



(19) **United States**

(12) **Patent Application Publication**
Hogue et al.

(10) **Pub. No.: US 2007/0143317 A1**

(43) **Pub. Date: Jun. 21, 2007**

(54) **MECHANISM FOR MANAGING FACTS IN A FACT REPOSITORY**

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

(76) Inventors: **Andrew Hogue**, Ho Ho Kus, NJ (US);
Jonathan Betz, Summit, NJ (US)

Continuation-in-part of application No. 11/356,765, filed on Feb. 17, 2006.

Publication Classification

(51) **Int. Cl.**
G06F 7/00 (2006.01)
(52) **U.S. Cl.** **707/100**

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

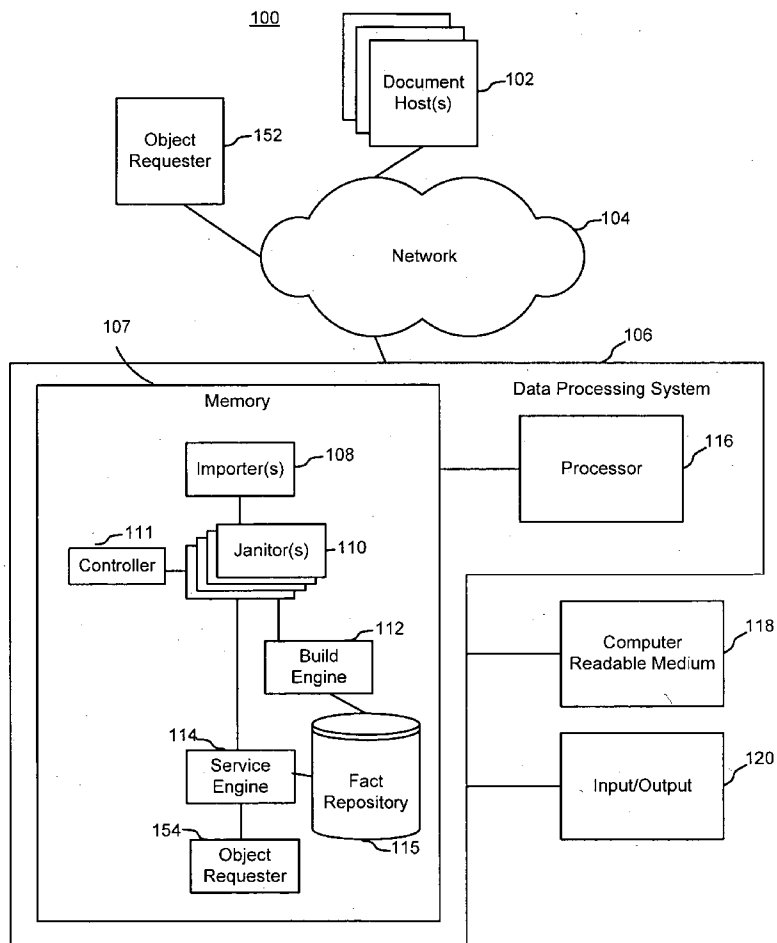
(57) **ABSTRACT**
Methods and systems for processing facts with one or more janitors. Facts are extracted from documents on the Internet or other sources. Facts can be any data or series of data in the documents including an attribute and a file. The data can be in the form of text, graphics, or multimedia content. Janitors transform facts responsive to inferring a certain condition associated with facts. The condition can be related to one or more of an attribute, a value, or an object of a fact being analyzed. For example, janitors can perform normalization, remove or merge similar or duplicate facts, segregate multiple values of a fact, and the like. An administrator can select which janitors are applied to facts and in which order.

(21) Appl. No.: **11/399,857**

(22) Filed: **Apr. 7, 2006**

Related U.S. Application Data

(63) Continuation-in-part of application No. 11/024,784, filed on Dec. 30, 2004.
Continuation-in-part of application No. 11/142,853, filed on May 31, 2005.
Continuation-in-part of application No. 11/341,069, filed on Jan. 27, 2006.



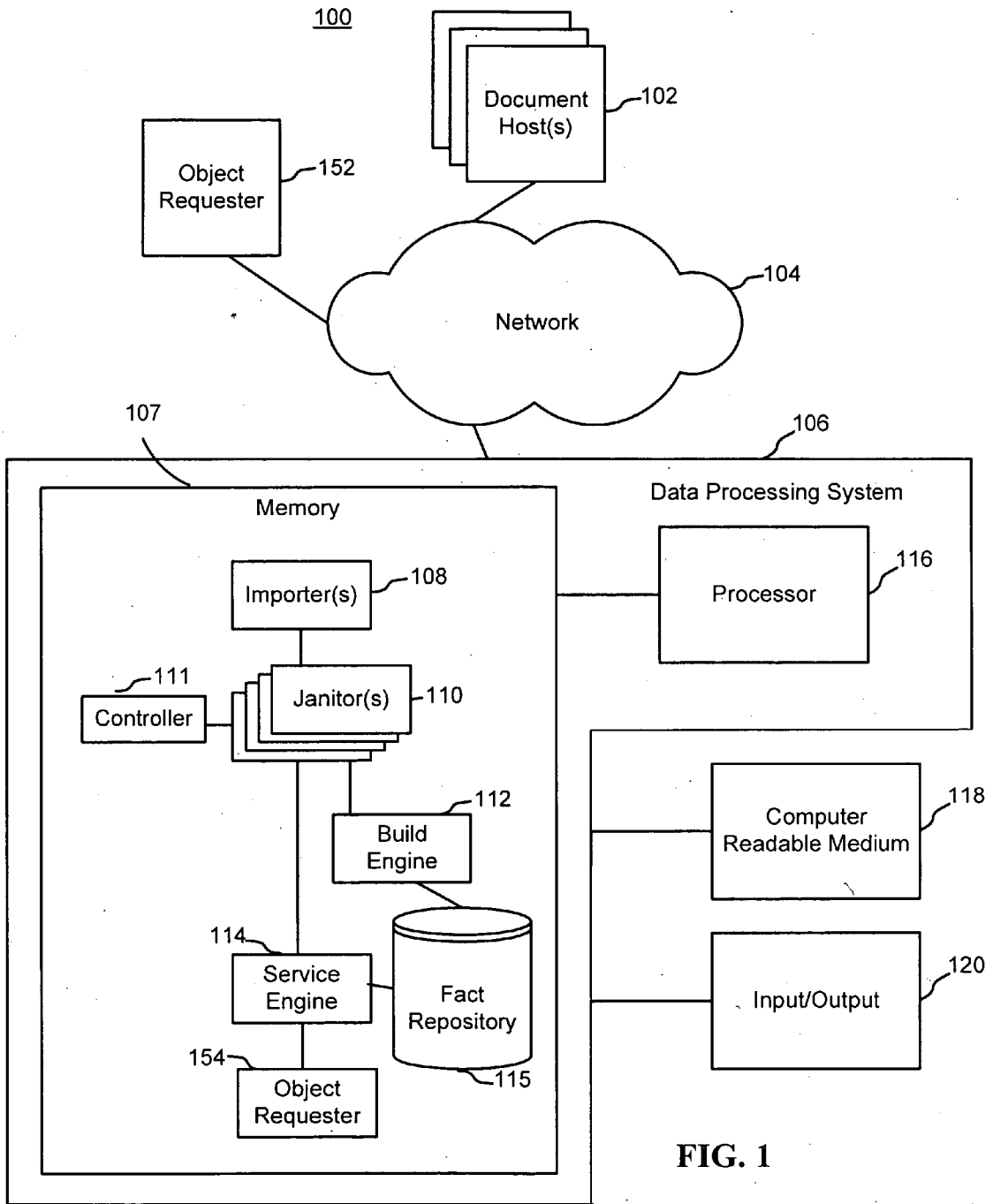


FIG. 1

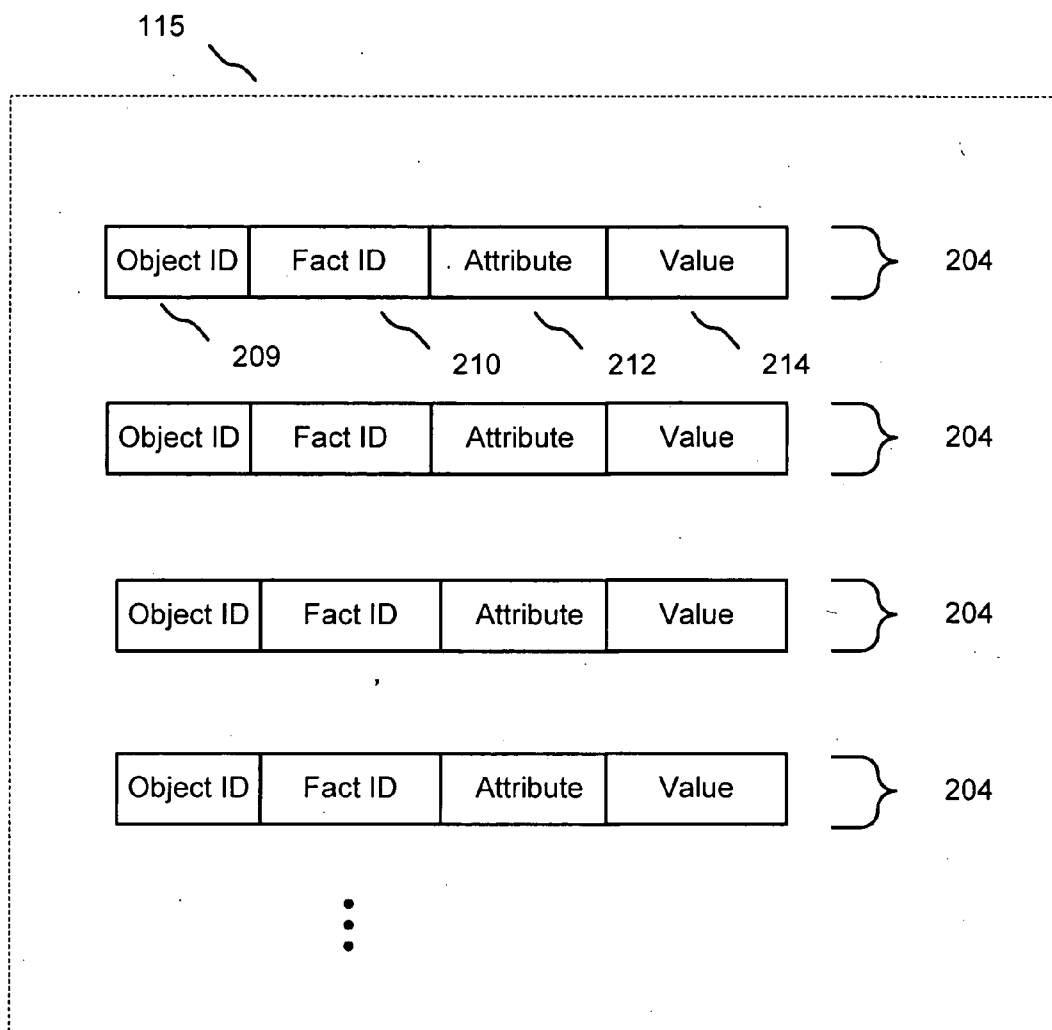


FIG. 2(a)
Example Format of Facts in Repository (each fact is associated with an object ID)

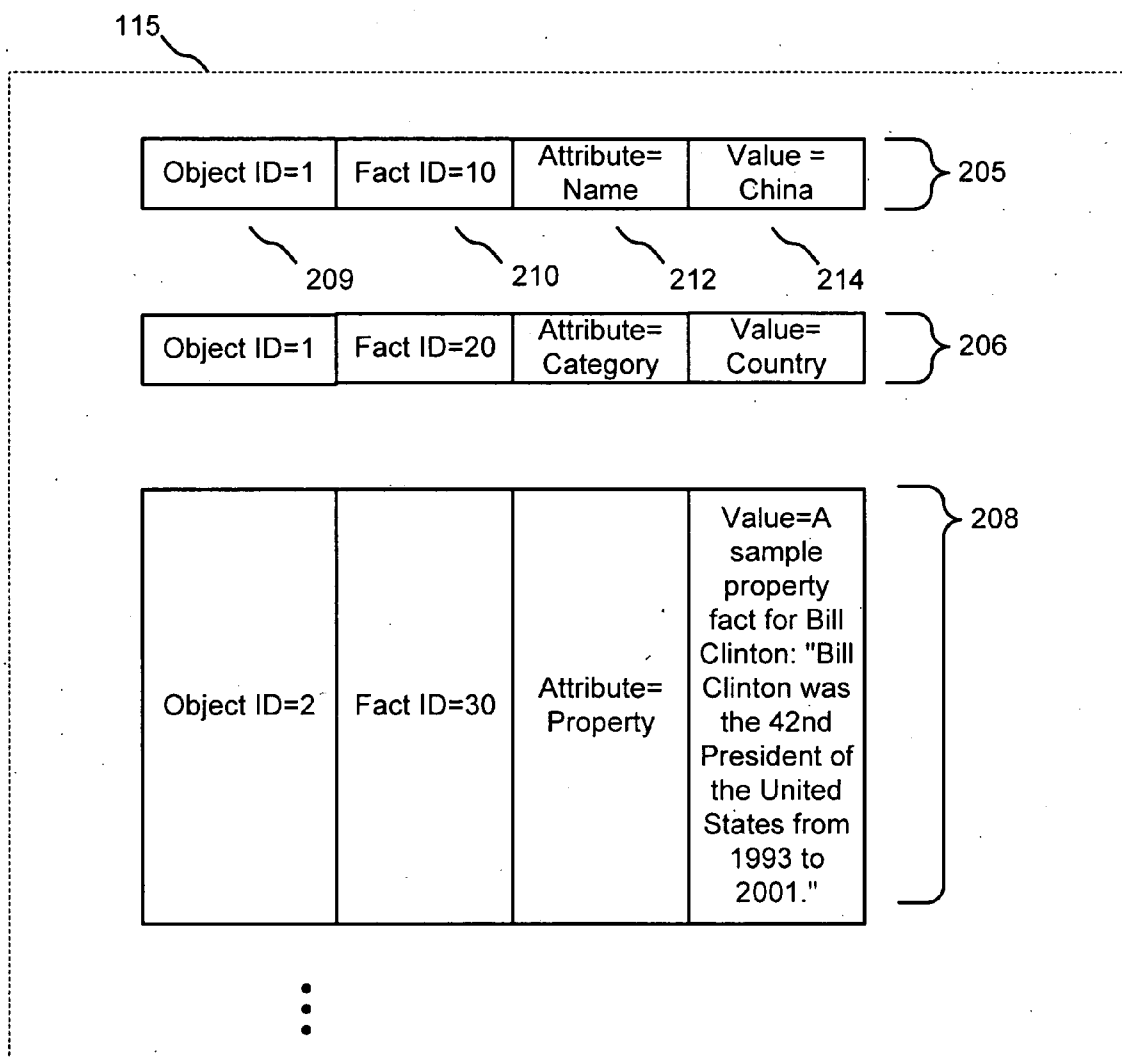


FIG. 2(b)
Example Facts in Repository (each fact is associated with an object ID)

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

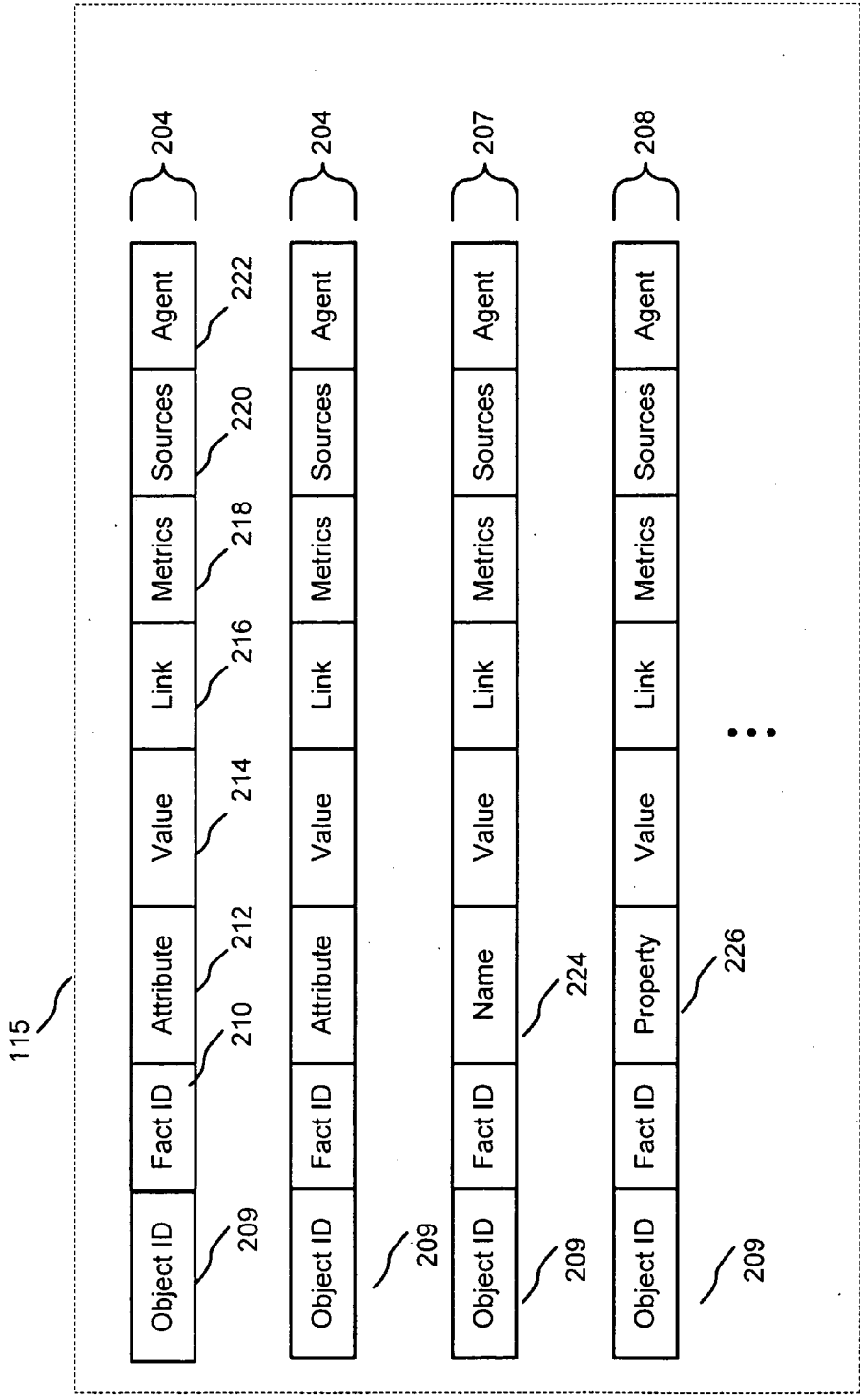


FIG. 2(d)
Example Format of Facts in
Repository (each fact is associated
with an object ID)

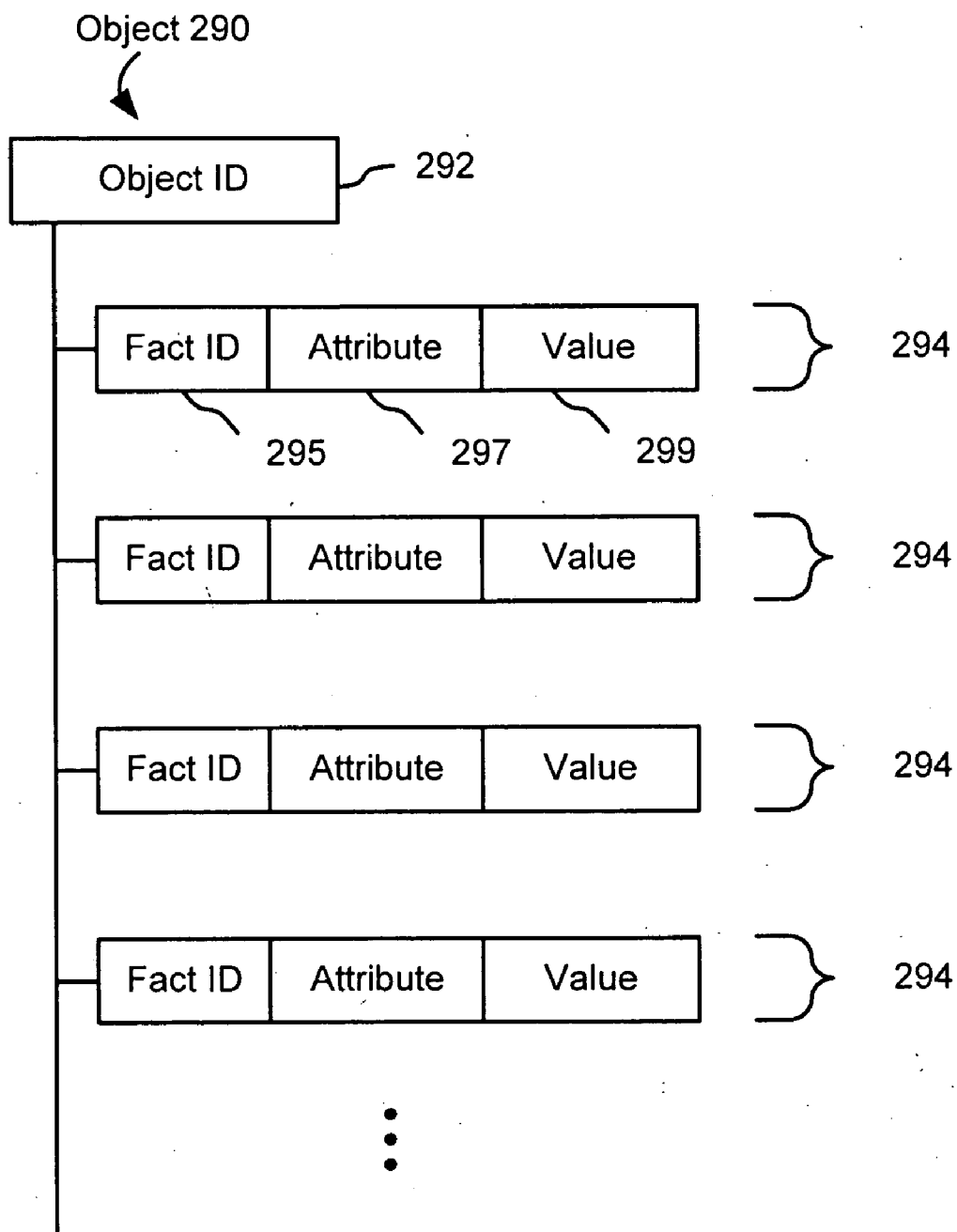


FIG. 2(e)
Example Objects

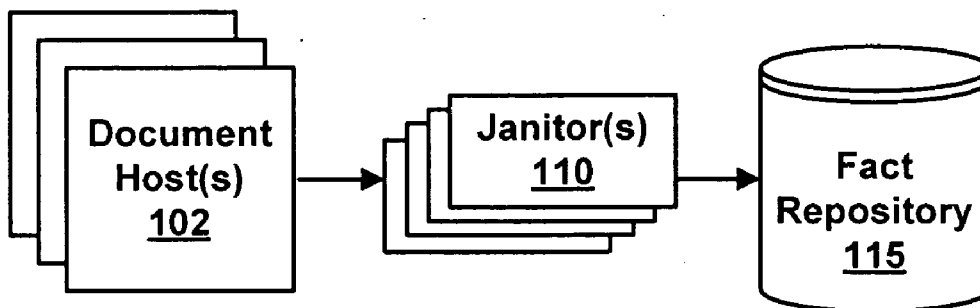


FIG. 3a

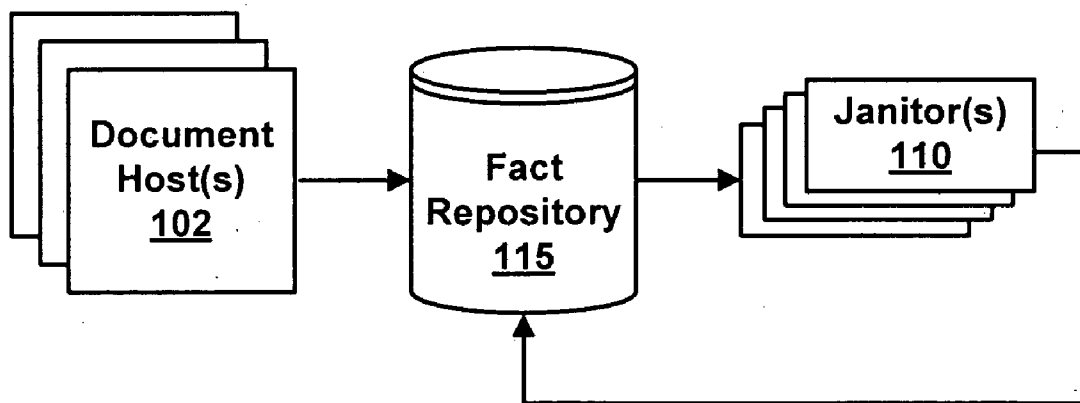


FIG. 3b

400

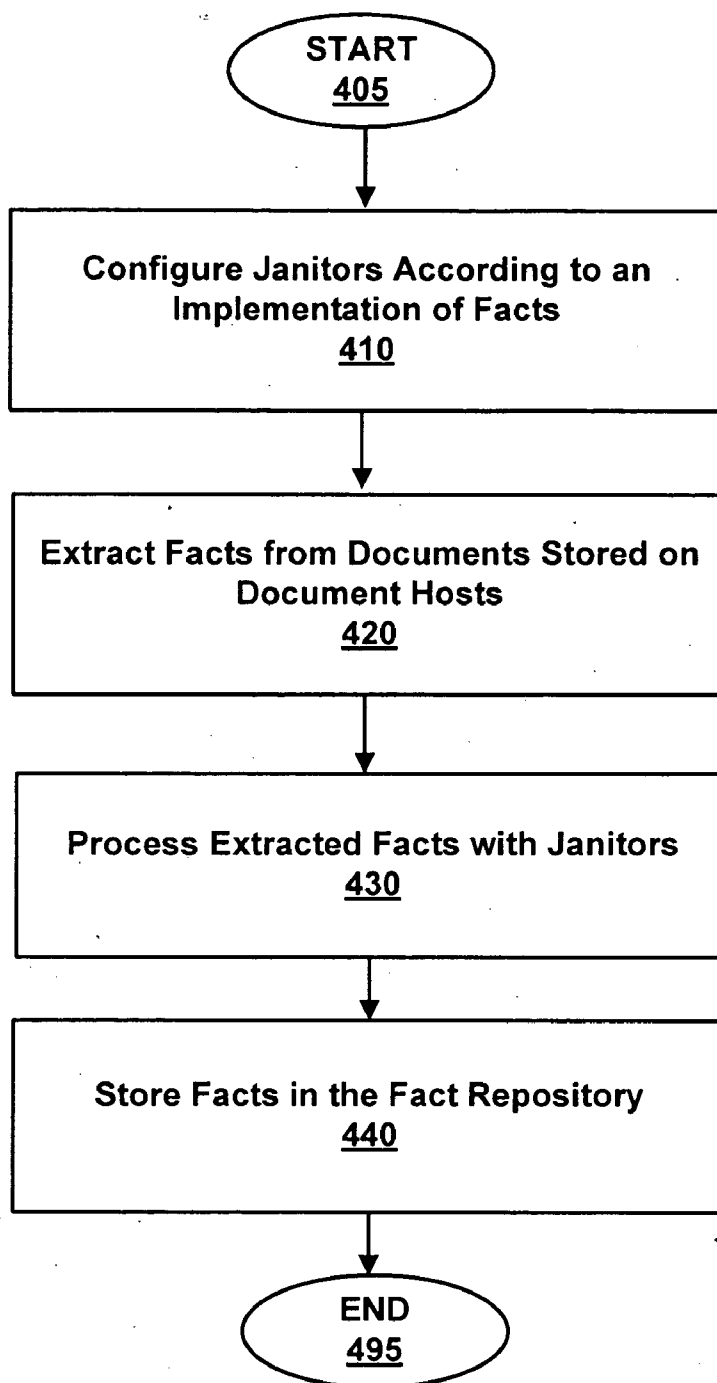


FIG. 4

500

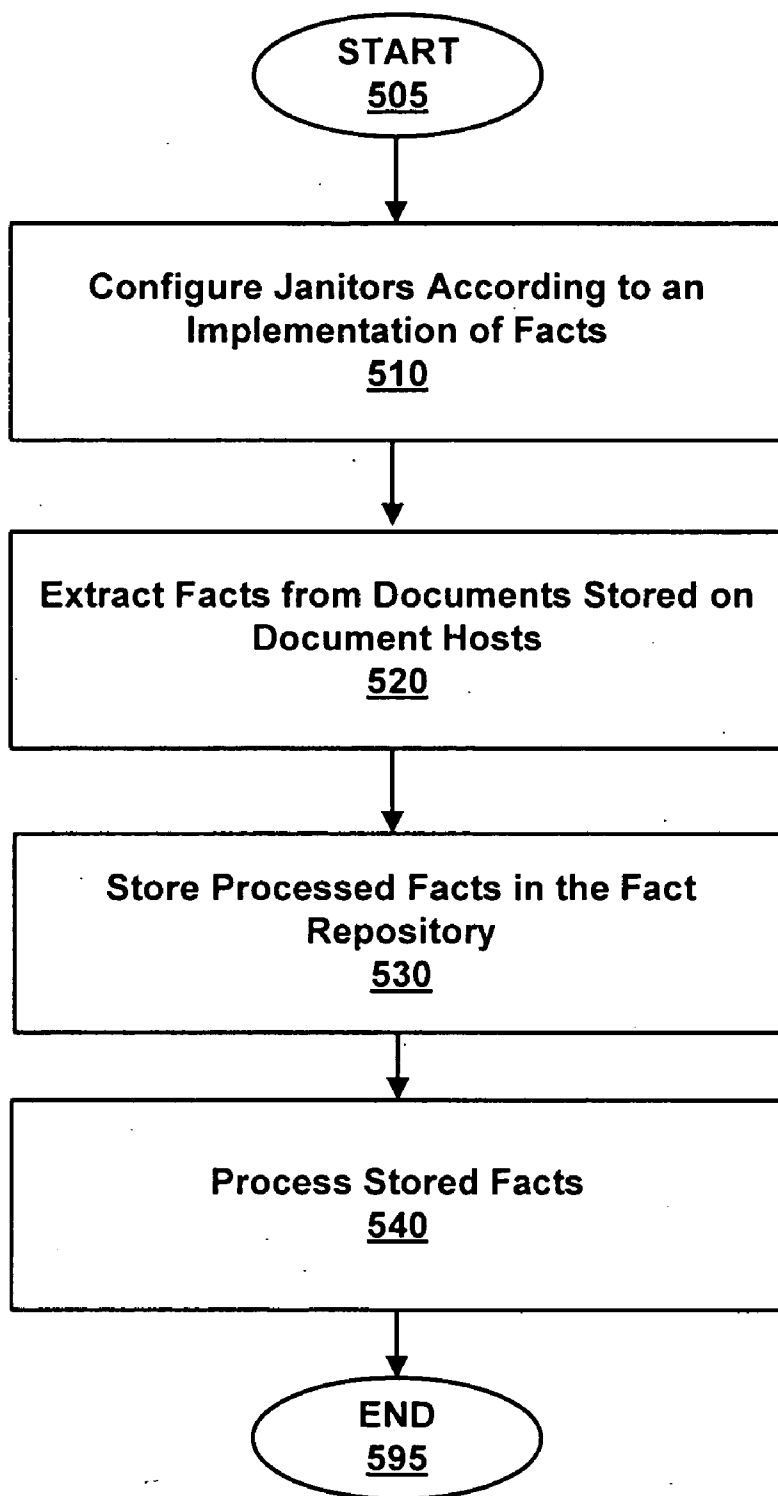


FIG. 5

600

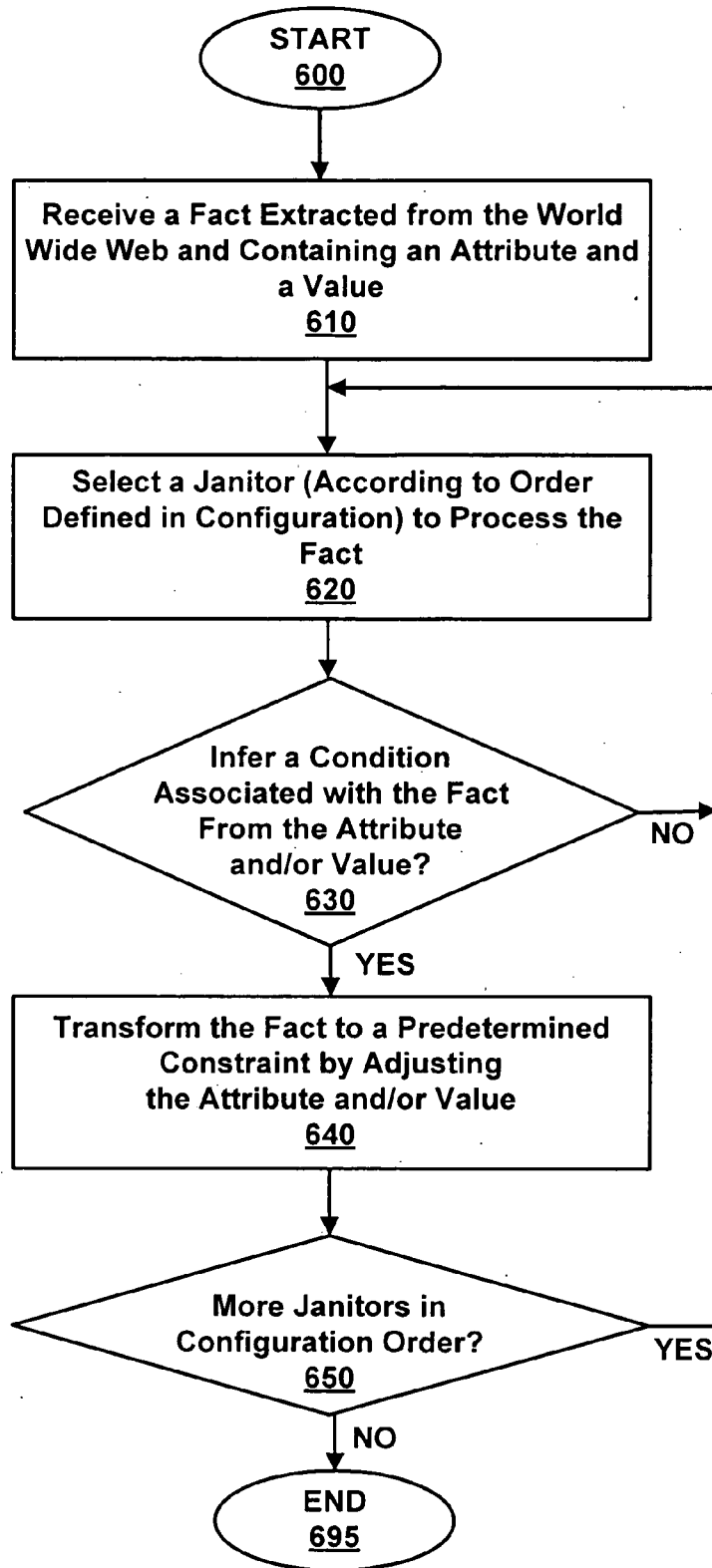


FIG. 6

MECHANISM FOR MANAGING FACTS IN A FACT REPOSITORY

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation-in-part the following applications, all of which are incorporated by reference herein:

[0002] U.S. application Ser. No. 11/024,784, entitled "Supplementing Search Results with Information of Interest", filed on Dec. 30, 2004, by Jonathan T. Betz;

[0003] U.S. application Ser. No. 11/142,853, entitled "Learning Facts from Semi-Structured Text", filed on May 31, 2005, by Shubin Zhao, Jonathan T. Betz;

[0004] U.S. application Ser. No. 11/341,069, entitled "Object Categorization for Information Extraction", filed on Jan. 27, 2006, by Jonathan T. Betz;

[0005] U.S. application Ser. No. 11/356,838, entitled "Modular Architecture for Entity Normalization", filed Feb. 17, 2006, by Jonathan T. Betz, Farhan Shamsi; and

[0006] 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;

[0007] This application is related to the following applications, all of which are incorporated by reference herein:

[0008] U.S. application Ser. No. 11/366,162, entitled "Generating Structured Information," filed Mar. 1, 2006, by Egon Pasztor and Daniel Egnor;

[0009] U.S. application Ser. No. 11/357,748, entitled "Support for Object Search", filed Feb. 17, 2006, by Alex Kehlenbeck, Andrew W. Hogue;

[0010] 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;

[0011] 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;

[0012] U.S. application Ser. No. 11/356,679, entitled "Query Language", filed Feb. 17, 2006, by Andrew W. Hogue, Doug Rohde;

[0013] 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;

[0014] U.S. application Ser. No. 11/356,851, entitled "Browseable Fact Repository", filed Feb. 17, 2006, by Andrew W. Hogue, Jonathan T. Betz;

[0015] U.S. application Ser. No. 11/356,842, entitled "ID Persistence Through Normalization", filed Feb. 17, 2006, by Jonathan T. Betz, Andrew W. Hogue;

[0016] U.S. application Ser. No. 11/356,728, entitled "Annotation Framework", filed Feb. 17, 2006, by Tom Ritchford, Jonathan T. Betz;

[0017] 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;

[0018] 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;

[0019] U.S. application Ser. No. _____, entitled "Entity Normalization Via Name Normalization", filed on Mar. 31, 2006, by Jonathan T. Betz, Attorney Docket No. 24207-11047;

[0020] U.S. application Ser. No. _____, entitled "Determining Document Subject by Using Title and Anchor Text of Related Documents", filed on Mar. 31, 2006, by Shubin Zhao, Attorney Docket No. 24207-11049;

[0021] U.S. application Ser. No. _____, entitled "Unsupervised Extraction of Facts", filed on Mar. 31, 2006, by Jonathan T. Betz and Shubin Zhao, Attorney Docket No. 24207-11056;

[0022] U.S. application Ser. No. _____, entitled "Anchor Text Summarization for Corroboration", filed on Mar. 31, 2006, by Jonathan T. Betz and Shubin Zhao, Attorney Docket No. 24207-11046; and

BACKGROUND OF THE INVENTION

[0023] 1. Field of the Invention

[0024] The present invention relates generally to database management, and more particularly, to managing data extracted from the World Wide Web.

[0025] 2. Background of the Invention

[0026] Data sources often present information in a manner that is not easily accessible by a user. For example, when the user queries web pages through a search engine, the user is burdened with reviewing individual search results for pertinent information. In other words, the information must be manually synthesized across several web pages.

[0027] Data stored on the web and similar hyperlinked networks has no set format and has no set content. Thus, data from the web or similar networks is often referred to as unstructured data because it is not received in a specific format and the documents contents are not necessarily identified as structured fields. Extraction and processing of data from unstructured sources, such as the World Wide Web presents unique challenges. Extraction of data from the Web is especially challenging due to the wide variety of topics covered and the almost infinite number of authors that are providing that information. In addition, not all information on the World Wide Web is factually accurate. In fact, just the opposite is true. It must be assumed that at least some of the data obtained from the Web is not true, is incomplete, or is outdated.

[0028] Conventional techniques for harvesting data from sources such as web pages also are limited by the variety of styles used to present information. The design of web pages using Hyper Text Markup Language, or HTML, is a creative process. Information can be presented in text paragraphs, tables, or across separate web pages of a domain. Furthermore, information such as a date can be presented in different formats such as “Dec. 2, 1981”, “Dec. 2, 1981”, and “12 Dec. 1981.” Moreover, similar information harvested from different sources can cause data duplication.

[0029] For these reasons, what is needed is a method and system for processing facts extracted from web-based documents to transform to predetermined constraints.

SUMMARY

[0030] The present invention provides methods and systems for using a janitor to process facts extracted from the World Wide Web. In one embodiment, janitors are software programs that transform facts into more useful data and/or provide functions to clean up and corroborate facts. Janitors can also process facts to detect and process duplicates. Janitors can transform facts responsive to inferring a certain condition associated with facts. Generally, a fact is information, data, or a series of data that can be represented as an attribute and a value. Facts can be in the form of text, graphics, or multimedia content. For example, a web page can list a series of presidents in a first column of a table and list their dates of births in another column. In one embodiment, facts are extracted from documents on the World Wide Web for storage in a fact repository. One or more janitors transform facts in accordance with constraints designed to improve the quality of facts. In one embodiment, facts can be processed as they are extracted from documents. In another embodiment, facts can be retrieved from the fact repository and processed after storage.

[0031] The condition can be related to one or more of an attribute, a value, or an object of a fact being analyzed. For example, janitors can perform normalization, remove or merge similar or duplicate facts, segregate multiple values of a fact, synthesize new facts from old, and the like. In one embodiment, an administrator can select which janitors are applied to facts. The administrator can choose to apply several janitors.

[0032] Advantageously, janitors improve the quality of facts extracted from the World Wide Web and stored in a fact repository. The improved facts are more useful and reliable to users.

[0033] The features and advantages described herein are not all inclusive, and, in particular, many additional features and advantages will be apparent to one skilled in the art in view of the drawings, specifications, and claims. Moreover, 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 circumscribe the claimed invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0034] The teachings of the present invention can be readily understood by considering the following detailed description in conjunction with the accompanying drawings. Like reference numerals are used for like elements in the accompanying drawings.

[0035] FIG. 1 is a block diagram of a system for gathering facts according to one embodiment of the present invention.

[0036] FIGS. 2(a)-(e) illustrate example data structures for facts within a fact repository.

[0037] FIGS. 3(a)-(b) illustrate exemplary data paths for fact processing according to one embodiment of the present invention.

[0038] FIG. 4 is a flow chart illustrating a method for processing facts according to one embodiment of the present invention.

[0039] FIG. 5 is a flow chart illustrating a method for processing facts according to another embodiment of the present invention.

[0040] FIG. 6 is a flow chart illustrating a method for transforming facts based on a condition according to one embodiment of the present invention.

[0041] The figures depict 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 EMBODIMENTS

[0042] Methods and systems for processing facts with janitors are described. Facts are extracted from documents on the Internet or other sources. Generally, facts are information, data, or a series of data that can be represented in a logical form of an attribute and a value. Facts can be in the form of text, graphics, or multimedia content. For example, a web page can list a series of presidents in a first column of a table and list their dates of births in another column. Janitors are used to transform facts into more useful data (e.g., to clean-up facts).

Exemplary Systems

[0043] FIG. 1 is a block diagram illustrating a system 100 for managing facts according to one embodiment of the present invention. System 100 comprises document hosts 102, object requestor 152, and data processing system 106. The components are communicatively coupled through a network 104 (e.g., a data network such as the Internet, a telephone network, etc.). At a high level, system 100 can gather and organize facts, and then retrieve facts in accordance with queries. For example, facts can be gathered from a set of web pages related to baseball players, and then presented in response to a query term such as “baseball”, “sports”, etc.

[0044] Document host 102 comprises one more hosts that store and provide access to documents. Document host 102 can be implemented in a computing device (e.g., personal computer, a workstation, mini-computer, or mainframe, or a PDA) including a processor and operating system. Document host 102 can communicate over network 104 via networking protocols (e.g., TCP/IP), and be configured to use application and presentation protocols (e.g., HTTP, HTML, SOAP, D-HTML, Java). A document comprises facts represented by any data that are discernable by a machine including any combination of text, graphics, multimedia content, etc. A document (e.g., an e-mail, a web

page, a file, news group posting, a blog, or a web advertisement) may be encoded in various formats such as a markup language (e.g., HTML), an interpreted language (e.g., JavaScript), an application-specific format (e.g., DOC format for Microsoft Word, or PDF format for Adobe Reader), or any other computer readable or executable format. A document can include references to other documents or other embedded information (e.g., hyperlinks). A document stored in a document host **102** may be accessed by a Uniform Resource Locator (URL), or Web address, or any other appropriate form of identification and/or location. The documents stored by document host **102** are typically held in a file directory, a database, or other data repository. 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.

[0045] Data processing system **106** includes one or more importers **108**, one or more janitors **110** with a controller **111**, a build engine **112**, a service engine **114**, and a fact repository **115**. Each of the components can be implemented as software modules (or programs) executed by a processor **116**.

[0046] Importers **108** can include one or more modules for different types of documents (e.g., an HTML importer, a PDF importer, etc.). Importers **108** processes documents received from document hosts **102** by parsing the data content of documents to identify facts, and extracting the identified facts from the documents. Importers **108** also determine the subject or subjects with which the facts are associated, and stores the facts in fact repository **115** as individual objects of data.

[0047] Janitors **110** can be self-contained software modules, or a software architecture with a functionality module that can be customized for a particular function. Janitors **110** manage facts by processing various combinations of objects, attributes, or values, according to janitor rules. Janitors **110** can include one or more modules that each perform a different data management function. An administrator can configure controller **111** (or a script) to call janitors **110** based on a specific ordering. For example, if only dates are extracted from documents, janitors **110** that specifically operate on dates can be used for processing date facts.

[0048] In one embodiment, janitors **110** infer a condition of a fact and, in response, transform an attribute and/or value of the fact in accordance with a predetermined constraint. Each janitor **110** can be configured to infer a certain condition of the fact. The fact is transformed to meet predetermined constraints. Generally, janitors **110** can perform functions such as data cleansing, object merging, fact merging, fact induction, and the like, as described in more detail below. For example, data cleansing can remove useless facts that have a low frequency of use. Object merging can combine duplicate objects that appear to represent the same entity. Fact merging can combine duplicate facts that have different formats. Fact induction can imply new facts from existing facts, such as implying that a capitalized name appearing before a comma and a state name is a city name. Some janitors **110** describe desired characteristics of a fact, such as a format or categorization of the attribute and/or value. One janitor **110** can normalize attribute names and values, and delete duplicate and near-duplicate facts so that

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 can be rewritten as Birthdate by one janitor **110** and then another janitor **110** can recognize that Dec. 2, 1981 and Dec. 2, 1981 are different forms of the same date. Janitor **110** transforms the dates to a preferred form. There are numerous rules that can be implemented, a particular set of which depends on a particular implementation. Specific rules are described in more detail below. Various embodiments of janitors **110** and methods operating therein are described in more detail below.

[0049] Referring again to system **100**, build engine **112** builds and manages repository **115**. Service engine **114** is an interface for querying repository **115**. Service engine **114** processes queries, scores matching objects, and returns them to the caller. Service engine **114** is also used by janitors **110**.

[0050] Fact repository **115** comprises a storage element such as a RAM or ROM device in combination with software such as a file system or a database manager. Fact repository **115** stores the facts extracted from the documents. The facts can be stored as a list, a file system, or database data. Exemplary data structures for storing facts in fact repository **115** are described in more detail below with respect to FIGS. 2(a)-(e).

[0051] Object requesters **152**, **154** are entities that request objects from fact 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 fact 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, janitors **120** or other entities considered to be part of data processing system **106** can function as object requester **154**, requesting the facts of objects from fact repository **115**.

[0052] 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 **126**. Memory **107** also includes fact repository **115**. Fact 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 **128** 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 Structures

[0053] FIGS. 2(a)-(e) show example data structures for the facts as stored. As shown in FIG. 2(a), 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.

[**0054**] 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), has 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.

[**0055**] FIG. **2(b)** shows an example of facts having respective fact IDs of **10**, **20**, and **30** in repository **215**. 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” could have a value including several paragraphs of text, numbers, or 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.

[**0056**] 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.

[**0057**] 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.

[**0058**] FIG. **2(d)** shows an example of a data structure for facts within repository **215**, 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 **215**, or a reference to the location (e.g., table row) for the object in the object reference table **210**. The object reference link **416** 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 **215**. 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 **416** that contains the object ID for the “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.

[**0059**] Each fact **204** also may include one or more metrics **218**. A metric provides an indication of 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.

[**0060**] 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.

[**0061**] The facts illustrated in FIG. **2(d)** include an agent field **222** that identifies which importer **108** extracted the fact. For example, importers **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 **208** that extracts facts from free text in documents throughout the Web, and so forth.

[**0062**] 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.

[0063] 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 **404**, also includes the same parameters (such as attribute, value, fact ID, etc.) as other facts **404**. The attribute field **426** of a property fact **408** 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)-(d) and described above are merely exemplary. The data structure of the repository **215** 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**.

[0064] 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 **206** 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 **215**, but have their object ID values cleared (or have their importance set to a negative value). However, the facts of the null object are treated as if they were removed from the repository **215**. In some other embodiments, facts of null objects are physically removed from repository **215**.

[0065] 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 **207**.

[0066] It should be appreciated that the components of document host and data processing system **406** can be distributed over multiple computers. For example, repository **215** can have components deployed over multiple servers. For convenience, however, the components of data processing system **206** are discussed as though they were implemented on a single computer.

Exemplary Data Paths

[0067] FIGS. 3(a)-(b) show alternative data paths for fact processing. As shown in FIG. 3(a), janitors **10** process facts as they are extracted from documents on document hosts **102** on the World Wide Web. Data stored on the web and similar hyperlinked networks has no set format and has no set content. Thus, data from the web or similar networks is often referred to as unstructured data. The processed facts are then stored in fact repository **115**. This embodiment may be ideal for janitors **110** that operate on one fact at a time to perform functions such as normalization. Some janitors such as those in FIG. 3(a) may also access facts already stored in the repository to process a newly extracted fact. For example, a janitor may compare the value of a new fact to values of facts that have been previously extracted, stored in the repository (and possibly indexed).

[0068] In FIG. 3(b), janitors **110** access facts from fact repository **115**, after the facts are extracted from document hosts **102**. Janitors **110** can thus be configured to process multiple facts previously stored in the repository **115**. This embodiment may be ideal for janitors **110** that operate on several facts at a time to perform functions such as merging, although it can also be used to "clean up," formats, spelling, etc of facts already in the repository.

[0069] It will be understood that some systems contain a combination of the types of janitors shown in FIGS. 3(a) and 3(b), so that janitors process facts when the facts are initially placed in the repository and also post-processes facts after the facts have initially been placed in the repository.

[0070] FIG. 4 is a flow chart illustrating a method **400** of processing facts during extraction, according to one embodiment of the present invention. Optionally, an administrator (which can be a human being or automated software) configures **410** janitors **110** according to an implementation of facts using a script or other controller. In one embodiment, a specific algorithm, or ordering, of janitors **110** is configured for a specific implementation. For example, janitors **110** performing normalization or other types of clean-up operations can be applied to facts early in an order. Subsequently, janitors **110** making inductions based on two or more facts can be applied. If janitors **110** making inductions were run first, they would not be as efficient, because similar facts may not be compatible until transformed to a common format. In another embodiment, a predetermined, ordered set of janitors **110** can be provided by controller **111**.

[0071] Importers **108** extract **420** facts from documents stored on document hosts **102**. Generally, the extraction

process analyzes documents for indicators of facts such as attribute value pairs. For example, a table is encoded using specific tags in HTML (e.g., <td>). Importers **108** can identify the table and determine whether column headers or row headers are appropriate attributes, and further, whether corresponding cells are appropriate values. Importers **108** can also be directed to documents known to contain facts under a known template. Fact repository **115** stores **420** facts.

[0072] Individual janitors **110** process **430** facts as described below with respect to FIG. **6**. More than one janitor **110** can process a fact. Additionally, one janitor **110** can process an object, associated with multiple facts or can compare the facts associated with multiple objects (e.g., during object merging and/or duplicate detection). After processing, the fact is stored **440** in fact repository **115**.

[0073] FIG. **5** is a flow chart illustrating a method **500** of processing facts already stored on fact repository **115**. An administrator configures **510** janitors as described. However, there may be differences in the number and/or type of janitors **110** applied to facts since facts are processed from the fact repository **115** rather than during extraction as above. Importers **108** extract **520** facts from documents stored on document hosts **102** and store **530** the facts in fact repository **115**. Janitors process **540** facts after extraction. However, after being processed by janitors **110**, facts can be stored under a different organization. For example, similar facts can be grouped under a single object.

[0074] FIG. **6** is a flow chart illustrating a method **600** of transforming facts with an inferred condition according to one embodiment of the present invention. A script or other controller receives **610** a fact (or more than one fact) containing an attribute and a value. The script or controller selects **620** a janitor to process the fact. As described, janitors **110** can be applied sequentially or according to a specific algorithm. In addition, a subset of all available janitors **110** can be used for processing certain facts.

[0075] A janitor **110** infers **630** a condition associated with the fact from the attribute and/or value. In one embodiment, inferences can be made from multiple facts associated with an object. Conditions of attributes and/or values can be birthdates, numerical values, names, cities, etc. A janitor **110** detecting fact for a Date of Birth and a fact for a Social Security Number, may infer that the facts concern a person. Because the fact concerns a person, the janitor **110** can apply specific constraints associated with persons such as the format of a person's name, or associate the fact with other person facts. In some embodiments, the janitor can also add a new fact explicitly indicating that the associated object represents a "person." Subsequently, additional janitors **110** configured to operate on persons can examine the fact to make additional inferences and adjustments. Thus, a janitor **110** may not perform any operation on the fact if the appropriate condition cannot be inferred. Facts typically require inferences since they are not specially formatted for fact repository **115** as is data that is generated for a particular database.

[0076] The janitor **110** transforms **640** the fact to a pre-determined constraint by adjusting the attribute and/or value. For example, the name of an attribute or format of a value can be changed as discussed above. If the fact has needs to be processed by more janitors in the configured order, the process repeats at the step selecting **620** a janitor.

[0077] The above paragraphs provide some general discussion and examples of janitors. The paragraphs that follow provide some specific examples of janitors. Different embodiments of the present invention may include some, all, or none of these example janitors. For the purpose of clarity, only a few types of janitors **110** have been described below. However, one of ordinary skill in the art will recognize that other types of janitors **110** are possible in addition to those described below.

[0078] In some embodiments, some janitors **110** reduce information in fact repository **115**. A singleton-attribute janitor **110** identifies attributes which should be unique per object, and eliminates all but one instance of that attribute on any given object. For example, a person should only have one date of birth. A blacklist janitor **110** reads in a list of patterns, and deletes any fact that matches a pattern. For example, blacklist janitor **110** can be used to remove curse words. A string-cleanup janitor **110** trims unuseful characters, such as @, #, %, or !, from the beginning or end of attributes. A name-group-threshold-match janitor **110** merges duplicate objects if they share a certain number of attributes, based on their entropy. An entropy is calculated for each value as described in further detail in U.S. application Ser. No. 11/356,765. Objects having similar facts can be merged if associated entropy values fall within an entropy threshold. The name-group-threshold-match janitor **110** is described in further detail in U.S. application Ser. No. 11/356,765. A near-duplicate-fact merger janitor **110** identifies duplicate facts within an object.

[0079] Thus, some janitors compare a first fact to a plurality of existing facts. The existing facts can be obtained from the repository or from any other appropriate source. In some janitors, a fact is compared to existing facts to determine whether the new fact should be stored in the fact repository. In one janitor, if the fact duplicates a threshold number of existing facts, the fact is not stored in the fact repository. In another janitor, if the fact is corroborated by a threshold number of existing facts, the fact is stored in the fact repository. In another janitor, if the fact is not corroborated by a threshold number of existing facts, the fact is not stored in the fact repository. Because the facts extracted from the world-wide web are from unstructured data, the facts can have many formats when they are initially extracted and some of the facts that are compared by the janitors may not have the same format. For example, dates can be in MMDDYY format, DDMMYY format, in formats where months are spelled out ("December"), and so on. Some janitors know about various formats, such as various date formats, and take those formats into account when comparing facts to facts in the repository. In some embodiments, the facts are normalized before they are stored in the repository. In some embodiments, a janitor may require that another janitor runs first in order to normalize formatting of the facts to be compared. Any of these situations allows a comparing janitor to compare facts that had different formats when they were extracted.

[0080] One set of janitors **110** is applied to delete certain facts. A persisted-id-fact-deleter janitor **110** deletes any fact from a previous repository that should no longer be kept as described in further detail in U.S. application Ser. No. 11/356,842. A stuttering-fact-deleter janitor **110** removes any fact whose attribute and value are the same. A reference-redirect-collapse janitor **110** collapses value links that point

to objects that have been merged. An invalid-fact-deleter janitor **110** removes any fact that fail some basic validity checks (e.g., the value is empty). A suspicious-fact-deleter janitor **110** removes facts with lengthy attributes (e.g., 3 words) and repeat information that appears elsewhere in the object. These facts can result from extraction problems. An invalid-language-deleter janitor **110** removes any fact in certain languages. This janitor **110** can be used to segregate facts by language. A legal-constraint janitor **110** enforces constraints on objects for legal purposes. For example, certain document can be limited as to how many facts should be extracted. An unlicensed-fact-finder janitor **110** removes any facts marked as being 'internal only' for legal or other reasons. A small-object-deleter janitor **110** removes any object with too few facts. A dangling-reference-deletion janitor **110** removes any fact with a value link that points at a non-existent object. An object can be missing when removed by another janitor **110**. A name-references-resolver janitor **110** identifies references to other objects in facts and creates search links to the other objects.

[0081] One set of janitors **110** can characterize preferred formats such as canonical forms. A place-cannonicalizer janitor **110** rewrites place names into canonical form. For example, the value "Trenton, N.J." can be rewritten to "Trenton, N.J." A date-cannonicalizer janitor **110** rewrites dates into a canonical form. For example, the date "2006-02-16" is rewritten to "16 Feb. 2006." A measurement-cleanup janitor **110** rewrites measurements to a canonical form. For example, the measurements "5'4"" or "5 ft. 4 in." can be rewritten to "5' 4". An attribute-cannonicalizer janitor **110** rewrites attributes. For example, "birthday", "birthdate", and "birth date" can be rewritten to "date of birth." An article-value-normalizer janitor **110** rewrites values with articles to a readable format. For example, the value "Foo, The" can be rewritten to "The Foo."

[0082] Other janitors **110** can be implemented as well. A type-identifier janitor **110** assigns type values to objects based on a subset of janitors **110**. For example, every fact with a "date of birth" attribute is assigned a type value of "person." A born-died cleanup janitor **110** splits facts associated with birth and death dates into several facts. For example, the fact "Born: 14 Jul. 1960 in Scranton, Pa." can be split into a fact for date of birth and another fact for place of birth. A near-duplicate-fact-merger janitor **110** combines duplicate facts. A value-dereferencer janitor **110** identifies a fact having a value which is a link to another object, and updates a display value of the fact to be the name of the object.

[0083] The order in which the steps of the methods of the present invention are performed is purely illustrative in nature. The steps can be performed in any order or in parallel, unless otherwise indicated by the present disclosure. The methods of the present invention may be performed in hardware, firmware, software, or any combination thereof operating on a single computer or multiple computers of any type. Software embodying the present invention may comprise computer instructions in any form (e.g., source code, object code, interpreted code, etc.) stored in any computer-readable storage medium (e.g., a ROM, a RAM, a magnetic media, a compact disc, a DVD, etc.). Such software may also be in the form of an electrical data signal

embodied in a carrier wave propagating on a conductive medium or in the form of light pulses that propagate through an optical fiber.

[0084] While particular embodiments of the present invention have been shown and described, it will be apparent to those skilled in the art that changes and modifications may be made without departing from this invention in its broader aspect and, therefore, the appended claims are to encompass within their scope all such changes and modifications, as fall within the true spirit of this invention.

[0085] In the above description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the invention. It will be apparent, however, to one skilled in the art that the invention can be practiced without these specific details. In other instances, structures and devices are shown in block diagram form in order to avoid obscuring the invention.

[0086] Reference in the specification to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the invention. The appearances of the phrase "in one embodiment" in various places in the specification are not necessarily all referring to the same embodiment.

[0087] Some portions of the detailed description are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

[0088] It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0089] The present invention also relates to an apparatus for performing the operations herein. This apparatus can be specially constructed for the required purposes, or it can comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program can be stored in a computer

readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

[0090] The algorithms and modules presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems can be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatuses to perform the method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages can be used to implement the teachings of the invention as described herein. Furthermore, as will be apparent to one of ordinary skill in the relevant art, the modules, features, attributes, methodologies, and other aspects of the invention can be implemented as software, hardware, firmware or any combination of the three. Of course, wherever a component of the present invention is implemented as software, the component can be implemented as a standalone program, as part of a larger program, as a plurality of separate programs, as a statically or dynamically linked library, as a kernel loadable module, as a device driver, and/or in every and any other way known now or in the future to those of skill in the art of computer programming. Additionally, the present invention is in no way limited to implementation in any specific operating system or environment.

[0091] It will be understood by those skilled in the relevant art that the above-described implementations are merely exemplary, and many changes can be made without departing from the true spirit and scope of the present invention. Therefore, it is intended by the appended claims to cover all such changes and modifications that come within the true spirit and scope of this invention.

What is claimed is:

1. A computer-implemented method for processing facts extracted from web-based-documents, comprising:

extracting a fact from a plurality of web-based documents stored on document hosts, the fact comprising an attribute and a value, at least two of the web-based documents presenting the fact in different formats;

applying two or more janitors to the fact, each janitor inferring one or more conditions associated with the fact from at least one of the attribute and value, and responsive to inferring the one or more conditions, each of the two or more janitors transforming the fact in accordance with a different predetermined constraint for the condition by adjusting at least one of the attribute and value; and

storing the transformed fact in a fact repository.

2. The method of claim 1, wherein the applying two or more janitors comprises:

applying a first and a second janitor in a predetermined order to infer a first condition and a second condition,

respectively, wherein the second janitor is able to detect the second condition responsive to applying the first janitor.

3. A computer-implemented method for processing facts extracted from web-based-documents, comprising:

extracting a fact from an unstructured web-based document, the fact containing an attribute and a value;

inferring a condition associated with the fact from at least one of the attribute and value;

responsive to inferring the condition, transforming the fact to a predetermined constraint for the condition by adjusting at least one of the attribute and value; and

storing the transformed fact in a fact repository.

4. The method of claim 3, wherein:

inferring the condition comprises inferring a type for the fact, and

transforming the fact comprises transforming the fact to a predetermined constraint for the fact type.

5. The method of claim 3, further comprising:

receiving another fact,

wherein inferring the condition comprises inferring that the facts have the same condition, and

wherein transforming the fact comprises comparing the facts.

6. The method of claim 3, further comprising:

receiving another fact containing another attribute and another value,

wherein inferring the condition comprises detecting that at least one of the attributes or the values are similar,

wherein transforming the fact comprises merging the facts.

7. The method of claim 3, wherein:

transforming the fact comprises reformatting the value according to the predetermined constraint.

8. The method of claim 3, wherein:

inferring the condition comprises inferring that the value includes two or more independent values, and

transforming the fact comprises generating two or more new facts, each one of the two or more new facts having one of the two or more independent values.

9. The method of claim 3, wherein:

inferring the condition comprises inferring the condition based on an object associated with the fact, the object being common to both the fact and at least one other fact.

10. The method of claim 3, further comprising:

retrieving the fact from storage in the fact repository.

11. The method of claim 3, further comprising:

receiving the fact during extraction from the web-based document.

12. The method of claim 3, wherein the web-based document is encoded in Hypertext Markup Language, and the fact is extracted from an HTML table in the document.

13. A computer-implemented method for processing facts extracted from documents, comprising:

- extracting a fact from an unstructured web-based document, the fact containing an attribute and a value;
- applying a collection of janitors in a predetermined order to process the fact, each janitor configured to infer a different condition associated with a fact from at least one of the attribute and value, and responsive to the janitor inferring the specific condition, the janitor transforming the fact to a different predetermined constraint for the condition by adjusting at least one of the attribute and value, and responsive to the janitor failing to infer the specific condition, the janitor discontinuing processing of the fact; and
- storing the fact in a fact repository of the computer.
- 14.** A computer program product stored on a computer readable medium and configured to perform a method for processing facts extracted from unstructured web-based documents and stored in a repository, comprising:
- extracting a fact containing an attribute and a value, the fact extracted from an unstructured web-based document;
- inferring a condition associated with the fact from at least one of the attribute and value;
- responsive to inferring the condition, transforming the fact to a predetermined constraint for the condition by adjusting at least one of the attribute and value; and
- storing the facts in the fact repository.
- 15.** The computer program product of claim 14, wherein:
- inferring the condition comprises inferring a type for the fact, and
- transforming the fact comprises transforming the fact to a predetermined constraint for the fact type.
- 16.** The computer program product of claim 14, further comprising:
- receiving another fact,
- wherein inferring the condition comprises inferring that the facts have the same condition, and
- wherein transforming the fact comprises normalizing the facts.
- 17.** The computer program product of claim 14, further comprising:
- receiving another fact containing another attribute and another value,
- wherein inferring the condition comprises detecting that at least one of the attributes or the values are similar,
- wherein transforming the fact comprises merging the facts.
- 18.** The computer program product of claim 14, wherein:
- transforming the fact comprises reformatting the value according to the predetermined constraint.
- 19.** The computer program product of claim 14, wherein:
- inferring the condition comprises inferring that the value includes two or more independent values, and
- transforming the fact comprises generating two or more new facts, each one of the two or more new facts having one of the two or more independent values.
- 20.** The computer program product of claim 14, wherein:
- inferring the condition comprises inferring the condition based on an object associated with the fact, the object used as a common indexer for both the fact and at least one other fact.
- 21.** The computer program product of claim 14, further comprising:
- retrieving the fact from storage in the fact repository.
- 22.** The computer program product of claim 14, further comprising:
- receiving the fact during extraction from the document.
- 23.** The computer program product of claim 14, wherein the web-based document is encoded in Hypertext Markup Language, and the fact is extracted from a table as indicated within the encoding.
- 24.** A system for processing facts extracted from documents, comprising:
- an extractor to extract facts, the facts each containing an attribute and a value, the facts extracted from a plurality of unstructured web-based documents;
- two or more janitors configured to receive a fact, each of the two or more janitors configured to operate in a predetermined order and to infer a different specific condition associated with the fact from at least one of the attribute and value, the two or more janitors configured to transform the fact to a predetermined constraint for the condition by adjusting at least one of the attribute and value, and store the fact.
- 25.** The system of claim 24, further comprising:
- a script module, in communication with the one or more janitors, the script to describe an order for the one or more janitors to process the fact.
- 26.** A method performed by a software janitor in a data processing system, comprising:
- receiving a first fact containing an attribute and a value, the value having a first format when it was been extracted from an unstructured web-based document;
- receiving a plurality of existing facts, the plurality of facts each containing an attribute and a value, each value having been previously extracted from an unstructured web-based document, and at least one of the values having had a second format different from the first format at the time the existing fact was extracted from the web-based document; and
- comparing the value of the first fact to values of the existing facts to determine whether the first fact should be stored in a fact repository of the data processing system.
- 27.** The method of claim 26, wherein comparing to determine whether the first fact should be stored in the fact repository comprises: not storing the fact in the fact repository if it duplicates a threshold number of the existing facts.
- 28.** The method of claim 26, wherein comparing to determine whether the first fact should be stored in the fact repository comprises storing the fact in the fact repository if it is corroborated by a threshold number of the existing facts.
- 29.** The method of claim 26, wherein comparing to determine whether the first fact should be stored in the fact

repository comprises refraining from storing the fact in the fact repository if it is not corroborated by a threshold number of the existing facts.

30. The method of claim 26, further comprising: normalizing the first fact and the at least one existing fact prior to comparing the first fact and the existing facts.

31. The method of claim 26, wherein comparing the first fact and the existing facts further comprises being able to compare facts in the first and second formats.

32. The method of claim 26, wherein receiving the plurality of existing facts comprises receiving the plurality of existing facts from the fact repository.

* * * * *