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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 January 2012 (05.01.2012)

PCT

(10) International Publication Number
WO 2012/000778 A1

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

(21) **International Application Number:**
PCT/EP201 1/059762

(22) **International Filing Date:**
13 June 201 1 (13.06.201 1)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) **Priority Data:**
2706743 30 June 2010 (30.06.2010) CA

(71) **Applicant** *(for all designated States except US):* **INTER¬NATIONAL BUSINESS MACHINES CORPORA¬TION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(71) **Applicant** *(for MG only):* **IBM UNITED KINGDOM LIMITED** [GB/GB]; PO Box 41, Portsmouth Hampshire P06 3AU (GB).

(72) **Inventors; and**

(75) **Inventors/ Applicants** *(for US only):* **AYOUB, Khalil, Andrew** [CA/CA]; IBM Canada, MD 03/Y2Y/1/KAN, 1, Hines Road, Kanata, Ontario KTK3C7 (CA). **ALY, Hosam** [CA/CA]; IBM Canada, MD 03/Y3D/1/KAN, 1, Hines Road, Kanata, Ontario KTK3C7 (CA). **WALSH, Jason, Matthew** [CA/CA]; IBM Canada, MD 03/Y2Y/1/KAN, 1, Hines Road, Kanata, Ontario KTK3C7 (CA).

(74) **Agent: ROBERTS, Scott;** IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester Hampshire S021 2JN (GB).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available):* AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available):* ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: PAGE UNIQUENESS DETECTION

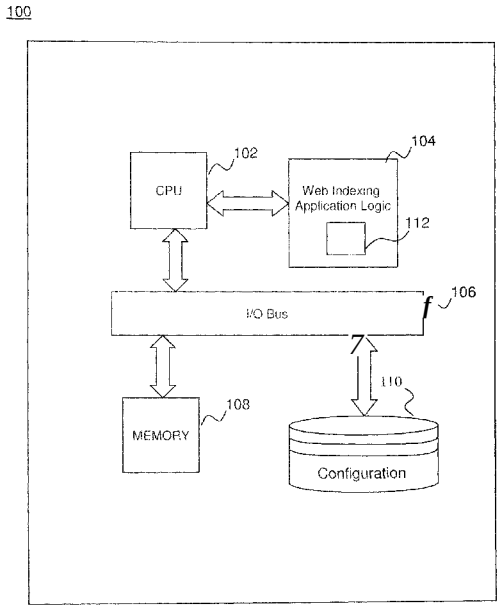

FIG. 1

(57) Abstract: DOM based unique ID generation, including receiving a hypertext markup language (HTML) page at a computer, and identifying HTML page elements in response to the receiving, the HTML page elements comprising parent nodes, the parent nodes comprising child nodes. The method further comprising processing each of the HTML page elements, the processing comprising: grouping the child nodes by parent node into a group of child nodes, detecting patterns in the group of child nodes in response to the grouping, reducing the group of child nodes to text strings in response to the detecting, storing the text strings as text values in the parent nodes, and generating a unique identifier (ID) of the HTML page in response to the processing.

WO 2012/000778 A1

Published:

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

PAGE UNIQUENESS DETECTION

FIELD OF THE INVENTION

This invention relates generally to processing within a computing environment, and more particularly to document object model (DOM) based page uniqueness detection.

BACKGROUND

Web Crawlers, such as those used by page indexing search engines, and security scanning applications often need to determine if a page has already been visited. To do this, those applications attempt to identify a page as unique using information on the page. This information is used to determine if the next page being visited is a new page or a duplicate of one visited previously. Web Crawlers and security scanning applications must use techniques in order to prevent them from entering an infinite loop (i.e. exploring a series of pages over and over again) while ensuring that the relevant pages of the website are indexed. These applications may use key elements of the page in order to determine the uniqueness of the page. They may use, for example, the url of the page, the parameters passed to the page and cookies (i.e. information stored on a browser from a web server), etc. in order to uniquely identify the page. This process will assist in avoiding creating an infinite loop. One problem with this type of implementation is that it often makes it impossible to crawl Web 2.0 applications. Web 2.0 applications make extensive use of JavaScript and XmlHttpRequest which may modify page content without changing the url, parameters, or cookies of the page thereby making identifying a page more difficult.

US patent publication US20090049062 discloses a method of identifying duplicate web pages based on document object model (DOM) tree representation of the web pages. Techniques are described for organizing structurally similar web pages for a website. Fingerprints are made of the structure of the web pages using shingling by placing the web page's HTML tags and attributes in sequence and encoding the tags and attributes using a standard encoding technique. Fixed-size portions of the encoded sequence are taken and a set of values extracted using independent hash functions to compute the shingles.

Alternatively, a DOM tree representation of HTML of the web page is generated and each path of the DOM tree encoded and values extracted using independent hash functions to compute the shingles. A specified number of shingles are retained as the fingerprint. The pages are then clustered based upon the URL and the similarity of the shingles. The clustered hierarchal organization of pages is further pruned by various criteria including similarity of shingles or support of the cluster node in the hierarchy.

US patent publication US7698317 discloses a method of identifying duplicate web pages based on shingles computed for the web pages, wherein the shingles may be computed based on DOM tree corresponding to the web pages. Techniques are disclosed for detecting web pages with duplicate content. In one embodiment, a set of shingles is computed for each page of a group of pages. An aggregate set of shingles is determined based on the sets of shingles computed for the group of pages. A first subset from the aggregate set of shingles is determined by selecting, from the aggregate set, shingles whose frequencies in the aggregate set exceed a specified threshold. A modified set of shingles is generated for each page of the group of pages by removing, from the set of shingles for that page, any shingle included in the first subset. One or more duplicate pages in the group of pages are determined based at least in part on the modified sets of shingles generated for the group of pages.

US patent document US20 100076954 discloses a method of identifying if a newly crawled document is a duplicate of a previously crawled document by comparing content identifiers associated the newly crawled document and the previously crawled document in a web crawler system. Duplicate documents are detected in a web crawler system. Upon receiving a newly crawled document, a set of documents, if any, sharing the same content as the newly crawled document is identified. Information identifying the newly crawled document and the selected set of documents is merged into information identifying a new set of documents. Duplicate documents are included and excluded from the new set of documents based on a query independent metric for each such document. A single representative document for the new set of documents is identified in accordance with a set of predefined conditions.

BRIEF SUMMARY OF THE INVENTION

Embodiments of the invention include methods for page based unique ID generation, the methods comprising receiving a hypertext markup language (HTML) page at a computer, and identifying HTML page elements in response to the receiving, the HTML page elements comprising parent nodes, the parent nodes comprising child nodes. The method further comprising processing each of the HTML page elements, the processing comprising: grouping the child nodes by parent node into a group of child nodes, detecting patterns in the group of child nodes in response to the grouping, reducing the group of child nodes to text strings in response to the detecting, storing the text strings as text values in the parent nodes, and generating a unique identifier (ID) of the HTML page in response to the processing.

Additional embodiments of the invention include systems for page based unique ID generation, the system comprising a host system in communication with at least one client system over a network, a page based unique ID generation application for execution on the host system, the page based for unique ID generation application including logic for implementing a method comprising receiving a hypertext markup language (HTML) page at a computer, and identifying HTML page elements in response to the receiving, the HTML page elements comprising parent nodes, the parent nodes comprising child nodes. The system further comprising processing each of the HTML page elements, the processing comprising: grouping the child nodes by parent node into a group of child nodes; detecting patterns in the group of child nodes in response to the grouping; reducing the group of child nodes to text strings in response to the detecting; and storing the text strings as text values in the parent nodes; and generating a unique identifier (ID) of the HTML page in response to the processing.

Further embodiments of the invention include computer program products comprising a non-transitory storage medium storing instructions, which when executed by a computer implement page based unique ID generation, the computer program product implementing a method, the method comprising receiving a hypertext markup language (HTML) page at a computer, and identifying HTML page elements in response to the receiving, the HTML page elements comprising parent nodes, the parent nodes comprising child nodes. The

method further comprising processing each of the HTML page elements, the processing comprising grouping the child nodes by parent node into a group of child nodes, detecting patterns in the group of child nodes in response to the grouping, reducing the group of child nodes to text strings in response to the detecting, and storing the text strings as text values in the parent nodes. The method further comprising generating a unique identifier (ID) of the HTML page in response to the processing.

Yet other embodiments of the invention include an apparatus for implementing page based unique ID generation, the apparatus comprising web indexing application logic communicatively coupled to a computer processor and configured to receive a hypertext markup language (HTML) page at a computer, identify HTML page elements in response to the receiving, the HTML page elements comprising parent nodes, the parent nodes comprising child nodes, and process each of the HTML page elements. The processing comprising grouping the child nodes by parent node into a group of child nodes, detecting patterns in the group of child nodes in response to the grouping, reducing the group of child nodes to text strings in response to the detecting, and storing the text strings as text values in the parent nodes. The web indexing application logic further configured to generate a unique identifier (ID) of the HTML page in response to the processing.

Additional features and advantages are realized through the techniques of the present invention. Other embodiments and aspects of the invention are described in detail herein and are considered a part of the claimed invention. For a better understanding of the invention with advantages and features, refer to the description and to the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the present invention will now be described, by way of example only, with reference to the following drawings in which:

FIG. 1 depicts a block diagram of a computer system that may be implemented by an exemplary embodiment;

FIG. 2 depicts a block diagram of a data processing system that may be implemented by an exemplary embodiment;

FIG. 3 depicts a block diagram of a client/server network environment that may be implemented by an exemplary embodiment;

FIG. 4 depicts a web page that may be processed by an exemplary embodiment;

FIG. 5 depicts an additional web page that may be processed by an exemplary embodiment;

FIG. 6 depicts an additional version of a web page that may be processed by an exemplary embodiment;

FIG. 7 depicts a detailed block diagram of an exemplary embodiment of the invention; and

FIG. 8 depicts a detailed block diagram of an additional exemplary embodiment of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

An exemplary embodiment of the present invention provides for document object model (DOM) based page uniqueness detection.

Web Crawlers are used to search and index pages on the Internet. Security scanning applications crawl pages as well, however, when they discover a page that they have not visited, they scan the page for security vulnerabilities such as cross site scripting, or SQL injection vulnerabilities. Both applications index pages in order to determine if they have already visited the page. Methods of indexing pages include using common page identifiers such as page uniform resource locators (urls) of the web pages (i.e. the address of the web page), parameters passed to the web page (e.g. form parameters submitted to the server), or cookies (i.e. information passed from the webserver to a browser for storage and retrieval). For static web pages, such as standard Hyper Text Markup Language (HTML) pages, these parameters may sufficiently identify pages. However, with the advent of Web 2.0 applications (i.e. software applications with dynamic functions written in HTML) and the increased uses of javascript, including XmlHttpRequests, the urls, parameter, cookies, or even the content of the page may not be enough to uniquely identify them. Javacript is a programming language used for scripting on the client side browser. Standard HTML may

be generated dynamically by a webserver, however once the browser receives the HTML page generated by the webserver, its content becomes static. Javascipt allows a properly coded HTML page to be modified by a web browser after being sent from the webserver. Javascript includes a series of functions and properties standardized by the World Wide Web Consortium (W3C) and implemented by web browser developers. One such Javascript command is the XmlHttpRequest command. The XmlHttpRequest command allows a web browser to make additional requests of the webserver, in the background, after the web page has been loaded. Using other Javascript commands the web browser can modify the content of the page based on the new content received as a result of the XmlHttpRequest command, user input through a keyboard and/or mouse, timers, or any combination of theses methods. As a result of the possible way that a page can change, the prior methods of searching and indexing pages breaks down.

For example, a page using the same url, parameters, and cookies may contain very different content over time even without user intervention. In addition, two pages with different urls may have identical content. Furthermore, two pages may share the same content but different layouts based on user preferences or user manipulation. Therefore, the typical methods of using urls, parameters, cookies and content can result in duplicate pages being indexed separately, the different pages being indexed as the same, and the same pages with different layouts to be indexed as two different pages.

DOM based page uniqueness (DOMBPU), offers another method of identifying a unique page, which will identify new pages within Web 2.0 and AJAX web sites. DOMBPU process pages as a human would look at it in order to determine that the page is in fact a new page. For example, a human would look at a page and see each of the elements on the page and be able to determine that it is a new page, or that the page is the same page but with different elements in the page. Web crawlers have difficulty making that distinction. DOMBPU looks at the key elements of the page, the HTML, in order to determine what a person sees. People know when they are looking at a list if it is a menu, for example, or a list of items. A person would not think the page has changed simply because the menu has another item in it, or a list has been filtered slightly. DOMBPU will attempt to determine which parts of a page are lists, identify the different sections of a page, and determine a

unique identifier for the page. If another page is found with the same unique identifier, DOMBPU will know that the page is the same merely with some different items in the page.

DOM based page uniqueness detection works by reducing all of the elements on a page to a single functional item. The reduction is repeatable and may be configured to create the same functional item based on a page with varying layouts as will be described in more detail below.

Turning now to FIG. 1, a system 100 for implementing DOM based page uniqueness detection will now be described. In an exemplary embodiment, the system 100 includes a computer processing unit (CPU) 102 executing computer instructions for DOM based page uniqueness detection. The CPU 102 is communicatively coupled to an I/O bus 106 for communicating with I/O systems such as memory 108 and a storage device 110. The memory 108 is communicatively coupled to the I/O bus 106 and may be any memory capable of high-speed storage and retrieval of data as is known in the art. The storage device 110 is communicatively coupled to the I/O bus 106 and may be any storage device capable of storing data such as a magnetic storage disk, non-volatile solid-state drive, or any other device capable of storing data as is known in the art. The CPU 102 is also communicatively coupled to web indexing application logic 104. The web indexing application logic 104 executes a web indexing application 112 for providing web indexing application 112 services as will be described in more detail below.

It will be understood that the DOM based page uniqueness detection system 100 described in FIG. 1 may be implemented in hardware, software, or a combination thereof.

Turning now to FIG. 2, an additional system 200 for implementing DOM based page uniqueness detection will now be described. In an exemplary embodiment, the system 200 includes a host system 202 executing computer instructions for DOM based page uniqueness detection. Host system 202 may operate in any type of environment that is capable of executing a software application. Host system 202 may comprise a high-speed computer-processing device, such as a mainframe computer, to manage the volume of operations governed by an entity for which the DOM based page uniqueness detection is executing. In

an exemplary embodiment, the host system 202 is part of an enterprise (e.g., a commercial business) that implements the DOM based page uniqueness detection.

In an exemplary embodiment, the system 200 depicted in FIG. 2 includes one or more client systems 204 through which users at one or more geographic locations may contact the host system 202. The client systems 204 are coupled to the host system 202 via one or more networks 206. Each client system 204 may be implemented using a general-purpose computer executing a computer program for carrying out the processes described herein. The client systems 204 may be personal computers (e.g., a lap top, a personal digital assistant) or host attached terminals. If the client systems 204 are personal computers, the processing described herein may be shared by a client system 204 and the host system 202 (e.g., by providing an applet to the client system 204). Client systems 204 may be operated by authorized users (e.g., programmers) of the DOM based page uniqueness detection described herein.

The networks 206 may be any type of known network including, but not limited to, a wide area network (WAN), a local area network (LAN), a global network (e.g., Internet), a virtual private network (VPN), and an intranet. The networks 206 may be implemented using a wireless network or any kind of physical network implementation known in the art. A client system 204 may be coupled to the host system 202 through multiple networks (e.g., intranet and Internet) so that not all client systems 204 are coupled to the host system 202 through the same network. One or more of the client systems 204 and the host system 202 may be connected to the networks 206 in a wireless fashion. In one embodiment, the networks 26 include an intranet and one or more client systems 204 execute a user interface application (e.g., a web browser) to contact the host system 202 through the networks 206. In another exemplary embodiment, the client system 204 is connected directly (i.e., not through the networks 206) to the host system 202 and the host system 202 contains memory for storing data in support of DOM based page uniqueness detection. Alternatively, a separate storage device (e.g., storage device 212) may be implemented for this purpose.

The DOM based page uniqueness detection storage device (storage device) 212 includes a data repository with data relating to DOM based page uniqueness detection by the system

200, as well as other data/information desired by the entity representing the host system 202 of FIG. 2. The storage device 212 is logically addressable as a consolidated data source across a distributed environment that includes networks 206. Information stored in the storage device 212 may be retrieved and manipulated via the host system 202 and/or the client systems 204. The data repository includes one or more databases containing, e.g., corresponding configuration parameters, values, methods, and properties, as well as other related information. It will be understood by those of ordinary skill in the art that the data repository may also comprise other structures, such as an XML file on the file system or distributed over a network (e.g., one of networks 206), or from a data stream from another server (not shown) located on a network. In addition, the storage device 212 may alternatively be located on a client system 204.

The host system 202 depicted in the system 200 of FIG. 2 may be implemented using one or more servers operating in response to a computer program stored in a storage medium accessible by the server. The host system 202 may operate as a network server (e.g., a web server) to communicate with the client systems 204. The host system 202 handles sending and receiving information to and from the client systems 204 and can perform associated tasks. The host system 202 may also include a firewall to prevent unauthorized access to the host system 202 and enforce any limitations on authorized access. For instance, an administrator may have access to the entire system and have authority to modify portions of the system. A firewall may be implemented using conventional hardware and/or software as is known in the art.

The host system 202 may also operate as an application server. The host system 202 executes one or more computer programs to provide the DOM based page uniqueness detection. Host system 202 includes the web indexing application 112 for DOM based page uniqueness detection as will be described in more detail below.

As indicated above, processing may be shared by the client systems 204 and the host system 202 by providing an application (e.g., java applet) to the client systems 204. Alternatively, the client system 204 can include a stand-alone software application for performing a portion or all of the processing described herein. As previously described, it is understood that

separate servers may be utilized to implement the network server functions and the application server functions. Alternatively, the network server, the firewall, and the application server may be implemented by a single server executing computer programs to perform the requisite functions.

It will be understood that the DOM based page uniqueness detection system 200 described in FIG. 2 may be implemented in hardware, software, or a combination thereof.

FIG. 3 is a block diagram of an exemplary embodiment of the web indexing application 112 executing on the host system 202 of FIG. 2 or the web indexing application logic 104 of FIG. 1 operating over a network 206. The web indexing application 112 is communicatively coupled to a network such the Internet 304 as described above. The web indexing application 112 sends page requests over the Internet 304 to a webserver 302. The webserver 302 receives the page request and returns a web page 306 to the web indexing application 112 over the Internet 304. The webserver 302 generates the web page 306 from content retrieved from other servers or data stores (not shown) serves the web page 306 from its local data store (not shown) or from a local datastore (not shown). The web page 306 includes page elements 308.

Page elements 308 are fragments of HTML (also referred to herein as nodes) that give structure to the web page 306. Some examples of page elements 308 are HTML tables (i.e. a TABLE node). An HTML table organizes data in rows (designated by a TR node) and columns (designated by a TD node). TR nodes are nested within TABLE nodes and TD nodes are nested within TR nodes. Other examples of nodes are DIV nodes which organize data in blocks, <P> nodes which organize text into paragraphs, the anchor node (A node) which indicates a link, and an image node (IMG node) which indicates an image. In addition to organization page elements 308 also provide mechanisms for interactivity (i.e. a SCRIPT node) and formatting (i.e. a STYLE node). These HTML nodes are nested within each other and share a single root page element called HTML. One such example is listed in Table 1.

```
<html>
  <head>
      <title>News Site</title>
  <head>
  <body>
      <table>
          <tr onmouseover="..." onmouseout="..." onkeyup="" id="...">
              <td>
                  <table class="..." cellpadding="..." cellspacing="...">
                      <tr>
                          <td style="...">
                              <a class="..." href="" style="...">
                                  <span id="menu">Menu Item l</span>
                              </a>
                          </td>
                      </tr>
                  </table>
              </td>
          </tr>
      </table>
  </body>
</html>
```

TABLE 1

The HTML nodes are arranged in a tree structure with the parent node containing child nodes. Turning to Table 1, the HTML node is the root element as stated above. It serves as the parent node for the head and body nodes. In that example, the head and body nodes are child nodes to the HTML node. The head and body nodes may be parent nodes to their child nodes. Turning back to Table 1, the head node is the parent node to the title node, which is a child node of the head node. The title node has no child nodes and is therefore considered a leaf node. In the example in Table 1, both the title and the span nodes are leaf nodes. Each node may contain attributes and text. The span node of Table 1, for example, contains an

attribute id with a value of "menu." The span node also contains the text "Menu Item 1." Table 1 is one example embodiment of an HTML page, HTML pages may contain any number of additional elements and attributes and other markup as is known in the art, the exemplary embodiment is not meant to be limiting.

Web 2.0 web pages 306 may also be modified with subsequent XmlHttpRequests. In an exemplary embodiment the web indexing application 112, detects one or more SCRIPT nodes, or associated javascript files as is known in the art, with one or more XmlHttpRequest commands. The web indexing application 112 may then execute each of the XmlHttpRequest commands and, using other commands from the SCRIPT node, or an associated javascipt file as is known in the art, reformats the web page 306 so that it matches the configuration it would have taken in a web browser. FIG. 4 depicts a web page 400 before the XMLHttpRequests were executed, and FIG. 5 depicts the same web page 500 once all of the XmlHttpRequests have been executed by the web indexing application 112. Although both web pages 400 and 500 share the same url, parameters, and cookies, the page elements 308 are different. FIG. 6 depicts a web page 600 with the same page elements 308 as web page 500, but with the page elements 308 in a different layout. The page elements 308 of web pages 400 and 500 are movable HTML page elements. The page elements 308 can be moved around the web pages 400 and 500 by a user of the web pages 400 and 500 or by configuration settings on, for example, the webserver 302. Although the web pages 400 and 500 have the same page elements 308, they are ordered differently. Based on configuration settings, the web indexing application 112 may determine that these are the same pages and generate the same ID for each of them. Although the network has been described herein as the Internet 304, it will be understood that in additional embodiments of the present invention the web indexing application 112 requests pages from a server over an Intranet (not shown) or other suitable network as is known in the art.

FIG. 7 depicts a process flow that may implement by the web indexing application 112 executing on the host system 202 or web indexing application logic 104 in an exemplary embodiment of the current invention. At block 702 the web indexing application 112 receives an HTML page from the webserver 302 of FIG. 2. The HTML page may be in the form of a static HTML page, or a Web 2.0 HTML page (e.g. FIGs. 4-6) with embedded

Javascript. At block 704 the HTML is parsed (i.e. the HTML page elements 308 are read and interpreted) and all of the page elements 308 are identified and stored in memory while retaining their original layout and order. At block 706 the page elements 308 which are not significant are filtered out. In an exemplary embodiment the filtered page elements 308 are removed from memory. In alternate embodiments the filtered page elements are left in memory but ignored in future processing steps.

The elements to be filtered are configured by configuration settings (also refered to herein as filter criteria) stored in the storage device 110 and/or 212 as filters. In an exemplary embodiment the filters comprise a list of HTML nodes (e.g. img, a) that are to be filtered. In one embodiment, a series of filters may be used to exclude some elements while including others. For example, a filter may be configured to eliminate all <li> nodes while a second filter may be created to include all <li> nodes which contain an attribute "important." By using both exclusive filters and inclusive filters all <li> nodes may be excluded except for <li> nodes with the attribute of "important." The filters may comprise a list of element names, and/or one or more XML Path Language (XPath) instructions. The XPath instructions comprise one or more strings of instructions indicating a particular HTML node by node name, attribute name, attribute value, or HTML element value including display text within the HTML page. The XPath command may be used to explicitly exclude particular segments of the HTML as is known in the art. In addition, the filters may comprise regular expression (regex) instructions as is known in the art. Regex instruction provide the text parsing and filtering capabilities which may be used to filter the HTML page by text and/or wildcard strings as is known in the art. Of course any other method of identifying elements to be excluded or included as is known in the art may be used to configure filters. In an additional embodiment, the filters may comprise a combination of one or more of a list of HTML elements, XPath instructions, and regex instructions in combination.

Returning now to block 708 of FIG. 7, the elements of the HTML page are traversed from parent to child until a leaf node is found and then the leaf nodes of the last parent node of the tree structure are processed. At block 710 if the node is not the last leaf node, the previously traversed leaf nodes are inspected for a pattern at block 716. A pattern is detected if a series of leaf nodes forms a consecutive repeating pattern. For example, given a series of nodes A,

B, C, A, B, C, a pattern of repeating A, B, C nodes is detected. In another example, given a series of leaf nodes A, B, C, D, A, B no pattern would be detected because although nodes A and B repeat in the pattern they are not consecutive. These examples of pattern detection are for illustrative purposes only and are not meant to be limiting. In an exemplary embodiment any pattern detection process may be used to detect patterns in leaf nodes.

Returning to block 716 of FIG. 7, if a pattern is detected the leaf nodes are reduced at block 718. Returning to the previous example, if the leaf nodes A, B, C, A, B, C, were processed at block 718, the leaf nodes would be reduced to eliminate the repeating pattern, the leaf nodes A, B, C, A, B, C would become for example A, B, C. In another example, given the leaf nodes D, E, D, E, G the reduction of block 718 would produce D, E, G by for example reducing the repeating leaf nodes D, E, D, E to leaf nodes D, E. Of course, these examples of pattern detection and reduction are for illustrative purposes only and are not meant to be limiting. In an exemplary embodiment any pattern detection process may be used to detect patterns in leaf nodes. Returning to block 718 of FIG. 7, once the leaf nodes are reduced the next leaf node is processed at block 720 and the steps of blocks 710, 716, 718, and 720 are repeated as described above.

Returning now to block 710, if the last leaf node has been processed for the given parent node, the entire set of leaf nodes is reduced to the parent node at block 712. In an exemplary embodiment the leaf nodes are converted to a string of text and placed as a text value of the parent node. Table 2a-2c shows one example of a parent node with child nodes throughout the reduction process blocks 710, 712, 716, and 718.

…

<Parent>

<A>

<B>

<C>

<A>

<B>

<C>

<E>

<D>
<E>
</Parent>
…

TABLE 2a

Table 2a depicts a parent node with several leaf nodes. One pattern is detected <A><B><C>. These leaf nodes are reduced through several iterations of block 718 eventually creating the parent and leaf nodes of Table 2b.

…
<Parent>
<A>
<B>
<C>
<E>
<D>
<E>
</Parent>
…

TABLE 2b

Table 2b depicts a parent node once the last reduction step for the leaf nodes has been performed. The original set of nine leaf nodes of Table 2a is now reduced to six leaf nodes. These leaf nodes are reduced to the parent node as text at block 712 as depicted in Table 2c.

…
<Parent><A><B><C><E><D><E></Parent>
…

TABLE 2c

Table 2c depicts a single parent node containing the text "<A><B><C><E><D><E>" and no leaf elements. Note that the parent element has now become a leaf node of its parent (not shown) because it no longer contains child nodes. The examples of Tables 2a-2c are examples for illustrative purposes and are not meant to be limiting in any way. It will be understood that any number or pattern of child elements may be reduced without impacting the efficacy of the invention.

Returning now to block 714 of FIG. 7, once all of the leaf nodes of a parent node have been reduced, if there are additional parent nodes, the processing continues on the next parent node in the tree structure at block 708. The processing blocks 708-712 and 716-720 are repeated until all of the leaf elements have been reduced to text patterns in the parent nodes and there is only a single root node (also referred to herein as the last parent node) with one text element and no child nodes. At block 714, if there are no more parent nodes to process, a unique page identifier is generated at block 722.

The unique page identifier may be created by any method of processing the last parent node into an identifier (ID) that can be repeated by subsequent processing of the same parent node such that when the page is processed a second time the same unique ID is produced. In an exemplary embodiment the last parent node is processed using a hashing algorithm as is known in the art to produce a hash string as is known in the art. The hashing algorithm is a set of instructions that create the same compressed string from a longer strong of text, such that a hashing algorithm processing of character string X will always produce hash string Y. In an alternate embodiment, the text value of the last parent node will be used as the unique key. The embodiments listed herein are some examples of a number of possible ways of generating a unique ID from the last parent node and are not meant to be limiting.

It will be understood that the process blocks of FIG. 7 produce a unique ID that may be independent of the visual content of a web page, such as, for example, the text or images on the page. The unique ID may also be independent of the menu items of the page, the advertisements, or any other content specific elements of the page. It will be understood that by modifying the configuration settings stored in the storage device 110 and/or 212 a user of the system 100 and/or 200 may configure which page elements become part of the unique ID

of the page and as a result determine the fidelity of the DOM based page uniqueness detection.

FIG. 8 depicts a process flow that may implement by the web indexing application 112 executing on the host system 202 or web indexing application logic 104 in an additional exemplary embodiment of the current invention. At block 802 the web indexing application 112 receives an HTML page from the webserver 302 of FIG. 2. The HTML page may be in the form of a static HTML page, or a Web 2.0 HTML page (e.g. FIGs. 4-6) with embedded Javascript. At block 804 the HTML is parsed (i.e. the HTML page elements 308 are read and interpreted) and all of the page elements 308 are identified and stored in memory while retaining their original layout and order. At block 806 the page elements 308 which are not significant are filtered out. In an exemplary embodiment the filtered page elements 308 are removed from memory. In alternate embodiments the filtered page elements are left in memory but ignored in future processing steps.

The elements to be filtered are configured by configuration settings stored in the storage device 110 and/or 212 as filters. In an exemplary embodiment the filters comprise a list of HTML nodes (e.g. img, a) that are to be filtered. In one embodiment, a series of filters may be used to exclude some elements while including others. For example, a filter may be configured to eliminate all <li> nodes while a second filter may be created to include all <li> nodes which contain an attribute "important." By using both exclusive filters and inclusive filters all <li> nodes may be excluded except for <li> nodes with the attribute of "important." The filters may comprise a list of element names, and/or one or more XML Path Language (XPath) instructions. The XPath instructions comprise one or more strings of instructions indicating a particular HTML node by node name, attribute name, attribute value, or HTML element value including display text within the HTML page. The XPath command may be used to explicitly exclude particular segments of the HTML as is known in the art. In addition, the filters may comprise regular expression (regex) instructions as is known in the art. Regex instruction provide the text parsing and filtering capabilities which may be used to filter the HTML page by text and/or wildcard strings as is known in the art. Of course any other method of identifying elements to be excluded or included as is known in the art may be used to configure filters. In an additional embodiment, the filters may

comprise a combination of one or more of a list of HTML elements, XPath instructions, and regex instructions in combination.

Returning now to block 808 of FIG. 8, the elements of the HTML page are traversed from parent to child until a leaf node is found and then the leaf nodes of the last parent in the tree structure are processed. At block 816 if the node is not the last leaf node, the previously traversed leaf nodes are inspected for a pattern at block 818. A pattern is detected if a series of leaf nodes forms a consecutive repeating pattern. For example, given a series of nodes A, B, C, A, B, C, a pattern of repeating A, B, C nodes is detected. In another example, given a series of leaf nodes A, B, C, D, A, B no pattern would be detected because although nodes A and B repeat in the pattern they are not consecutive. These examples of pattern detection are for illustrative purposes only and are not meant to be limiting. In an exemplary embodiment any pattern detection process may be used to detect patterns in leaf nodes.

Returning to block 818 of FIG. 8, if a pattern is detected the leaf nodes are reduced at block 820. Returning to the previous example, if the leaf nodes A, B, C, A, B, C, were processed at block 820, the leaf nodes would be reduced to eliminate the repeating pattern, the leaf nodes A, B, C, A, B, C would become for example A, B, C. In another example, given the leaf nodes D, E, D, E, G the reduction of block 820 would produce D, E, G by for example reducing the repeating leaf nodes D, E, D, E to leaf nodes D, E. Of course, these examples of pattern detection and reduction are for illustrative purposes only and are not meant to be limiting. In an exemplary embodiment any pattern detection process may be used to detect patterns in leaf nodes. Returning to block 820 of FIG. 8, once the leaf nodes are reduced the next leaf node is processed at block 822 and the steps of blocks 816, 818, 820, and 822 are repeated as described above.

Returning now to block 816, if the last leaf node has been processed for the given parent node, the entire set of leaf nodes is sorted at block 810. The leaf nodes are sorted in order to neutralize differences in a page based on the same elements being rearranged on the page such as those depicted in FIGs. 5 and 6 as described above. The sorting of leaf nodes is illustrated below in tables 3a-3b.

```
…
<Parent>
  <A>
  <B>
  <C>
</Parent>
…
```

TABLE 3a

Table 3 a depicts a set of nodes representing a series of page elements 308 <A>, <B> and <C>. These elements may be, for example, a series of paragraphs on an html page (eg. FIG. 5). In one example, the elements may be arranged in the page in a different order (e.g. FIG. 6) such as, for example, the order depicted in Table 3b.

```
…
<Parent>
  <C>
  <A>
  <B>
</Parent>
…
```

TABLE 3b

Table 3b depicts the same nodes arranged in a different order. Sorting the leaf nodes of Table 3b results in the order of Table 3c.

```
…
<Parent>
  <A>
  <B>
  <C>
```

</Parent>

…

TABLE 3c

Note that the leaf nodes of Table 3c are the in the same order as Table 3a. By sorting the leaf elements two pages with the same page elements 308 ordered differently may be indexed as the same page.

Returning now to block 812 of FIG. 8, once all of the leaf nodes are sorted the entire set of leaf nodes is reduced to the parent node. In an exemplary embodiment the leaf nodes are converted to a string of text and placed as a text value of the parent node. Table 4a-4c shows one example of a parent node with child nodes throughout the reduction process blocks 810, 816, 818, 820, and 822.

…

<Parent>

<A>

<B>

<C>

<A>

<B>

<C>

<E>

<D>

<E>

</Parent>

…

TABLE 4a

Table 4a depicts a parent node with several leaf nodes. The leaf nodes are reduced at block 820 resulting in the parent and leaf nodes depicted in Table 4b.

…

```
<Parent>
  <A>
  <B>
  <C>
  <E>
  <D>
  <E>
</Parent>
…
```

TABLE 4b

One pattern was detected <A>,<B>,<C>. These leaf nodes are now sorted at block 810 resulting in the parent and leaf nodes of Table 4c.

```
…
<Parent>
  <A>
  <B>
  <C>
  <D>
  <E>
  <E>
</Parent>
…
```

TABLE 4c

Table 4c depicts a parent node once the sorting step for the leaf nodes has been performed. The original set of nine leaf nodes of Table 4a is now reduced to six sorted leaf nodes. These leaf nodes are reduced to the parent node as text at block 812 as depicted in Table 4d.

```
…
<Parent>&lt;A&gt;&lt;B&gt;&lt;C&gt;&lt;D&gt;&lt;E&gt;&lt;E&gt;</Parent>
```

…

TABLE 4d

Table 4c depicts a single parent node containing the text "<A><B><C><D><E><E>" and no leaf elements. Note that the parent element has now become a leaf node of its parent (not shown) because it no longer contains child nodes. The examples of Tables 4a-4d are examples for illustrative purposes and are not meant to be limiting in any way. It will be understood that any number or pattern of child elements may be reduced without impacting the efficacy of the invention.

Returning now to block 814 of FIG. 8, once all of the leaf nodes of a parent node have been reduced, if there are additional parent nodes, the processing continues on the next parent node in the tree structure at block 808. The processing blocks 808-812 and 816-822 are repeated until all of the leaf elements have been reduced to text patterns in the parent nodes and there is only a single root node (also referred to herein as the last parent node) with one text element and no child nodes. At block 814, if there are no more parent nodes to process, a unique page identifier is generated at block 824.

The unique page identifier may be created by any method of processing the last parent node into an ID that can be repeated by subsequent processing of the same parent node such that when the page is processed a second time the same unique ID is produced. In an exemplary embodiment the last parent node is processed using a hashing algorithm as is known in the art to produce a hash string as is known in the art. The hashing algorithm is a set of instructions that create the same compressed string from a longer strong of text, such that a hashing algorithm processing of character string X will always produce hash string Y. In an alternate embodiment, the text value of the last parent node will be used as the unique key. The embodiments listed herein are some examples of a number of possible ways of generating a unique ID from the last parent node and is not meant to be limiting.

It will be understood that the process blocks of FIG. 8 produce a unique ID that may be independent of the visual content of a web page 306, such as, for example, the text or images

on the page. The unique ID may also be independent of the arrangement of the elements on a page, the menu items of the page, the advertisements, or any other content specific elements of the page. It will be understood the by modifying the configuration settings stored in the storage device 110 and/or 212 a user of the system may configure which page elements 308 become part of the unique ID of the page and as a result determine the fidelity of the DOM based page uniqueness detection.

Technical effects and benefits include providing the capability detecting the uniqueness of DOM based web pages 306 even when the page content and order of page elements 308 changes dynamically over time. The uniqueness is determined by reducing all of the elements of a page to a single element in a reproducible way. The reduction may generate the same single element for a page even if the items on the page are in different locations on the page.

The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of the invention. As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. It will be further understood that the terms "comprises" and/or "comprising," when used in this specification, specify the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof.

The corresponding structures, materials, acts, and equivalents of all means or step plus function elements in the claims below are intended to include any structure, material, or act for performing the function in combination with other claimed elements as specifically claimed. The description of the present invention has been presented for purposes of illustration and description, but is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope of the invention. The embodiment was chosen and described in order to best explain the principles of the invention and the practical application, and to enable others of ordinary skill in the art to understand the invention for

various embodiments with various modifications as are suited to the particular use contemplated.

As will be appreciated by one skilled in the art, aspects of the present invention may be embodied as a system, method or computer program product. Accordingly, aspects of the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a "circuit," "module" or "system." Furthermore, aspects of the present invention may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied thereon.
Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the context of this document, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device.

A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof A computer readable signal medium may be any computer readable medium that is not a computer

readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device.

Program code embodied on a computer readable medium may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc., or any suitable combination of the foregoing.

Computer program code for carrying out operations for aspects of the present invention may be written in any combination of one or more programming languages, including an object oriented programming language such as Java™, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the internet 304 using an Internet Service Provider).

These computer program instructions may also be stored in a computer readable medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer readable medium produce an article of manufacture including instructions which implement the function/act specified in the flowchart and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

As described above, embodiments can be embodied in the form of computer-implemented processes and apparatuses for practicing those processes. In exemplary embodiments, the invention is embodied in computer program code executed by one or more network elements. Embodiments include a computer program product encoded on a computer usable medium with computer program code logic containing instructions embodied in tangible media as an article of manufacture. Exemplary articles of manufacture for computer usable medium may include floppy diskettes, CD-ROMs, hard drives, universal serial bus (USB) flash drives, or any other computer-readable storage medium, wherein, when the computer program code logic is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention. Embodiments include computer program code logic, for example, whether stored in a storage medium, loaded into and/or executed by a computer, or transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via electromagnetic radiation, wherein, when the computer program code logic is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention. When implemented on a general-purpose microprocessor, the computer program code logic segments configure the microprocessor to create specific logic circuits.

Aspects of the present invention are described above with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present invention. In this regard, each block in the

flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or fiowchart illustration, and combinations of blocks in the block diagrams and/or fiowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

CLAIMS

1. A computer implemented method for page based unique ID generation, the method comprising:

receiving a hypertext markup language (HTML) page at a computer;

identifying HTML page elements in response to the receiving, the HTML page elements comprising parent nodes, the parent nodes comprising child nodes;

processing each of the HTML page elements, the processing comprising:

grouping the child nodes by parent node into a group of child nodes;

detecting patterns in the group of child nodes in response to the grouping;

reducing the group of child nodes to text strings in response to the detecting; and

storing the text strings as text values in the parent nodes; and

generating a unique identifier (ID) of the HTML page in response to the processing.

2. The method of claim 1, wherein the processing further comprising:

sorting the group of child nodes in response to the reducing.

3. The method of claim 1, wherein the HTML page is a Web 2.0 page, the Web 2.0 page comprising content, the content generated dynamically.

4. The method of claim 2, wherein the HTML page comprises a visible page, the visible page comprising moveable HTML page elements, the moveable HTML page elements configured to occupy any location on the visible page, and the unique ID of the HTML page is the same for all locations of the moveable HTML page elements on the visible page.

5. The method of claim 1, wherein the HTML page comprises a uniform resource locator (URL), and the unique ID of the HTML page is the same for a plurality of HTML pages, the plurality of HTML pages each comprising a different URL.

6. The method of claim 1, further comprising filtering HTML page elements in response to the identifying, the filtering removing the child nodes and the parent nodes that meet filter criteria, the filter criteria comprising on or more of:

extensible markup language path language instructions;

regular expression instructions; and

a list of html nodes.

7. A system for page based unique ID generation, the system comprising:

a host system in communication with at least one client system over a network;

a page based unique ID generation application for execution on the host system, the page based for unique ID generation application including logic for implementing a method comprising:

receiving a hypertext markup language (HTML) page at a computer;

identifying HTML page elements in response to the receiving, the HTML page elements comprising parent nodes, the parent nodes comprising child nodes;

processing each of the HTML page elements, the processing comprising:

grouping the child nodes by parent node into a group of child nodes;

detecting patterns in the group of child nodes in response to the grouping;

reducing the group of child nodes to text strings in response to the detecting; and

storing the text strings as text values in the parent nodes; and

generating a unique identifier (ID) of the HTML page in response to the processing.

8. The system of claim 7, wherein the processing further comprising:

sorting the group of child nodes in response to the reducing.

9. The system of claim 7, wherein the HTML page is a Web 2.0 page, the Web 2.0 page comprising content, the content generated dynamically.

10. The system of claim 8, wherein the HTML page comprises a visible page, the visible page comprising moveable HTML page elements, the moveable HTML page elements configured to occupy any location on the visible page, and the unique ID of the HTML page is the same for all locations of the moveable HTML page elements on the visible page.

11. The system of claim 7, wherein the HTML page comprises a uniform resource locator (URL), and the unique ID of the HTML page is the same for a plurality of HTML pages, the plurality of HTML pages each comprising a different URL.

12. The system of claim 7, further comprising filtering HTML page elements in response to the identifying, the filtering removing the child nodes and the parent nodes that meet filter criteria, the filter criteria comprising on or more of:

extensible markup language path language instructions;

regex instructions; and

a list of html nodes.

13. A computer program product comprising a non-transitory storage medium storing instructions, which when executed by a computer implement page based unique ID generation, the computer program product implementing a method, the method comprising:

receiving a hypertext markup language (HTML) page at a computer;

identifying HTML page elements in response to the receiving, the HTML page elements comprising parent nodes, the parent nodes comprising child nodes;

processing each of the HTML page elements, the processing comprising:

grouping the child nodes by parent node into a group of child nodes;

detecting patterns in the group of child nodes in response to the grouping;

reducing the group of child nodes to text strings in response to the detecting; and

storing the text strings as text values in the parent nodes; and

generating a unique identifier (ID) of the HTML page in response to the processing.

14. The computer program product of claim 13, wherein the processing further comprising:

sorting the group of child nodes in response to the reducing.

15. The computer program product of claim 13, wherein the HTML page is a Web 2.0 page, the Web 2.0 page comprising content, the content generated dynamically.

16. The computer program product of claim 14, wherein the HTML page comprises a visible page, the visible page comprising moveable HTML page elements, the moveable HTML page elements configured to occupy any location on the visible page, and the unique ID of the HTML page is the same for all locations of the moveable HTML page elements on the visible page.

17. The computer program product of claim 13, wherein the HTML page comprises a uniform resource locator (URL), and the unique ID of the HTML page is the same for a plurality of HTML pages, the plurality of HTML pages each comprising a different URL.

18. The computer program product of claim 13, further comprising filtering HTML page elements in response to the identifying, the filtering removing the child nodes and the parent nodes that meet filter criteria, the filter criteria comprising on or more of:

extensible markup language path language instructions;

regex instructions; and

a list of html nodes.

19. An apparatus for implementing page based unique ID generation, the apparatus comprising:

web indexing application logic communicatively coupled to a computer processor and configured to receive a hypertext markup language (HTML) page at a computer, identify HTML page elements in response to the receiving, the HTML page elements comprising parent nodes, the parent nodes comprising child nodes, and process each of the HTML page elements, the processing comprising:

grouping the child nodes by parent node into a group of child nodes;

detecting patterns in the group of child nodes in response to the grouping;

reducing the group of child nodes to text strings in response to the detecting; and

storing the text strings as text values in the parent nodes; and

the web indexing application logic further configured to generate a unique identifier (ID) of the HTML page in response to the processing.

20. The apparatus of claim 19, wherein the processing further comprising:

sorting the group of child nodes in response to the reducing.

21. The apparatus of claim 19, wherein the HTML page is a Web 2.0 page, the Web 2.0 page comprising content, the content generated dynamically.

22. The apparatus of claim 20, wherein the HTML page comprises a visible page, the visible page comprising moveable HTML page elements, the moveable HTML page elements configured to occupy any location on the visible page, and the web indexing application logic configured to generate the same unique ID of the HTML page for all locations of the moveable HTML page elements on the visible page.

23. The apparatus of claim 19, wherein the HTML page comprises a uniform resource locator (URL), and the unique ID of the HTML page is the same for a plurality of HTML pages, the plurality of HTML pages each comprising a different URL.

24. The apparatus of claim 19, wherein the web indexing application logic is further configured to filter HTML page elements in response to the identifying, the filtering removing the child nodes and the parent nodes that meet filter criteria, the filter criteria comprising on or more of:

extensible markup language path language instructions;

regular expression instructions; and

a list of html nodes.

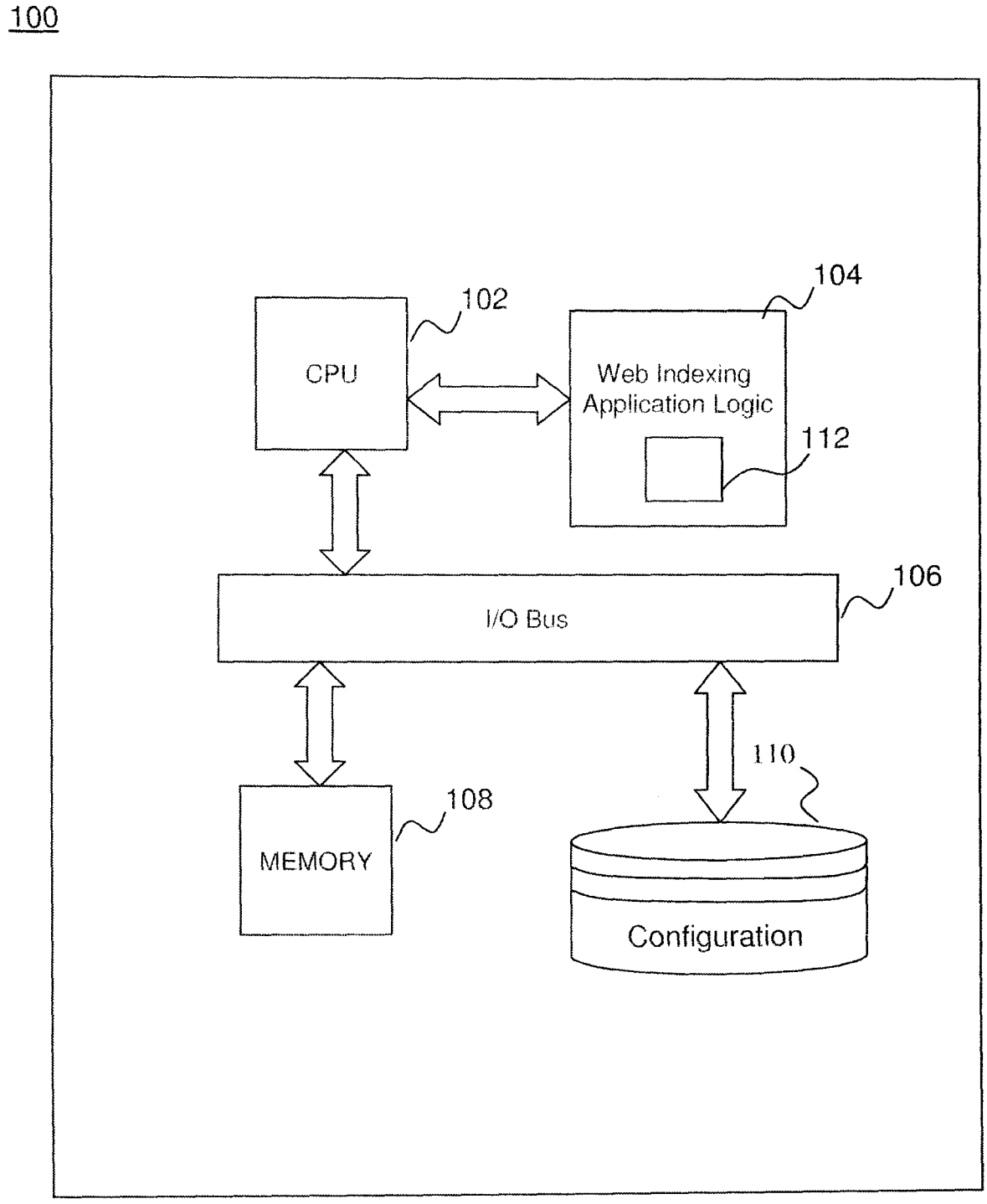
100
102
CPU
104
Web Indexing
Application Logic
112
106
I/O Bus
108
MEMORY
110
Configuration

FIG. 1

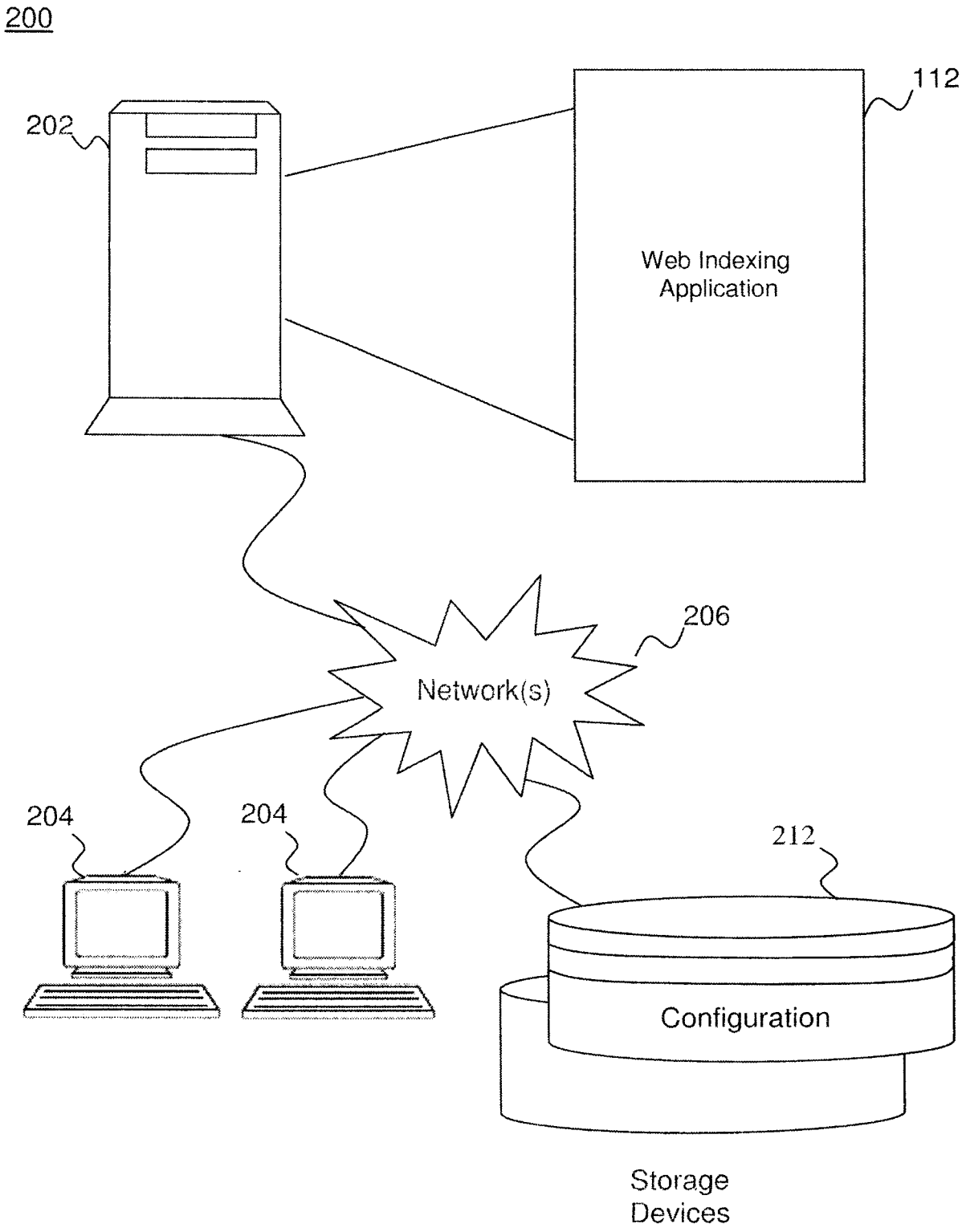
200
202
112
Web Indexing Application
206
Network(s)
204
204
212
Configuration
Storage Devices

FIG. 2

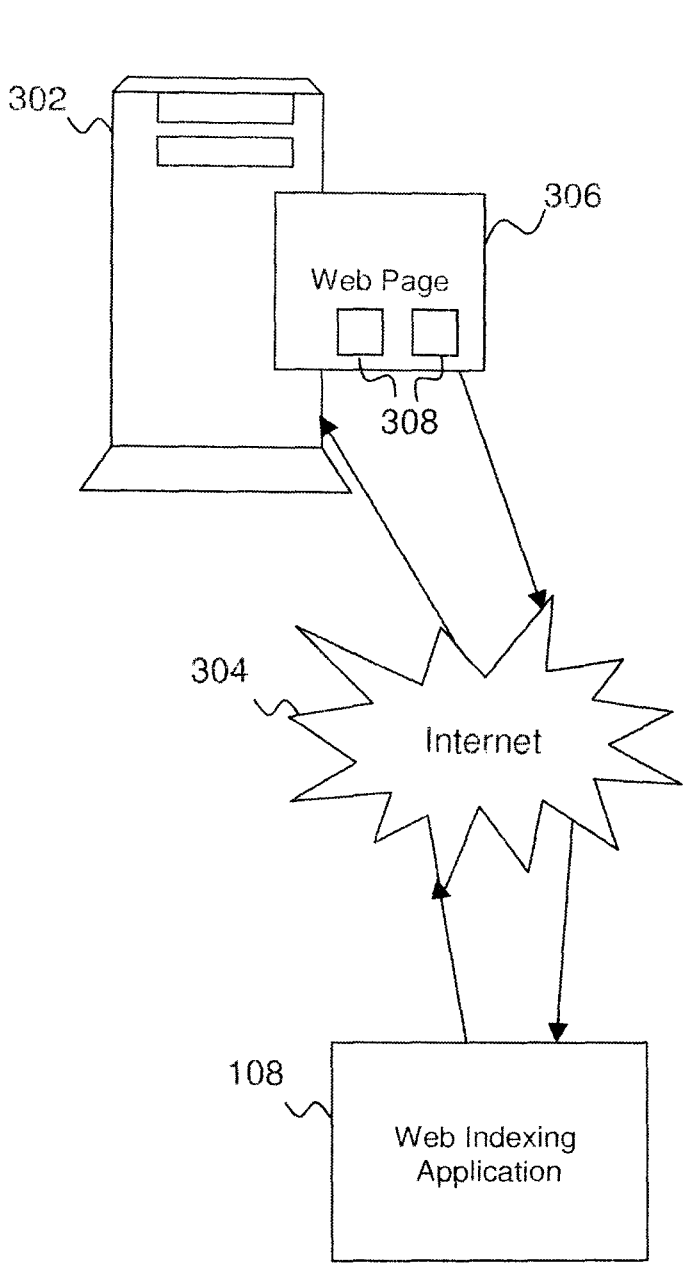
300
302
306
Web Page
308
304
Internet
108
Web Indexing Application

FIG. 3

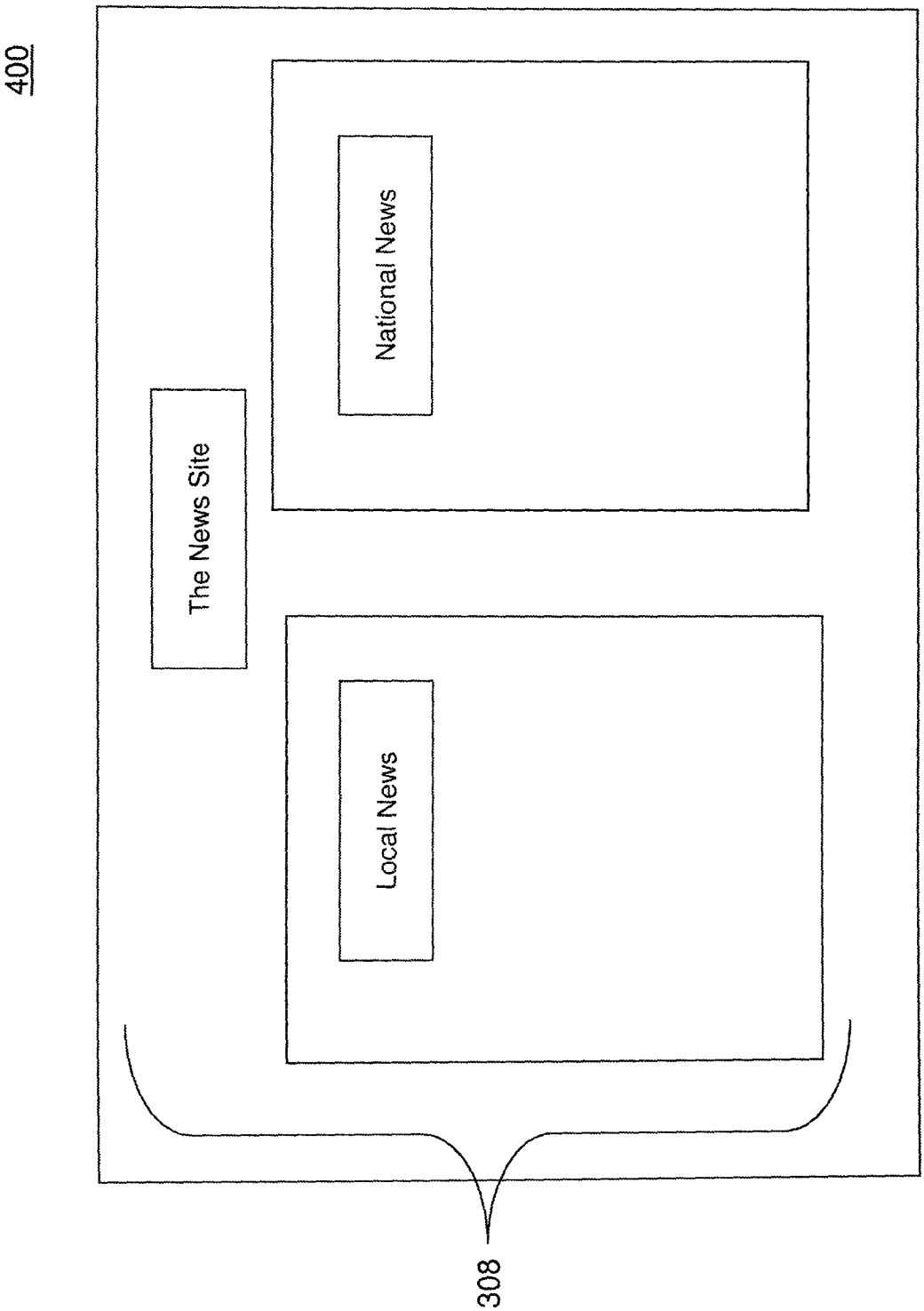

FIG. 4

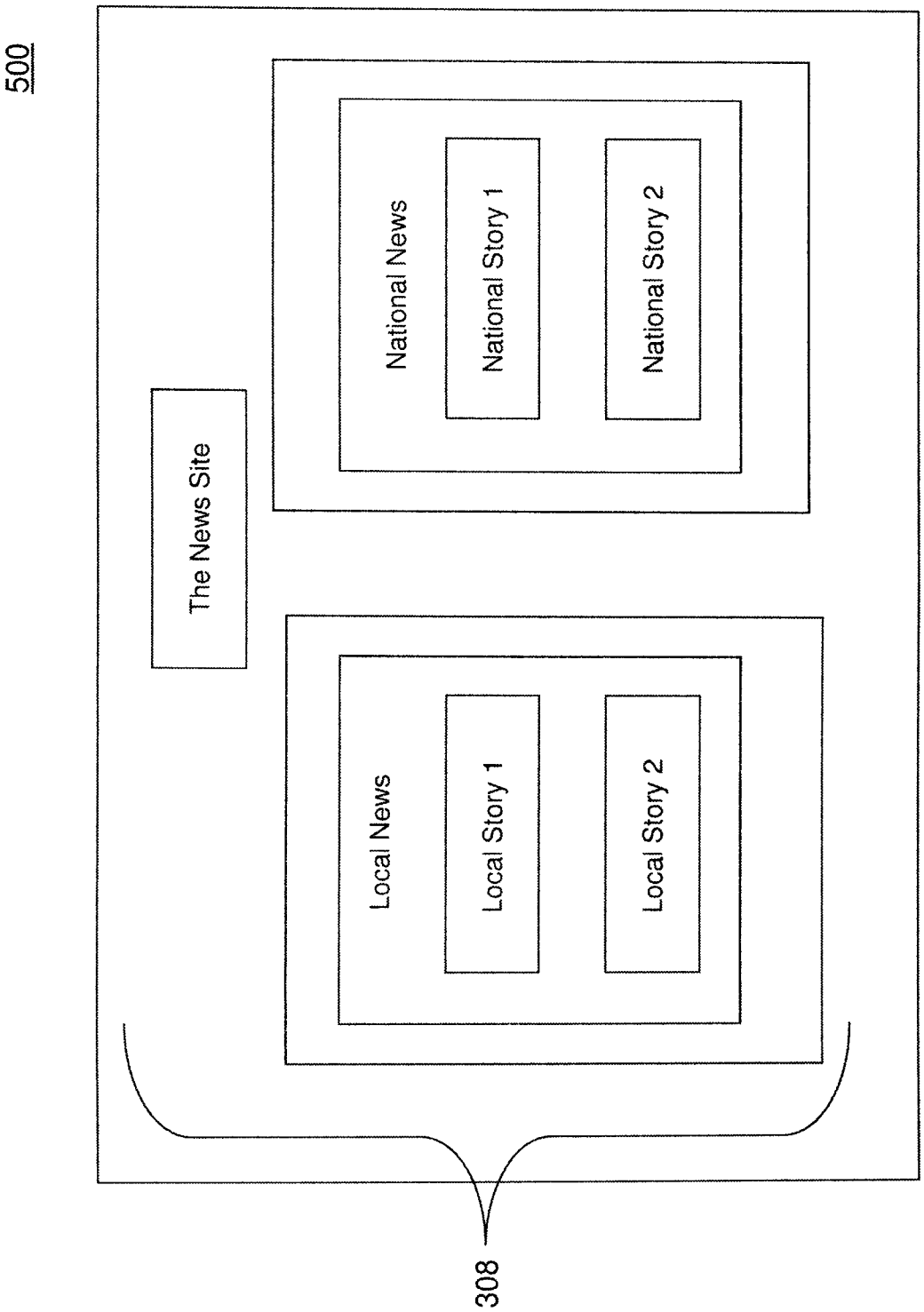
500
The News Site
National News
National Story 1
National Story 2
Local News
Local Story 1
Local Story 2
308

FIG. 5

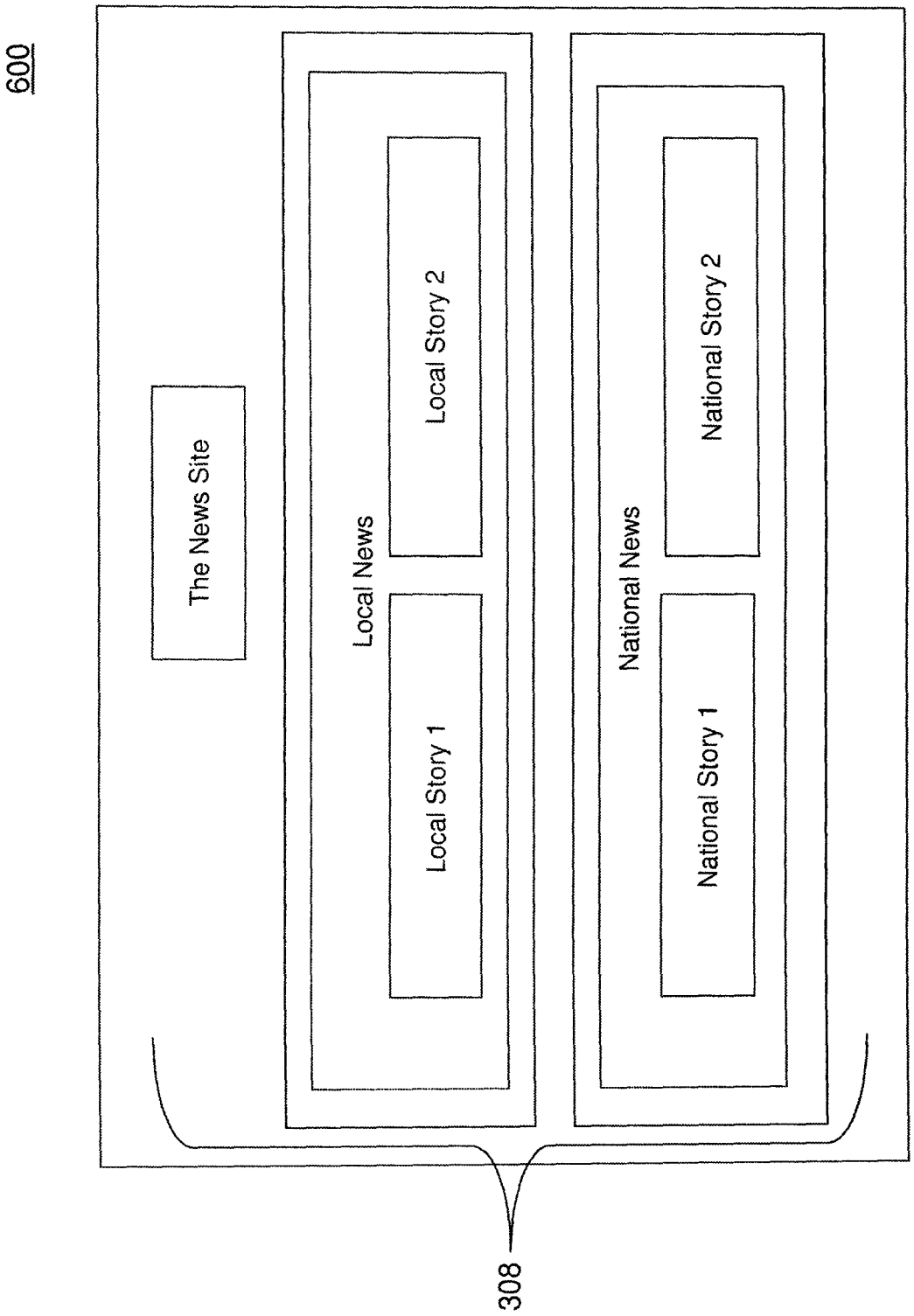
600
The News Site
Local News
Local Story 1
Local Story 2
National News
National Story 1
National Story 2
308

FIG. 6

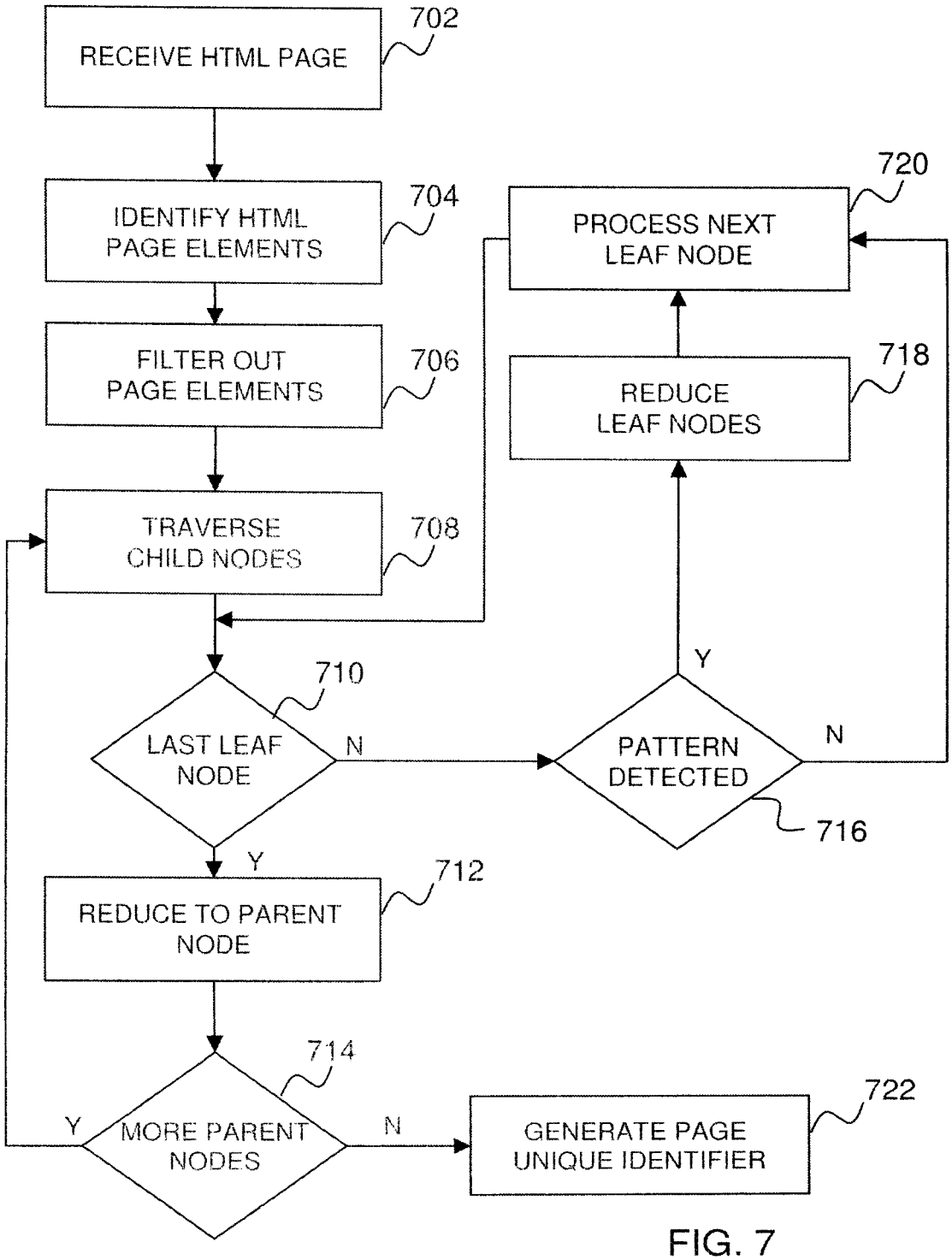
RECEIVE HTML PAGE
702
IDENTIFY HTML PAGE ELEMENTS
704
FILTER OUT PAGE ELEMENTS
706
TRAVERSE CHILD NODES
708
LAST LEAF NODE
710
N
Y
REDUCE TO PARENT NODE
712
MORE PARENT NODES
714
Y
N
PATTERN DETECTED
716
Y
N
REDUCE LEAF NODES
718
PROCESS NEXT LEAF NODE
720
GENERATE PAGE UNIQUE IDENTIFIER
722

FIG. 7

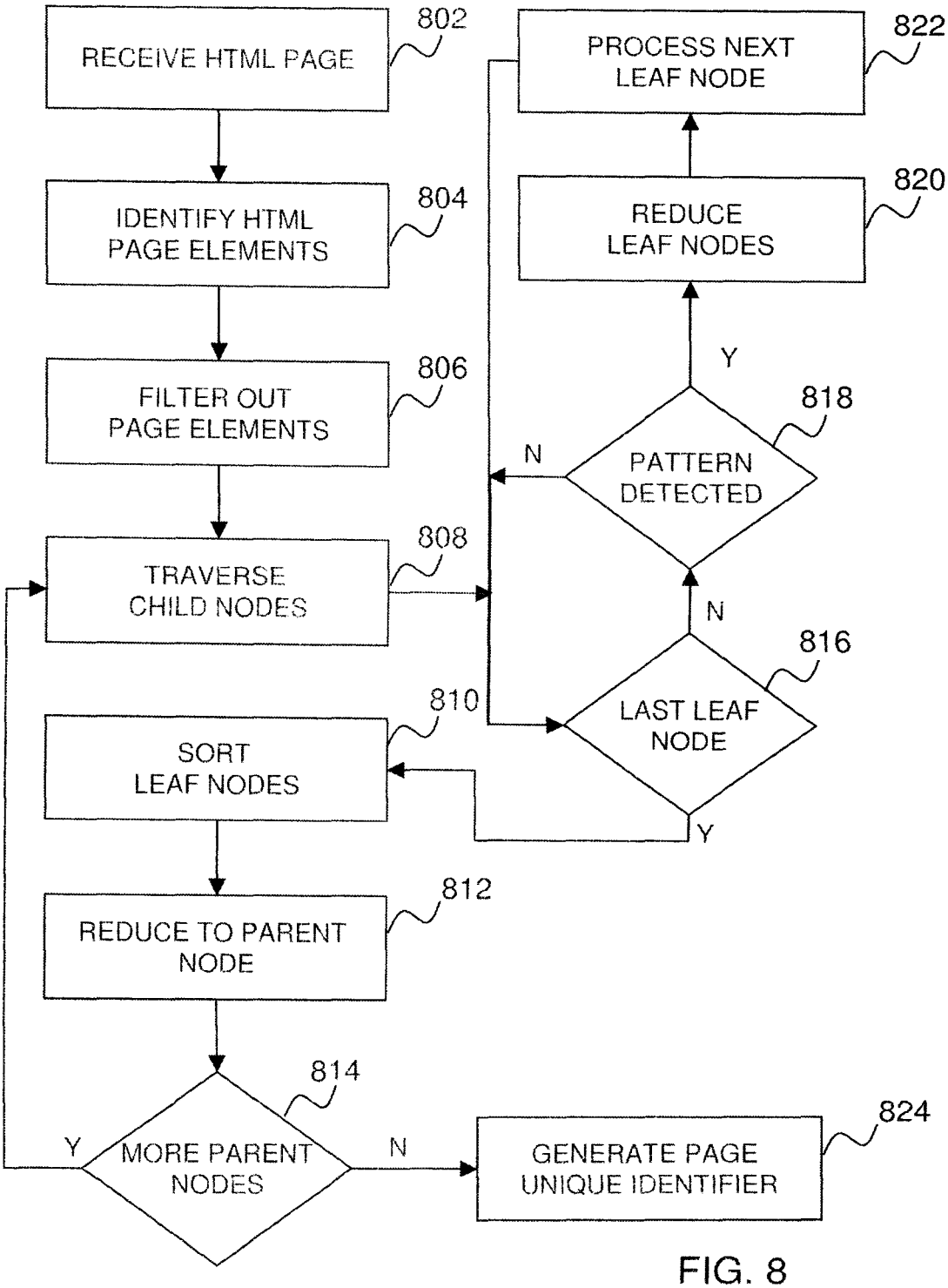
RECEIVE HTML PAGE
802
IDENTIFY HTML PAGE ELEMENTS
804
FILTER OUT PAGE ELEMENTS
806
TRAVERSE CHILD NODES
808
SORT LEAF NODES
810
REDUCE TO PARENT NODE
812
MORE PARENT NODES
814
Y
N
LAST LEAF NODE
816
N
Y
PATTERN DETECTED
818
N
Y
REDUCE LEAF NODES
820
PROCESS NEXT LEAF NODE
822
GENERATE PAGE UNIQUE IDENTIFIER
824

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2011/059762

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F17/30
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal , WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2004/260676 Al (DOUGLIS FREDERICK [US] ET AL) 23 December 2004 (2004-12-23) abstract; figures 1,2A,3 ,4,7 paragraph [0046] - paragraph [0076] paragraph [0098] - paragraph [0109]	1-24
X	LAKSHMISH RAMASWAMY ET AL: "Automatic detection of fragments in dynamically generated web pages" , PROCEEDINGS OF THE 13TH CONFERENCE ON WORLD WIDE WEB , WWW '04 , 17 May 2004 (2004-05-17) , - 22 May 2004 (2004-05-22) , page 443 , XP55002984, New York, New York, USA D0I : 10. 1145/988672 .988732 ISBN : 978-1-58-113844-3 the whole document	1-24

-/- ·

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" documentwhich may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such docu¬ments, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 July 2011	26/07/2011

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Nazzaro, Antonio

3

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2008/263026 A1 (SASTURKAR AMIT [US] ET AL) 23 October 2008 (2008-10-23) paragraph [0020] - paragraph [0045]	1-24
X	US 2009/049062 A1 (CHITRAPURA KRISHNA PRASAD [IN] ET AL) 19 February 2009 (2009-02-19) abstract paragraph [0032] - paragraph [0047] figures 2B,3-4	1-24

3

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2011/059762

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004260676 A1	23-12-2004	NONE	
US 2008263026 A1	23-10-2008	NONE	
us 2009049062 A1	19-02-2009	NONE	