



US 20070198600A1

(19) **United States**

(12) **Patent Application Publication**

**Betz**

(10) **Pub. No.: US 2007/0198600 A1**

(43) **Pub. Date: Aug. 23, 2007**

(54) **ENTITY NORMALIZATION VIA NAME NORMALIZATION**

**Related U.S. Application Data**

(63) Continuation-in-part of application No. 11/356,838, filed on Feb. 17, 2006.

(76) Inventor: **Jonathan T. Betz**, Summit, NJ (US)

**Publication Classification**

(51) **Int. Cl.**  
**G06F 17/30** (2006.01)  
(52) **U.S. Cl.** ..... **707/201**

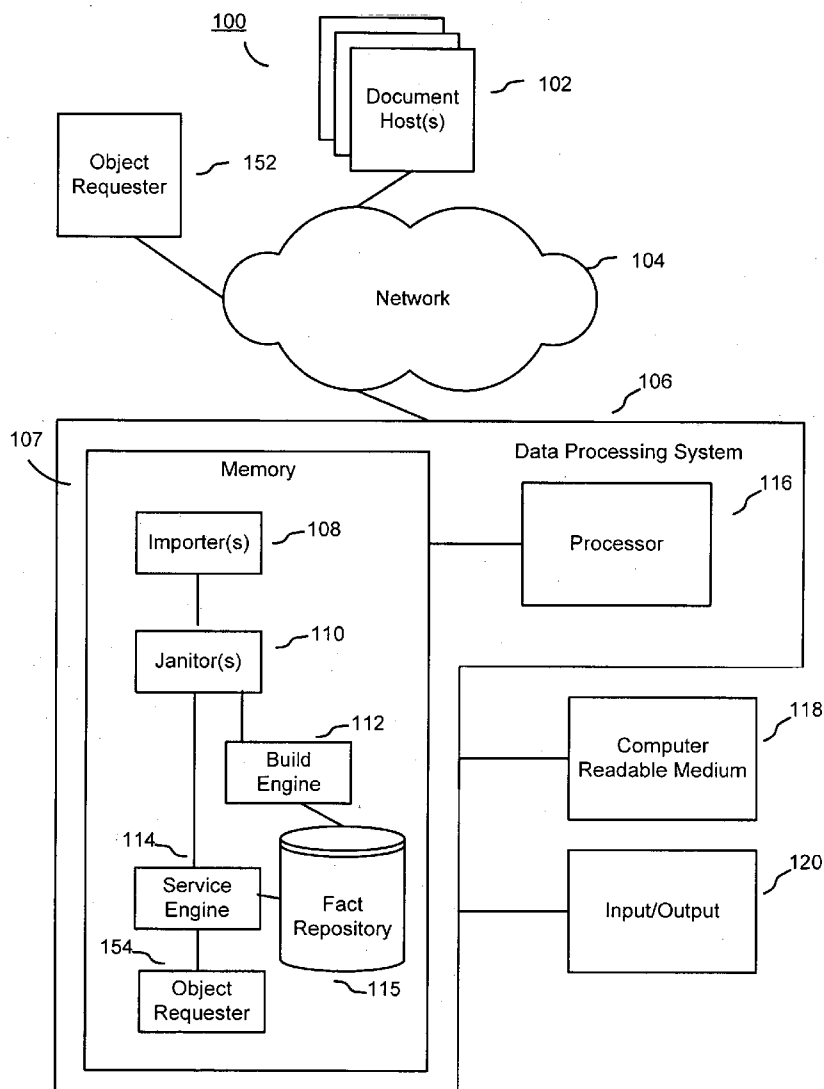
Correspondence Address:  
**GOOGLE / FENWICK**  
**SILICON VALLEY CENTER**  
**801 CALIFORNIA ST.**  
**MOUNTAIN VIEW, CA 94041 (US)**

(57) **ABSTRACT**

A system and method identifying duplicate objects from a plurality of objects. The system and method normalizes name values of objects, groups objects into buckets based at least in part on the normalized name values, matches objects within the same bucket based on a selected matcher, and identifies the matching objects as duplicate objects.

(21) Appl. No.: **11/394,508**

(22) Filed: **Mar. 31, 2006**



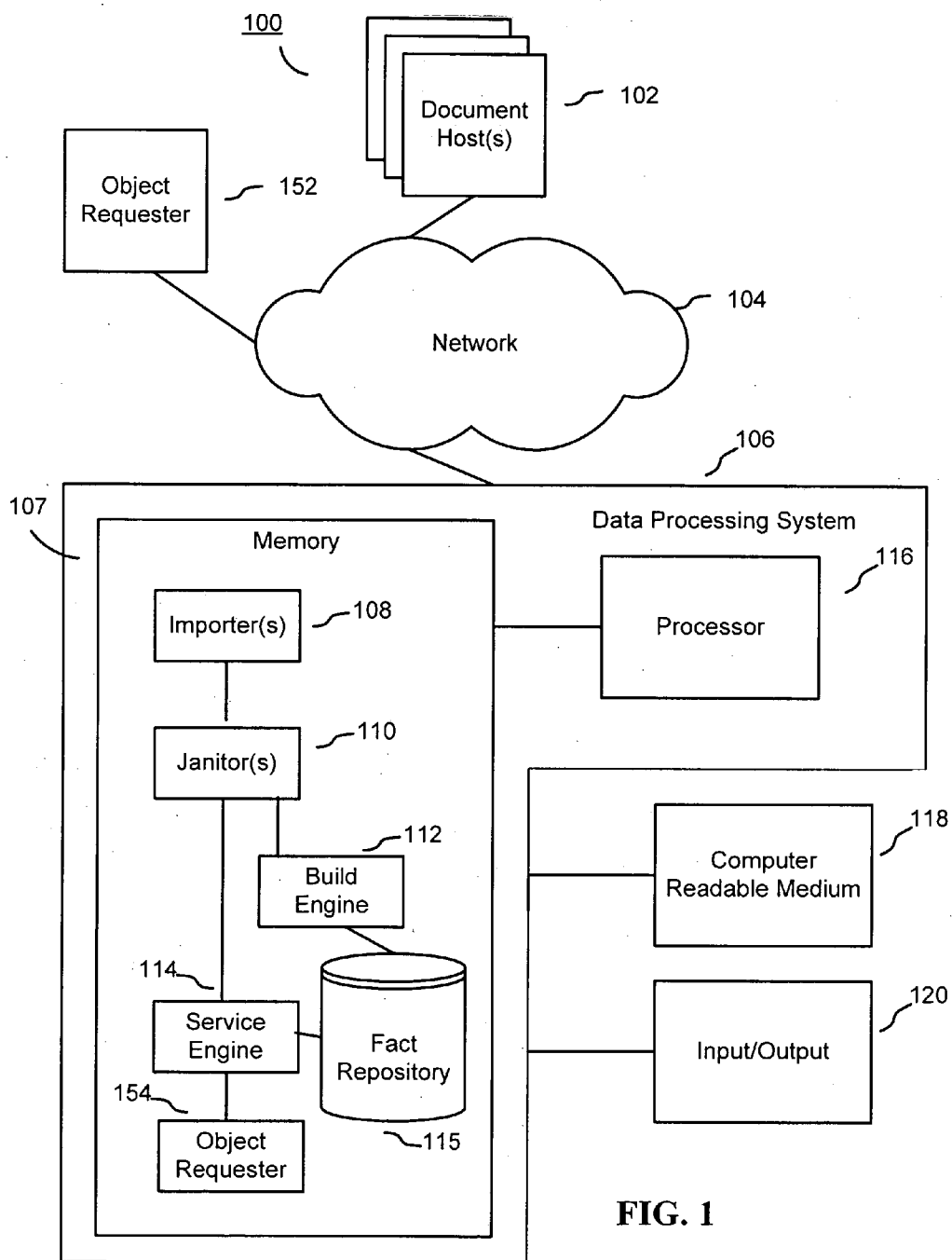
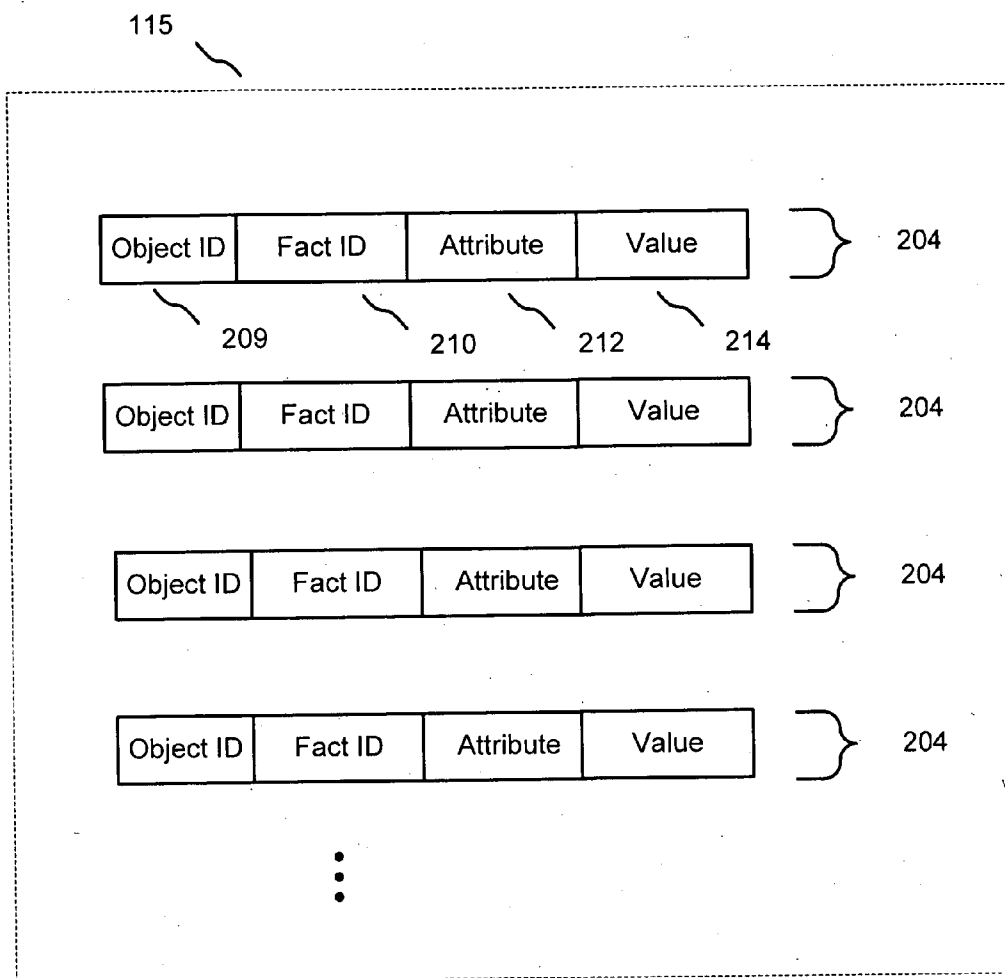
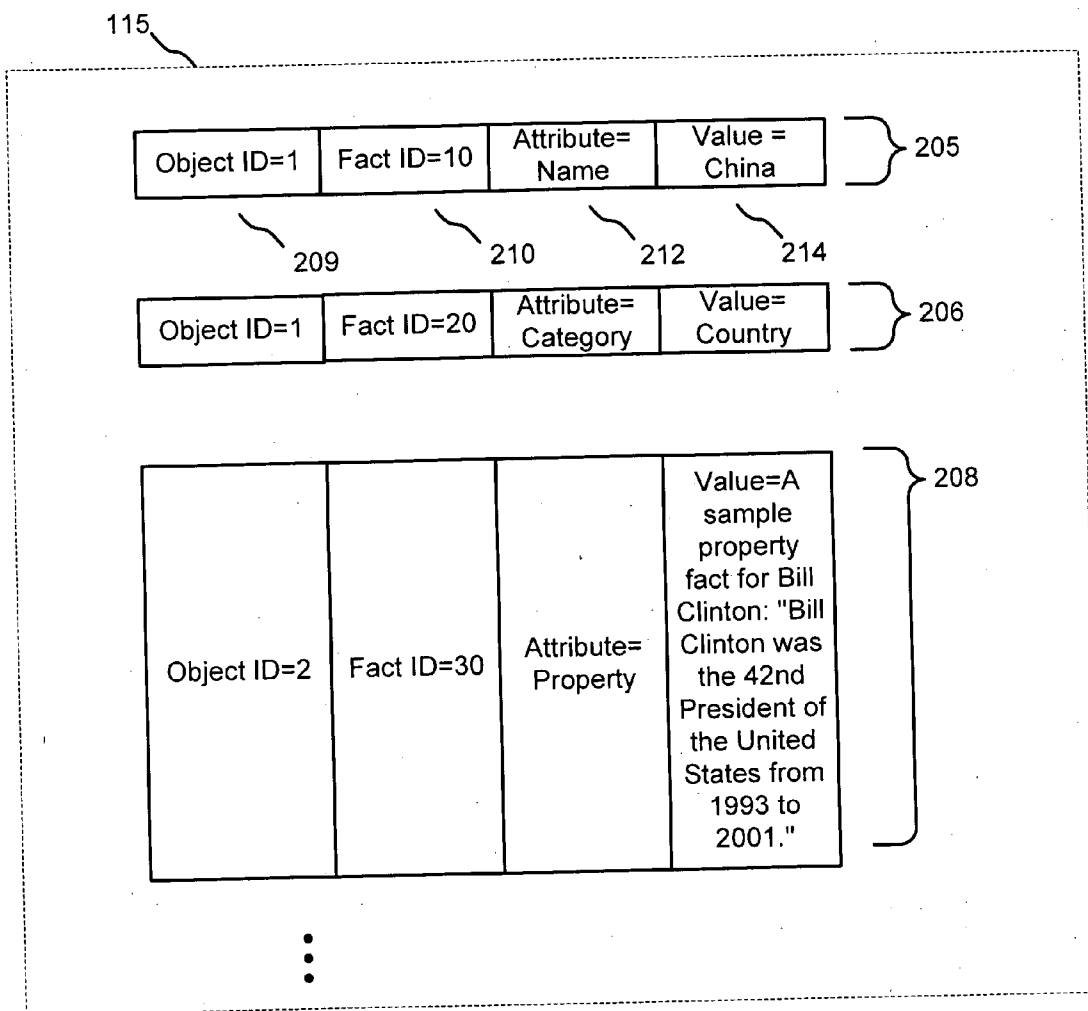


FIG. 1



**FIG. 2(a)**  
**Example Format of Facts in Repository (each fact is associated with an object ID)**



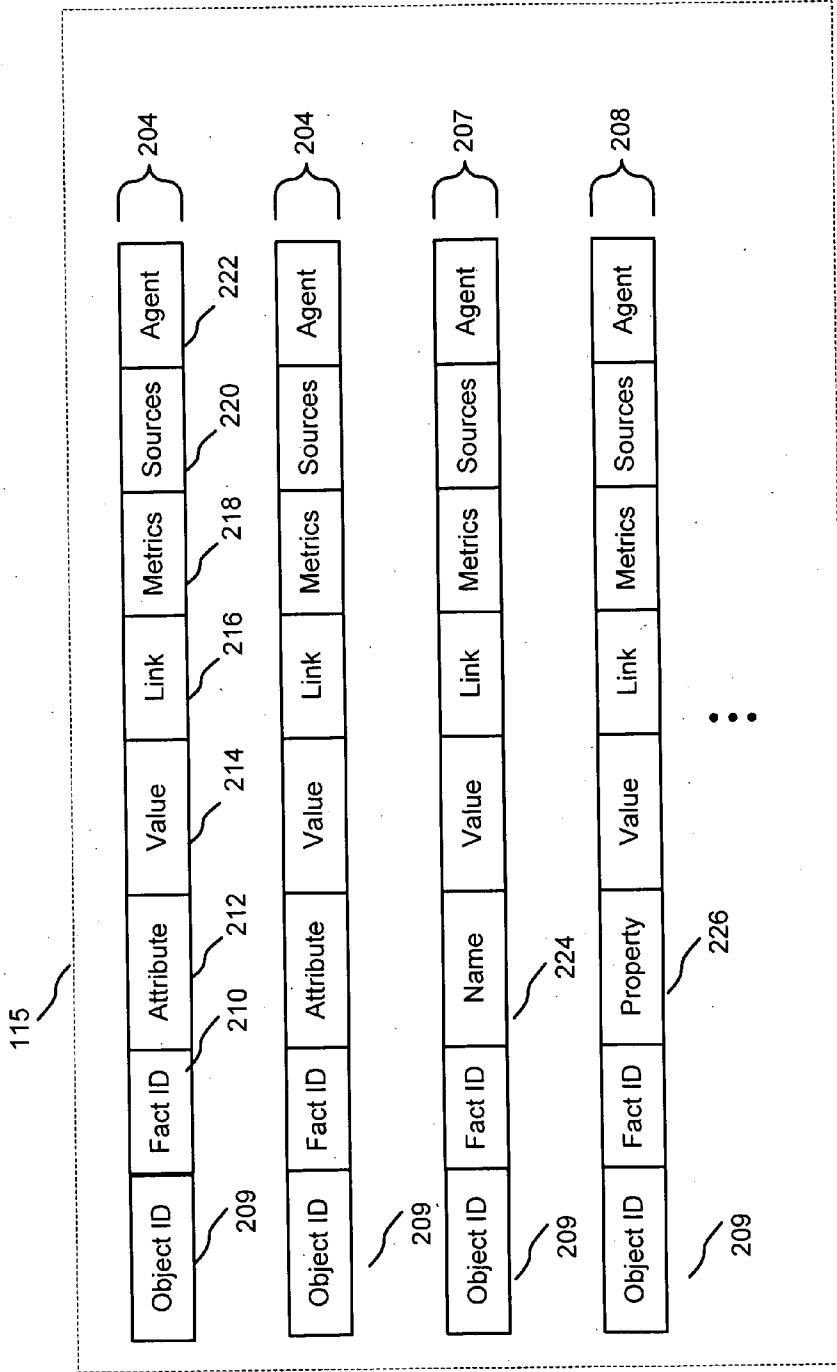
**FIG. 2(b)**  
**Example Facts in Repository (each fact is associated with an object ID)**

Object ID=1	Fact ID=10
Object ID=1	Fact ID=20
Object ID=1	Fact ID=30
Object ID=2	Fact ID=40

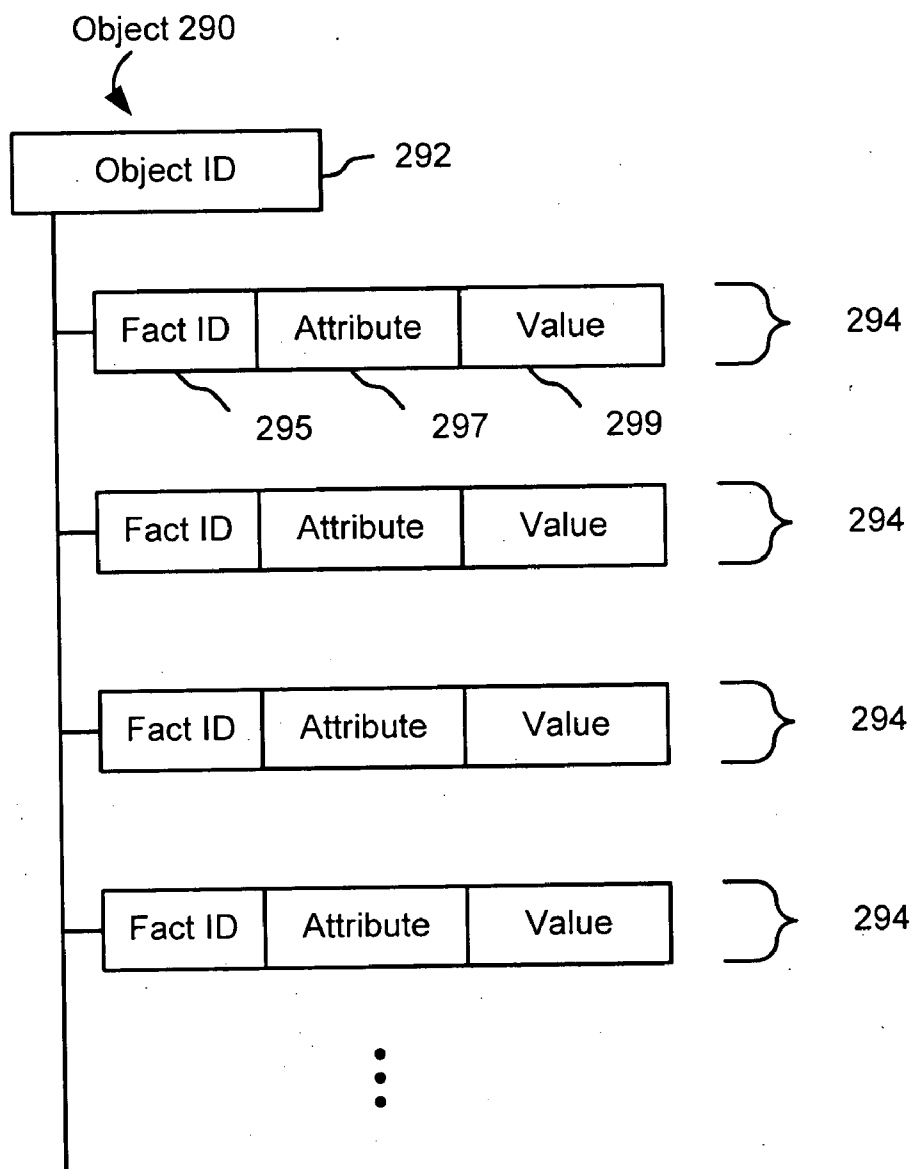
⋮

210

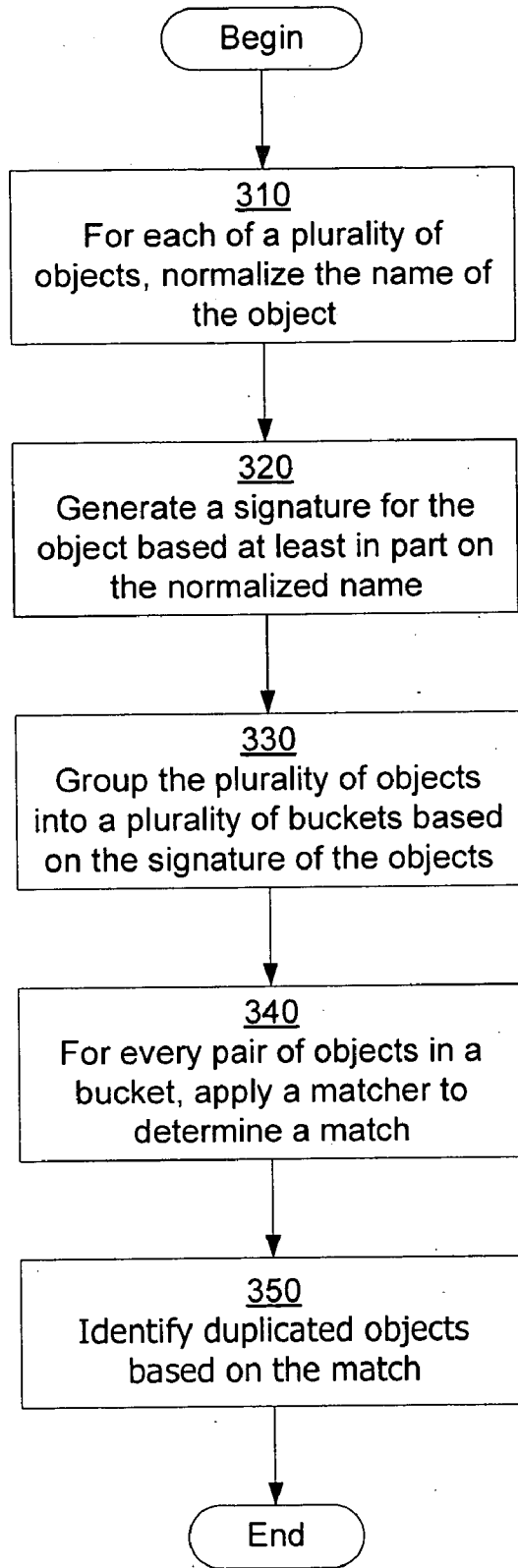
**FIG. 2(c)**  
**Example Object**  
**Reference Table**



**FIG. 2(d)**  
**Example Format of Facts in**  
**Repository (each fact is associated**  
**with an object ID)**



**FIG. 2(e)**  
**Example Objects**



**FIG. 3**

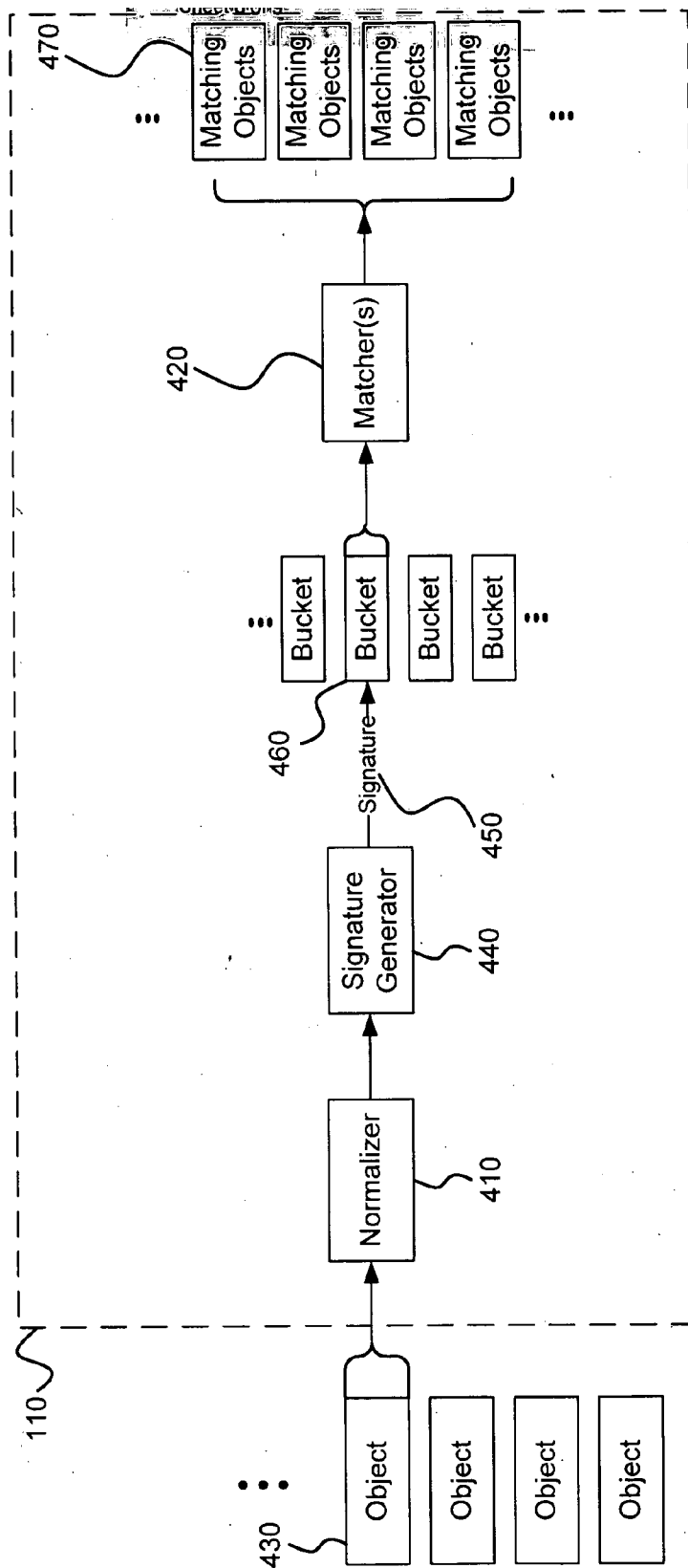


FIG. 4

Object ID \ Attribute	Name	Date of Birth	Address	Phone Number	Date of Death
O1	John W. Henry	1960		(719) 123-4567	
O2	John Henry	03/09/1976	4089 Iron Works Parkway, Lexington, KY 40511	(859) 233-4303	
O3	Mr. Henry, John		Lamar, CO 81052	719-123-4567	
O4	Senator John Henry	November 1750			December 16, 1798
O5	John F. Kennedy	05/29/1917			12/22/1963
O6	President John F. Kennedy	May 29, 1917			November 22, 1963

FIG. 5(a)

Object ID	Normalized Name Value
O1	john w. henry
O2	john henry
O3	mr. henry, john
O4	senator john henry
O5	john f. kennedy
O6	president john f. kennedy

FIG. 5(b)

Object ID	Normalized Name Value
O1	john w. henry
O2	john henry
O3	henry, john
O4	john henry
O5	john f. kennedy
O6	john f. kennedy

FIG. 5(c)

Object ID	Normalized Name Value
O1	henry john
O2	henry john
O3	henry john
O4	henry john
O5	john kennedy
O6	john kennedy

FIG. 5(d)

Normalized Name Value that Bucket Index is Based on	Object IDs of Objects Included in the Bucket
Henry john	O1, O2, O3, O4
John kennedy	O5, O6

FIG. 5(e)

**ENTITY NORMALIZATION VIA NAME NORMALIZATION**

**CROSS-REFERENCE TO RELATED PATENT APPLICATIONS**

[0001] This application is a continuation-in-part of U.S. patent application Ser. No. 11/356,838, entitled “Modular Architecture For Entity Normalization,” by Jonathan T. Betz and Farhan Shamsi, filed on Feb. 17, 2006, which is hereby incorporated by reference in its entirety.

[0002] This application is related to the following U.S. applications all of which are incorporated by reference herein:

[0003] U.S. application Ser. No. 11/366,162, entitled “Generating Structured Information,” filed Mar. 1, 2006, by Egon Pasztor and Daniel Egnor, Attorney Docket number 24207-11149;

[0004] U.S. application Ser. No. 11/357,748, entitled “Support for Object Search”, filed Feb. 17, 2006, by Alex Kehlenbeck, Andrew W. Hogue, Attorney Docket No. 24207-10945;

[0005] U.S. application Ser. No. 11/342,290, entitled “Data Object Visualization”, filed on Jan. 27, 2006, by Andrew W. Hogue, David Vespe, Alex Kehlenbeck, Mike Gordon, Jeffrey C. Reynar, David Alpert;

[0006] U.S. application Ser. No. 11/342,293, entitled “Data Object Visualization Using Maps”, filed on Jan. 27, 2006, by Andrew W. Hogue, David Vespe, Alex Kehlenbeck, Mike Gordon, Jeffrey C. Reynar, David Alpert;

[0007] U.S. application Ser. No. 11/356,679, entitled “Query Language”, filed Feb. 17, 2006, by Andrew W. Hogue, Doug Rohde, Attorney Docket No. 24207-10948;

[0008] U.S. application Ser. No. 11/356,837, entitled “Automatic Object Reference Identification and Linking in a Browseable Fact Repository”, filed Feb. 17, 2006, by Andrew W. Hogue, Attorney Docket No. 24207-10961;

[0009] U.S. application Ser. No. 11/356,851, entitled “Browseable Fact Repository”, filed Feb. 17, 2006, by Andrew W. Hogue, Jonathan T. Betz, Attorney Docket No. 24207-10949;

[0010] U.S. application Ser. No. 11/356,842, entitled “ID Persistence Through Normalization”, filed Feb. 17, 2006, by Jonathan T. Betz, Andrew W. Hogue, Attorney Docket No. 24207-10950;

[0011] U.S. application Ser. No. 11/356,728, entitled “Annotation Framework”, filed Feb. 17, 2006, by Tom Richford, Jonathan T. Betz, Attorney Docket No. 24207-10951;

[0012] U.S. application Ser. No. 11/341,069, entitled “Object Categorization for Information Extraction”, filed on Jan. 27, 2006, by Jonathan T. Betz, Attorney Docket No. 24207-10952;

[0013] U.S. application Ser. No. 11/356,838, entitled “Modular Architecture for Entity Normalization”, filed

Feb. 17, 2006, by Jonathan T. Betz, Farhan Shamsi, Attorney Docket No. 24207-10953;

[0014] U.S. application Ser. No. 11/356,765, entitled “Attribute Entropy as a Signal in Object Normalization”, filed Feb. 17, 2006, by Jonathan T. Betz, Vivek Menezes, Attorney Docket No. 24207-10954;

[0015] U.S. application Ser. No. 11/341,907, entitled “Designating Data Objects for Analysis”, filed on Jan. 27, 2006, by Andrew W. Hogue, David Vespe, Alex Kehlenbeck, Mike Gordon, Jeffrey C. Reynar, David Alpert;

[0016] U.S. application Ser. No. 11/342,277, entitled “Data Object Visualization Using Graphs”, filed on Jan. 27, 2006, by Andrew W. Hogue, David Vespe, Alex Kehlenbeck, Mike Gordon, Jeffrey C. Reynar, David Alpert;

[0017] U.S. application Ser. No. \_\_\_\_\_, entitled “Determining Document Subject by Using Title and Anchor Text of Related Documents”, filed on \_\_\_\_\_, by Shubin Zhao, Attorney Docket No. 24207-11049.

[0018] U.S. application Ser. No. \_\_\_\_\_, entitled “Anchor Text Summarization for Corroboration”, filed on \_\_\_\_\_, by Jonathan T. Betz and Shubin Zhao, Attorney Docket No. 24207-11046.

[0019] U.S. application Ser. No. \_\_\_\_\_, entitled “Unsupervised Extraction of Facts”, filed on \_\_\_\_\_, by Jonathan T. Betz and Shubin Zhao, Attorney Docket No. 24207-11056.

**TECHNICAL FIELD**

[0020] The disclosed embodiments relate generally to fact databases. More particularly, the disclosed embodiments relate to identifying duplicate objects in an object collection.

**BACKGROUND**

[0021] Data is often organized as large collections of objects. When objects are added over time, there are often problems with data duplication. For example, a collection may include multiple objects that represent the same entity. As used herein, the term “duplicate objects” refers to objects representing the same entity. The names used to describe the represented entity are not necessarily the same among the duplicate objects.

[0022] Duplicate objects are undesirable for many reasons. They increase storage cost and take a longer time to process. They lead to inaccurate results, such as an inaccurate count of distinct objects. They also cause data inconsistency.

[0023] Conventional approaches identifying duplicate objects assume a homogeneity in the input set of objects (all books, all products, all movies, etc). Identifying duplication for objects of different type requires looking at different fields for different type. For example, when identifying duplicate objects in a set of objects representing books, traditional approaches match the ISBN value of the objects; when identifying duplicate objects in objects representing people, traditional approaches match the SSN value of the objects. One drawback of the conventional approaches is that they are only effective for specific types of objects, and

tend to be ineffective when applied to a collection of objects with different types. Also, even if the objects in the collection are of the same type, these approaches tend to be ineffective when the objects include incomplete or inaccurate information.

[0024] What is needed is a method and system that identifies duplicate objects in a large number of objects having different types and/or incomplete information.

#### SUMMARY

[0025] The invention is a system and method for identifying duplicate objects from a plurality of objects. For each object, the name used to describe the represented entity is normalized. A signature is generated for each object based on the normalized name. Objects are grouped into buckets based on the signature of the objects. Objects within the same bucket are compared to each other using a matcher to identify duplicate objects. The matcher can be selected from a collection of matchers.

[0026] This approach normalizes names used by objects to describe the represented entity. Therefore, objects representing the same entity share the same normalized name. As a result, this approach can identify duplicate objects even if the associated names initially are different. This approach is also computationally cost-efficient because objects are pairwise matched only within a bucket, rather than being pairwise matched across all buckets.

[0027] These features and benefits are not the only features and benefits of the invention. In view of the drawings, specification, and claims, many additional features and benefits will be apparent.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0028] FIG. 1 shows a network, in accordance with a preferred embodiment of the invention.

[0029] FIGS. 2(a)-2(d) are block diagrams illustrating a data structure for facts within a repository of FIG. 1 in accordance with preferred embodiments of the invention.

[0030] FIG. 2(e) is a block diagram illustrating an alternate data structure for facts and objects in accordance with preferred embodiments of the invention.

[0031] FIG. 3 is a flowchart of an exemplary method for identifying duplicate objects in accordance with a preferred embodiment of the invention.

[0032] FIG. 4 is a simplified diagram illustrating an object being processed for identification of its duplicate objects in accordance with a preferred embodiment of the invention.

[0033] FIGS. 5(a)-(e) illustrate an example of identifying duplicate objects, in accordance with a preferred embodiment of the invention.

[0034] The figures depict various embodiments of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention described herein.

#### DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

##### System Architecture

[0035] FIG. 1 shows a system architecture 100 adapted to support one embodiment of the invention. FIG. 1 shows components used to add facts into, and retrieve facts from a repository 115. The system architecture 100 includes a network 104, through which any number of document hosts 102 communicate with a data processing system 106, along with any number of object requesters 152, 154.

[0036] Document hosts 102 store documents and provide access to documents. A document is comprised of any machine-readable data including any combination of text, graphics, multimedia content, etc. A document may be encoded in a markup language, such as Hypertext Markup Language (HTML), i.e., a web page, in an interpreted language (e.g., JavaScript) or in any other computer readable or executable format. A document can include one or more hyperlinks to other documents. A typical document will include one or more facts within its content. A document stored in a document host 102 may be located and/or identified by a Uniform Resource Locator (URL), or Web address, or any other appropriate form of identification and/or location. A document host 102 is implemented by a computer system, and typically includes a server adapted to communicate over the network 104 via networking protocols (e.g., TCP/IP), as well as application and presentation protocols (e.g., HTTP, HTML, SOAP, D-HTML, Java). The documents stored by a host 102 are typically held in a file directory, a database, or other data repository. A host 102 can be implemented in any computing device (e.g., from a PDA or personal computer, a workstation, mini-computer, or mainframe, to a cluster or grid of computers), as well as in-any processor architecture or operating system.

[0037] FIG. 1 shows components used to manage facts in a fact repository 115. Data processing system 106 includes one or more importers 108, one or more janitors 110, a build engine 112, a service engine 114, and a fact repository 115 (also called simply a "repository"). Each of the foregoing are implemented, in one embodiment, as software modules (or programs) executed by processor 116. Importers 108 operate to process documents received from the document hosts, read the data content of documents, and extract facts (as operationally and programmatically defined within the data processing system 106) from such documents. The importers 108 also determine the subject or subjects with which the facts are associated, and extract such facts into individual items of data, for storage in the fact repository 115. In one embodiment, there are different types of importers 108 for different types of documents, for example, dependent on the format or document type.

[0038] Janitors 110 operate to process facts extracted by importer 108. This processing can include but is not limited to, data cleansing, object merging, and fact induction. In one embodiment, there are a number of different janitors 110 that perform different types of data management operations on the facts. For example, one janitor 110 may traverse some set of facts in the repository 115 to find duplicate facts (that is, facts that convey the same factual information) and merge them. Another janitor 110 may also normalize facts into standard formats. Another janitor 110 may also remove unwanted facts from repository 115, such as facts related to

pornographic content. Other types of janitors **110** may be implemented, depending on the types of data management functions desired, such as translation, compression, spelling or grammar correction, and the like.

[0039] Various janitors **110** act on facts to normalize attribute names, and values and delete duplicate and near-duplicate facts so an object does not have redundant information. For example, we might find on one page that Britney Spears' birthday is "Dec. 2, 1981" while on another page that her date of birth is "Dec. 2, 1981." Birthday and Date of Birth might both be rewritten as Birthdate by one janitor and then another janitor might notice that Dec. 2, 1981 and Dec. 2, 1981 are different forms of the same date. It would choose the preferred form, remove the other fact and combine the source lists for the two facts. As a result when you look at the source pages for this fact, on some you'll find an exact match of the fact and on others text that is considered to be synonymous with the fact.

[0040] Build engine **112** builds and manages the repository **115**. Service engine **114** is an interface for querying the repository **115**. Service engine **114**'s main function is to process queries, score matching objects, and return them to the caller but it is also used by janitor **110**.

[0041] Repository **115** stores factual information extracted from a plurality of documents that are located on document hosts **102**. A document from which a particular fact may be extracted is a source document (or "source") of that particular fact. In other words, a source of a fact includes that fact (or a synonymous fact) within its contents.

[0042] Repository **115** contains one or more facts. In one embodiment, each fact is associated with exactly one object. One implementation for this association includes in each fact an object ID that uniquely identifies the object of the association. In this manner, any number of facts may be associated with an individual object, by including the object ID for that object in the facts. In one embodiment, objects themselves are not physically stored in the repository **115**, but rather are defined by the set or group of facts with the same associated object ID, as described below. Further details about facts in repository **115** are described below, in relation to FIGS. 2(a)-2(d).

[0043] It should be appreciated that in practice at least some of the components of the data processing system **106** will be distributed over multiple computers, communicating over a network. For example, repository **115** may be deployed over multiple servers. As another example, the janitors **110** may be located on any number of different computers. For convenience of explanation, however, the components of the data processing system **106** are discussed as though they were implemented on a single computer.

[0044] In another embodiment, some or all of document hosts **102** are located on data processing system **106** instead of being coupled to data processing system **106** by a network. For example, importer **108** may import facts from a database that is a part of or associated with data processing system **106**.

[0045] FIG. 1 also includes components to access repository **115** on behalf of one or more object requesters **152**, **154**. Object requesters are entities that request objects from repository **115**. Object requesters **152**, **154** may be understood as clients of the system **106**, and can be implemented

in any computer device or architecture. As shown in FIG. 1, a first object requester **152** is located remotely from system **106**, while a second object requester **154** is located in data processing system **106**. For example, in a computer system hosting a blog, the blog may include a reference to an object whose facts are in repository **115**. An object requester **152**, such as a browser displaying the blog will access data processing system **106** so that the information of the facts associated with the object can be displayed as part of the blog web page. As a second example, janitor **110** or other entity considered to be part of data processing system **106** can function as object requester **154**, requesting the facts of objects from repository **115**.

[0046] FIG. 1 shows that data processing system **106** includes a memory **107** and one or more processors **116**. Memory **107** includes importers **108**, janitors **110**, build engine **112**, service engine **114**, and requester **154**, each of which are preferably implemented as instructions stored in memory **107** and executable by processor **116**. Memory **107** also includes repository **115**. Repository **115** can be stored in a memory of one or more computer systems or in a type of memory such as a disk. FIG. 1 also includes a computer readable medium **118** containing, for example, at least one of importers **108**, janitors **110**, build engine **112**, service engine **114**, requester **154**, and at least some portions of repository **115**. FIG. 1 also includes one or more input/output devices **120** that allow data to be input and output to and from data processing system **106**. It will be understood that data processing system **106** preferably also includes standard software components such as operating systems and the like and further preferably includes standard hardware components not shown in the figure for clarity of example.

#### Data Structure

[0047] FIG. 2(a) shows an example format of a data structure for facts within repository **115**, according to some embodiments of the invention. As described above, the repository **115** includes facts **204**. Each fact **204** includes a unique identifier for that fact, such as a fact ID **210**. Each fact **204** includes at least an attribute **212** and a value **214**. For example, a fact associated with an object representing George Washington may include an attribute of "date of birth" and a value of "Feb. 22, 1732." In one embodiment, all facts are stored as alphanumeric characters since they are extracted from web pages. In another embodiment, facts also can store binary data values. Other embodiments, however, may store fact values as mixed types, or in encoded formats.

[0048] As described above, each fact is associated with an object ID **209** that identifies the object that the fact describes. Thus, each fact that is associated with a same entity (such as George Washington), will have the same object ID **209**. In one embodiment, objects are not stored as separate data entities in memory. In this embodiment, the facts associated with an object contain the same object ID, but no physical object exists. In another embodiment, objects are stored as data entities in memory, and include references (for example, pointers or IDs) to the facts associated with the object. The logical data structure of a fact can take various forms; in general, a fact is represented by a tuple that includes a fact ID, an attribute, a value, and an object ID. The storage implementation of a fact can be in any underlying physical data structure.

[0049] FIG. 2(b) shows an example of facts having respective fact IDs of 10, 20, and 30 in repository 115. Facts 10 and 20 are associated with an object identified by object ID “1.” Fact 10 has an attribute of “Name” and a value of “China.” Fact 20 has an attribute of “Category” and a value of “Country.” Thus, the object identified by object ID “1” has a name fact 205 with a value of “China” and a category fact 206 with a value of “Country.” Fact 30208 has an attribute of “Property” and a value of “Bill Clinton was the 42nd President of the United States from 1993 to 2001.” Thus, the object identified by object ID “2” has a property fact with a fact ID of 30 and a value of “Bill Clinton was the 42nd President of the United States from 1993 to 2001.” In the illustrated embodiment, each fact has one attribute and one value. The number of facts associated with an object is not limited; thus while only two facts are shown for the “China” object, in practice there may be dozens, even hundreds of facts associated with a given object. Also, the value fields of a fact need not be limited in size or content. For example, a fact about the economy of “China” with an attribute of “Economy” would have a value including several paragraphs of text, numbers, perhaps even tables of figures. This content can be formatted, for example, in a markup language. For example, a fact having an attribute “original html” might have a value of the original html text taken from the source web page.

[0050] Also, while the illustration of FIG. 2(b) shows the explicit coding of object ID, fact ID, attribute, and value, in practice the content of the fact can be implicitly coded as well (e.g., the first field being the object ID, the second field being the fact ID, the third field being the attribute, and the fourth field being the value). Other fields include but are not limited to: the language used to state the fact (English, etc.), how important the fact is, the source of the fact, a confidence value for the fact, and so on.

[0051] FIG. 2(c) shows an example object reference table 210 that is used in some embodiments. Not all embodiments include an object reference table. The object reference table 210 functions to efficiently maintain the associations between object IDs and fact IDs. In the absence of an object reference table 210, it is also possible to find all facts for a given object ID by querying the repository to find all facts with a particular object ID. While FIGS. 2(b) and 2(c) illustrate the object reference table 210 with explicit coding of object and fact IDs, the table also may contain just the ID values themselves in column or pair-wise arrangements.

[0052] FIG. 2(d) shows an example of a data structure for facts within repository 115, according to some embodiments of the invention showing an extended format of facts. In this example, the fields include an object reference link 216 to another object. The object reference link 216 can be an object ID of another object in the repository 115, or a reference to the location (e.g., table row) for the object in the object reference table 210. The object reference link 216 allows facts to have as values other objects. For example, for an object “United States,” there may be a fact with the attribute of “president” and the value of “George W. Bush,” with “George W. Bush” being an object having its own facts in repository 115. In some embodiments, the value field 214 stores the name of the linked object and the link 216 stores the object identifier of the linked object. Thus, this “president” fact would include the value 214 of “George W. Bush,” and object reference link 216 that contains the object

ID for the for “George W. Bush” object. In some other embodiments, facts 204 do not include a link field 216 because the value 214 of a fact 204 may store a link to another object.

[0053] Each fact 204 also may include one or more metrics 218. A metric provides an indication of the some quality of the fact. In some embodiments, the metrics include a confidence level and an importance level. The confidence level indicates the likelihood that the fact is correct. The importance level indicates the relevance of the fact to the object, compared to other facts for the same object. The importance level may optionally be viewed as a measure of how vital a fact is to an understanding of the entity or concept represented by the object.

[0054] Each fact 204 includes a list of one or more sources 220 that include the fact and from which the fact was extracted. Each source may be identified by a Uniform Resource Locator (URL), or Web address, or any other appropriate form of identification and/or location, such as a unique document identifier.

[0055] The facts illustrated in FIG. 2(d) include an agent field 222 that identifies the importer 108 that extracted the fact. For example, the importer 108 may be a specialized importer that extracts facts from a specific source (e.g., the pages of a particular web site, or family of web sites) or type of source (e.g., web pages that present factual information in tabular form), or an importer 108 that extracts facts from free text in documents throughout the Web, and so forth.

[0056] Some embodiments include one or more specialized facts, such as a name fact 207 and a property fact 208. A name fact 207 is a fact that conveys a name for the entity or concept represented by the object ID. A name fact 207 includes an attribute 224 of “name” and a value, which is the name of the object. For example, for an object representing the country Spain, a name fact would have the value “Spain.” A name fact 207, being a special instance of a general fact 204, includes the same fields as any other fact 204; it has an attribute, a value, a fact ID, metrics, sources, etc. The attribute 224 of a name fact 207 indicates that the fact is a name fact, and the value is the actual name. The name may be a string of characters. An object ID may have one or more associated name facts, as many entities or concepts can have more than one name. For example, an object ID representing Spain may have associated name facts conveying the country’s common name “Spain” and the official name “Kingdom of Spain.” As another example, an object ID representing the U.S. Patent and Trademark Office may have associated name facts conveying the agency’s acronyms “PTO” and “USPTO” as well as the official name “United States Patent and Trademark Office.” If an object does have more than one associated name fact, one of the name facts may be designated as a primary name and other name facts may be designated as secondary names, either implicitly or explicitly.

[0057] A property fact 208 is a fact that conveys a statement about the entity or concept represented by the object ID. Property facts are generally used for summary information about an object. A property fact 208, being a special instance of a general fact 204, also includes the same parameters (such as attribute, value, fact ID, etc.) as other facts 204. The attribute field 226 of a property fact 208 indicates that the fact is a property fact (e.g., attribute is

“property”) and the value is a string of text that conveys the statement of interest. For example, for the object ID representing Bill Clinton, the value of a property fact may be the text string “Bill Clinton was the 42nd President of the United States from 1993 to 2001.” Some object IDs may have one or more associated property facts while other objects may have no associated property facts. It should be appreciated that the data structures shown in FIGS. 2(a)-2(d) and described above are merely exemplary. The data structure of the repository 115 may take on other forms. Other fields may be included in facts and some of the fields described above may be omitted. Additionally, each object ID may have additional special facts aside from name facts and property facts, such as facts conveying a type or category (for example, person, place, movie, actor, organization, etc.) for categorizing the entity or concept represented by the object ID. In some embodiments, an object’s name(s) and/or properties may be represented by special records that have a different format than the general facts records 204.

[0058] As described previously, a collection of facts is associated with an object ID of an object. An object may become a null or empty object when facts are disassociated from the object. A null object can arise in a number of different ways. One type of null object is an object that has had all of its facts (including name facts) removed, leaving no facts associated with its object ID. Another type of null object is an object that has all of its associated facts other than name facts removed, leaving only its name fact(s). Alternatively, the object may be a null object only if all of its associated name facts are removed. A null object represents an entity or concept for which the data processing system 106 has no factual information and, as far as the data processing system 106 is concerned, does not exist. In some embodiments, facts of a null object may be left in the repository 115, but have their object ID values cleared (or have their importance to a negative value). However, the facts of the null object are treated as if they were removed from the repository 115. In some other embodiments, facts of null objects are physically removed from repository 115.

[0059] FIG. 2(e) is a block diagram illustrating an alternate data structure 290 for facts and objects in accordance with preferred embodiments of the invention. In this data structure, an object 290 contains an object ID 292 and references or points to facts 294. Each fact includes a fact ID 295, an attribute 297, and a value 299. In this embodiment, an object 290 actually exists in memory 107.

#### Overview of Methodology

[0060] Referring now to FIG. 3, there is shown a flow diagram depicting a method for identifying duplicate objects in accordance with a preferred embodiment of the invention. The steps of the process illustrated in FIG. 3 may be implemented in software, hardware, or a combination of hardware and software.

[0061] In one embodiment, the present invention is implemented in a janitor 110 to identify duplicate objects so that the duplicate objects can be merged together. Duplicate objects are objects representing the same entity but having a different object ID. Janitor 110 examines object reference table 210, and reconstructs the objects based on the associations between object IDs and fact IDs maintained in object reference table 210. Alternatively, janitor 110 can retrieve objects by asking service engine 114 for the infor-

mation stored in repository 115. Depending how object information is stored in repository 115, janitor 110 needs to reconstruct the objects based on the facts and object information retrieved.

[0062] The flowchart shown in FIG. 3 will now be described in detail, illustrated by the diagram in FIG. 4 and the example in FIGS. 5(a)-(e). The process commences with a set of objects that may contain duplicate objects. For example, there may be multiple objects that represent the entity U.S. President George Washington. Each object has an associated set of facts. As illustrated in FIG. 2(a), each fact 204 has an attribute 212 and a value 214 (also called fact value).

[0063] As shown in FIG. 5(a), objects O1 and O3 are duplicate objects representing the same entity, a Mr. John W. Henry living in Lamar CO 81052. O1 is associated with three facts with the following attributes: name, date of birth, and phone number. O3 is associated with three facts: name, address, and phone number. Object O2 represents a race horse named John Henry. O2 is associated with four facts: name, date of birth, address, and phone number (the phone number of the horse park where the race horse lives). Object O4 represents a Senator John Henry. O4 is associated with three facts: name, date of birth, and date of death. Objects O5 and O6 are duplicate objects representing US President John F. Kennedy. O5 is associated with three facts: name, date of birth, and date of death. O6 is also associated with three facts: name, date of birth, and date of death. Note that among the duplicate objects, there are considerable variations in the associated name values. A name value is the value of a name fact, a fact with attribute name. A preferred embodiment of the present invention can be used on collections of objects numbering from tens of thousands, to millions, or more.

[0064] Referring to FIGS. 3 and 4, for each object 430, janitor 110 normalizes the associated name value by calling a normalizer 410. As observed above, there are considerable variations in the name values associated with duplicate objects. Normalizer 410 is designed to standardize the name values such that the normalized name values of duplicate objects are the same. It is noted that the above normalization rules can be applied to fact values other than the name values.

[0065] In one embodiment, normalizer 410 normalizes a name value by applying a set of normalization rules to the name value. A normalization rule can remove from the name value information unnecessary to describe the represented entity (e.g., removing the from the United States). Alternatively, a normalization rule can standardize the format of the name value (e.g., changing a person’s name from a last name first order to a first name first order, such as from Washington, George to George Washington). Some of the normalization rules are language specific while others are universally applicable to name values in different languages. Some embodiments allow a name fact to indicate that the associated name value is an exception to one or more of the normalization rules. When normalizer 410 identifies such indication it will not apply the normalization rules indicated. For example, an object with a name value of J. F. K. may indicate that the associated name value is an exception to a single-letter-word removal rule.

[0066] One example of the normalization rules, uppercase-to-lowercase conversion rule, converts uppercase char-

acters in a name value to corresponding lowercase characters, such as from “America” to “america.” The name values of some duplicate objects may use capital characters to describe the represented entity while the name values of others may ignore capital characters. For example, one object representing the Apple computer iMac may have a name value of iMac, while other objects representing the same entity may have a name value of Imac, imac, or IMAC. Each of the above four distinct name values describes the same entity—the Apple computer iMac. By applying the uppercase-to-lowercase conversion rule, all four name values are standardized to be imac. Applying the uppercase-to-lowercase conversion rule to the name values of the set of objects illustrated in FIG. 5(a), the resulting normalized name values are shown in FIG. 5(b).

[0067] Another example of the normalization rules, stop-words removal rule, removes stop words from name values. Stop words are small or frequently used words that are generally overlooked by the search engines. Common stop words are words such as the, a, an, this, and that. Stop words tend to convey no additional value in describing the represented entity, therefore the name values of some objects include stop words while the name values of others do not. For example, for two duplicate objects describing the United Nations, one may have a name value of the United Nations while the other may have a name value of United Nations. By applying the above stop-words-removal rule, the two duplicate objects’ name values are standardized to be United Nations. Normalizer 410 can dynamically update the collection of words it deems as stop words.

[0068] Another example of the normalization rules, social-titles removal rule, removes social titles from name values. Social titles are identifying appellations signifying status of the entity described. Common social titles are words such as Mr., Ms., Mrs., Miss, Sir, etc. Because social titles are not essential in identifying the represented entity, the name values of some objects do not include them. For example, for two duplicate objects representing the English mathematician and physicist Isaac Newton, one may have a name value of Sir Isaac Newton while the other may have a name value of Isaac Newton. By applying the above social-titles removal rule, the two duplicate objects’ name values are standardized to be Isaac Newton. Similarly, the normalization rules can include a honorific-titles removal rule which removes honorific words such as General, President, Congressman, Senator from the name values.

[0069] Applying the uppercase-to-lowercase conversion rule to the normalized name values shown in FIG. 5(b), the resulting normalized name values are shown in FIG. 5(c). Normalizer 410 removes the social title name from the normalized name value of O3. Similarly, normalizer 410 removes the honorific-titles senator and president from the normalized name values of O4 and O6 accordingly.

[0070] Another example of the normalization rules, single-letter-word removal rule, removes single letter words from name values. When identifying an entity, certain non-essential words are often omitted or only shown their initial characters. One example of such non-essential words is a person’s middle name. Some objects representing a person includes the person’s middle name initial in the associated name value while others do not. For example, for two duplicate objects representing a John Henry, one may have

a name value of John W. Henry while the other may have a name value of John Henry. By applying the above single-letter-word removal rule and a punctuation-marks removal rule as described below, the two duplicate objects’ name values are standardized to be John Henry.

[0071] Another example of the normalization rules, alphabetic sort rule, sorts the words in a name value in alphabetic order. When identifying an entity, the name of the entity can be in one of several different formats. For example, China can be either called People’s Republic of China or China, People’s Republic. Also, a person can either be addressed in a first-name first way or in a more formal last-name first way. Correspondingly, for two duplicate objects representing a person named John Henry, the name value of one object can be John Henry while the name value of the other can be Henry, John. By applying the alphabetic sort rule and a punctuation-marks removal rule as described below, the two duplicate objects’ name values are standardized to be Henry John.

[0072] Yet another example of the normalization rules, punctuation-marks removal rule, removes punctuation marks from name values. Punctuation marks are used to clarify meaning by indicating separation of words into clauses and phrases. Because punctuation marks are not essential in identifying an entity, some objects omit them in the associated name values. Also, punctuation marks in a fact value may become unnecessary after normalizer 410 applies one or more normalization rules to the fact value. For example, after applying the alphabetic sorting rule to a name value of Henry, Bill, the name value becomes Bill Henry, and the comma mark becomes unnecessary. The punctuation removal rule removes the extra comma sign and standardizes the name value to be Bill Henry.

[0073] Applying the single-letter-word removal rule, the alphabetic sort rule, and the punctuation-marks removal rule to the normalized name values shown in FIG. 5(c), the resulting normalized name values are shown in FIG. 5(d). The normalized name value of O1 first becomes john.henry after normalizer 410 applies the single-letter-word removal rule, then becomes henry john. after the alphabetic sort rule, and eventually becomes henryjohn after the punctuation-marks removal rule removes the period mark. Similarly, the normalized name values of O2, O3, and O4 become henry john, and those of O5 and O6 become john kennedy.

[0074] Referring to FIGS. 3 and 4, janitor 110 generates 320 a signature 450 for object 430 by calling a signature generator 440. Signature generator 440 is a function/module designed to generate an identical signature for duplicate objects even if the facts associated with the objects are not duplicates. Janitor 110 then groups 330 objects 430 into a plurality of buckets 460 in the following fashion. Janitor 110 puts an object 430 into an existing bucket 460 indexed by signature 450. If there is no such bucket 460 then a new bucket 460 is created, the signature 450 is assigned as the index of the bucket 460, and the object 430 is put into the bucket 460. When all objects 430 are processed, those objects sharing a signature are in the same bucket.

[0075] The purpose of generating a signature is to optimize the object normalization process. In general, normalizing a repository of objects requires comparing all possible pairs of objects in the repository, which is computationally impractical for a large collection of objects. As an optimi-

zation, it's desirable to design the signature generator 440 to always create the same signature for duplicate objects. As a result, only objects sharing the same signature need to be compared to identify duplicate objects and normalize the repository of objects. In order for the signature generator 440 to always create the same signature for duplicate objects, it needs to be inclusive and ignore minor differences among the objects.

[0076] In one embodiment, signature generator 440 generates 320 signatures 450 based solely on the name values of objects 430. For example, signature generator 440 can generate 320 the signature 450 by removing any white space in the name value of object 430. Janitor 110 then places object 430 into a bucket 460 in accordance with signature 450.

[0077] It is noted that signatures 450 generated by signature generator 440 can be a null signature, a signature with an empty value. In one embodiment, janitor 110 does not place an object 430 with a null signature into any bucket 460. As a result, objects with null signatures are neither compared nor merged with other objects. Signature generator 440 can generate a null signature because the object does not have a name fact. Signature generator 440 can also generate a null signature because the normalized name value of the object is empty (e.g., the original name value consists only of stop words, and the normalizer 410 removes all the stop words from the name value). Alternatively, the signature generator 440 can purposefully generate a null signature for certain objects to prevent the objects from being considered for merger.

[0078] FIG. 5(e) shows in which buckets the objects 430 ultimately placed. Applying the above signature generator 440, objects O1 to O4 are properly grouped into a bucket indexed by a signature 450 based on henryjohn, the normalized name value of objects O1-O4. Objects O5 and O6 are placed in a bucket indexed by a signature 450 based on john kennedy, the normalized name value of both objects.

[0079] Alternatively, signature generator 440 can generate 320 signatures 450 based on a combination of name values and other fact values of the associated objects 430. In one example, the signature generator 440 applies some normalization rules similar to the ones described above to the other fact values before generating 320 the signature 450.

[0080] By normalizing 310 the name values of each object, janitor 110 can detect duplicate objects with different name values describing the same represented entity. Objects 430 created from different data sources may not share the same name value, even if they represent the same entity. For example, an object 430 representing George Washington created based on a webpage devoted to his childhood may have a name value of George Washington, while another object 430 also representing George Washington created based on a webpage dedicated to his years of presidency probably would have a different name value of President George Washington. By normalizing the name values of each object, janitor 110 can standardize the name values such that objects representing the same entity share the same normalized name value. For example, the normalizes name value of both of the above objects are george Washington.

[0081] Because signature 450 is based on the normalized name value, signature generator 440 generates the same

signature 450 for duplicate objects. Because janitor 110 groups objects 430 based on their associated signatures, duplicate objects tend to be grouped 330 into the same bucket 460. For example, as illustrated in FIG. 5(e), duplicate objects O1 and O3 are placed in the same bucket and duplicate objects O5 and O6 are placed in the same bucket.

[0082] After all objects are grouped 430 into buckets 460, for every bucket 460 created, janitor 110 applies 340 a matcher 420 to every two objects in bucket 460, and identifies 350 the matching objects 470 as duplicate objects. Matcher 420 is designed to match duplicate objects based on the similarity of facts with the same attribute associated with the two objects (also called simply common facts). Similarity between two corresponding facts can be determined in a number of ways. For example, two facts are determined to be similar when the fact values are identical. In another example, two facts can be determined to be similar when the fact values are lexically similar, such as "U.S.A." and "United States." Alternatively, two facts are determined to be similar when the fact values are proximately similar, such as "176 pounds" and "176.1 pounds." In another example, two facts are determined to be similar when the fact values are similar based on string similarity measure (e.g., edit distance, Hamming Distance, Levenshtein Distance, Smith-Waterman Distance, Gotoh Distance, Jaro Distance Metric, Dice's Coefficient, Jaccard Coefficient to name a few).

[0083] For example, matcher 420 determines whether two objects match based on the number of common facts with similar values (also called simply similar common facts) and the number of common facts with values that are not similar (also called simply dissimilar common facts). In one such matcher 420, two objects are deemed to match when there is more similar common fact than dissimilar common facts. Because the name values are used to generate 320 the signatures of each object 430, matcher 420 does not consider name fact when determining whether two objects match.

[0084] When the above matcher 420 is applied to the buckets shown in FIG. 5(e), O1 and O3 are determined to match because they have one similar common fact: the fact with attribute phone number, and no dissimilar common facts. O5 and O6 are also determined to match because there are two similar common facts: the facts associated with attributes of date of birth and date of death, and no dissimilar common facts. It is noted that matcher 420 ignores the format difference between the fact value associated with attribute phone number of O1 ((719) 123-4567) and O3 (719-123-4567). Similarly, the format difference between the fact value associated with attributes of date of birth and date of death of O5 and O6 is also ignored by matcher 420. As a result, the janitor 110 properly identifies O1 and O3 as duplicate objects and O5 and O6 as duplicate objects. More examples of how to determine whether two objects are duplicate objects can be found in U.S. Utility patent application Ser. No. 11/356,838 (Attorney Docket No. 10953), for "Modular Architecture For Entity Normalization," by Jonathan Betz, et al., filed Feb. 17, 2006.

[0085] In another embodiment, janitor 110 does not first apply matcher 420 to every two objects in bucket 460 and then identify matching objects 470 as duplicate objects. Instead, janitor 110 applies matcher 420 to two objects in bucket 460. If matcher 420 indicates the two objects to be matching objects 470, janitor 110 merges them, keeps the

merged object in bucket 460, and removes the other object(s) out of bucket 460. Then, janitor 110 restarts the process by applying matcher 420 to two objects in bucket 460 that have not been matched before. This process continues until matcher 420 has been applied to every pair of objects in bucket 460.

[0086] Janitor 110 can merge two objects in several different ways. For example, janitor 110 can choose one of the two objects as the merged object, add facts only present in the other object to the merged object, and optionally reconcile the dissimilar common facts of the merged object. Alternatively, janitor 110 can create a new object as the merged object, and add facts from the two matching objects to the merged object.

[0087] In another embodiment, a matcher 420 can be a function or a module. The system selects matcher 420 from a collection of matcher functions/modules. The collection of matcher functions/modules includes functions/modules provided by a third party and functions/modules previously created. By providing the ability to select a matcher 420, janitor 110 can reuse the existing well-tested functions/modules, and select matcher 420 based on the specific needs.

[0088] There are many ways for janitor 110 to select a matcher function/module. For example, janitor 110 can select matcher 420 based on system configuration data. Alternatively, the selection can be determined at run time based on information such as grouper 410 selected. For example, if the resulting buckets of grouper 410 include many objects, janitor 110 selects a matcher function/module requiring a higher entropy threshold.

[0089] After identifying 350 the matching objects as duplicate objects, janitor 110 can merge the duplicate objects into a merged object, so that each entity is represented by no more than one object and each fact that is associated with a same entity will have the same object ID.

[0090] Finally, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

What is claimed is:

1. A method of identifying duplicate objects in a plurality of objects, each of the plurality of objects having one or more associated facts, each of the one or more facts having a value, the method comprising:

for each of the plurality of objects, normalizing the value of a name fact, the name fact being among the one or more facts associated with the object;

grouping the plurality of objects into a plurality of buckets in accordance with the normalized value of the name facts of the plurality of objects; and

applying a matcher to a pair of objects in one of the plurality of buckets to determine if the pair of objects are duplicates, one of the pair of objects having an associated fact that is not a common fact of the pair of objects.

2. The method of claim 1, wherein normalizing the value of the name fact comprises:

normalizing the value of the name fact by applying a group of normalization rules to the value of the name fact.

3. The method of claim 2, wherein the group of normalization rules comprises at least one rule selected from a group consisting of:

removing social titles; and

removing honorific words.

4. The method of claim 2, wherein the group of normalization rules comprises at least one rule selected from a group consisting of:

removing single letter words;

removing punctuation marks;

removing stop words; and

converting uppercase characters into lowercase.

5. The method of claim 2, wherein the group of normalization rules comprises sorting the value in alphabetic order.

6. The method of claim 1, wherein grouping comprises:

generating a signature for each of the plurality of objects based at least in part on the normalized value of the name fact of each of the plurality of objects; and

responsive to an identifier of an existing bucket being the same as the signature of an object, adding the object to the existing bucket, otherwise establishing a new bucket including the object, an identifier of the new bucket being same as the signature of the object.

7. The method of claim 1, wherein applying the matcher to a pair of objects in one of the plurality of buckets comprises:

for each common fact of the pair of objects, determining a similarity of the values of the common fact based on a similarity measure; and

determining that the pair of objects are duplicates based on the similarity.

8. The method of claim 7, wherein determining that the pair of objects are duplicates comprises:

determining that the pair of objects are duplicates based on the number of the common facts with similar values and the number of common facts.

9. The method of claim 1, wherein applying the matcher to each pair of objects in one of the plurality of buckets to determine if the pair of objects are duplicates.

10. The method of claim 1, further comprising:

selecting the matcher from a collection of matchers, wherein applying the matcher includes applying the selected matcher to a pair of objects in one of the plurality of buckets to determine if the pair of objects are duplicates.

11. A system for identifying duplicate objects in a plurality of objects, each of the plurality of objects having one or more associated facts, each of the one or more facts having a value, the system comprising:

a processor for executing programs; and

a subsystem executable by the processor, the subsystem including:

instructions for normalizing the value of a name fact of each of the plurality of objects, the name fact being among the one or more facts associated with the object;

instructions for grouping the plurality of objects into a plurality of buckets in accordance with the normalized value of the name facts of the plurality of objects; and

instructions for applying a matcher to a pair of objects in one of the plurality of buckets to determine if the pair of objects are duplicates, one of the pair of objects having an associated fact that is not a common fact of the pair of objects.

12. A computer program product for use in conjunction with a computer system, the computer program product comprising a computer readable storage medium and a

computer program mechanism embedded therein, the computer program mechanism including:

instructions for normalizing the value of a name fact of each of the plurality of objects, the name fact being among the one or more facts associated with the object;

instructions for grouping the plurality of objects into a plurality of buckets in accordance with the normalized value of the name facts of the plurality of objects; and

instructions for applying a matcher to a pair of objects in one of the plurality of buckets to determine if the pair of objects are duplicates, one of the pair of objects having an associated fact that is not a common fact of the pair of objects.

\* \* \* \* \*