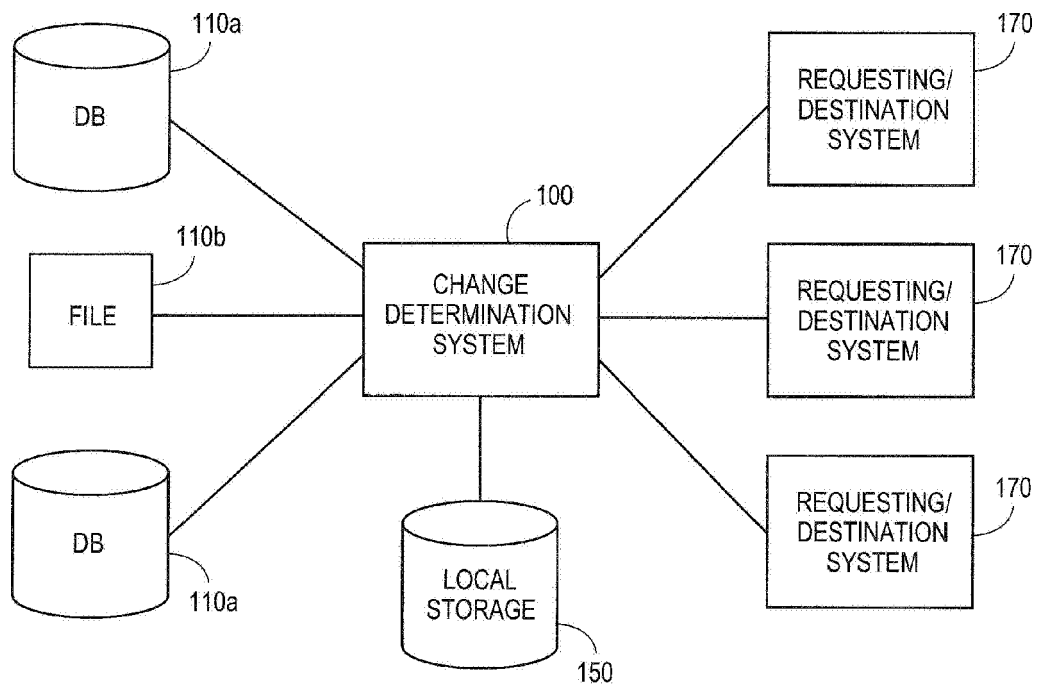


FIG. 1

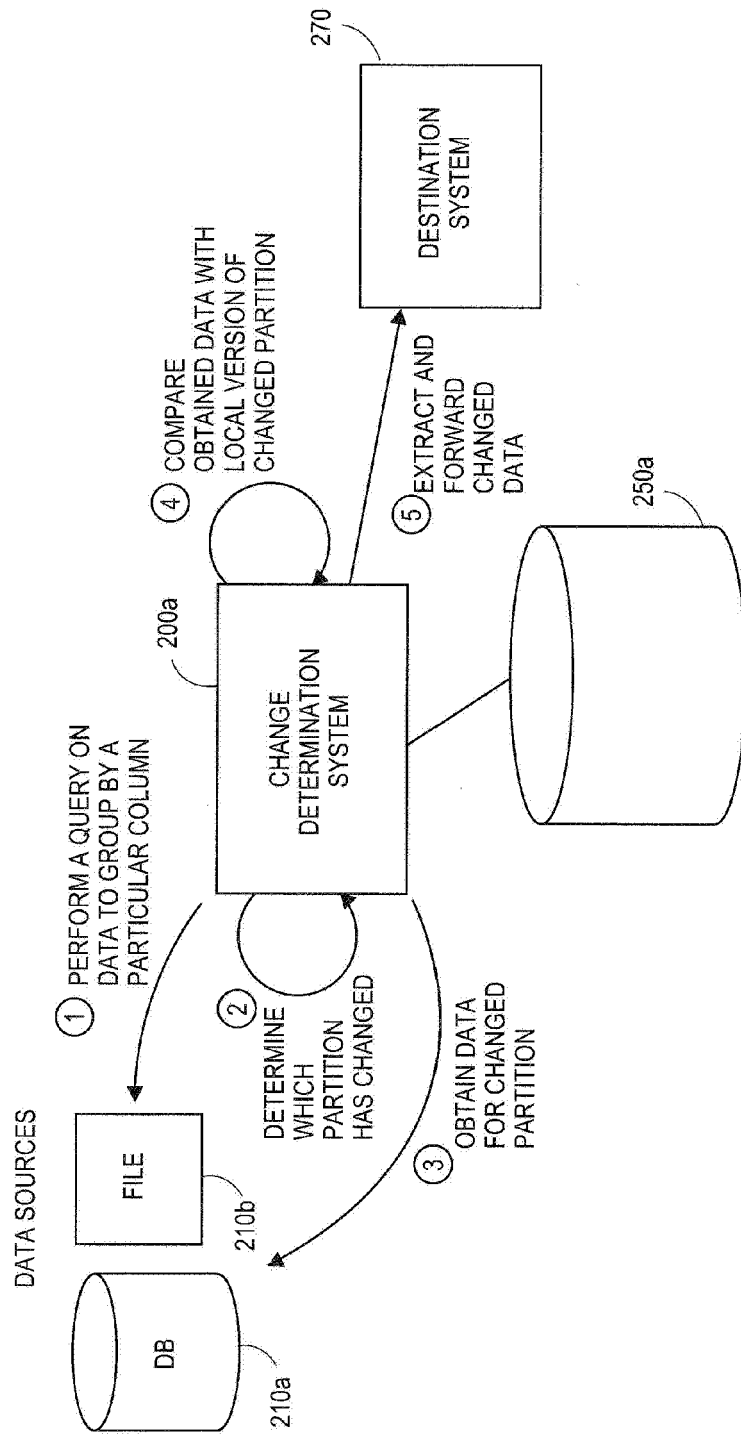
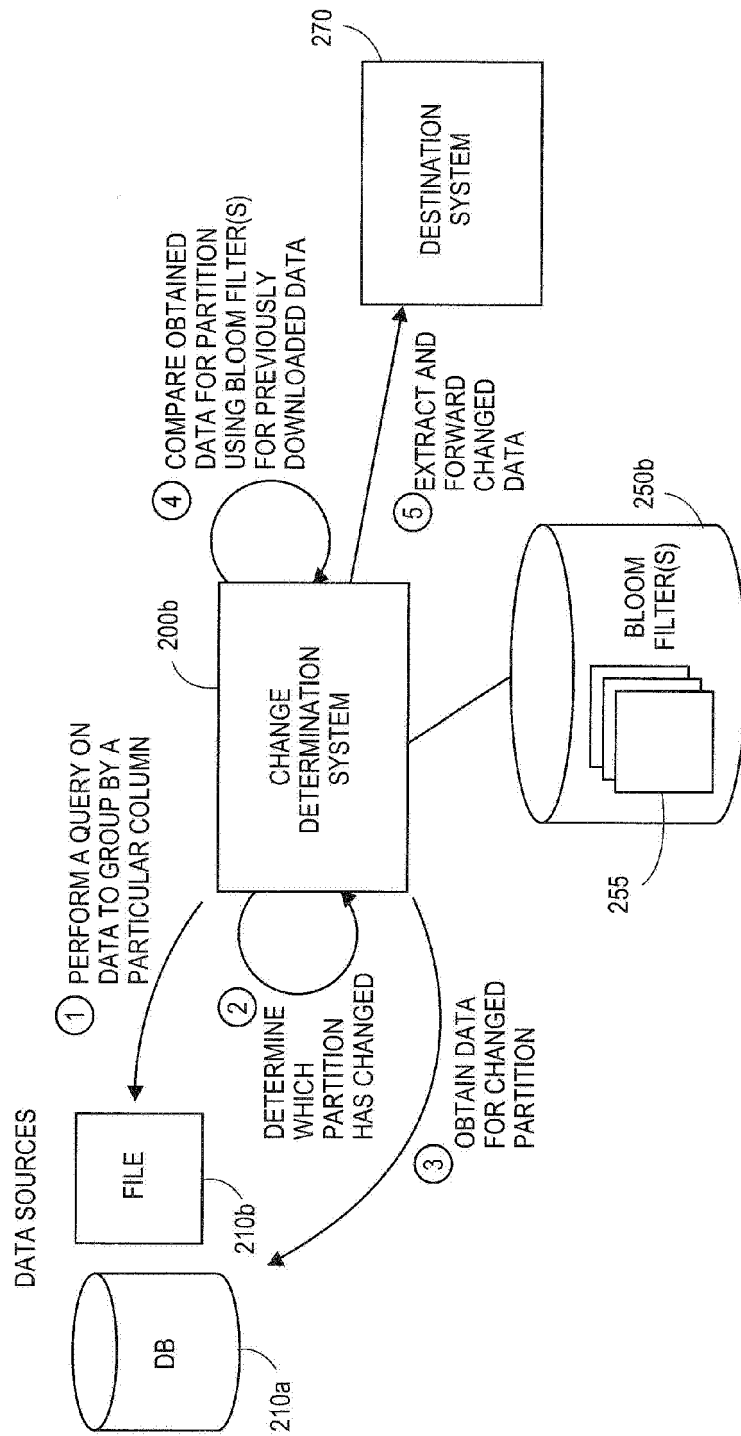
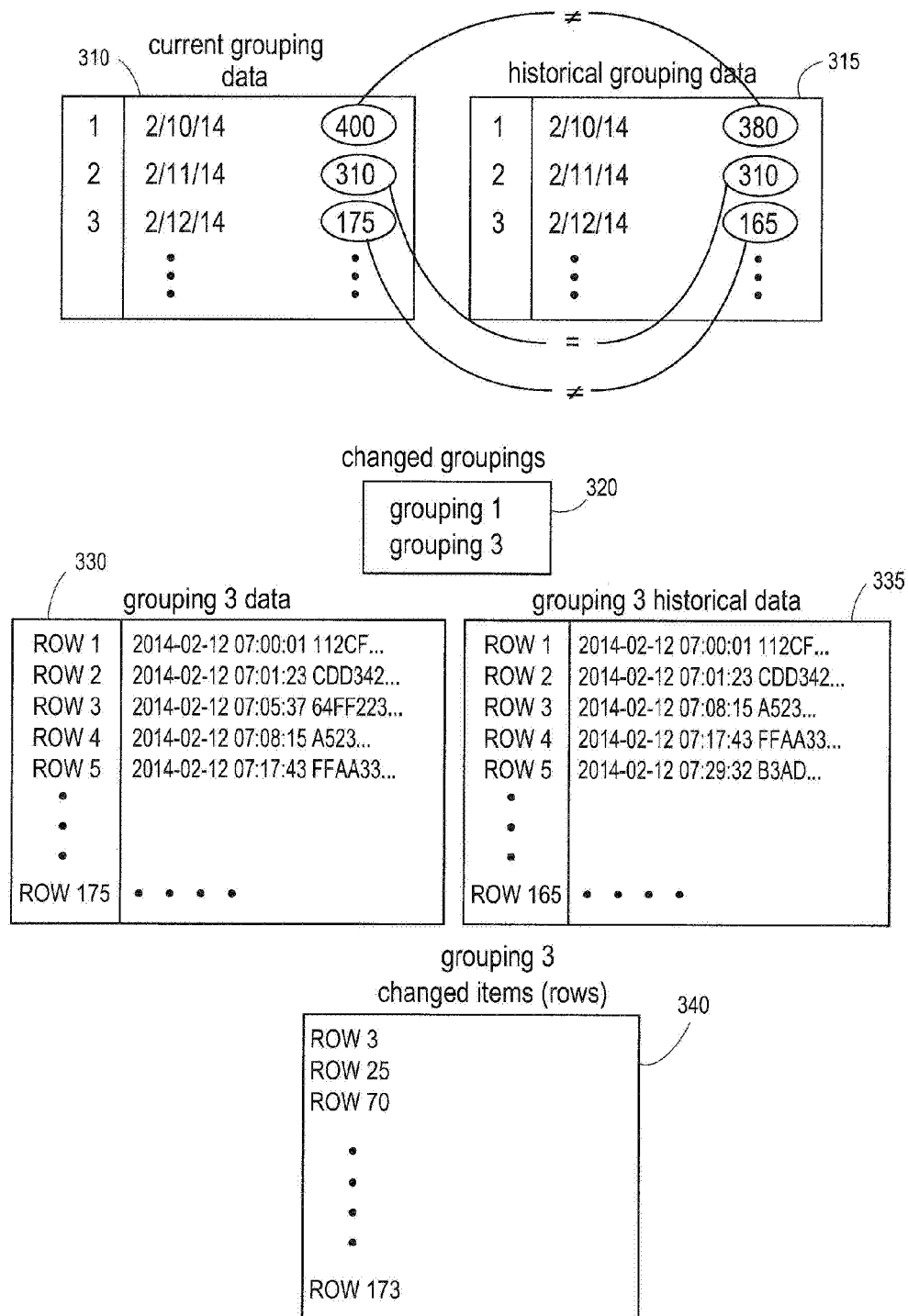
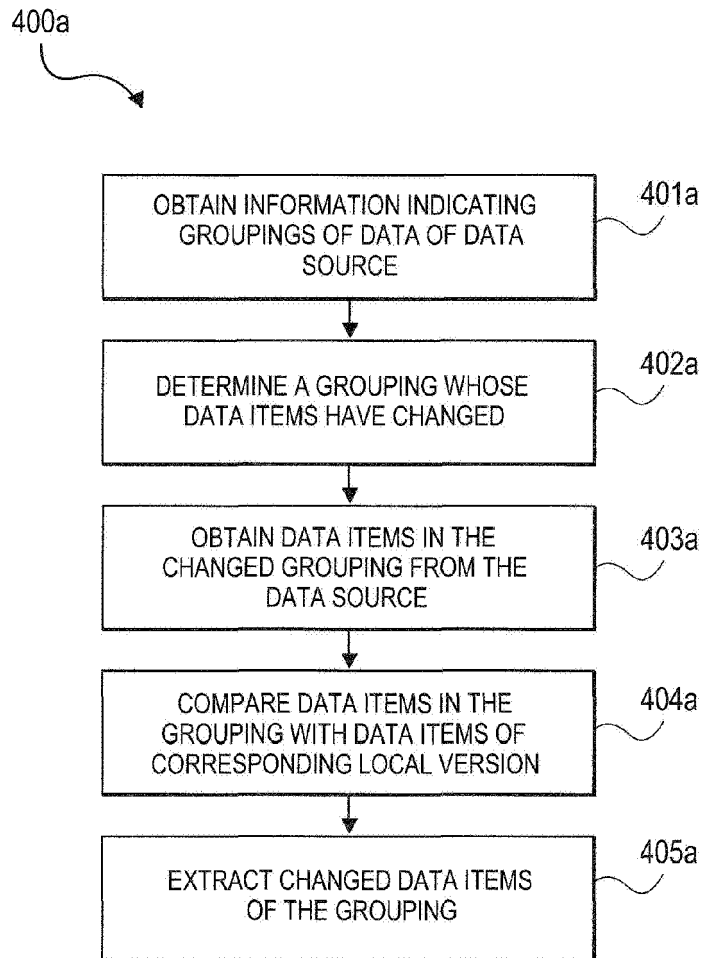
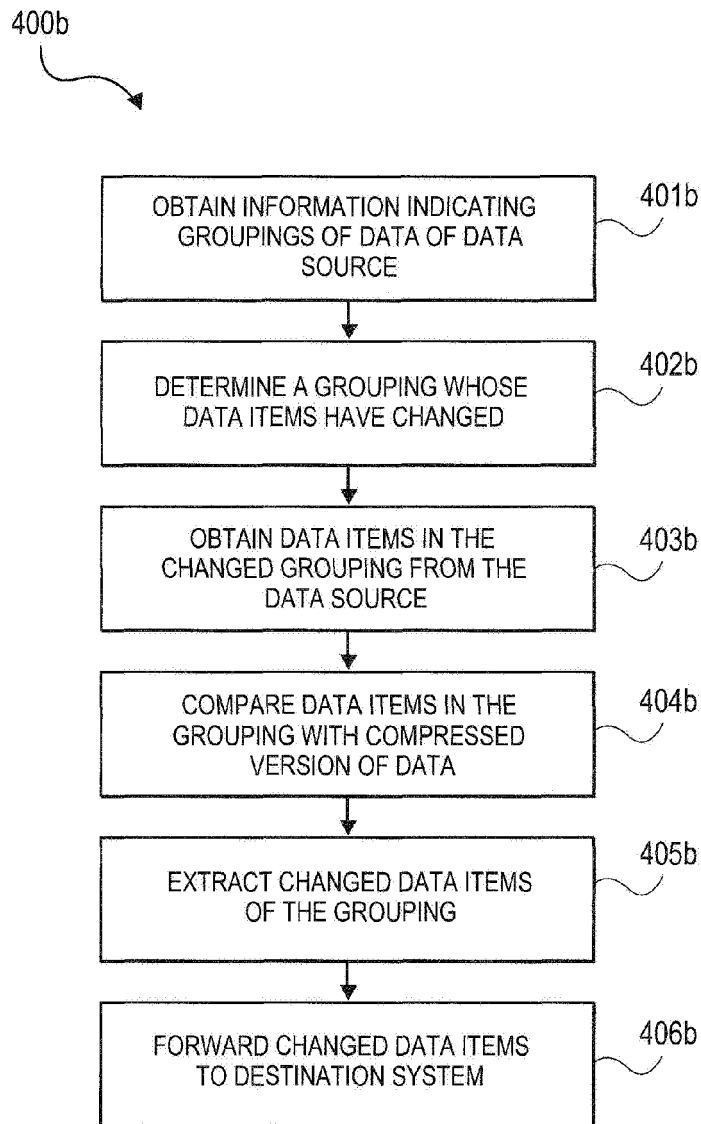


FIG. 2A

**FIG. 2B**

**FIG. 3**

**FIG. 4A**

**FIG. 4B**

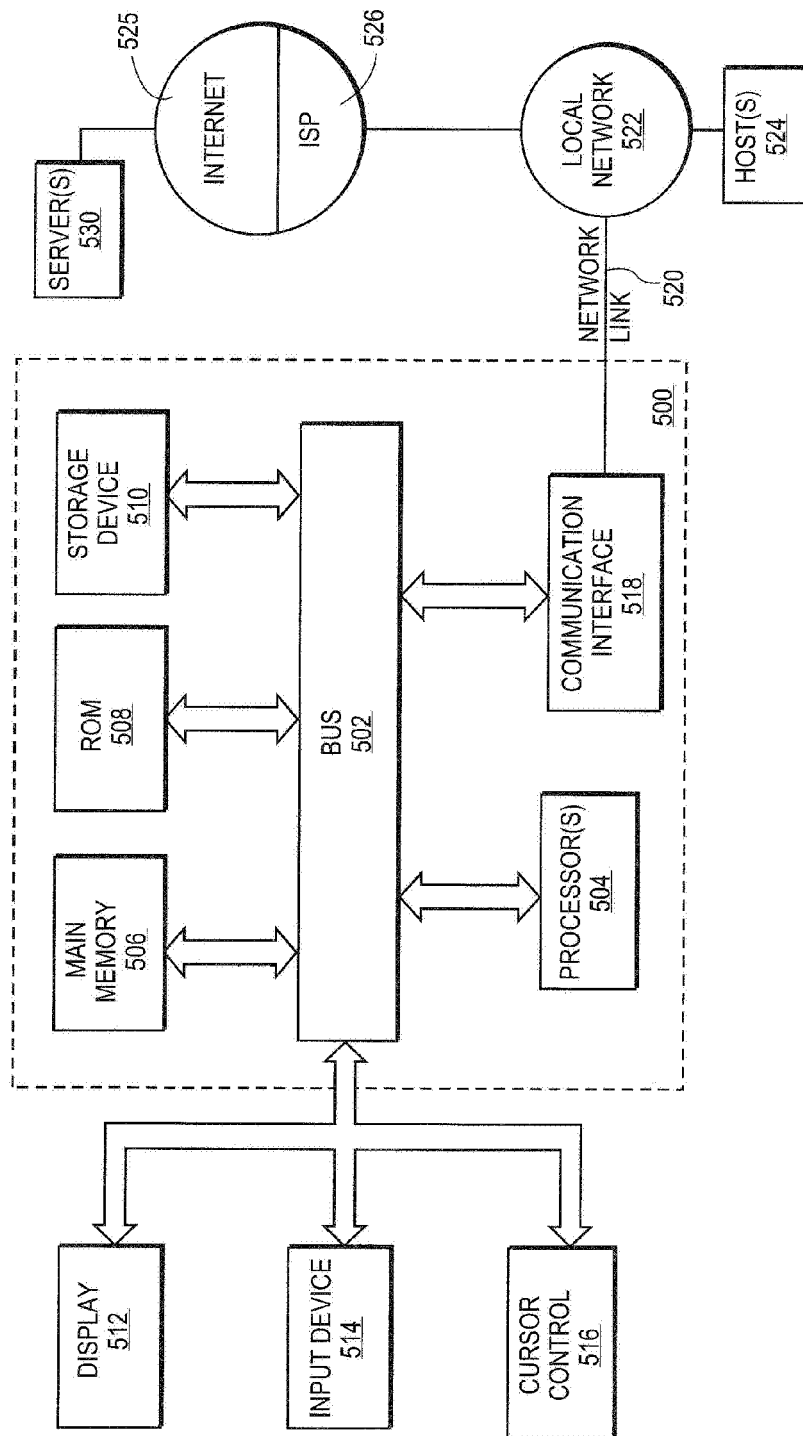


FIG. 5