US 20130046730A1

(19) **United States**

(12) **Patent Application Publication**

**Sood et al.**

(10) **Pub. No.: US 2013/0046730 A1**

(43) **Pub. Date: Feb. 21, 2013**

(54) **METHODS AND SYSTEMS FOR ACCESSING DATA**

(76) Inventors: **Manish Sood**, San Mateo, CA (US); **Xiaofeng Oian**, Toronto (CA)

(21) Appl. No.: **13/589,066**

(22) Filed: **Aug. 17, 2012**

**Related U.S. Application Data**

(63) Continuation of application No. 11/957,398, filed on Dec. 14, 2007, now Pat. No. 8,271,477.

(60) Provisional application No. 60/951,187, filed on Jul. 20, 2007, provisional application No. 60/970,257, filed on Sep. 5, 2007.

**Publication Classification**

(51) **Int. Cl.**

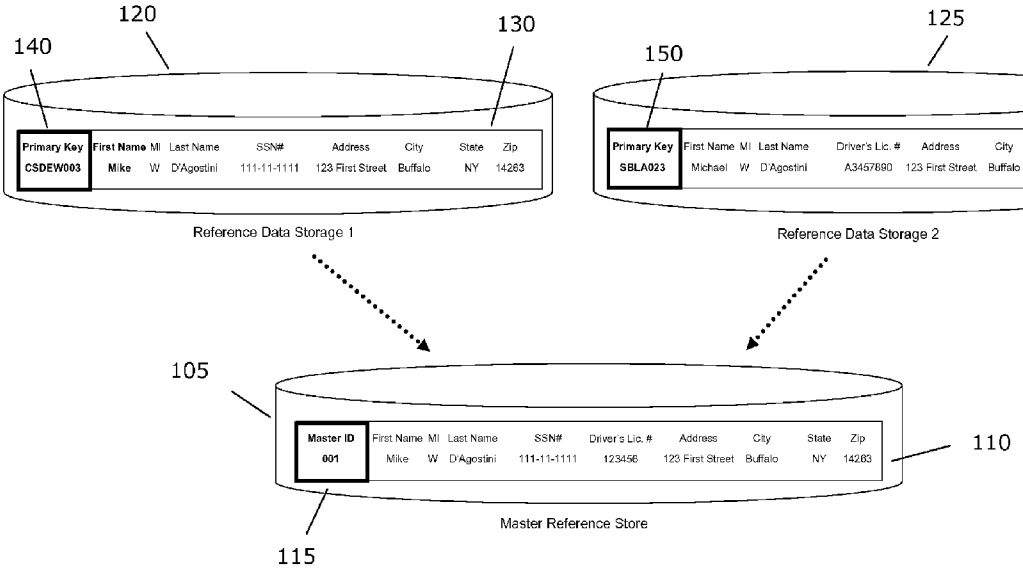
***G06F 17/30*** (2006.01)

***G06F 3/048*** (2006.01)

(52) **U.S. Cl.** ..... **707/609**; 715/764; 707/E17.005

(57) **ABSTRACT**

Some embodiments of the invention provide a user interface that allows a user to specify one or more attributes that should be included in a master reference data set, and identify which of these attributes should serve as enterprise specified identifiers that can be used to identify the particular master reference data set in an enterprise data storage. Some embodiments of the invention provide a method that allows the master reference data set to be accessed and updated in the data storage through the use of the enterprise specified identifiers.



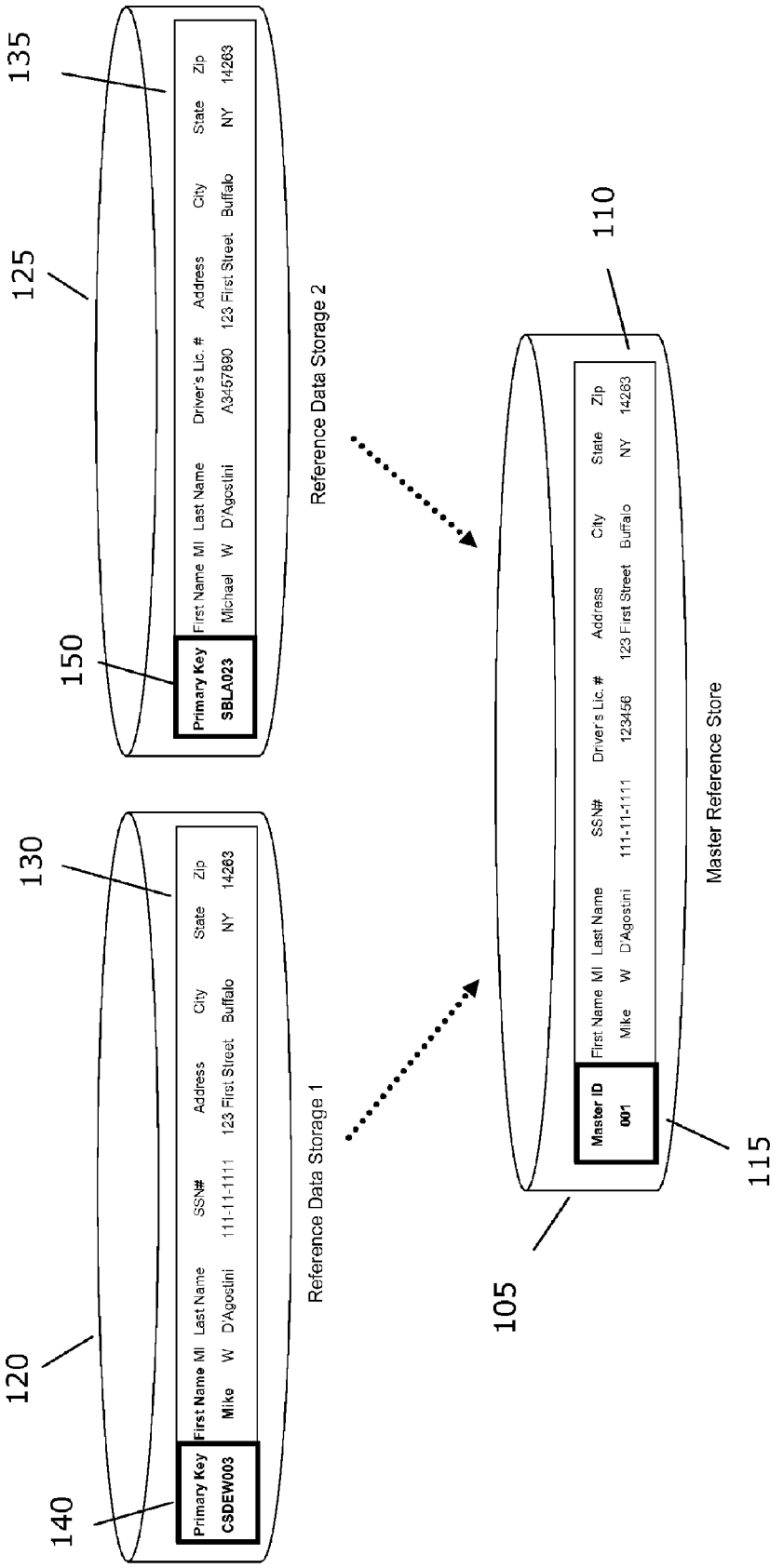

135
125
150
Primary Key
SBLA023
First Name MI Last Name Driver's Lic. # Address City State Zip
Michael W D'Agostini A3457890 123 First Street Buffalo NY 14263
Reference Data Storage 2
110
Master ID
001
First Name MI Last Name SSN# Driver's Lic. # Address City State Zip
Mike W D'Agostini 111-11-1111 123456 123 First Street Buffalo NY 14263
Master Reference Store
115
105
120
130
140
Primary Key
CSDEW003
First Name MI Last Name SSN# Address City State Zip
Mike W D'Agostini 111-11-1111 123 First Street Buffalo NY 14263
Reference Data Storage 1

*Figure 1*

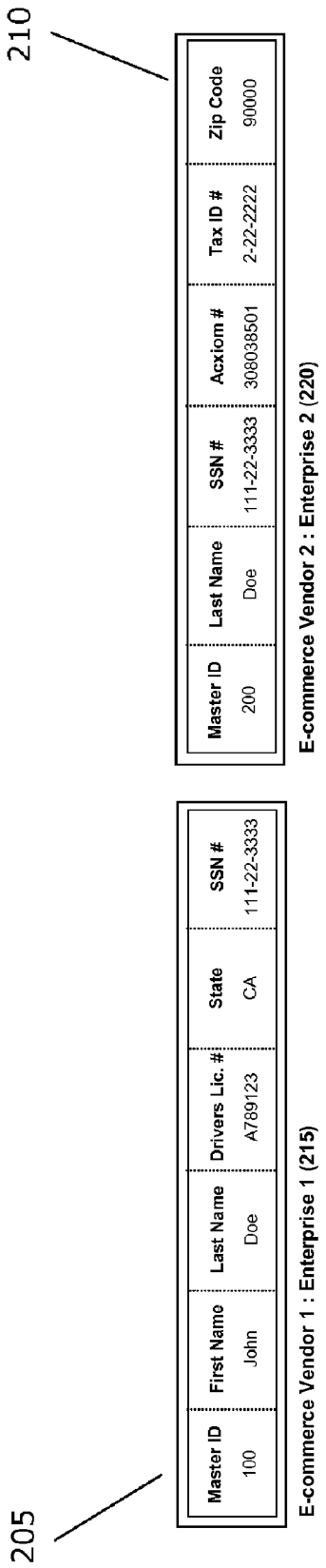

210
205
Master ID
200
Last Name
Doe
SSN #
111-22-3333
Acxiom #
308038501
Tax ID #
2-22-2222
Zip Code
90000
E-commerce Vendor 2 : Enterprise 2 (220)
Master ID
100
First Name
John
Last Name
Doe
Drivers Lic. #
A789123
State
CA
SSN #
111-22-3333
E-commerce Vendor 1 : Enterprise 1 (215)

*Figure 2*

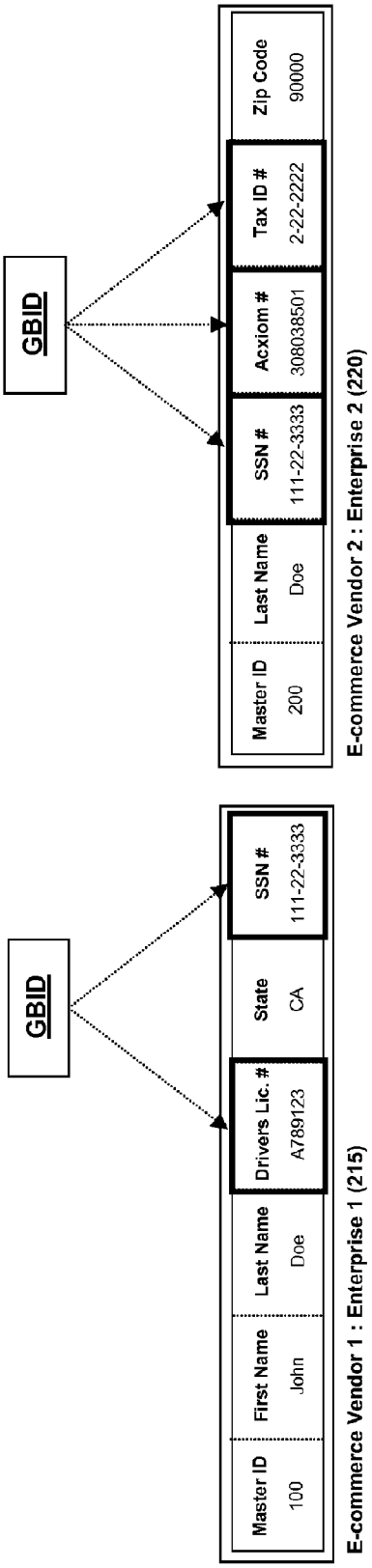

GBID
Master ID
100
First Name
John
Last Name
Doe
Drivers Lic. #
A789123
State
CA
SSN #
111-22-3333
E-commerce Vendor 1 : Enterprise 1 (215)
GBID
Master ID
200
Last Name
Doe
SSN #
111-22-3333
Acxiom #
308038501
Tax ID #
2-22-2222
Zip Code
90000
E-commerce Vendor 2 : Enterprise 2 (220)

*Figure 3*

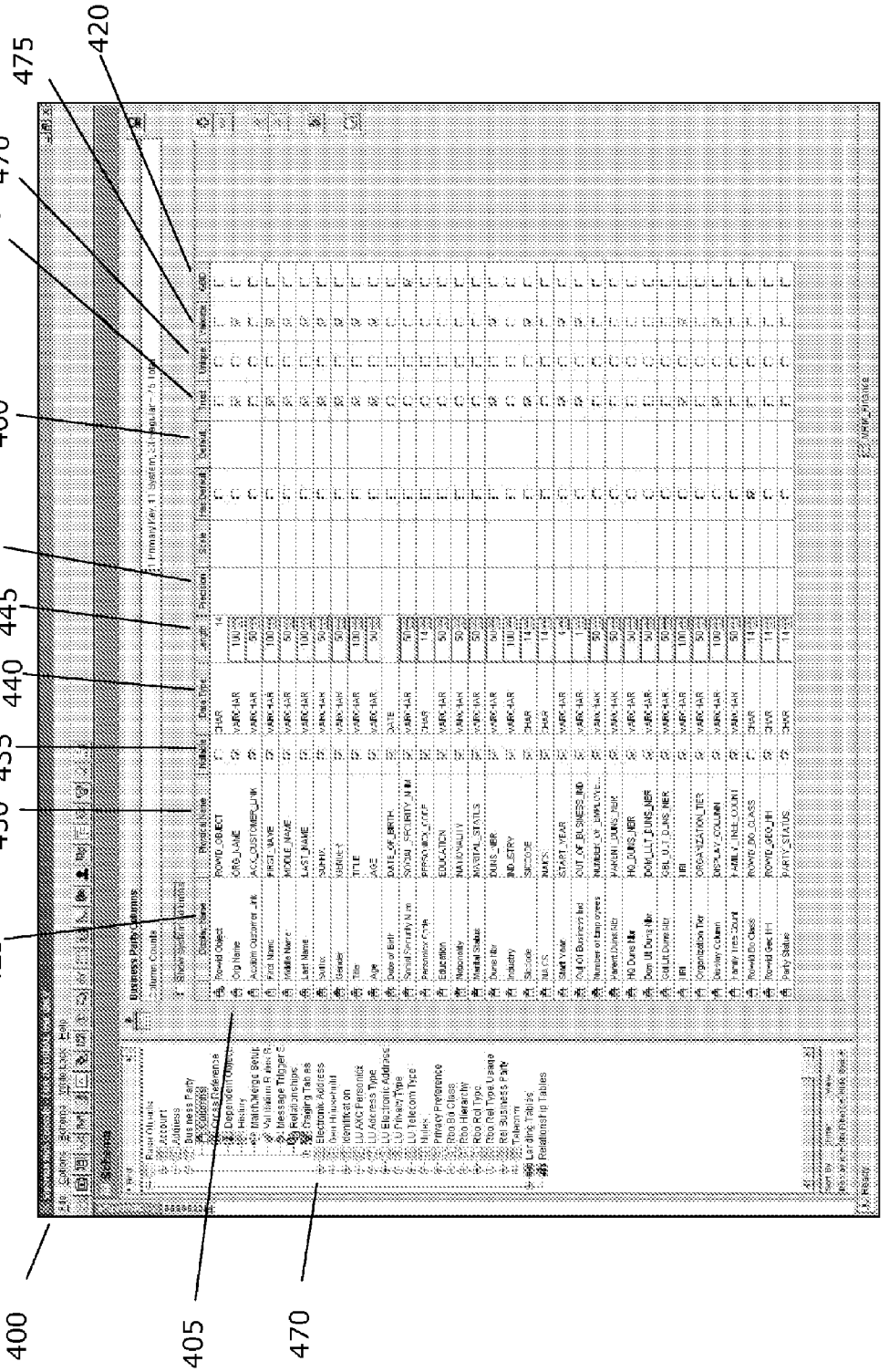



Figure 4

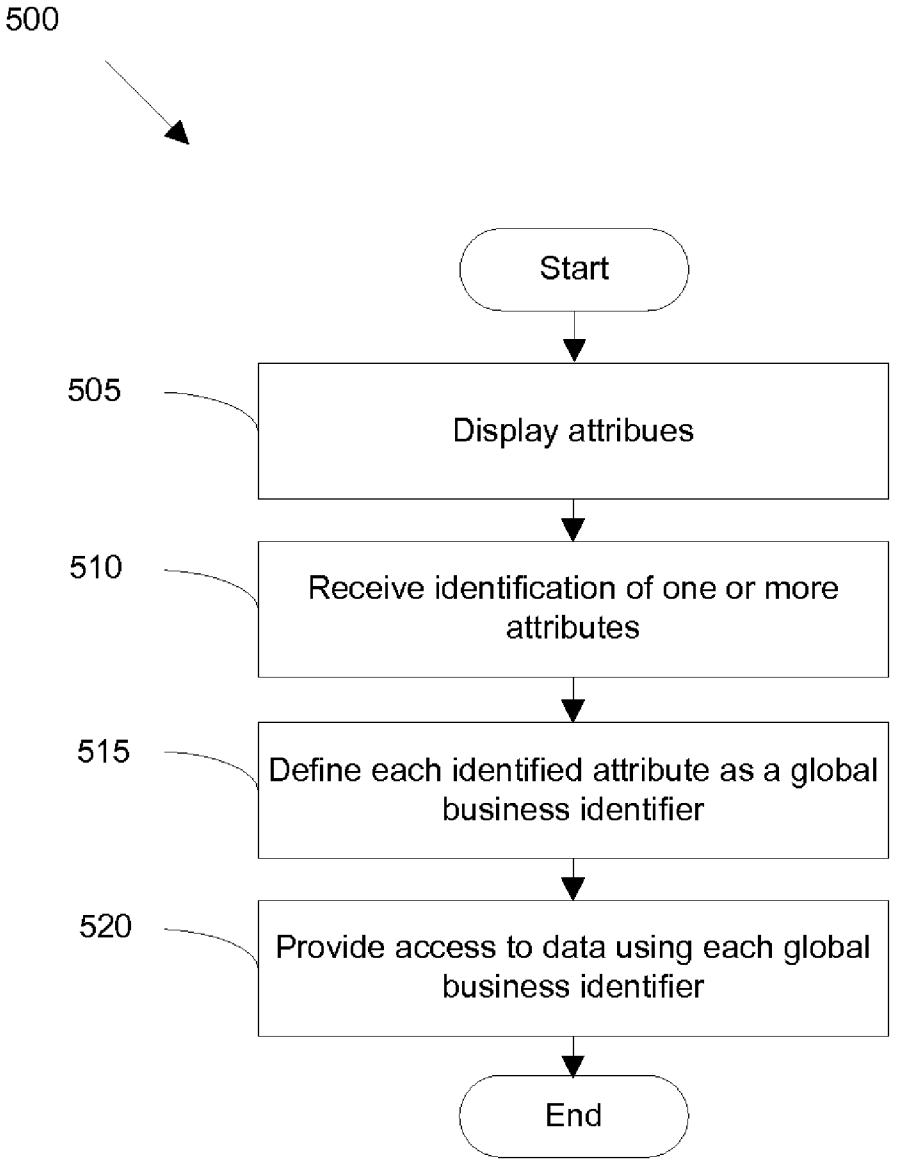

500
Start
505
Display attribues
510
Receive identification of one or more attributes
515
Define each identified attribute as a global business identifier
520
Provide access to data using each global business identifier
End

Figure 5

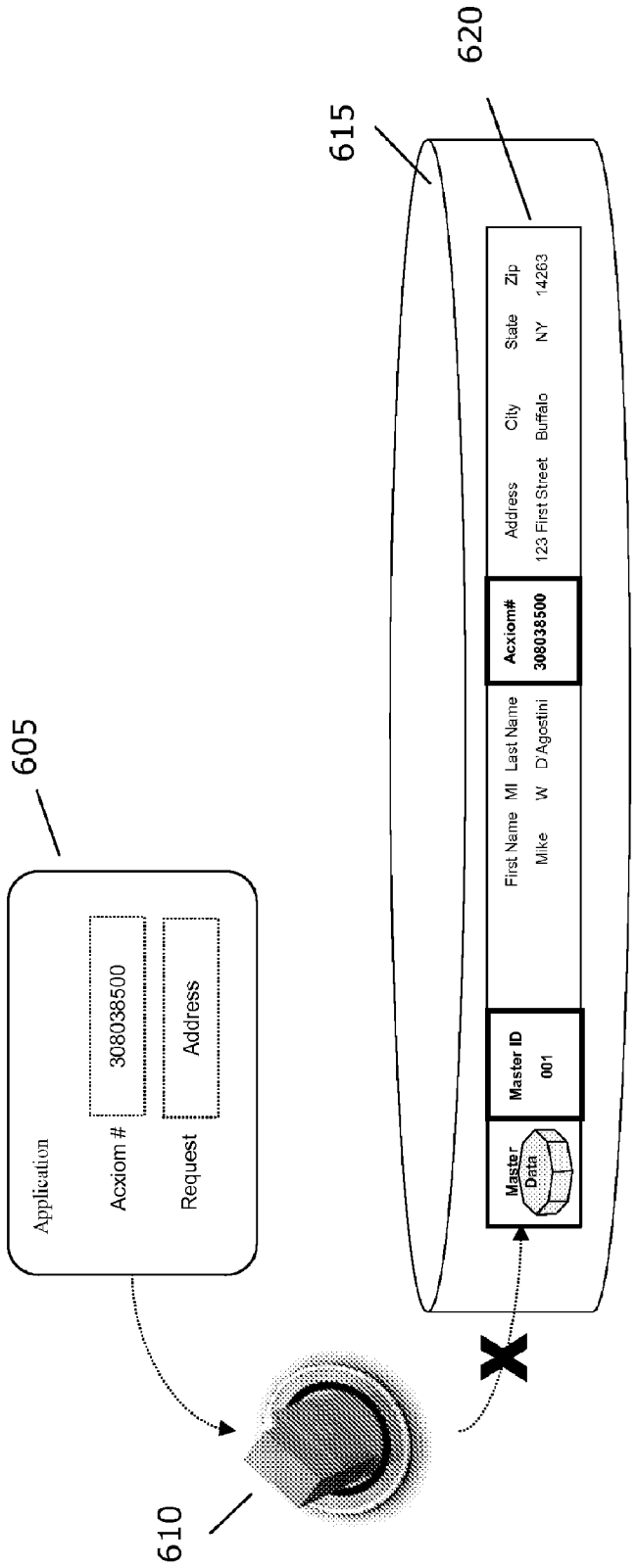

605
Application
Acxiom #
308038500
Request
Address
610
615
620
Master Data
Master ID
001
First Name MI Last Name
Mike W D'Agostini
Acxiom#
308038500
Address City State Zip
123 First Street Buffalo NY 14263

Figure 6

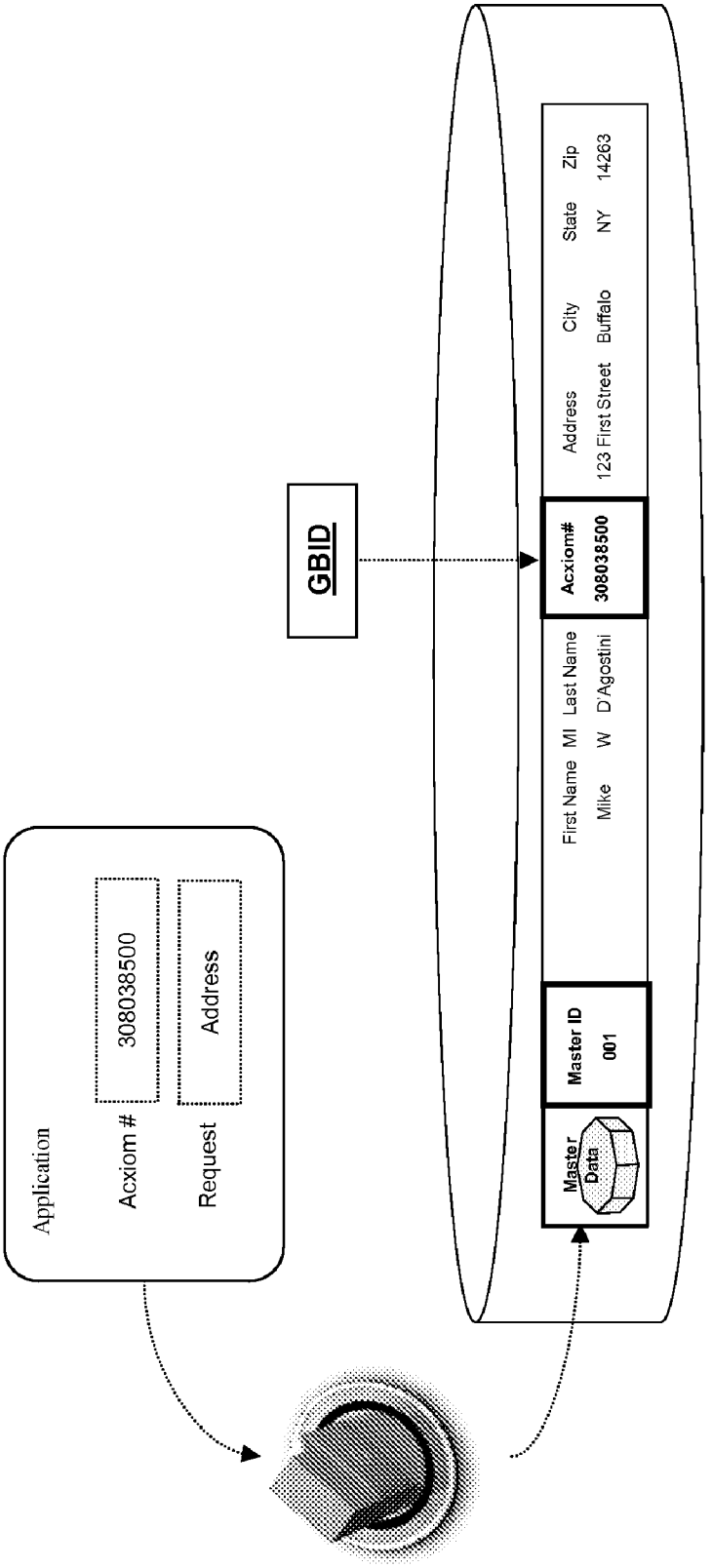

Application
Acxiom #
308038500
Request
Address
GBID
Master Data
Master ID
001
First Name MI Last Name
Mike W D'Agostini
Acxiom#
308038500
Address City State Zip
123 First Street Buffalo NY 14263

Figure 7

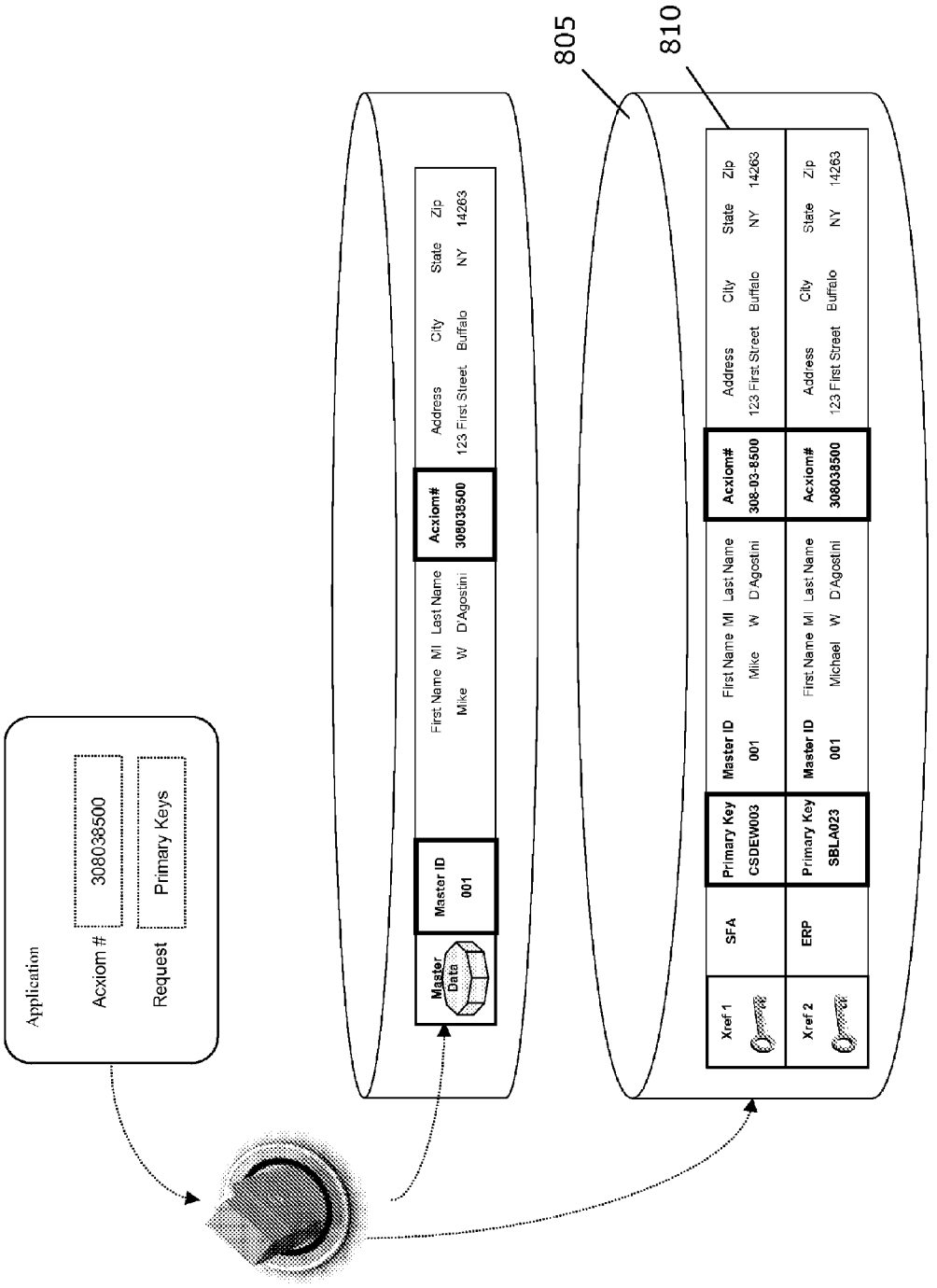

805
810
Application
Acxiom #
3080038500
Request
Primary Keys
Master Data
Master ID
001
First Name MI Last Name
Mike W D'Agostini
Acxiom#
308038500
Address City State Zip
123 First Street Buffalo NY 14263
Xref 1
SFA
Primary Key
CSDEW003
Master ID
001
First Name MI Last Name
Mike W D'Agostini
Acxiom#
308-03-8500
Address City State Zip
123 First Street Buffalo NY 14263
Xref 2
ERP
Primary Key
SBLA023
Master ID
001
First Name MI Last Name
Michael W D'Agostini
Acxiom#
308038500
Address City State Zip
123 First Street Buffalo NY 14263

*Figure 8*

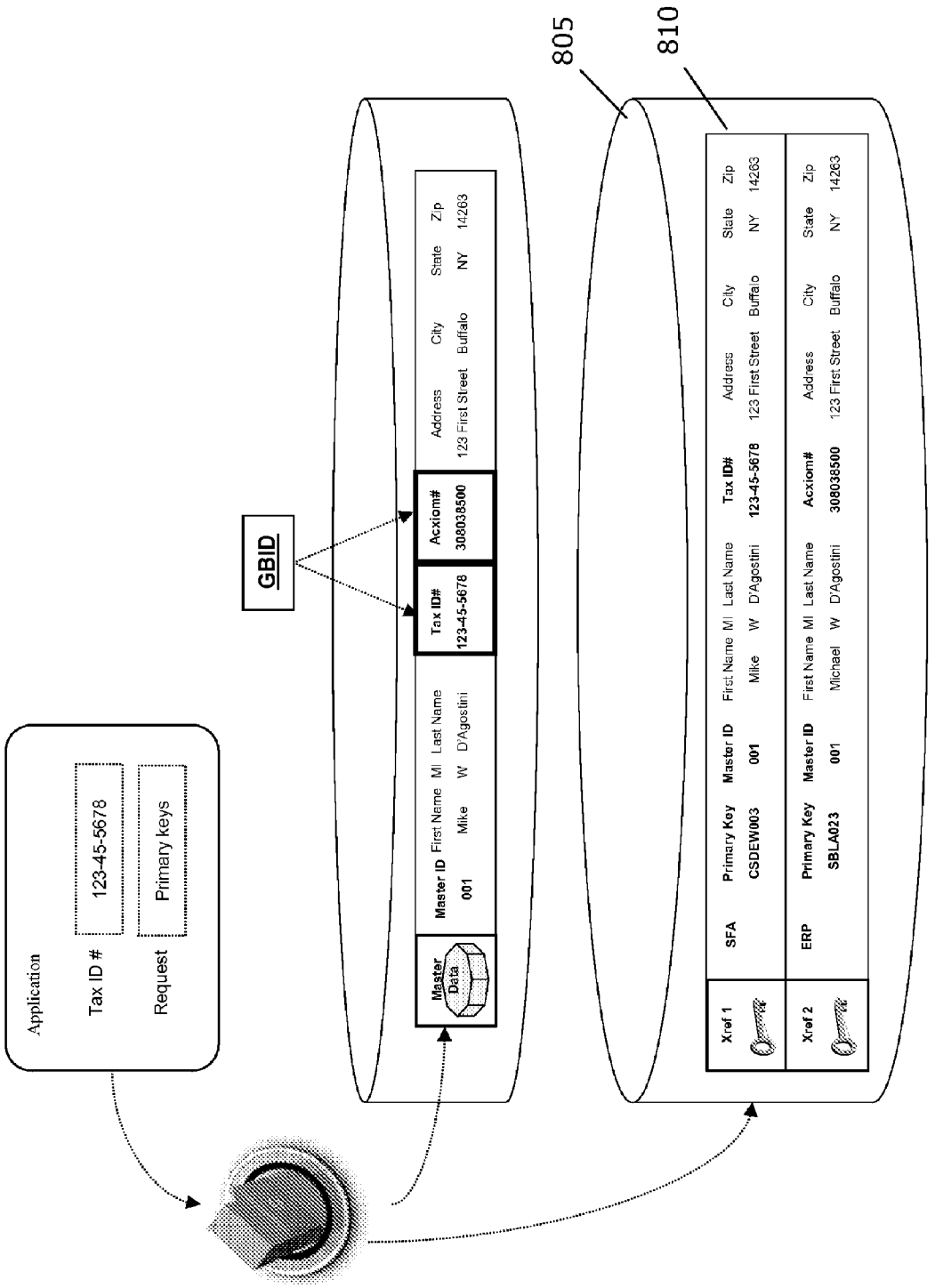

805
810
Application
Tax ID #
123-45-5678
Request
Primary keys
GBID
Master Data
Master ID
First Name
MI
Last Name
001
Mike
W
D'Agostini
Tax ID#
123-45-5678
Acxiom#
308038500
Address
123 First Street
City
Buffalo
State
NY
Zip
14263
Xref 1
SFA
Primary Key
CSDEW003
Master ID
001
First Name
MI
Last Name
Mike
W
D'Agostini
Tax ID#
123-45-5678
Address
123 First Street
City
Buffalo
State
NY
Zip
14263
Xref 2
ERP
Primary Key
SBLA023
Master ID
001
First Name
MI
Last Name
Michael
W
D'Agostini
Acxiom#
308038500
Address
123 First Street
City
Buffalo
State
NY
Zip
14263

*Figure 9*

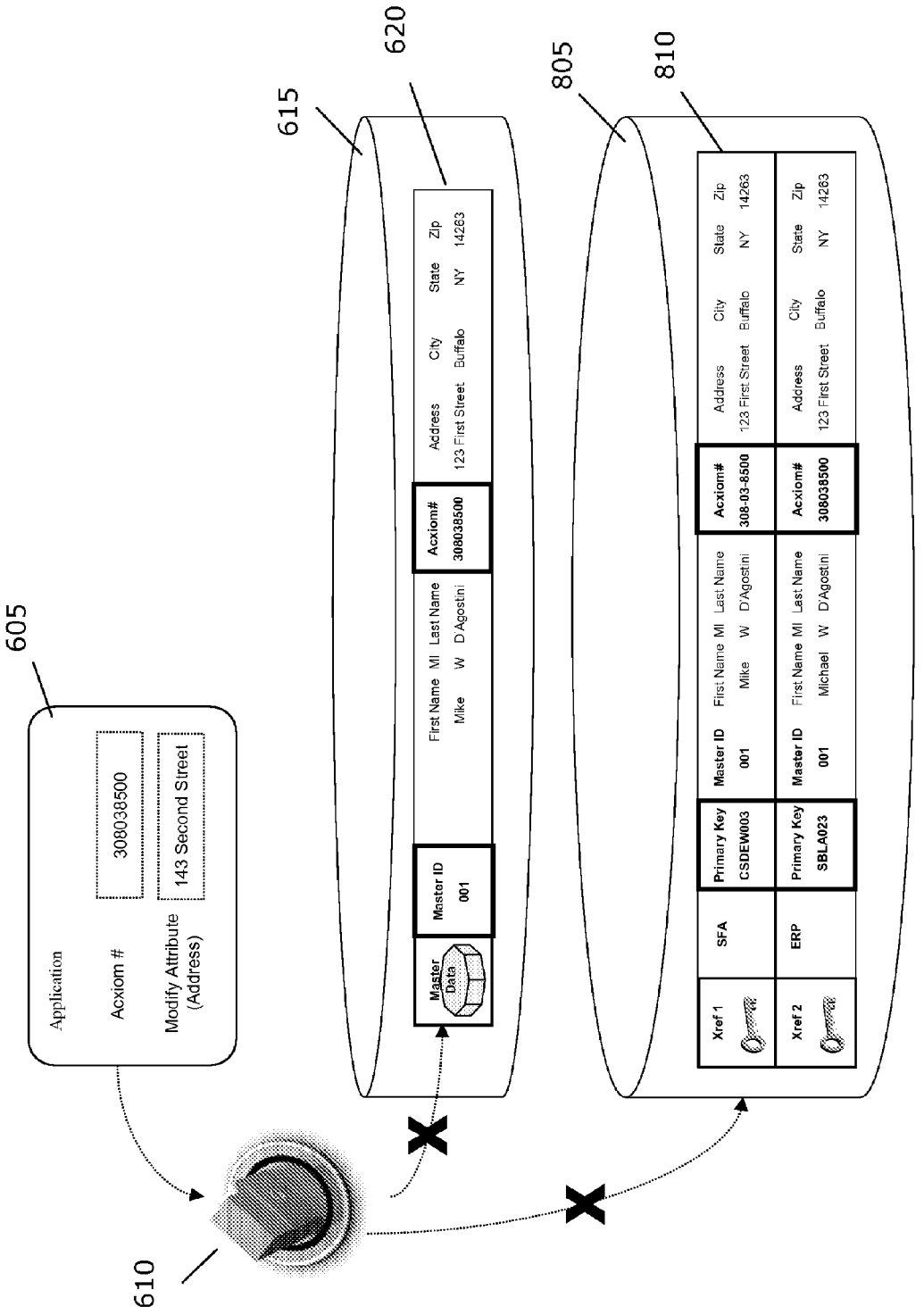

605
Application
Acxiom #
308038500
Modify Attribute (Address)
143 Second Street
610
615
620
Master Data
Master ID
001
First Name MI Last Name
Mike W D'Agostini
Acxiom#
308038500
Address City State Zip
123 First Street Buffalo NY 14263
805
810
Xref 1
SFA
Primary Key
CSDEW003
Master ID
001
First Name MI Last Name
Mike W D'Agostini
Acxiom#
308-03-8500
Address City State Zip
123 First Street Buffalo NY 14263
Xref 2
ERP
Primary Key
SBLA023
Master ID
001
First Name MI Last Name
Michael W D'Agostini
Acxiom#
308038500
Address City State Zip
123 First Street Buffalo NY 14263

*Figure 10*

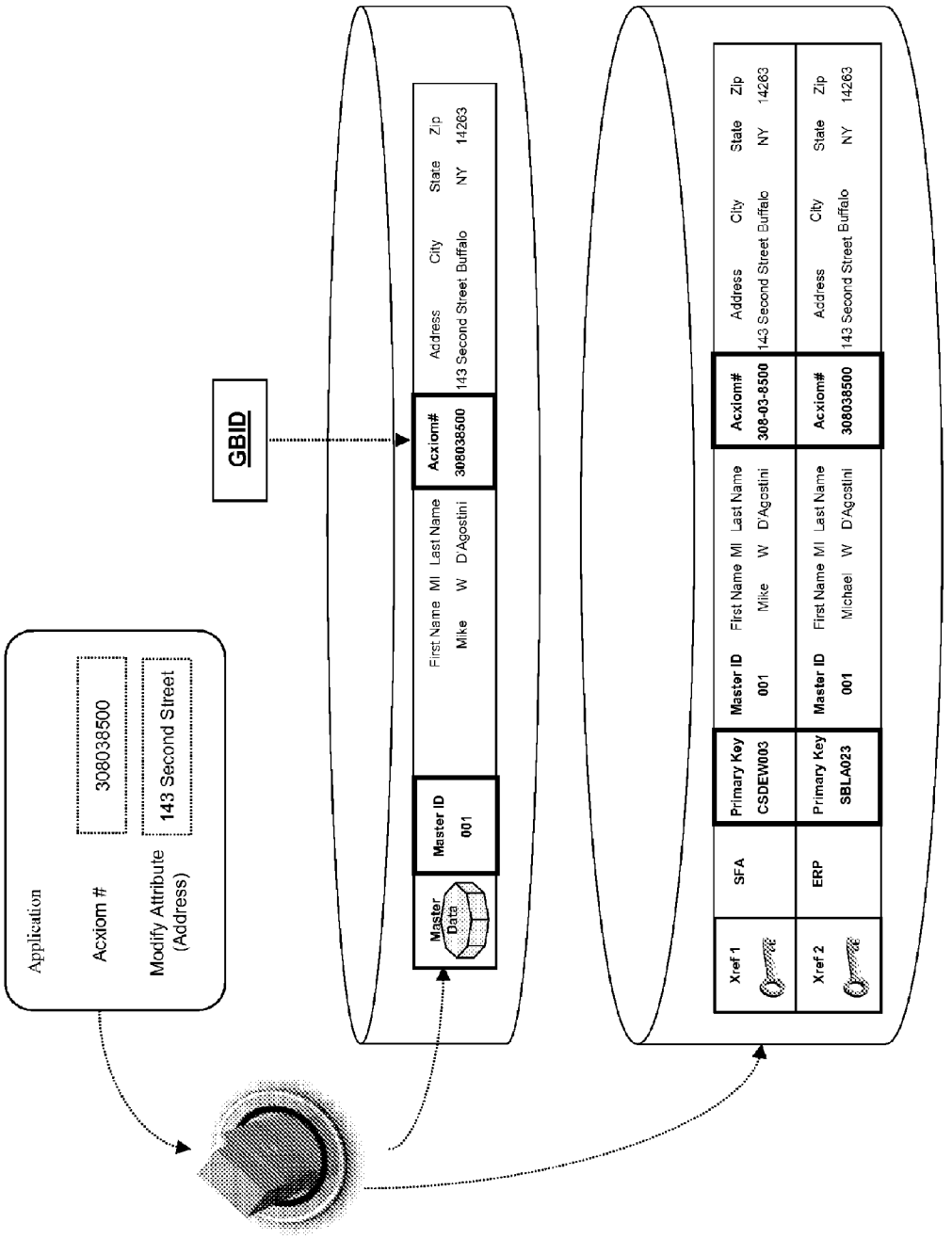

Application
Acxiom #
308038500
Modify Attribute (Address)
143 Second Street
GBID
Master Data
Master ID
001
First Name MI Last Name
Mike W D'Agostini
Acxiom#
308038500
Address City State Zip
143 Second Street Buffalo NY 14263
Xref 1
SFA
Primary Key
CSDEW003
Master ID
001
First Name MI Last Name
Mike W D'Agostini
Acxiom#
308-03-8500
Address City State Zip
143 Second Street Buffalo NY 14263
Xref 2
ERP
Primary Key
SBLA023
Master ID
001
First Name MI Last Name
Michael W D'Agostini
Acxiom#
308038500
Address City State Zip
143 Second Street Buffalo NY 14263

Figure 11

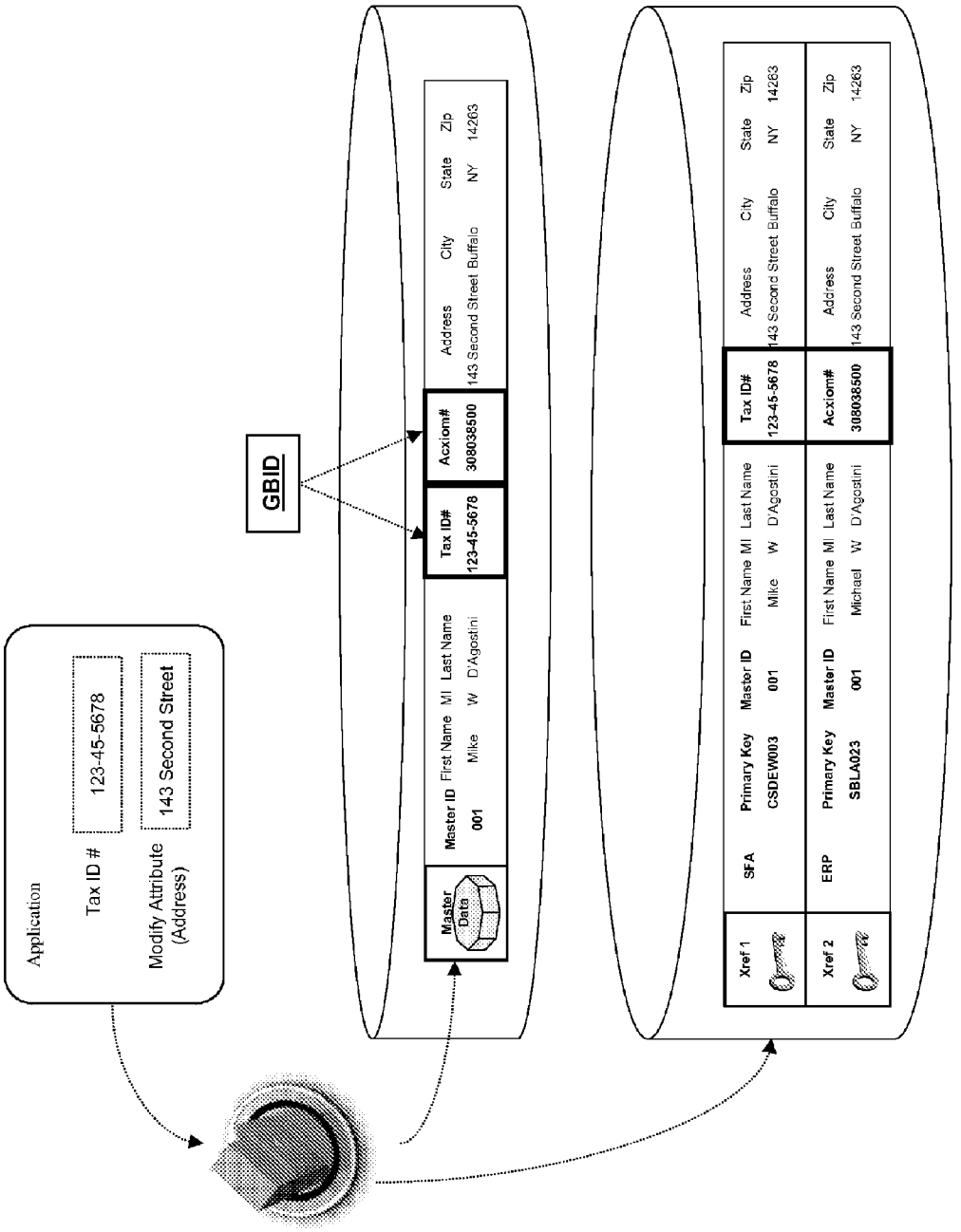

Application
Tax ID #
123-45-5678
Modify Attribute (Address)
143 Second Street
GBID
Master Data
Master ID
First Name
MI
Last Name
001
Mike
W
D'Agostini
Tax ID#
123-45-5678
Acxiom#
308038500
Address
City
State
Zip
143 Second Street
Buffalo
NY
14263
Xref 1
SFA
Primary Key
CSDEW003
Master ID
001
Xref 2
ERP
Primary Key
SBLA023
Master ID
001
Michael

Figure 12

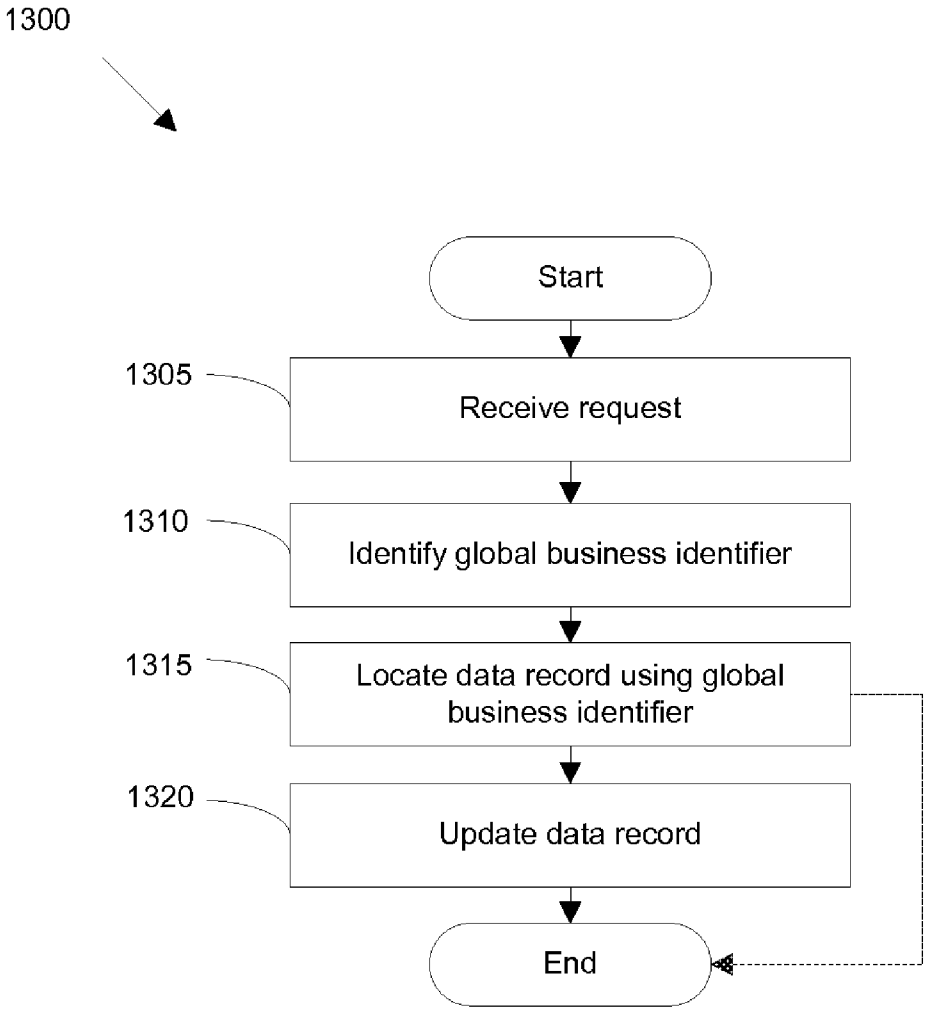

1300
Start
1305
Receive request
1310
Identify global business identifier
1315
Locate data record using global business identifier
1320
Update data record
End

Figure 13

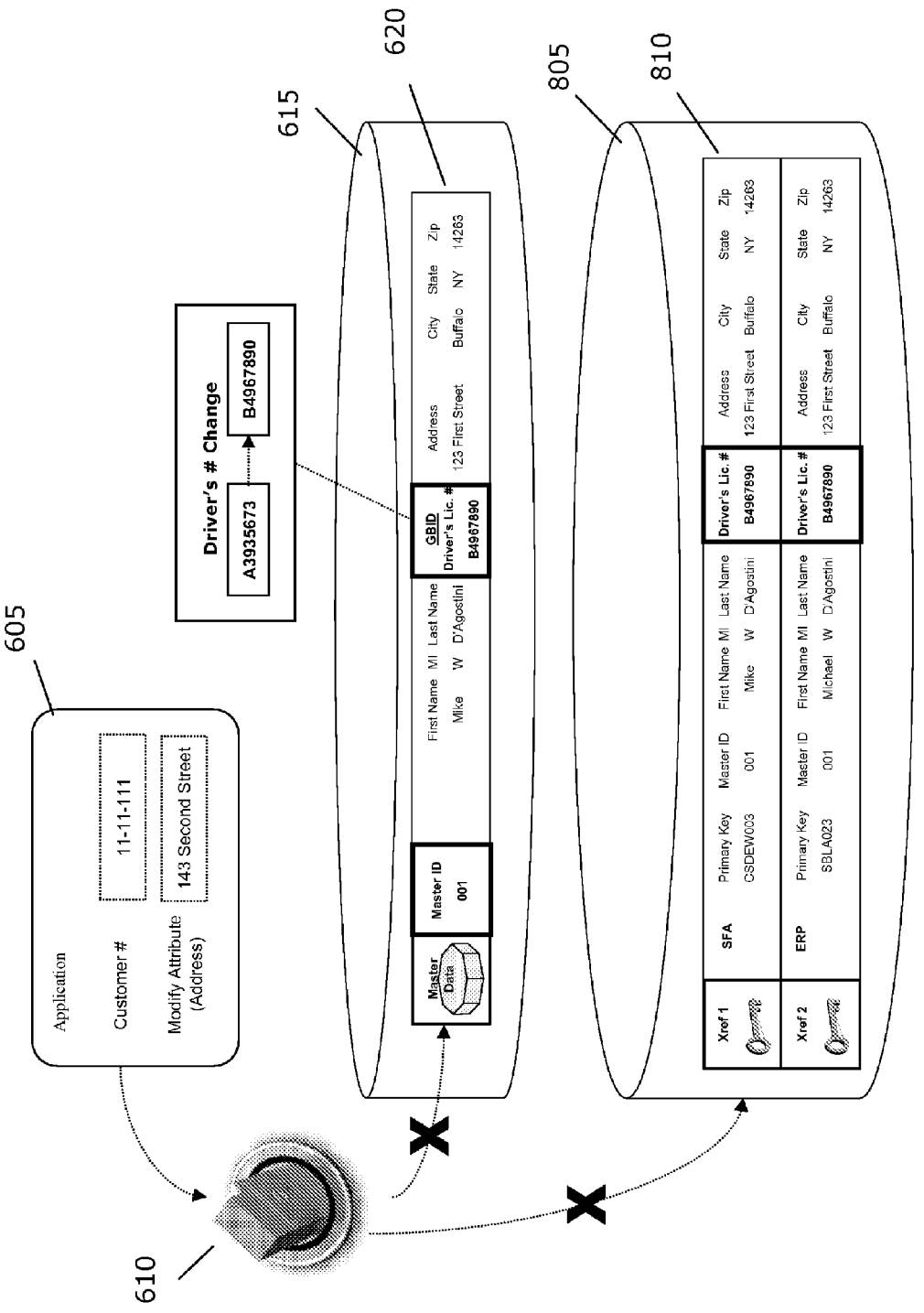

605
Application
Customer #
11-11-111
Modify Attribute (Address)
143 Second Street
610
615
620
Driver's # Change
A3935673
B4967890
Master Data
Master ID
001
GBID
Driver's Lic. #
B4967890
First Name MI Last Name
Mike W D'Agostini
Address City State Zip
123 First Street Buffalo NY 14263
805
810
Xref 1
Xref 2
SFA
ERP
Primary Key Master ID First Name MI Last Name
CSDEW003 001 Mike W D'Agostini
SBLA023 001 Michael W D'Agostini
Driver's Lic. #
B4967890
Address City State Zip
123 First Street Buffalo NY 14263

Figure 14

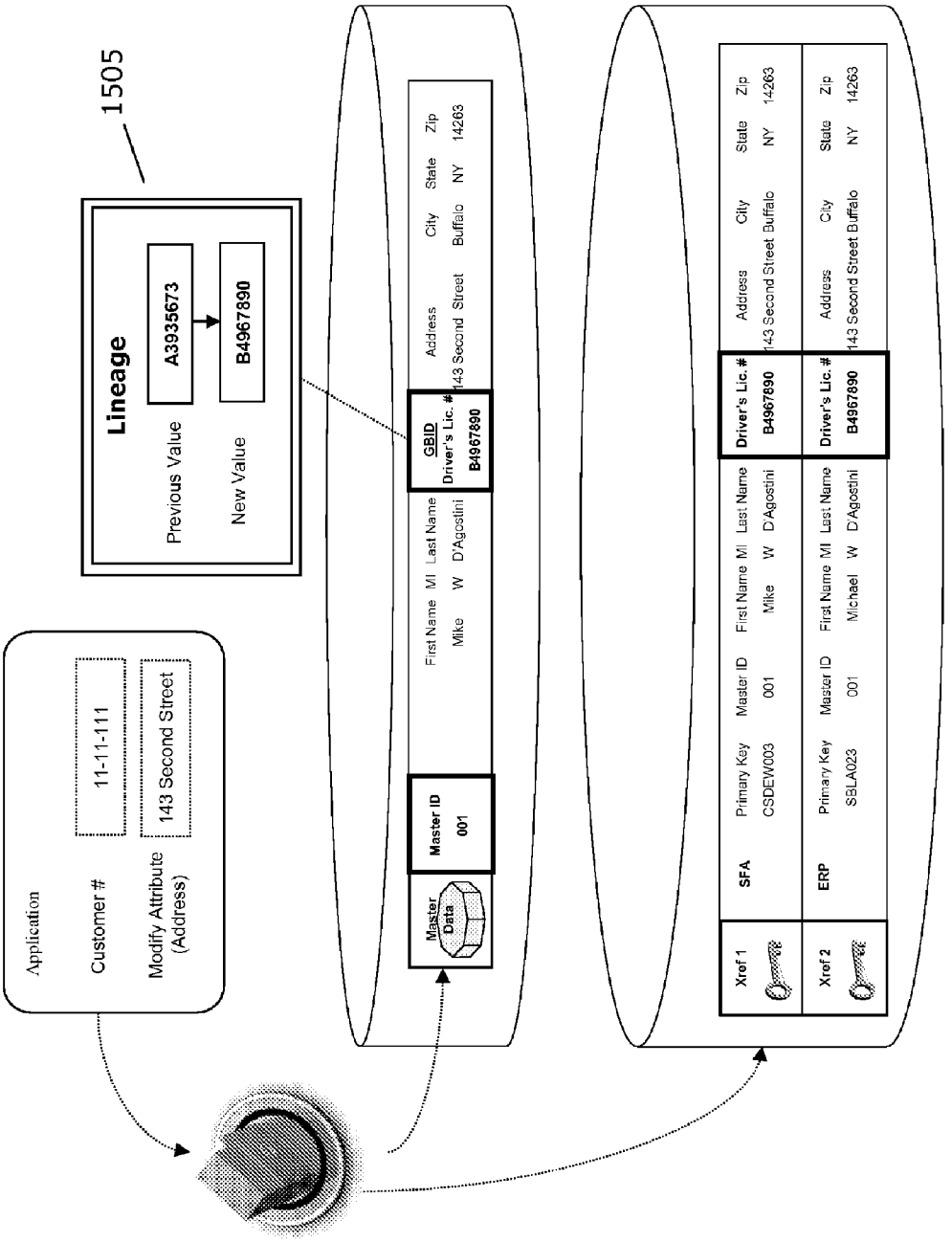

1505
Lineage
Previous Value
A3935673
New Value
B4967890
Application
Customer #
11-11-111
Modify Attribute (Address)
143 Second Street
Master Data
Master ID
001
First Name MI Last Name
Mike W D'Agostini
GBID
Driver's Lic. #
B4967890
Address City State Zip
143 Second Street Buffalo NY 14263
Xref 1
SFA
Primary Key Master ID First Name MI Last Name
CSDEW003 001 Mike W D'Agostini
Driver's Lic. #
B4967890
Address City State Zip
143 Second Street Buffalo NY 14263
Xref 2
ERP
Primary Key Master ID First Name MI Last Name
SBLA023 001 Michael W D'Agostini
Driver's Lic. #
B4967890
Address City State Zip
143 Second Street Buffalo NY 14263

*Figure 15*

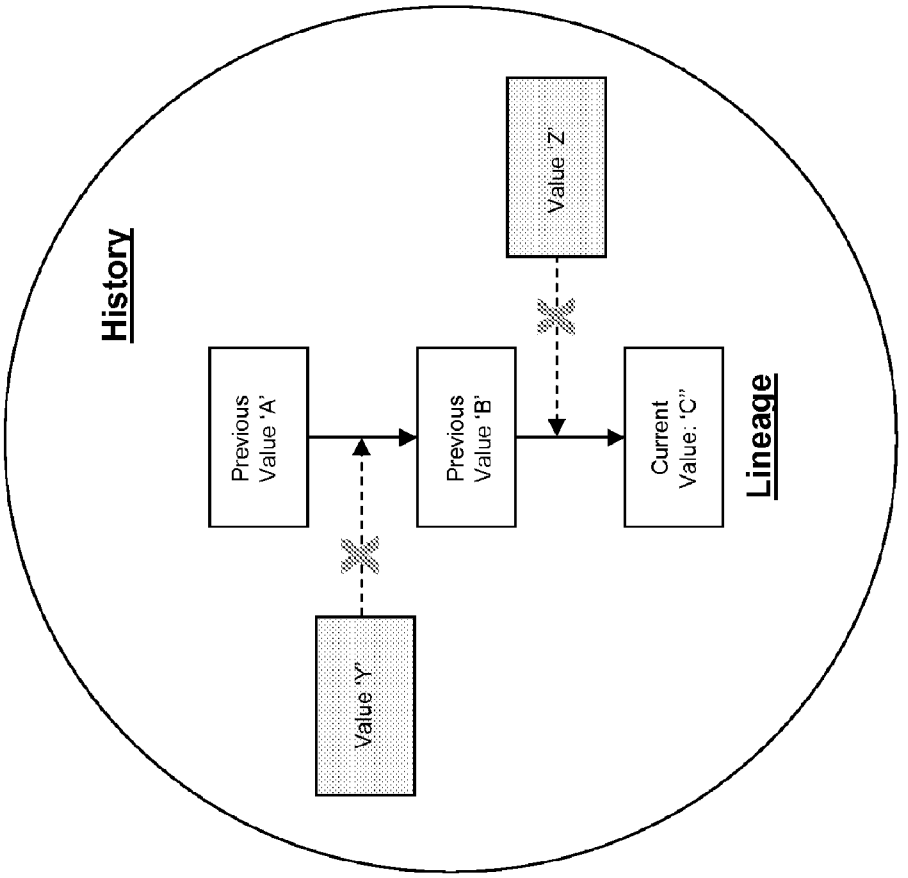

History
Previous Value 'A'
Previous Value 'B'
Current Value: 'C''
Lineage
Value 'Y'
Value 'Z'

Figure 16

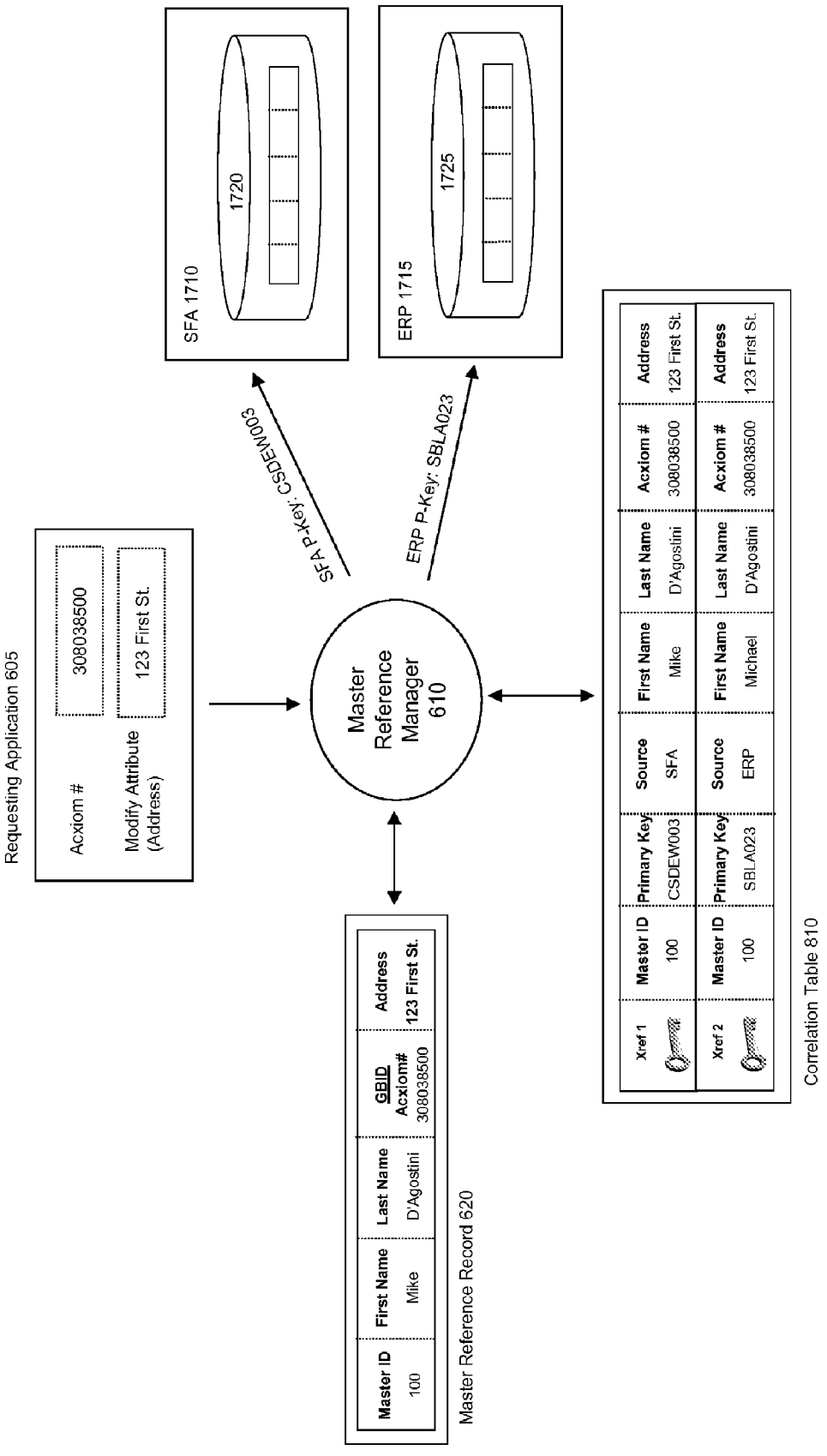

Requesting Application 605
Acxiom #
308038500
Modify Attribute (Address)
123 First St.
SFA 1710
1720
ERP 1715
1725
SFA P-Key: CSDEW003
ERP P-Key: SBLA023
Master Reference Manager 610
Master ID
100
First Name
Mike
Last Name
D'Agostini
GBID Acxiom#
308038500
Address
123 First St.
Master Reference Record 620
Xref 1
Master ID
100
Primary Key
CSDEW003
Source
SFA
First Name
Mike
Last Name
D'Agostini
Acxiom #
308038500
Address
123 First St.
Xref 2
Master ID
100
Primary Key
SBLA023
Source
ERP
First Name
Michael
Last Name
D'Agostini
Acxiom #
308038500
Address
123 First St.
Correlation Table 810

*Figure 17*

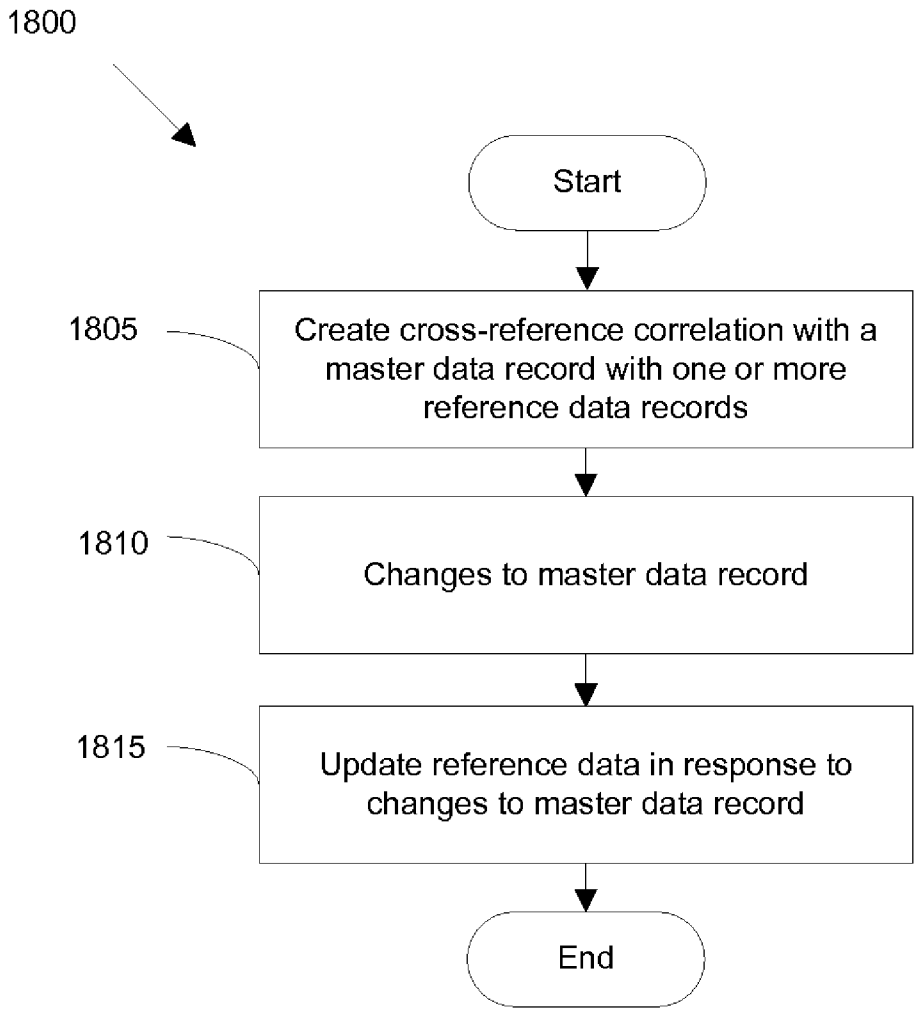

1800
Start
1805
Create cross-reference correlation with a master data record with one or more reference data records
1810
Changes to master data record
1815
Update reference data in response to changes to master data record
End

Figure 18

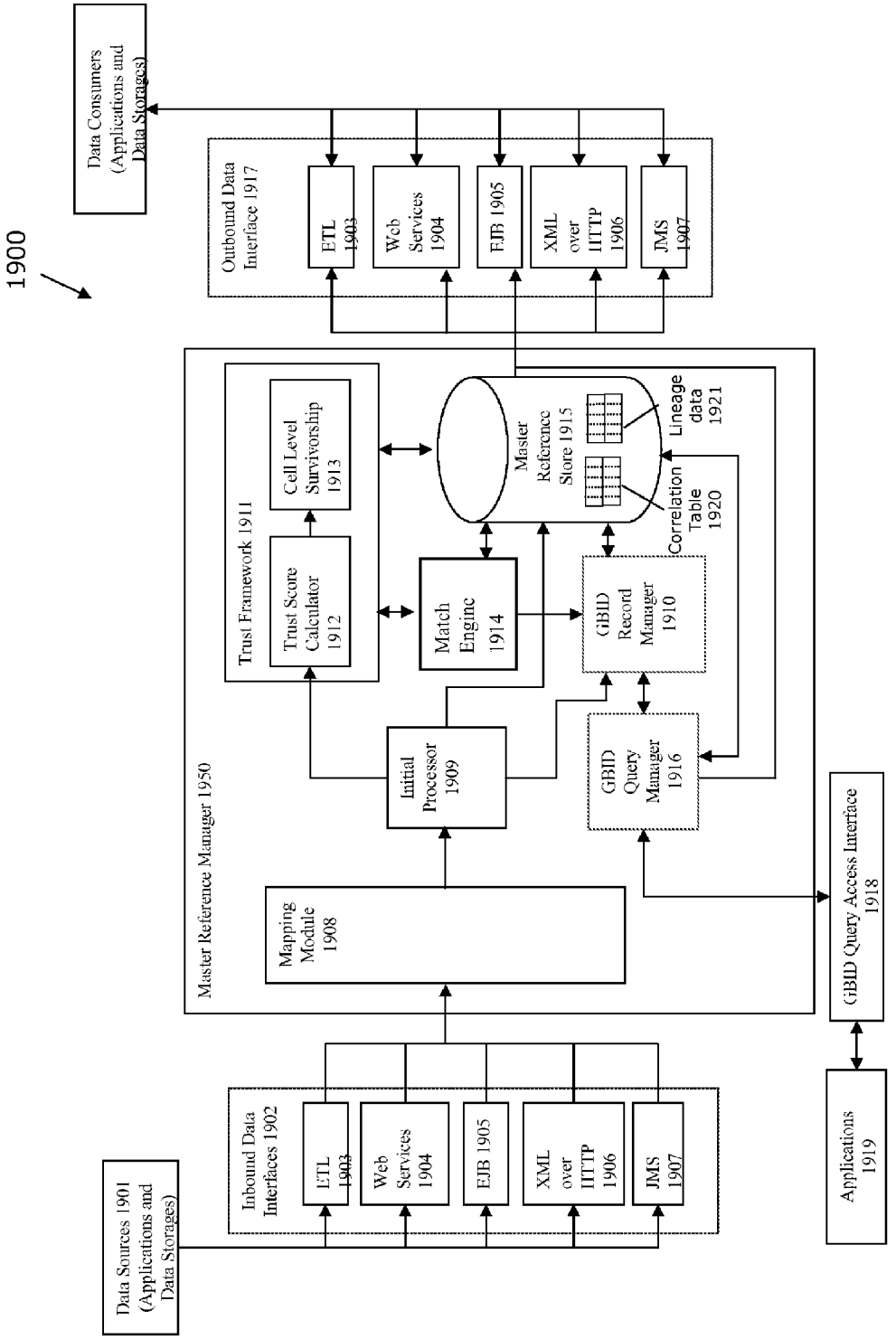

1900
Data Consumers (Applications and Data Storages)
Outbound Data Interface 1917
ETL 1903
Web Services 1904
EJB 1905
XML over HTTP 1906
JMS 1907
Master Reference Manager 1950
Trust Framework 1911
Cell Level Survivorship 1913
Trust Score Calculator 1912
Master Reference Store 1915
Lineage data 1921
Correlation Table 1920
Match Engine 1914
GBID Record Manager 1910
Initial Processor 1909
GBID Query Manager 1916
Mapping Module 1908
GBID Query Access Interface 1918
Applications 1919
Inbound Data Interfaces 1902
ETL 1903
Web Services 1904
EJB 1905
XML over HTTP 1906
JMS 1907
Data Sources 1901 (Applications and Data Storages)

Figure 19

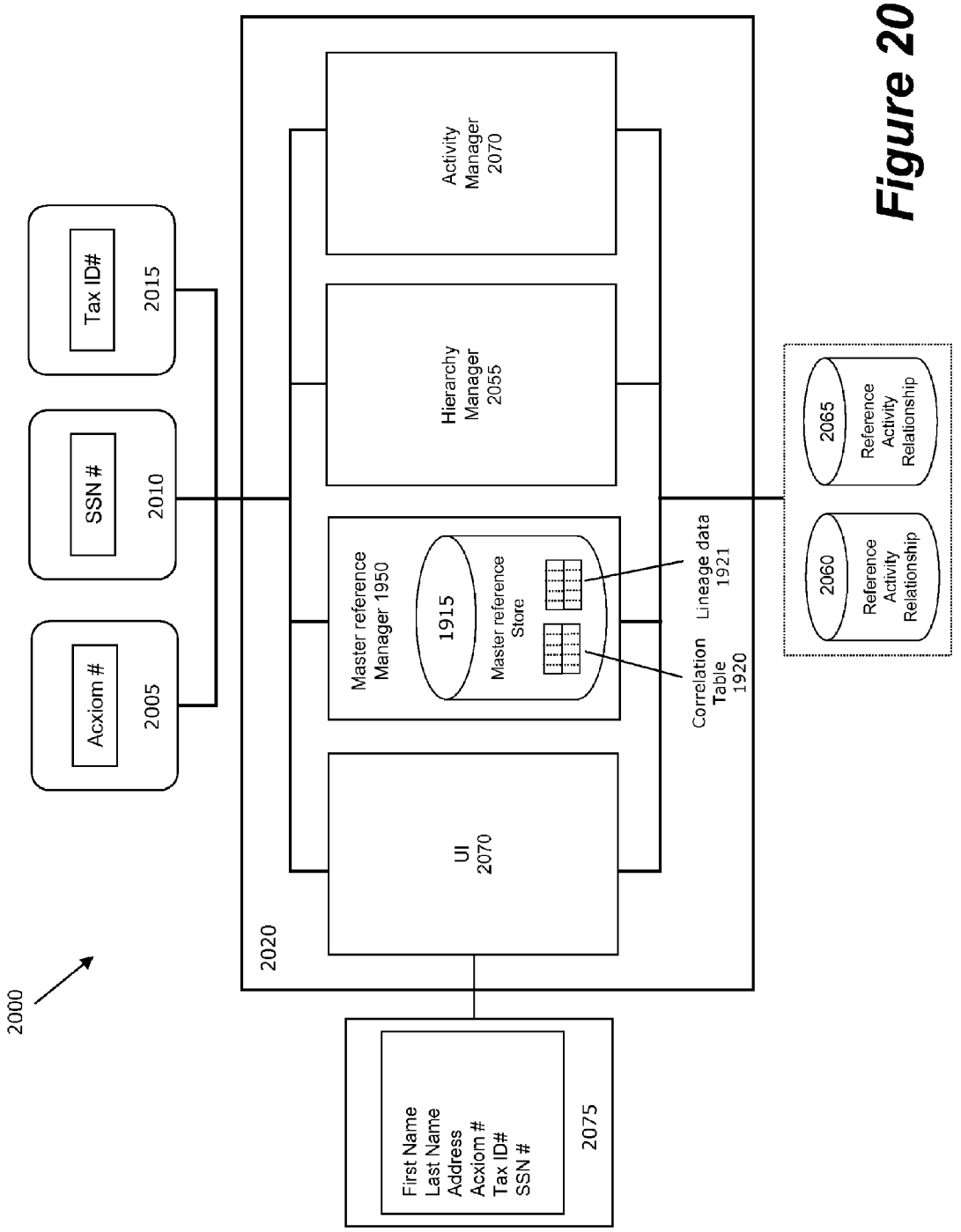

2000
Acxiom #
2005
SSN #
2010
Tax ID#
2015
2020
UI
2070
Master reference
Manager 1950
1915
Master reference
Store
Correlation
Table
1920
Lineage data
1921
Hierarchy
Manager
2055
Activity
Manager
2070
2060
Reference
Activity
Relationship
2065
Reference
Activity
Relationship
First Name
Last Name
Address
Acxiom #
Tax ID#
SSN #
2075

Figure 20

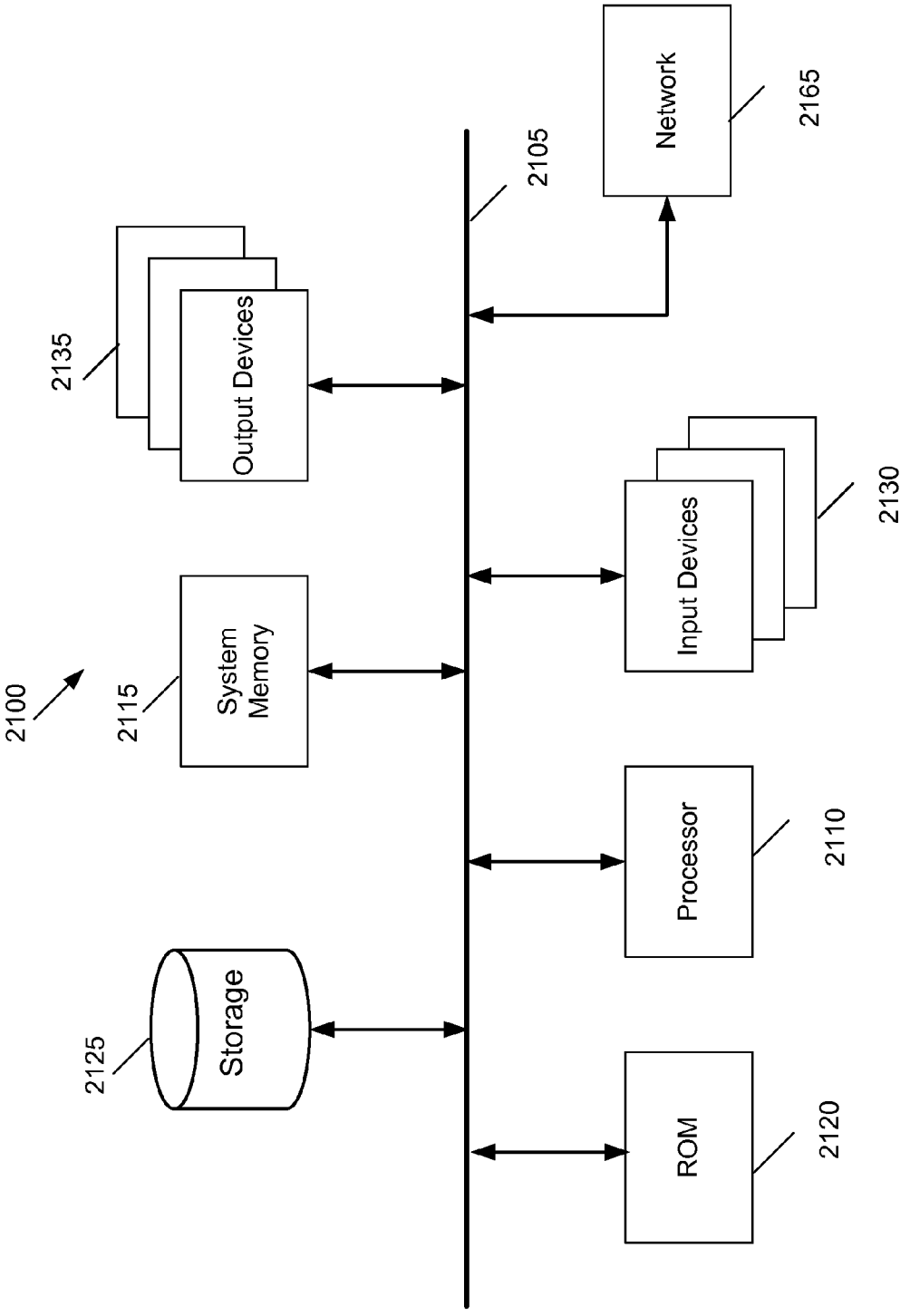

2100
2105
Network
2165
Output Devices
2135
System Memory
2115
Input Devices
2130
Processor
2110
Storage
2125
ROM
2120

Figure 21

# METHODS AND SYSTEMS FOR ACCESSING DATA

## CLAIM OF BENEFIT TO PRIOR APPLICATIONS

**[0001]** This application claims benefit to U.S. Provisional Application 60/951,187, filed Jul. 20, 2007, and also claims benefit to U.S. Provisional Application 60/970,257, filed Sep. 5, 2007. These Provisional Applications are incorporated herein by reference.

## FIELD OF THE INVENTION

**[0002]** The present invention relates to the field of data integration and management, and in particular, to methods and systems for accessing data.

## BACKGROUND OF THE INVENTION

**[0003]** In many different enterprises, multiple applications and data sources exist which creates challenges in reconciling and managing a unified view of a data entity across an enterprise. Master data represents the common or shared entities to the transactions that record the operations of the enterprise across the various applications and data storages. Although the list is non-exhaustive, examples of entities include organizations, enterprises, companies, customers, individuals, services, accounts, products, etc. When an enterprise attempts data integration or data sharing, a fundamental problem with master data is how it is identified by the applications and data sources, as well the users (e.g., business users) trying to access this data.

**[0004]** A conventional approach taken by many data integration solution providers to solve the data integration and sharing problem is to standardize using a single unifying identifier that uniquely identifies an entity across all source systems. However, although it appears practical at a high level, this approach is not without its share of problems. For instance, the approach requires significant changes to the existing systems and the underlying business processes to adhere to and accommodate the unifying identifier. Any such pervasive changes across an enterprise come at a significant financial cost and run the risk of breaking the existing systems, including the underlying business processes.

**[0005]** Further, any subsequent changes to a data model require extensive customization and coding across many of the existing applications and data storages. Such an approach to standardize may also lead to an enterprise losing data. For example, some enterprises have "legacy" data silos, with many different applications built on top of them, that are resistant to change and cannot easily be standardized.

**[0006]** Moreover, such pervasive changes across an enterprise ignore the business users of the various applications and data sources as it forces the designated identifier on those business users. For instance, such change may require the business users to start utilizing an identifier that is not specific for their business needs. For the users, it is easier for them to remember and utilize the identifiers they use in a business context (e.g., driver's license number, Dun & Bradstreet number, tax identification number), rather than an identifier implemented during the integration process.

**[0007]** Accordingly, there is a need in the art for a data integration system that can easily integrate data from several disparate sources. Also, there is a need in the art for a data management system that can flexibly manage data.

## SUMMARY OF THE INVENTION

**[0008]** Some embodiments of the invention provide a user interface ("UI") that allows a user to specify one or more attributes that should be included in a master reference data set, and identify which of these attributes should serve as enterprise specified identifiers ("ESIDs") that can be used to identify the particular master reference data set (e.g., master reference record) in an enterprise data storage (e.g., in a database). Some embodiments of the invention provide a method that allows the master reference data set to be accessed and updated in the data storage through the use of the ESIDs.

**[0009]** For each particular ESID of a master reference data set, some embodiments maintain not only the current value associated with the ESID but also the prior value or values associated with the ESID. In some cases, these prior historical values are not stored for each ESID of the master reference data set but are only stored for one or more of the ESIDs. Some embodiments allow the master reference record to be accessed and updated not only through the current value associated with the ESID but also though one or more of the previous values associated with the ESID. In other words, these embodiments can query a data storage with an old value of an ESID and still be able to identify and update the particular master reference data set.

**[0010]** Several examples are given below where the enterprise is a business organization. In these contexts, the ESID is referred to as a global business identifier ("GBID") because, once they are identified, they are used to drive various business processes of the business organization.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0011]** The novel features of the invention are set forth in the appended claims. However, for purpose of explanation, several embodiments are set forth in the following figures.

**[0012]** FIG. **1** provides an illustrative example of a master reference record.

**[0013]** FIG. **2** provides an illustrative example of several template master reference data sets in different enterprises.

**[0014]** FIG. **3** provides an illustrative example of several enterprises that specify GBIDs.

**[0015]** FIG. **4** provides an example of a UI that allows an enterprise to identify which attributes of a master reference data set should serve as GBIDs.

**[0016]** FIG. **5** illustrates a process that some embodiments utilize to provide an access path using GBIDs.

**[0017]** FIGS. **6-9** provide several illustrative examples of data records that are located and retrieved through the use of GBIDs.

**[0018]** FIGS. **10-12** provide several illustrative examples of data records that are updated through the use of GBIDs.

**[0019]** FIG. **13** illustrates a process that some embodiments utilize to access data sets in a data storage.

**[0020]** FIGS. **14-15** provides several illustrative examples regarding accessing and updating a master data record through lineage that is maintained for a GBID.

**[0021]** FIG. **16** provides a conceptual illustration of historical that can be maintained by some embodiments.

**[0022]** FIG. **17** presents an illustrative example of updating reference records in response to an update to the master reference record.

**[0023]** FIG. **18** illustrates a process that some embodiments utilize to update reference data sets in response to an update to a master reference data set.

**[0024]** FIG. **19** illustrates a data flow for a system according to some embodiments of the invention.

**[0025]** FIG. **20** provides an illustrative example of a system that implements some embodiments of the invention.

**[0026]** FIG. **21** conceptually illustrates a computer system with which some embodiments of the invention are implemented.

## DETAILED DESCRIPTION OF THE INVENTION

**[0027]** In the following description, numerous details are set forth for the purpose of explanation. However, one of ordinary skill in the art will realize that the invention may be practiced without the use of these specific details. In other instances, well-known structures and devices are shown in block diagram form in order not to obscure the description of the invention with unnecessary detail.

**[0028]** Some embodiments of the invention provide a UI that allows a user to specify one or more attributes that should be included in a master reference data set, and identify which of these attributes should serve as GBIDs that can be used to identify the particular master reference data set (e.g., master reference record) in an enterprise data storage (e.g., in a database). Some embodiments of the invention provide a method that allows the master reference data set to be accessed and updated in the data storage through the use of the GBIDs.

**[0029]** For each particular GBID of a master reference data set, some embodiments maintain not only the current value associated with the GBID but also the prior value or values associated with the GBID. In some cases, these prior historical values are not stored for each GBID of the master reference data set but are only stored for one or more of the GBIDs. Some embodiments allow the master reference record to be accessed and updated not only through the current value associated with the GBID but also though one or more of the previous values associated with the GBID. In other words, these embodiments can query a data storage with an old value of a GBID and still be able to identify and update the particular master reference data set.

**[0030]** In the discussion below, Section I provides an overview of defining a master reference data set and providing a direct access path to the master reference data set through the use of GBIDs. Section II provides several examples related to accessing data sets through the use of GBIDs. Section II also introduces and provides examples related to the concepts of access by lineage and cross-reference correlation. Section III provides a data flow for a system according to some embodiments of the invention. Section IV provides an exemplary system that implements some embodiments of the invention. Section V describes a computer system with which some embodiments are implemented.

### I. Overview

**[0031]** In many enterprises, data related to an entity such as customer or product is strewn across various systems (e.g., applications and data storages). Several of these systems have different data models and thus store their own version of the entity data. In other words, the data storages of an enterprise might store different data records for a particular entity. This redundant data may cause problems for the enterprise that uses the data. Therefore, some embodiments of the invention store a master reference data set (e.g., master reference record) or a "best version" of the reference data records for at least some of the entities in a master reference store.

**[0032]** FIG. **1** provides an illustrative example of a master reference record for an individual. The figure includes a master reference record **110** that is stored in a master reference store **105** and several reference records (**130** and **135**) that are stored in reference data storages (**120** and **125**). Each reference record has a primary key field (**140** or **150**) and several other data fields (e.g., first name, last name, address, etc.). The primary key is a unique internal key associated with a particular data record that is used in accessing (e.g., locating, updating) that particular record. The data fields are attributes that store data values that define the data record. In the example illustrated, the reference records (**130** and **135**) have different data fields for the same individual. For instance, the reference data record **130** in the first data storage **120** includes a field for a social security number but does not include a field for a driver's license number. Also, these reference records (**130** and **135**) store a different data value for the same field. For instance, the reference data record **130** of the first data storage **120** store "Mike" in the first name field, while the other reference data record **135** stores "Michael" in the same field.

**[0033]** The master reference record **110** has a master ID field **115** and several other data fields (e.g., first name, last name, address, etc.). The master ID is a unique identifier associated with a master reference record that is generated by the internal logic of some embodiments. Similar to the primary key of a database, the master ID is utilized by some embodiments in accessing and updating a particular master reference record. The data fields of the master reference record **110** are fields specified by the enterprise (e.g., data administrator, business user). In some embodiments, these specified fields are based on the fields of reference data sets. For example, in the example illustrated, the specified fields include all the fields in both reference records (**130** and **135**) such as social security number, driver's license number, etc.

**[0034]** As described above, in some cases, the master reference data set maintained in the master reference store **110** is the "best version" of several reference data sets. This concept of maintaining the most reliable data set is also conceptually illustrated in FIG. **1**. When reference data records are consolidated (e.g., matched and merged) into a single master reference record, some embodiments provide a master reference manager that determines which one of data values of the same record fields (e.g., address field, first name field) contain the most reliable data value. In some embodiments, the master reference manger applies a concept of "trust" to these data values to determine the most reliable data value. For instance, in the example illustrated, the master reference manager has determined that "Mike" is the more reliable name for the individual than "Michael". Hence, "Mike" is the name that is stored in the master reference record **110**.

**[0035]** FIG. **2** provides an illustrative example of several template master reference data sets for an individual in different enterprises. Specifically, this figure illustrates that different enterprises can create a data model for an entity differently. The figure includes a first template master reference data set **205** from one enterprise **215** and a second template master reference data set **210** from another enterprise **220**. Each master reference data set (**205** or **210**) has several enterprise specified data fields and a master ID.

**[0036]** In the example illustrated in FIG. **2**, the data fields for the template master reference data set specified by the first enterprise **215** are different from those specified by the second enterprise **220**. For instance, the first enterprise **215** has specified a first name field, a last name field, a driver's license number field, a state field, and a social security number field, while the second enterprise **220** has specified a last name field, a social security number field, an Acxiom number field, a tax identification number field, and a zip code field. Hence, the first enterprise **215** has created a data model for an individual that is different from the one specified by the second enterprise **220**.

**[0037]** As described above, the master ID is a unique identifier associated with a particular master reference record that can be used in accessing that particular record. Some embodiments provide a direct access path to the particular master reference record through one or more of the other fields. Specifically, some embodiments allow an enterprise to specify or select one or more attributes of a template master reference data set as GBIDs. Once specified, like the master ID, these GBIDs can be used in accessing (e.g., locating, updating, deleting, merging, etc.) the particular master reference record. FIG. **3** provides an illustrative example of several enterprises that specify GBIDs. This figure includes the master reference data sets (**205** and **210**) from the first and second enterprises (**215**-**220**), as illustrated in FIG. **2**.

**[0038]** As illustrated, as GBIDs, the first enterprise **215** has specified the driver's license number and the social security number, while the second enterprise **220** has specified the social security number, the Acxiom number, and the tax identification number. As stated above, once specified, these GBIDs can be used in accessing the master reference data sets. For instance, in the first enterprise **215**, either the driver's license number or the social security number can be used equally in accessing the master data record for "John Doe". Also, in the second enterprise, any one of the social security number, the customer number, or the tax identification number can be used equally in accessing the master data record for the same individual. Furthermore, in each enterprise, the master IDs can also be used to access the master reference data sets.

**[0039]** In some embodiments, the attributes specified by an enterprise are business identifiers (e.g., social security number, driver's license number) that can uniquely identify a single data record out of several data records. This characteristic of business identifiers to identify a single data record is important because it allows an enterprise to build processes (e.g., business processes) around these identifiers. For instance, a first name attribute of an individual can't be used reliably to build business processes because a data storage (e.g., master reference store) might store multiple data records with the same first name. In this instance, a business process might be broken because an update command from a business user providing the first name might lead to a different master reference record being updated than what was intended by the user. However, a social security number can be used reliably to build business processes because there is only one individual with a particular social security number.

**[0040]** In some embodiments, the business identifiers specified by the enterprise as GBIDs are those generated by applications in the enterprise (e.g., Systems Applications and Products ("SAP") number, Siebel number). In some embodiments, the business identifiers specified are those provided by data providers (e.g., Data Universal Numbering System ("DUNS") number, Acxiom number). In some embodiments, the business identifiers specified are those associated with an entity and are not tied to a particular data record of a database. In other words, these business identifiers are different from data record keys (e.g., primary key) that can be erased by deleting one or more database records. Examples of such business identifiers include tax identification number, driver's license number, social security number, etc.

**[0041]** Some embodiments of the invention provide a UI that allows an enterprise to identify which attributes of a master reference data set should serve as GBIDs. FIG. **4** provides an illustrative example of one such UI. This figure includes a configuration utility **400** that has an object browser **470** and a display area **405**. The object browser **470** allows a user to select a data object (e.g., customer, company) from the available data objects. For instance, in the exemplary UI illustrated, the object browser **470** lists several objects such as an account object and a business party object. When a data object is selected, the configuration utility provides one or more sets of controls in the display area **405** that allows a user to define the data object. Examples of different sets of controls that can be displayed in the display area **405** are a set of controls for defining attributes, cross-reference, dependent objects, relationship with other objects, history, match and merge rules, validation rules, message trigger, and data mapping. The concepts of matching and merging data, and data mapping are described below with respect to FIG. **20**.

**[0042]** The set of controls for defining message triggers allows the user to specify what changes to the data object can fire messages to different applications. Also, the set of controls for defining cross-reference allows the user to define an association between the selected object and other data objects in different data storages. In some embodiments, this cross-reference correlation is defined to allow reference data sets to be automatically updated in response to an update to a master reference data set (e.g., master reference record). In some embodiments, this cross-reference correlation is defined to provide access to different types of data in several different data storages. These concepts of propagating changes and accessing different types of data in several different data storages will further be described below with respect to FIGS. **17**-**20**.

**[0043]** The set of control for defining relationship allows the user to specify not only the relationship between objects but attributes of the relationship (e.g., relationship type, start and end dates, etc.). Also, the set of controls for defining dependent object allows the user to specify any dependent objects for the selected object. Dependent object are those objects that relies on another object for existence. For instance, as a dependent object, a customer object might include an address object that contains the address, state, and zip code. When the customer object is deleted, the address object will also be deleted. Hence, the address object is a dependent object of the customer object.

**[0044]** The set of controls for defining history allows the user to specify whether history should be maintained for the selected object and type of historical data (e.g., lineage that includes only the current value and any previous values, history that includes the previous values and information such as the source of those values). In some embodiments, this historical data is used to restore the selected object to a previous state (e.g., restore through unmerging). For enterprise specified GBIDs, some embodiments automatically maintain the historical data to provide access to data set by lineage.

**[0045]** In the example illustrated, the display area **405** displays the set of controls for defining the attributes of a business party object. The display area **405** is organized into several rows and columns, where each row represents a particular attribute of the data object. For instance, the display area **405** includes rows for a first name, a last name, a social security number, a DUNS number, etc.

**[0046]** The columns of the display **405** have several fields for a user to specify different aspects of a particular attribute that define that particular attribute. These fields include (1) a display name field **425** for specifying a display name (e.g., the logical name that is displayed in UIs), (2) a physical name field **430** for specifying an internal name (e.g., the actual name used internally), (3) a nullable field **435** for specifying whether the attribute can have no value, (4) a data type field **440** for specifying the data type of the attribute (e.g., variable characters, date, etc.), (5) a length field **445** for specifying the number of values (e.g., the total number of characters that the field accepts), (6) a precision field **450** for specifying the number of digits, (7) a default value field **460** for specifying a given value, (8) a trust field **410** for specifying whether to compute a trust score in determining the reliability of a received value, (9) a unique field **470** for determining whether the attribute stores a unique value, (10) a validate field **475** for specifying whether to perform validation logic on a received value, and (11) a GBID field **420** for specifying GBIDs.

**[0047]** As illustrated in FIG. **4**, the social security number is identified by the enterprise as a GBID. Specifically, a user has identified the social security number by selecting the GBID field **420** (e.g., checking the GBID field using a cursor controller) in the social security number row. Although only the social security number is identified as a GBID, any number of different attributes can be identified as GBIDs. For instance, the DUNS number or any other attribute of the template master reference data set can be identified by the enterprise as GBIDs.

**[0048]** When one or more GBIDs are identified by an enterprise, some embodiments provide an access path to the master reference data set through the use of each identified GBID. FIG. **5** illustrates a process **500** that some embodiments utilize to provide the access path using GBIDs. This process **500** will be described with reference to FIG. **4**. The process begins (at **505**) when it displays in the display area **400** one or more attributes of a data object. Next, the process (at **510**) receives selection from a user one or more attributes such as the social security number or the DUNS number.

**[0049]** After receiving selection of one or more attributes, the process (at **515**) defines each selected attribute as a GBID. In some embodiments, the process defines the selected identifier by creating and maintaining data (e.g., metadata). For example, metadata that lists the specified GBIDs for the data object can be created and stored. Next, the process (at **520**) provides the direct access path access through the use of GBIDs. In some embodiment, the process provides the access path in an automated manner by generating services around those identified GBIDs. For instance, the metadata that list the GBIDS can be processed to automatically generate the services that provide the access (e.g., locate, update) capabilities. This means that different applications in an enterprise can instantly update master reference data set using any one of the identified GBIDs.

**[0050]** Also, the access path is provided to the various applications and data storages in an enterprise without customization. For instance, the applications do not have to be custom coded to use one particular identifier. Also, these applications do not have to be custom coded to query and retrieve a data record key (e.g., primary key, master ID) associated with a data record to modify that data record. Instead, the applications can continue using the same identifiers that are being utilized. Hence, some embodiment of the invention provides the direct access path that allows all the different applications and data storages in the enterprise to not only coexist and remain as they are but also be connected to each other.

II. Examples

**[0051]** The following section provides several examples and processes related to locating and updating data sets through the use of GBIDs. This section also provides examples of accessing and updating a data set through the lineage of the GBID. Also, this section provides examples related to the concept of cross-reference correlation. In some embodiments, this cross-reference correlation allows changes to a master reference data set to be synchronized across reference data sets in several different data storages.

**[0052]** A. Global Business Identifiers

**[0053]** Some embodiments of the invention allow the master reference data set to be accessed (e.g., located and retrieved) through the use of the GBIDs. FIGS. **6-7** provide an illustrative example of a master data record that is located and retrieved through the use of GBIDs. These figures include a master reference manager **610**, an application **605**, and a master reference record **620** stored in a master reference store **615**.

**[0054]** As described above, some embodiments of the master reference manager **610** stores and maintains the "best version" of the reference data set records for at least some of the entities in the master reference store **615**. In the examples illustrated in FIGS. **6-7**, the master reference manager also provides the access path to the master reference records in the master reference store **615**. Also, the application **605** is any one of the various applications in an enterprise. Specifically, in these examples, the application **605** is an application that sends a request for an address stored in the master reference store **615**.

**[0055]** FIG. **6** provides an illustrative example of retrieving the master reference record **620** before the implementation of GBIDs. Specifically, before GBIDs, the master reference manager **610** provides a direct access path to the master reference record **620** in the master reference store **615** through the use of the master ID. Therefore, in this example, when the application **605** sends a request for the address using the Acxiom number, the master reference manager **610** cannot translate or understand the request, as the request does not include the master ID of the master reference record **620**. Typically, for the application **605** to receive the address using the Acxiom number, the application would have to be customized. For example, the application **605** would have to be custom coded to send one or more queries that identify the master ID associated with the Acxiom number.

**[0056]** FIG. **7** provides an illustrative example of retrieving the master reference record **620** after implementation of GBIDs. After implementation, the master reference manager **610** provides a direct access to the master reference record through not only the master ID but through one or more GBIDs. In this example, the Acxiom number has been designated by the enterprise as a GBID. Therefore, when the request with the Acxiom number is received, the master ref-

erence manager **610** is able to understand the request and retrieve the address for the application **605**. Moreover, the application **605** is able to receive the address without having to send one or more queries to identify the master ID associated with the Acxiom number.

**[0057]** Some embodiments of the invention allow not only the master reference data sets to be accessed through GBIDs but other reference data sets as well. For instance, some embodiments of the master reference manager **610** store a cross-correlation table for each entity (e.g., customer, company, product). This cross-correlation table stores a reference data set of the entity from various applications and data sources. Some embodiments provide access to these data sets through the use of GBIDs. FIG. **8** illustrates one such example of accessing a data set in the cross-correlation table. This figures include the master reference manager **610**, the application **605**, and the master reference record **620** stored in the master reference store **615**, as illustrated in FIGS. **6-7**. In addition, this figure includes the cross-reference correlation table **810** stored in a data storage **805**.

**[0058]** The cross-reference correlation table **810** has several correlation records from different data sources. Each correlation records include a primary key field and several other data fields. In some embodiments, the primary key value stored in the primary key field is not a primary key value generated by the data storage **805**. Instead, the primary key value stored in the primary key field is a primary key of a record from a different data storage. Hence, the primary keys stored in the cross-reference correlation table are also referred to as cross-reference keys.

**[0059]** In some embodiments, the correlation data set are utilized by the master reference manager **610** to propagate changes to the master reference record **620** back to the data sources. For instance, when the master reference record **620** is updated with a new address, the update to the master reference record is broadcast to source data storages (that store the reference records) based on the records in the correlation table **810**. In the examples illustrated, the correlation table **810** is stored in the data storage **805**. However, in some embodiments, the correlation table **810** is stored in the master reference store **615**. Specific example of propagating changes to a master reference record back to source records will further be described below with respect to FIG. **17**.

**[0060]** As illustrated in FIG. **8**, after implementation of GBIDs, the master reference manager **610** provides the access path to correlation records in the cross-reference correlation table **810** through the use of each designated GBID, as well as the master ID. In this example, the Acxiom number has been designated by the enterprise as a GBID. Therefore, when the request from the application **605** with the Acxiom number is received, the master reference manager **610** is able to understand the request. Moreover, the master reference manager **610** is able to retrieve the list of primary keys from the correlation table **810**. Some embodiments also provide the access path to the master reference data set or correlation data set through the use of the primary keys (i.e., cross-reference keys) stored in the correlation table **810**. For instance, in the example illustrated, the application **605** is able to retrieve the master reference record **620** by supplying any one of the primary keys stored in the correlation table **810**.

**[0061]** FIG. **9** provides an illustrative example of accessing data records after several GBIDs has been designated by the enterprise. Specifically, the enterprise has designated the Acxiom number and the tax identification attributes of the template master reference data set as GBIDs. Therefore, when the request from the application **605** with the either Acxiom or tax identification number is received, the master reference manager **610** is able to understand the request and retrieve the list of primary keys in the cross-reference table **810** stored in the data storage **805**.

**[0062]** Some embodiments allow data records to be not only located but also updated in the data storage through the use of the GBIDs. FIGS. **10-12** presents illustrative examples of updating data records using GBIDs. These figures include the master reference manager **610**, the application **605**, the master reference record **620** stored in the master reference store **615**, and the correlation table **810** stored in the data storage **805**, as illustrated in FIGS. **8-9**. However, in these figures, the application **605** is an application that sends a request to update the address of the master reference record **620**.

**[0063]** FIG. **10** provides an illustrative example of updating the master reference record **620** before the implementation of GBIDs. Specifically, before GBIDs, the master reference manager **610** provides a direct update path to update the master reference record **620** in the master reference store **615** through the use of the master ID. Therefore, in this example, when the application **605** sends a request to update the address using the Acxiom number, the master reference manager **610** cannot translate or understand the request, as the request does not include the master ID of the master reference record **620**. Typically, for the application **605** to update the address using the Acxiom number, the application would have to be customized. For example, the application **605** would have to be custom coded to send one or more queries that identify the master ID associated with the Acxiom number.

**[0064]** FIG. **11** provides an illustrative example of updating the master reference record **620** after implementation of GBIDs. After implementation, the master reference manager **610** provides a direct update path to the master reference record through not only the master ID but also through one or more GBIDs. In this example, the Acxiom number has been designated by the enterprise as a GBID. Therefore, when the request with the Acxiom number is received, the master reference manager **610** is able to understand the request and update the address associated with the master reference record **620**.

**[0065]** Some embodiments of the invention allow not only the master reference data sets to be updated through the GBIDs but other reference data sets as well. This is also illustrated in FIG. **11**. For example, when the application **605** sends the request to update the addresses, the addresses in the correlation table are also updated by the master reference manager **610**.

**[0066]** FIG. **12** provides an illustrative example of updating the master reference record after several attributes of the template master reference data set has been designated as the GBIDs. Specifically, the enterprise has identified the Acxiom number and the tax identification of the template master reference data set as GBIDs. Therefore, in this example, the application **605** is able to update the master reference record **620** and the correlation records by sending either the Acxiom number or the tax identification number. In addition, the application can also update the master reference record using the master ID, and any one of the cross-reference keys stored in the correlation table.

**[0067]** FIG. **13** illustrates a process **1300** that some embodiments utilize to locate or update data sets in a data storage (e.g., master reference data set in a master reference store **615**). In some embodiments, this process is performed by the master reference manager **610**. The process begins (at **1305**) when a request to a locate or update a data record is received. The request includes a data value for a GBID. When the request is an update, the request might include other data values to update fields of the data record. For example, as illustrated in FIGS. **11-12**, the request from the application **605** includes the new address for the address field of the master data record **620** associated with the Acxiom number.

**[0068]** After receiving the request, the process (at **1310**) identifies the GBID that is included in the request. In some embodiments, the process identifies the GBID by evaluating the name associated with the GBID data value. For example, with reference to FIG. **11**, when the request from the application **605** is transmitted through a web interface, the GBID data value in the request might include an associated name that identifies the data value as an Acxiom number.

**[0069]** Next, the process (at **1315**) locates the data record in the data storage using the GBID data value. In some embodiments, when the process determines that the data record is not accessible using the GBID data value, the process ends. However, the process of some embodiments examines the history or lineage data that is maintained for the GBID. The process examines the history data to identify a current value associated with the GBID. Specific examples of locating data records through the history data of the GBID will further be described with respect to FIGS. **14-15**.

**[0070]** When a matching record is located in the data storage using the GBID data value, the process (at **1320**) updates the data record using one or more of the data values received with the request. For example, in FIGS. **11-12**, the address of the master reference record **620** is updated with the new address provided by the application **605**. However, when the request is not an update request but a locate request, the process **1300** sends the matching record to the requesting application and ends.

**[0071]** B. Access by Lineage

**[0072]** For each GBID identified by an enterprise, some embodiments maintain not only the current value associated with a GBID but also a prior value or values associated with the GBID. In some cases, these prior historical values are not stored for each GBID of a master reference data set but are only stored for one or more of the GBIDs. Some embodiments allow the master reference record to be accessed and updated not only through the current value associated with the GBID but also though one or more of the previous values associated with the GBID. In other words, these embodiments can query a data storage with an old value of the GBID and still be able to identify and update the particular master reference data set.

**[0073]** FIGS. **14-15** provides several illustrative examples regarding accessing (e.g., identifying and updating) data records through lineage data of a GBID. These figures include the master reference manager **610**, the application **605**, the master reference record **620** stored in the master reference store **615**, and the correlation table **810** stored in the data storage **805**, as illustrated in FIGS. **10-12**. In these examples, the driver's license number of an individual in the master reference record **620** has changed from a previous data value to a new data value. The application **605** is an application that sends a request to update the address field of the master reference record **620** using the previous data value. Also, the master reference record **620** includes a driver's number field that has been designated by the enterprise as the GBID.

**[0074]** FIG. **14** provides an illustrative example of updating a master reference record before implementation of access by lineage. In this example, when the application sends the request, the master reference manager **610** is able to understand the request because it includes the GBID. However, the master reference record **620** cannot be located and updated because the request from the application **605** has the previous data value that is no longer associated with the master reference record **620**. Thus, an access path to resolve the change to the GBID does not exist. In some cases, this can cause duplicate data records to be created in a data storage. For example, in such situations, a new master reference record for the same individual as the one stored in the master reference record **620** might be created in the master reference store **615**.

**[0075]** FIG. **15** presents an illustrative example of updating the master reference record **620** after the implementation of access by lineage. For illustrative purposes, this figure provides conceptual illustration of a lineage data **1505**. The lineage data **1505** includes one or more previous values and the current value associated with the GBID. In some embodiments, the master reference manager **610** stores the lineage data **1505** in the master reference store **615**.

**[0076]** When a master reference record cannot be located using the GBID, some embodiments of the master reference manager **610** identify the current data value associated with the GBID using the lineage data **1505**. In the example illustrated, the lineage data **1505** is used to identify the current data value associated with the driver's license number. After the current data is identified, the master reference manager **610** uses the address value in the request to update the address fields of the master reference record **620** and the correlation records in the correlation table **810**.

**[0077]** In some embodiments, the lineage data **1505** is updated each time a data attribute identified as a GBID is updated. For example, if the driver's license number data value in the master reference record **620** changes, then the master reference manager **610** updates the lineage data **1505** with the new value. Also, to provide access by lineage for master IDs, some embodiments maintain not only the current value associated with a master ID but a prior value or values associated with the master ID. This allows various applications to access a master reference record of an entity, even though the original master reference record created for the entity has been deleted or merged with another master reference record.

**[0078]** Some embodiments maintain not only the lineage but other historical data. FIG. **15** provides a conceptual illustration of other types of historical data that can be maintained for a GBID. While lineage includes the current value and any previous value or values, history includes several other factors that could affect the data at each point in time. For instance, history might includes values of each field in the master reference record, and the source of this value, irrespective of whether the value was ever merged into a master reference record. This distinction between the history data and the lineage data is illustrated in FIG. **16**.

**[0079]** As illustrated, the history data not only includes previous and current values (i.e., values 'A', 'B', 'C') stored in a master reference record but also includes other values (i.e., 'Y', 'Z') that could have affected the master reference record. In some embodiments, maintaining history and lineage pro-

vide for a un-merge procedure, as will be further described below. In some embodiments, the historical data is provided to the enterprise for data management (e.g., maintenance).

**[0080]** C. Cross-Reference Correlation

**[0081]** In response to an update to a master reference data set, some embodiments update reference data sets in different data storages. Some embodiments maintain a correlation table to automatically propagate the changes to the master reference data set back to the source data storages.

**[0082]** FIG. **17** presents an illustrative example of updating reference records in response to an update to the master reference record. The figure includes the master reference manager **610**, the master reference record **620**, the correlation table **810** with several correlation records, and the application **605**, as illustrated in FIGS. **11-12**. This figure also has a first data storage **1720** of a sales force automation ("SFA") application **1710**, and a second data storage **1725** of an enterprise resource and planning ("ERP") application **1715**. In this example, the master reference record **620** and the correlation table **810** are a part of the master reference manager **610**. Also, the requesting application **605**, the SFA application **1710**, and the ERP application **1715** are different applications in the enterprise.

**[0083]** The correlation table **810** has several correlation records, where each record has a set of fields. In the example illustrated, the set of fields includes (1) a x-ref field, (2) a master ID field, (3) a primary key field, (4) a source field, (5) a first name field, (6) a last name field, (7) an Acxiom number field, and (8) an address field. The x-ref field stores a generated data value. In some embodiments, this generated value is used internally by the master reference manager **610** to identify a particular correlation record.

**[0084]** The master ID field of the correlation record stores a master ID of a particular master reference record. In some embodiments, this field associates the correlation record with the particular master reference record. In other words, this master ID field allows a one-to-many relationship to be created between a master reference record and one or more correlation records. For example, the master ID fields of several different correlation records in the correlation table might store one master ID value. Also, the source field stores a data value that identifies the source of the record. For example, in FIG. **17**, the source field of the first correlation record identifies the source as the SFA application **1710**, and the second correlation record identifies the source as the ERP application **1715**.

**[0085]** In some embodiments, the primary key stored in the primary key field of the correlation table **810** is not an internally generated primary key. Instead, the primary key (i.e., natural key) is a foreign key from a reference data record from a different data storage than the one storing the correlation table **810**. Hence, the primary key in the cross-reference correlation table **810** can also be referred to as a cross-reference key. When a master reference record is updated, the update message sent to each source system (e.g., applications, data storages) includes a particular cross-reference key from the correlation table. This is because the cross-reference key is the natural key that the source data storage understands to identify the reference record.

**[0086]** The first name field, the last name field, and the Acxiom number field store attributes related to an entity such as customer. In some embodiments, the attribute fields are used to determine if a particular data source should receive the update message when a master reference record is updated. For example, if the first correlation record of FIG. **17** does not include an address field, then the SFA application might not receive the update message. In the example illustrated, each correlation record includes the same attribute fields (i.e., first name, last name, Acxiom number). However, in some embodiments, the attribute fields of two different correlation record might differ as reference records can have different fields.

**[0087]** As illustrated in FIG. **17**, the master reference record **620** includes an Acxiom number field that has been designated by the enterprise as a GBID. When the application **605** sends a request to update the address of the master reference record **620** using the GBID, the request is received by the master reference manager **610**. The master reference manager **610** performs several operations that include (1) identifying a matching master reference record in the master reference store based on the request, (2) identifying each correlation record, (3) sending the update message to one or more data sources based on the identified correlation records. For example, the master reference manager **610** identifies the master reference record using the received GBID value in the request. Once a matching record is identified, the manager **610** identifies each associated correlation record in the correlation table **810**. In some embodiments, the master reference manager identifies the correlation records by locating any correlation records that has the master ID of the matching master reference record. Some embodiments of the master reference manager identify the correlation records using the received GBID value.

**[0088]** After identifying one or more correlation records, the master reference manager **610** sends the update messages to the source data storages (**1720** and **1725**). In some embodiments, the update messages are sent using the cross-reference keys or primary keys stored in the cross-reference table. For example, in FIG. **17**, the first update message sent to the SFA application **610** includes the cross-reference key stored in the first correlation record, and the second update message sent to the ERP application **1725** includes the cross-reference key stored in the second correlation record.

**[0089]** FIG. **18** illustrates a process **1800** that some embodiments utilize to update reference data sets in response to an update to a master reference data set. The process **1800** of some embodiments is performed by the master reference manager **610**. The process begins (at **1805**) when a cross-reference correlation is created between a master data record and a reference data record. In the example illustrated in FIG. **17**, the cross-reference correlation is created through the cross-correlation table **810**. Specifically, the cross-reference correlation is created by storing correlation records in the cross-correlation table **810** and storing a master ID of a master reference record in the correlation records.

**[0090]** Next, the process (at **1810**) receives an update to a master reference record. Some embodiments identify a matching master reference record in a master reference store using the GBID data value. After identifying the matching record, the process of some embodiments identifies each correlation record associated with the matching master reference record. Finally, the process (at **1815**) sends an update message to one or more data sources storing reference data sets. As mentioned above, this update message is sent to each system with the primary key or cross-reference key stored in the correlation table **610**.

III. Data Flow

**[0091]** FIG. **19** illustrates a data flow for a system **1900** according to some embodiments of the invention. System **1900** includes a master reference manager **1950**, data sources **1901**, data consumers **1923**, applications **1919**, inbound data interfaces **1902**, and outbound data interfaces **1923**. The master reference manager **1950** includes (1) a mapping module **1908**, (2) an initial processor **1909**, (3) a GBID record manager **1910**, (4) a match engine **1914**, (5) a trust framework **1911**, (6) a GBID query manager **1916**, and (7) and a master reference store **1915**.

**[0092]** In some embodiments, the data sources **1901** include applications and data storages, such as the applications (**610** and **1715**) and data storages (**1720** and **1725**) of FIG. **17**. Similarly, in some embodiments, the data consumers **1923** include applications (e.g., **610** and **1715**) and data storages (e.g., **1720** and **1725**). In some cases, the data sources **1901** and the data consumers **1923** may overlap such that a source of data is also a consumer of data.

**[0093]** As illustrated in FIG. **19**, the reference data from each of the data sources **1901** enters the master reference manager **1950** through the inbound data interfaces **1902**. The inbound data interfaces **1902** include a variety of different pathway processes. For instance, data from data sources **1901** (such as a file) can enter the master reference manager **1950** through an extract-transform-load ("ETL") process **1903**, which can be a typical batch or scheduled processes. The data from the data sources **1901** can also enter the master reference manager **1950** through several processes that typically operate in real-time. For instance, the data can enter the master reference manager **1950** through a web services process **1904** such as web browser requests. Also, the data can enter through application server objects such those of an Enterprise Java Bean ("EJB") process **1905**. This data can also enter through a XML over HTTP process **1906** (e.g., through an XML message in a HTTP request). Also, the data can enter through a Java Message Service ("JMS") process **1907** that provide a messaging interface for sending and receiving messages.

**[0094]** A. Mapping Incoming Data

**[0095]** Irrespective of whether the ETL process **1903**, XML over the HTTP process **1906**, or the web services process **1904** is used to receive data from the data sources **1901**, the data first enters the master reference manager **1950** through the mapping module **1908**, as shown in FIG. **19**. When the mapping module **1908** receives the data, it performs one or more operations to internalize the data for processing and storing. These operations can include (1) delta detection, (2) data cleansing, (3) standardization, and (4) data mapping. In some embodiments, delta detection initially confirms whether the received data has actually been changed or is different from previously received data. Hence, delta detection may reduce unnecessary processing of unchanged data.

**[0096]** In some embodiments, mapping module **1908** performs a data cleansing operation that normalizes data for processing and storage. For example, the cleansing operation might remove extraneous blanks or other noise character, change case for an attribute, etc. In some embodiments, this format is the same format that is used to store reference data in the master reference store **1915**.

**[0097]** In some embodiments, standardization is the process of homogenizing different data codifications. For example, in one application-centric body of data, the male gender may be codified as 'M'. In another application-centric body of data, the male gender may be codified as '1'. During standardization, the mapping module **1908** assimilates these two different coding styles into a uniform coding style for the master reference store **1915**. For example, it may be that the master reference manager **1950** standardizes on male codification 'M', '1', '456', or some other value.

**[0098]** In some embodiments, the mapping module **1908** performs the mapping operation to map the incoming data to enterprise normalized data schema in the master reference store **1915**. Schema mapping is the process of projecting incoming data from the source schema into the target schema. Application-centric data store have schema, also known as a data model, which is most conducive to the specific application that they service. The master reference store schema, similarly, will be most suitable for its purpose of managing master reference data sets. Schema mapping determines the appropriate location in the target schema for each incoming data record.

**[0099]** B. Initial Processor

**[0100]** After the data are mapped to the target data model, the data is received by the initial processor. The initial processor determines whether the received data entering the initial processor **1909** updates a master reference data set stored in the master reference store **1915**. In some of embodiments, the initial processor uses the cross-reference key associated with the received data to determine whether the data updates a data set previously stored in the master reference store **1915**. In some embodiments, the initial processor **1909** searches the master reference store **1915** to determine whether a master reference data set in the master reference store **1915** also has a data set with the same set of associated cross-reference keys. Some embodiments search the correlation table **1920** to determine whether the same set of associated cross-reference keys is already stored. As illustrated in FIG. **19**, the cross-reference correlation table **1920** is stored in the master reference store **1915**. However, the cross-reference correlation table **1921** may be stored elsewhere in a different data storage. In some embodiments, the presence of the associated cross-reference keys indicates that the data entering the master reference manager **1950** updates a master reference data set.

**[0101]** In some embodiments, when the initial processor **1909** determines that the data is not an update or contains new data, the initial process stores the reference data without further processing. However, in some cases, the initial processor uses the trust score calculator **1912** of the trust framework **1911** to compute trust scores for one or more attributes of the received data set before storing the data set. Computation of trust scores are described below. On the other hand, when the initial processor **1909** determines that the received data updates a previously stored master reference data set, then initial processor (1) retrieves from the master reference store the previously stored master reference data set, and (2) has the trust framework **1911** perform the update operation on the retrieved and received reference data sets.

**[0102]** In some embodiments, the initial processor determines whether the reference data set entering the initial processor includes one or more GBIDs. Once the initial processor determines that reference data set includes one or more GBIDs, the initial processors sends the received data to the GBID record manager **1910** for GBID processing.

**[0103]** C. GBID Processing

**[0104]** After receiving the received data, the GBID record manager **1910** performs several GBID related operations. These operations include maintaining lineage or historical

data and maintaining correlation data for one or more of the received GBID data values. The correlation data (e.g., the cross-correlation table **1920**) maintained by the GBID record manager **1910** allows changes to the master reference data set to be reflected across one or more data storages of the data consumers **1923**. The lineage data **1921** maintained by the GBID record manager **1910** allows a master reference data set to be accessed not only through the current value associated with the GBID but also through one or more of the previous values associated with the GBID.

**[0105]** In some embodiments, the GBID record manager **1910** maintains the lineage data using several lineage tables. For example, if the GBID data value is a driver's license number, then the GBID record manager **1910** might store the previous driver's license number and the current driver's license number in a driver's license number table. As illustrated in FIG. **19**, the lineage data **1921** is stored in the master reference store **1915**. However, in some embodiments, the lineage data **1921** is stored elsewhere in a different data storage.

**[0106]** When the received data includes a new data record, some embodiments of the GBID record manager **1910** create a correlation record in the correlation table **1920**. As described above, the correlation record creates a cross-reference correlation between a reference data record and the master reference record. In some embodiments, this cross-reference correlation is created by storing the cross-reference key (i.e., primary key) of the reference record and the master ID of the master reference record in the correlation record. In some embodiment, the GBID record manager initially identifies the master ID by creating the master reference record for the received data. After identifying the master ID, the GBID record manager **1910** creates the correlation record in the correlation table **1920** that includes both the master ID and the cross-reference key. Also, when the received data includes a new data record, some embodiments of the GBID record manager **1910** create a lineage record in the lineage data **1921** for each GBID in the received data.

**[0107]** When the received data updates a previously stored master reference record, the GBID record manager **1910** identifies the stored master reference record through the GBID query manager **1916**. The GBID query manager **1916** retrieves the master reference record from the master reference store **1921** based on the received GBID data value. For example, the GBID query manager might locate the matching record using a social security number or a driver's license number in the received data. When the master reference record cannot be located using the GBID data value, some embodiments of the GBID query manager **1916** process the lineage data **1921** to locate the master reference record.

**[0108]** Once the master reference record is identified, the GBID record manager **1910** updates the master data record. When updating the matching master reference record, the GBID record manager **1910** of some embodiments uses the trust score calculator **1912** of the trust framework **1911** to compute trust scores for one or more attributes in the received data. When the received data updates one or more GBID values, the GBID record manager **1910** of some embodiments updates the lineage data **1921**. For example, if a new driver's license number is received for the master reference record, then the lineage data **1921** might be updated to include the new driver's license number and the previous driver's license number.

**[0109]** In some embodiments, when updating the master reference record, the GBID record manager **1910** identifies each correlation records associated with the master reference record. The GBID record manager **1910** identifies the correlations record to update reference records stored in source data storages. As described above, some embodiments identify the correlation record through the master ID of the master reference record. When the correlation record is identified, the GBID record manager **1910** updates the correlation record in the correlation table **1920** and sends the update message to the source data storage. This update message to the source data storage is sent through the cross-reference key that is stored in the correlation record. The update message is sent through the cross-reference key because it is the natural key that is understood by the source data storage.

**[0110]** As described above, the GBID query manager **1916** identifies a matching master reference record in the master reference store **1915** based on a GBID data value (e.g., the current data value, the previous data value). In some embodiments, the GBID query manager also identifies the master reference record based on a master ID or a cross-reference key. As illustrated, the GBID query manager **1916** communicates with the GBID query access interface **1918** to locate and retrieve data records for the applications **1919**. The applications **1919** are various applications in the enterprise that use GBIDs, master IDs, or cross-reference keys to locate and retrieve data records. As will be further discussed below with respect to FIG. **20**, these data records include not only reference records but also activity records and relationship records.

**[0111]** The GBID query access interface **1918** receives a request from an application and sends the request to the GBID query manager **1916**. In some embodiments, the GBID query access manager **1918** identifies a GBID, a master ID, or a cross-reference key in the request before sending the request to the GBID query manager **1916**. Also, the GBID query access interface **1918** may includes one or more communications interfaces for communicating with the applications **1919**, such as those listed for the inbound and outbound data interfaces (**1902** and **1917**). For example, the GBID query access interface **1918** might include a web service interface to communicate with an application across the internet.

**[0112]** As will be described below, when reference data is stored in the master reference store **1915**, some embodiments further consolidate the stored data record through a match and merge process. In some embodiments, this consolidation is performed periodically or in real-time by the match/merge engine **1914**. When data records are matched and merged through this engine, some embodiments of the GBID record manager **1910** perform any necessary operations to maintain the lineage and correlation. For instance, if two master reference records are merged into a single survivor, the GBID record manager **1910** might update the lineage data **1921** to include each GBID value that is no longer stored in the survivor. Also, the master ID stored in the cross-reference table **1920** may need to be updated with the master ID of the surviving master reference record.

**[0113]** D. Trust Framework

**[0114]** In some embodiments, the trust framework **1911** applies a concept of "trust" to update reference data set. The concept of trust involves a system of measuring the value and reliability of data. Trust may be represented as a numerical score as to the confidence of the system in that data. As illustrated in FIG. **19**, the trust framework includes two mod-

ules, a trust score calculator **1912** and a cell-level survivorship module **1913**. The trust score calculator **1912** compute the trust score for the received data set. In some cased, the trust score calculator **1912** re-compute the trust score for the previously stored data set in the master reference store **1915**. In some embodiments, the trust score calculator **1912** computes trust scores for some or all of the attributes (e.g., fields) in a data set. In some embodiments, the trust score is calculated for only those attributes that are identified by the enterprise. For example, through a UI as illustrated in FIG. **4**, a user can select one more attributes to compute trust scores.

**[0115]** This trust score calculator **1912** computes the trust score differently in different embodiments. In some embodiments, the trust score calculator **1912** computes the trust score based on certain parameters, algorithms, and rules. Examples of such parameters are source reliability weighting parameters that specify the reliability of the data source and fields from which the reference data records are provided. One example of trust rules are syntax validation rules that are used to determine the trust score of a data field based on the value of the data that is stored in that field. For instance, a syntax rule might reduce the trust score of a telephone number when the telephone number is not seven or ten digits long.

**[0116]** Examples of algorithms in the trust framework **1970** include data decay algorithms that express the diminishing value of data over time. Several examples of data decay profiles that can used by such data decay algorithms include (1) a linear reliability decay function, (2) a slow initial reliability decay function, and (3) a rapid initial reliability decay function. In some embodiments, the trust framework **1970** applies one of these three data reliability decay functions to the data entering the trust framework **1970** to determine the reliability of the data at a point in time. For instance, the rapid initial reliability decay function can be used to represent data that is expected to change frequently and thus become unreliable within a relatively short passage of time. This data would be expected to have an initial trust score that rapidly diminishes until its reliability (i.e., trust score) plateaus at a lower state. This feature, as represented by the rapid initial reliability decay function, can be attributed to data during scoring. Some embodiments administer data reliability decay functions and applicability to various types of data by using a data steward tool.

**[0117]** Thus, the trust framework **1970** includes a rule-based system of trust that includes various algorithms and parameters. The trust score calculator **1912**, in the embodiments described above, applies the system of trust to compute trust scores for data entering trust framework **1970**. Trust scoring begins when one or more reference records are received by the trust framework **1970** from the initial processor **1955**. In some embodiments, these records are cleansed records received by the initial processor **1955**. In some embodiments, these records also include stored records retrieved by the initial processor **1955** from the master reference store **1965** based on associated cross-reference keys.

**[0118]** Once the trust score calculator **1912** computes one or more trust scores for the reference data sets, the survivorship module **1913** consolidates the attributes of the reference data sets (i.e., the received and previously stored reference data sets) based on the computed trust scores. In some embodiments, the survivorship module **1913** will retain the attributes that have the higher calculated trust score.

**[0119]** The survivorship module **1913** of some embodiments maintains content metadata for each data attribute that it checks for consolidation. Two examples of content metadata are (1) the lineage of the new value and the source that provides this new value, and (2) the history of the replaced value and the source that provided the replaced value. Some embodiments maintain the full lineage and history for the replaced data fields and the data sources from which these fields emanated. Maintaining history and lineage for each field allows some embodiments to provide for a un-merge procedure, as further described below.

**[0120]** After the survivorship module updates any necessary content metadata, the initial processor **1909** stores the scored and/or consolidated reference data sets in the master reference store. This data then resides in the master reference store **1915** for additional updating by the master reference manager **1950**, retrieval by a data consumer, and/or matching by the match/merge engine **1914**. The match/merge engine **1914** will now be described.

**[0121]** E. Matching and Consolidation

**[0122]** Once reference data is stored in the master reference store **1915**, some embodiments further consolidate the stored data through a match and merge process. Such consolidation includes, for instance, removal of redundant reference records and resolution of conflicting reference records.

**[0123]** To consolidate reference records stored in the master reference store **1915**, the master reference manager **1950** includes the match/merge engine **1914**. The match/merge engine **1914** may operate periodically or in real time to consolidate reference records in the master reference store **1915**. The operation of the match/merge engine **1914** could be triggered by various events such as a change in a reference data record stored in the master reference store **1915**. The match/merge engine may also be triggered by an identification of one or more GBID by a user. The match/merge engine **1914** may also be triggered by a change in the rules and trust algorithms relating to the trust scoring of the reference records. The data steward may further trigger the match/merge engine **1914** to specifically perform matching at various times.

**[0124]** When scheduled or requested, the match/merge engine **1914** determines whether a reference data record matches one or more reference data records stored in the master reference store **1915**. To match existing records in the master reference store **1915**, the match/merge engine **1914** of some embodiments uses a method different from the system of cross-reference keys described above in relation to updating by the initial processor **1909**.

**[0125]** In some embodiments, the matching engine **1914** performs the matching process for each potentially matching pair of reference data sets in the master reference store **1915**. The match performs the matching process by initially determining whether two pair of reference data sets matches on a first set of criteria. In some embodiments, the first set of criteria includes whether a set of X fields match between the records. To perform this determination, the matching engine of some embodiments uses SSA-NAME3 from Identity Systems, an Intellisync Company.

**[0126]** If the matching engine **1914** determines that the first set of criteria is not met (e.g., the set of X fields do not match), then the match/merge engine **1914** determines whether a second set of criteria are met (e.g., a different set of Z fields match). For this operation, the matching engine of some embodiments can again use the matching modules SSA-NAME3 from Identity Systems, an Intellisync Company.

**[0127]** The second matching determination allows the match/merge engine **1914** to differentiate between both the number and the quality of field matches. For instance, in the case where the first set of criteria comprises a set of X field matches, the set of X fields might include both a high threshold number of field matches and some particular fields that are a strong indicator of a match (e.g., the direction of the reference, start date, end date, and role fields all match, or just the start date and end date fields match). This case may represent a correct match between the records for almost every instance where the set of X fields match, and thus meeting this condition indicates a highly probable or "absolute" match.

**[0128]** On the other hand, the second set of criteria for the set of Z fields can include a lesser number and a lesser quality of field matches (e.g., only the direction of reference matches). If only the set of Z fields match, then there is only a possibility of a record match in this instance and this "possible" match should be queued for an individual to inspect the data and/or perform a manual-merge.

**[0129]** Accordingly, when the match/merge engine determines that a manual-merge might be appropriate, then the match/merge engine of some embodiments queues up the two reference data sets for a data steward to review and determine whether the two reference data sets are a match. On the other hand, when the mat/merge engine determines that the two reference data sets do not meet the second set of matching criteria (i.e., determines that a manual-merge would not be appropriate), the match/merge engine specifies that the two reference data sets are not a match.

**[0130]** When the match/merge engine determines that the two reference data sets meet a first set of matching criteria (i.e., have a sufficient number and/or quality of fields that match), there is a highly probability (i.e., virtually absolute) that the two reference data sets match. Accordingly, the match/merge engine directs the trust score calculator to compute trust scores for the attributes of the matching reference data sets.

**[0131]** In some embodiments, the operation of the trust score calculator **1912** during the matching process **1400** is the same as the operation of this calculator during the update process **1190**. Once the trust score calculator **1912** computes one or more trust scores for the reference data set(s), the survivorship module **1913** consolidates the attributes of the reference data sets (i.e., the received and previously stored reference data sets) based on the computed trust scores. In some embodiments, the survivorship module **1913** will retain the attributes that have the higher calculated trust score. During this consolidation, the survivorship module **1913** again maintains content metadata (e.g., lineage and history) for each data attribute that it checks for consolidation. After processing by the trust framework, the match/merge engine stores the consolidate reference data set and its associated metadata in the master reference store, where it awaits for additional updating by the master reference manger, retrieval by a data consumer, and/or matching by a match/merge engine.

**[0132]** A data consumer can receive reference data sets from the master reference store **1915** through a variety of pathway process. In other words, reference data sets that are stored in the master reference store can be exported to outside systems such as an application, data warehouses, or application-centric data sources through the pathway process such as the ETL process **1903**, the EJB process **1905**, the XML over HTTP process **1906**, etc.

**[0133]** F. Un-Merge

**[0134]** The advantage of tracking content metadata (e.g., lineage and history of data) will be further described by reference to a un-merge functionality of some embodiments. At times, a merge procedure will combine reliable reference data with unreliable reference data. Unreliable data may contain an error or may simply be misinterpreted data. For instance, a reference with an entity "Smith" may be mistakenly interpreted to be the same reference as with entity "J. Smith". However, it may later be determined that "J. Smith" is actually a separate entity "J. Smith, Jr.". For this instance, some embodiments provide an unmerge procedure that allows the improperly merged reference for J. Smith Jr., to be extracted from the reference data for John Smith, and create a separate reference for J. Smith Jr. At other times, a change in the matching rules will cause previous merges to become obsolete, and require new merges to be performed. For these instances that result in undesired or inappropriate merges, some embodiments provide a sophisticated un-merge procedure.

**[0135]** The un-merge procedure will restore the various cells of a reference record for John Smith to a state prior to the merge and then re-apply all subsequent merges that did not include the (undesirable) reference data for J. Smith, Jr. Un-merge differs from a simple "undo" because it does not reverse the change to a single record. Rather, un-merge iterates through the content metadata (e.g., the history and lineage of data) to return a set of records and references affected by the un-merge to a state as if the merge with the incorrect and/or unreliable data had never occurred.

**[0136]** Thus, some embodiments provide a un-merge functionality that dramatically improves data reliability and quality. Moreover, some embodiments provide for several additional mechanisms such as updating, and a match and merge process, that promote a unified, consolidated view that is typically the best version of the available data. Further, these embodiments provide these functionalities and processes in real time.

IV. System

**[0137]** FIG. **20** provides an illustrative example of a system **2000** that implements some embodiments of the invention. An enterprise uses this system to integrate and mange data regarding various entities (e.g., customers, vendors, products, employees, etc.). As illustrated, the system **2000** includes a master reference manager **1950**, an activity manager **2060**, a UI **2070**, and a hierarchy manager **2055**, which run on one or more servers **2020**. The system further includes a design-time console **2045**, several applications (**2005**, **2010**, **2015**), and several data storages (**2060** and **2065**).

**[0138]** The design-time console **2075** provides a UI for a user to input application logic. An example of the UI is described above and illustrated in FIG. **4**. As described above, some embodiments of the UI allow the user to (1) specify attributes of the template master reference data set and (2) identify which attributes should serve as GBIDs. After the user perform either or both these operations, the user can send the configuration data to the master reference manager **1950** using the design-time console **2075**. In the exemplary system **2000**, the user has specified a first name, last name, address, Acxiom number, tax identification number, and social security number, as attributes that should be included in the template master reference data set. Also, the user has identified the tax identification, social security number, and tax ID

number, as GBIDs. As mentioned above with respect to FIG. **4**, some embodiments of the UI **2070** also include logic that allows the user to further define the template master reference data set (e.g., define cross-reference, define match and merge rules, define rules for mapping incoming data to target data, etc.).

**[0139]** The master reference manager **1950** receives the template master reference data set from the design-time console **2075** and creates a master reference data (e.g., one or more tables) in the master reference store **1915** according to the template. In some embodiments, the master reference manager **1950** automatically populates the master reference data with reference records from different data storages. For example, multiple reference data records from the data storages (**2060** and **2065**) may be matched and merged into the master reference data based on trust scores. In some embodiment, the correlation table **1920** and lineage data **1921** is also automatically created and updated in the master reference store **1915**. For example, the reference records from the data storages (**2060** and **2065**) and the GBID values in the reference records may be included in the correlation table **1920** and lineage data **1921**, respectively.

**[0140]** When the GBIDs have been identified by the user, the master reference manager **1950** receives the GBIDs and provides an access path to the master reference data set through the use of the GBIDs. For example, in FIG. **20**, the applications (**2005**, **2010**, and **2015**) are each able to use different identifiers (i.e., social security number, tax identification number, Acxiom number) to update the master reference data set in the master reference store **1915**. In some embodiments, this access path is provided in an automated manner by processing identified GBIDs from the design-time console **2045**. For example, when the GBID data for the template master reference data set is received, the master reference manager **1950** might process the data to generate the services that provide the access capabilities. This means that different applications (e.g., **2005**, **2010**, **2015**) in the enterprise can instantly access the master reference data set through any attribute identified by the user.

**[0141]** As described above, some embodiments allow not only the master reference data set to be accessed (e.g., located, updated) through the use the GBIDs but other reference data sets as well (e.g., reference data set in the correlation table **1920**). These reference data sets can be stored in different data storages in the enterprise. For instance, the application **2005** can use the Acxiom number to update a reference data record stored in the data storage **2060**. In some embodiments, this access to the reference data sets is provided based on correlation records in the cross-reference correlation table **1920**. For instance, when the application **2005** requests a reference data that is not stored in the master reference store, a federated query (i.e., distributed query) can be sent across to the different systems (e.g., data storages **2060** and **2065**) in the enterprise for that reference data. In some embodiments, the federated query is sent to the different systems using the cross-reference keys of the correlation records. The cross-reference key is used to send the distributed query because it is the natural key that is understood by a particular system (e.g., data storage **2060**).

**[0142]** Some embodiments allow not only the reference data to be accessed (e.g., located, updated) through the use the GBIDs but other types of data as well. These other types of data include activity data and relationship data. While the reference data identifies the entities that the system tracks for the enterprise, the activity data (i.e., transactional data) specifies the interaction of the entities with the enterprise, and the relationship data (i.e., hierarchical data) identifies the relationship between the entities. For instance, the activity data might include the account balance and recent transactions of a customer. Also, the relationship data might include relationship between a sales agent and the customer.

**[0143]** Some embodiments allow activity data to be accessed (e.g., located, updated) through the use the GBIDs. In some embodiments, this access path to the activity data is provided through the activity manger. The activity manager **2070** uses the reference data whenever an application initiates a particular interaction with the enterprise regarding a particular entity. In such a situation, the activity manager **2070** is responsible for providing a composite data object to the application, in order to allow the application to use the activity data regarding the particular interaction. A composite data object includes in some embodiments the reference data and the activity data. The reference data is provided to the activity manager **2070** from the master reference manager **1950**. This reference data is an instance of all or part of the master reference record stored in the master reference store **1945** for the particular entity.

**[0144]** The activity data is provided to the application by sending a federated query (i.e., distributed query) across to the different systems (e.g., data storages **2060** and **2065**) in the enterprise for the activity data. In some embodiments, this federated query is determined at run-time based on the correlation records in the correlation table **1920**. For example, when the application **2005** sends a request for the account balance and the recent transactions of a customer associated with the Acxiom number, the request is received by the master reference manager **1915**. The master reference manager identifies a master reference record and one or more correlation records associated with the master reference record. The activity manager **2070** will then receive the correlation records and coordinate the distribution of the federated query across the different system based on the correlation records.

**[0145]** Specifically, the activity manager will send the distributed query using the cross-reference keys in the correlation records. The cross-reference key is used to send the distributed query because it is the natural key that is understood by a particular system (e.g., data storage **2060**). In some embodiments, during design time, the user can define several connections to the activity data in the different systems based on the reference data using the activity manager **2070**.

**[0146]** Some embodiments allow relationship data to be accessed (e.g., located, updated) through the use the GBIDs. In some embodiments, this access path to the relationship data is provided through the hierarchy manager **2055**. Similar to the activity manager, the hierarchy manager **2055** provides relationship data by identifying a master reference record and locating any relationship data associated with the master reference record. In some embodiments, the hierarchical manager **2055** locates the relationship data through the correlation records in the correlation table. For example, when the application **2010** sends a request for a list of people related to the customer with social security number provided, the hierarchy manager **2055** will receives the correlations records and dynamically retrieves the relationship data based on these records. The relationship data provided to the application **2010** will, in some embodiments, include not just direct relationships but indirect relationship as well. For example, the customer entity might not be directly related to a sales man-

ager entity but may be related through one or more intermediary sales agent entities. In some embodiments, the accessed hierarchical data records are data records that are stored in a data storage managed by the hierarchy manager or master reference manager such as the master reference store.

V. Computer System

**[0147]** FIG. **21** conceptually illustrates a computer system with which some embodiments of the invention are implemented. The computer system **2100** includes a bus **2105**, a processor **2110**, a system memory **2115**, a read-only memory **2120**, a permanent storage device **2125**, input devices **2130**, and output devices **2135**.

**[0148]** The bus **2105** collectively represents all system, peripheral, and chipset buses that support communication among internal devices of the computer system **2100**. For instance, the bus **2105** communicatively connects the processor **2110** with the read-only memory **2120**, the system memory **2115**, and the permanent storage device **2125**.

**[0149]** From these various memory units, the processor **2110** retrieves instructions to execute and data to process in order to execute the processes of the invention. The read-only-memory (ROM) **2120** stores static data and instructions that are needed by the processor **2110** and other modules of the computer system. The permanent storage device **2125**, on the other hand, is a read-and-write memory device. This device is a non-volatile memory unit that stores instruction and data even when the computer system **2100** is off. Some embodiments of the invention use a mass-storage device (such as a magnetic or optical disk and its corresponding disk drive) as the permanent storage device **2125**. Other embodiments use a removable storage device (such as a floppy disk or Zip® disk, and its corresponding disk drive) as the permanent storage device.

**[0150]** Like the permanent storage device **2125**, the system memory **2115** is a read-and-write memory device. However, unlike storage device **2125**, the system memory is a volatile read-and-write memory, such as a random access memory. The system memory stores some of the instructions and data that the processor needs at runtime.

**[0151]** Instructions and/or data needed to perform processes of some embodiments are stored in the system memory **2115**, the permanent storage device **2125**, the read-only memory **2120**, or any combination of the three. For example, the various memory units may contain instructions for processing multimedia items in accordance with some embodiments. From these various memory units, the processor **2110** retrieves instructions to execute and data to process in order to execute the processes of some embodiments.

**[0152]** The bus **2105** also connects to the input and output devices **2130** and **2135**. The input devices enable the user to communicate information and select commands to the computer system. The input devices **2130** include alphanumeric keyboards and cursor-controllers. The output devices **2135** display images generated by the computer system. For instance, these devices display IC design layouts. The output devices include printers and display devices, such as cathode ray tubes (CRT) or liquid crystal displays (LCD).

**[0153]** Finally, as shown in FIG. **21**, bus **2105** also couples computer **2100** to a network **2165** through a network adapter (not shown). In this manner, the computer can be a part of a network of computers (such as a local area network ("LAN"), a wide area network ("WAN"), or an Intranet) or a network of networks (such as the Internet). Any or all of the components of computer system **2100** may be used in conjunction with the invention. However, one of ordinary skill in the art will appreciate that any other system configuration may also be used in conjunction with the invention.

**[0154]** While the invention has been described with reference to numerous specific details, one of ordinary skill in the art will recognize that the invention can be embodied in other specific forms without departing from the spirit of the invention. For instance, providing access (e.g., query, update) using different identifiers; providing access by lineage; and providing cross-reference capabilities can each be provided as part of separate computer program or components of a data management system.

**[0155]** In other places, various changes may be made, and equivalents may be substituted for elements described without departing from the true scope of the present invention. Thus, one of ordinary skill in the art would understand that the invention is not limited by the foregoing illustrative details but rather is to be defined by the appended claims.

What claimed is:

**1**. For an enterprise, a method of maintaining a master data set that includes a plurality of attributes, said method comprising:

a) from a user, receiving identification of at least one attribute of said master reference data set, said master reference data set representing the most reliable reference data set for an entity that is maintained by the enterprise;

b) designating each identified attribute as a key for accessing the master reference data set for the entity.

**2**. The method of claim **1**, wherein at least two different attributes of said master reference data set are designated as keys for accessing the master reference data set.

**3**. The method of claim **1**, wherein at least one designated attribute is a business identifier for the entity, wherein said business identifier for the entity is an identifier that is stored in a plurality of different data storages and can uniquely identify a single reference data set for the entity in each of the data storages.

**4**. The method of claim **1** further comprising providing access to said master reference data set using each designated identifier.

**5**. The method of claim **4** further comprising providing access to the master reference data set using a default access key, wherein said default access key is an internally generated key and the designated identifier is not an internally generated key.

**6**. The method of claim **1** further comprising providing access to relationship data sets for the entity using each designated identifier, said relationship data sets providing a composite view of the entity's relationship to other entities in the enterprise.

**7**. The method of claim **1** further comprising providing access to transactional data sets for the entity using each designated identifier.

**8**. The method of claim **7**, wherein said master reference data set is stored in a first data storage, and the transactional data sets are stored in another data storage in the enterprise that is apart from the first data storage.

**9**. The method of claim **1** further comprising providing access to the master reference data set through a previously stored value that is not the current value stored for the entity.

**10**. The method of claim **1** further comprising enabling cross-reference correlation for updating reference data sets in different data storages in response to an update to the master reference data set.

**11**. The method of claim **1** further comprising providing a user interface ("UI") for displaying the attributes of the master reference data set and receiving selection of the attributes to be designated as keys for accessing the master reference data set.

**12**. A graphical user interface ("GUI") comprising:

a) a display area for displaying attributes of a reference data set, wherein said reference data set represents a master reference data set for an entity that is based on several different reference data sets for the entity in an enterprise;

b) at least one widget for designating a set of attributes of the mater reference data set as a set of keys for accessing the reference data set.

**13**. The GUI of claim **12** further comprising at least one widget for specifying attributes that should be included in the reference data set.

**14**. A method comprising:

a) from an application, receiving a request to access a master reference data set for an entity, said request comprising at least one data value, wherein said master reference data set is based on at least two reference data sets maintained for the entity in an enterprise;

b) identifying a data value in the request as an attribute designated as a key for accessing the master reference data set;

c) providing access to the master reference data set using the identified data value in the request.

**15**. The method of claim **14**, wherein providing the access comprises identifying a previous data value associated with the data value to locate the master reference data set.

**16**. The method of claim **14** further comprising updating the master reference data set using at least one other data value in the request.

**17**. The method of claim **16**, wherein said master reference data set is stored in a first data storage, the method further comprising updating the reference data set in a second data storage that is apart from the first data storage in response to the update to the master reference data set.

**18**. The method of claim **14** further comprising providing access to transactional data sets for the entity, wherein providing the access to the transactional data sets comprises sending a distributed query across a set of data storages to locate the transactional data sets.

**19**. The method of claim **14** further comprising providing access to relationship data sets for the entity, said relationship data sets providing a composite view of the entity's relationship to other entities in the enterprise.

**20**. The method of claim **14**, wherein the identifying comprises identifying data value as the key from a plurality of other keys that are designated as the keys for accessing the master reference data set.

* * * * *