US 20080288486A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2008/0288486 A1**

(43) **Pub. Date: Nov. 20, 2008**

Kim et al.

(54) **METHOD AND SYSTEM FOR AGGREGATE WEB SITE DATABASE PRICE WATCH FEATURE**

(76) Inventors: **Sang-Heun Kim**, Mississauga (CA); **Charles Laurence Stinson**, Mississauga (CA)

Correspondence Address:
GOWLING LAFLEUR HENDERSON LLP
SUITE 1600, 1 FIRST CANADIAN PLACE, 100 KING STREET WEST
TORONTO, ON M5X 1G5 (CA)

(21) Appl. No.: **12/119,309**

(22) Filed: **May 12, 2008**

Related U.S. Application Data

(60) Provisional application No. 60/924,503, filed on May 17, 2007.

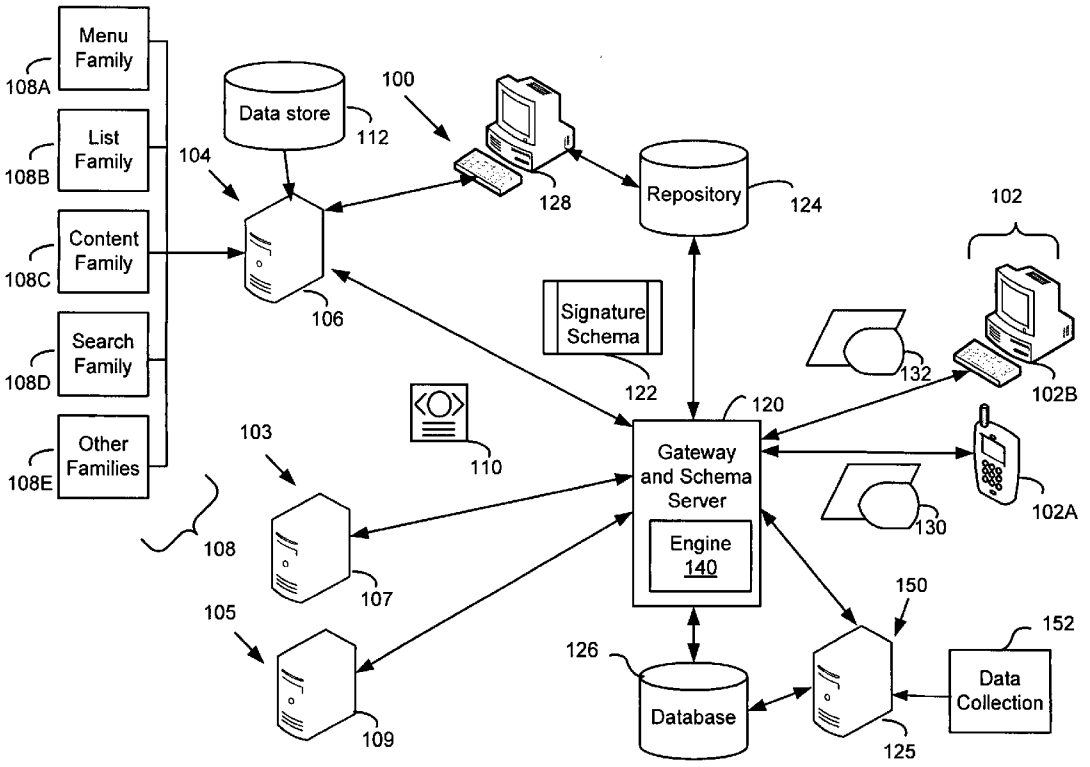
Publication Classification

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. Cl.** **707/5**; 707/E17.108

(57) **ABSTRACT**

Signature schema documents may be pre-defined using a query language to provide instructions for application by an engine to extract data from web pages of respective web sites. For a particular web page, signature schema instructions identify a web page family for the web page and extract desired data from the web page in accordance with its web page family. A server may receive data from a web site and apply signature schema instructions maintained in a repository coupled to the engine. Extracted data can be cached to a database coupled to the engine to facilitate querying of the data to enable aggregate results to be presented to a client machine (e.g. a wireless device). The aggregate database can be monitored based upon defined user criteria such as for price changes of an item and provide appropriate notification to the client machine when changes occur.



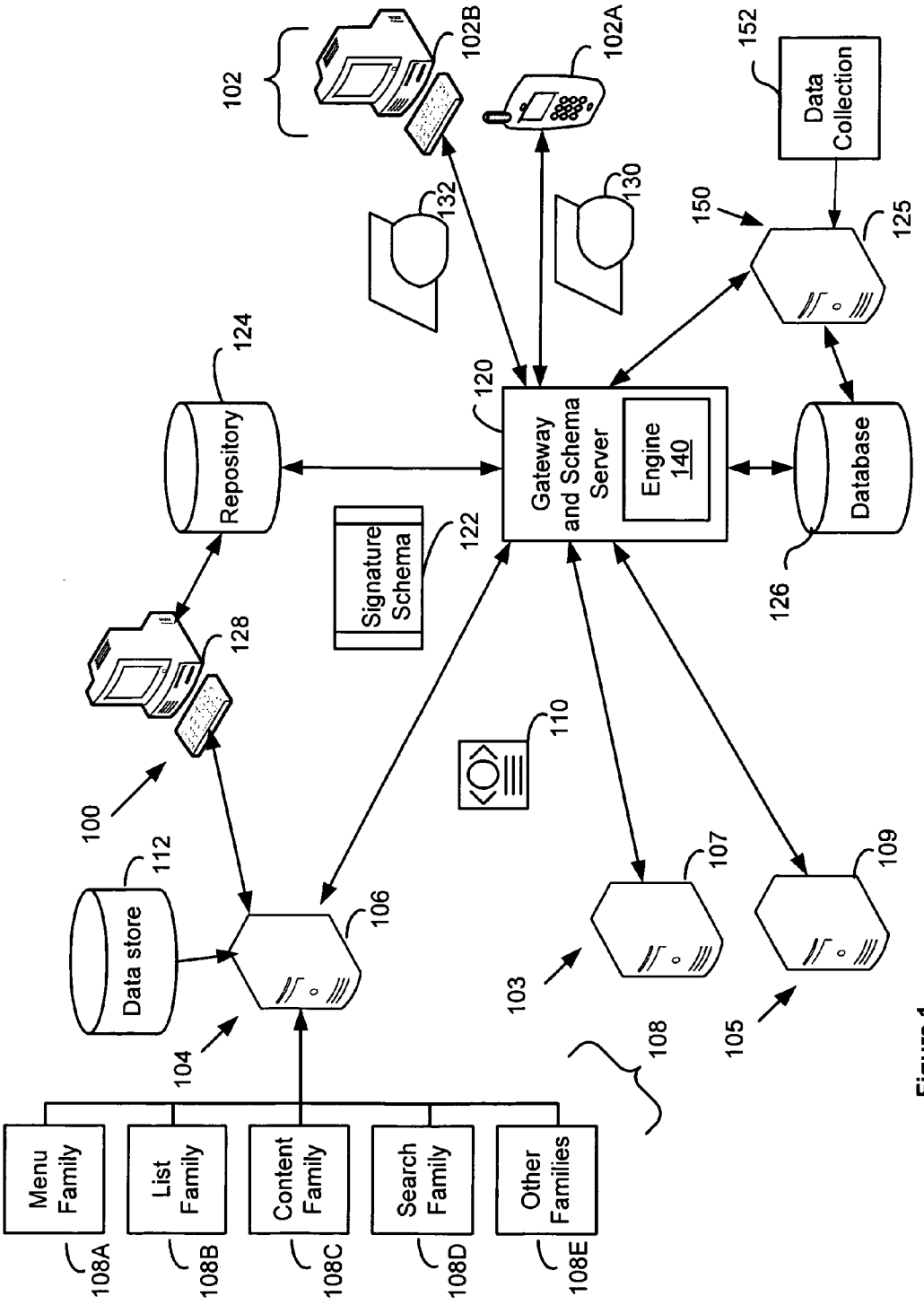

102
102B
102A
152
Data Collection
132
130
150
125
124
120
Repository
Gateway and Schema Server
Engine 140
Database
Signature Schema
122
126
128
110
100
112
Data store
106
107
109
103
105
104
108
Menu Family
List Family
Content Family
Search Family
Other Families
108A
108B
108C
108D
108E

Figure 1

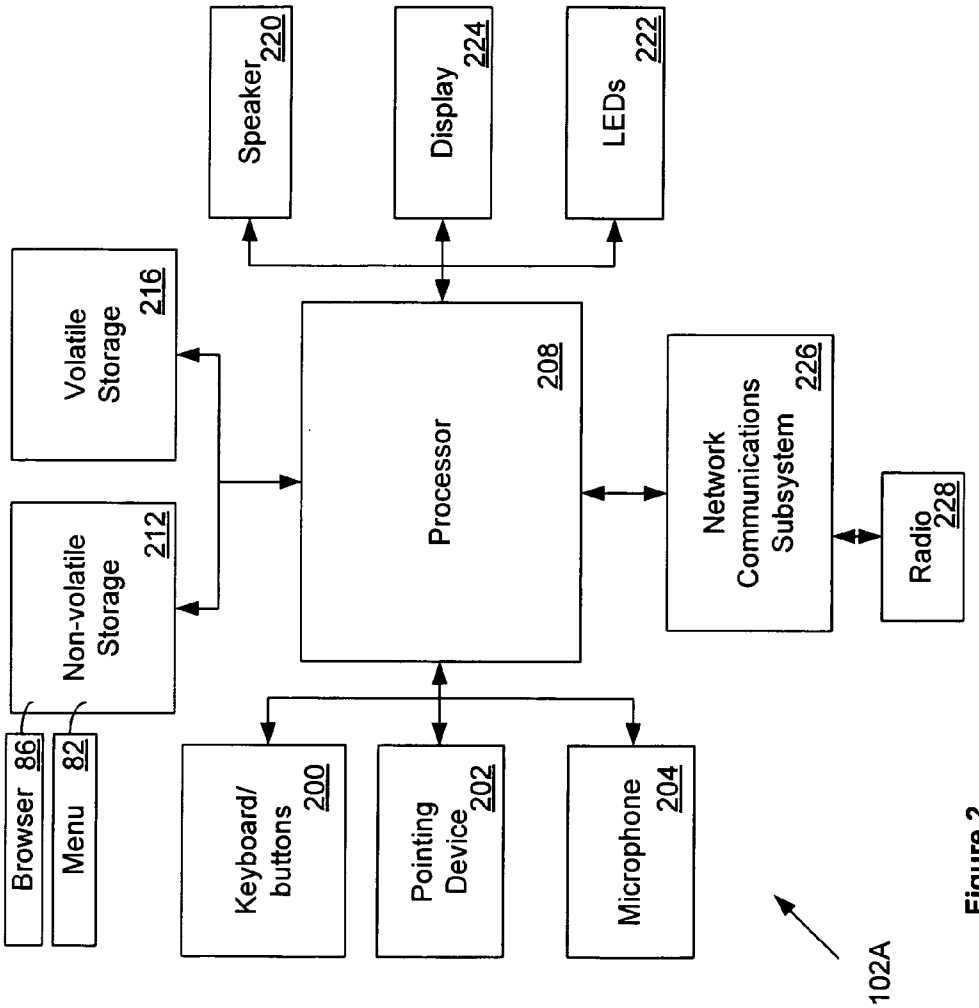

Speaker 220
Display 224
LEDs 222
Volatile Storage 216
Processor 208
Network Communications Subsystem 226
Radio 228
Non-volatile Storage 212
Browser 86
Menu 82
Keyboard/ buttons 200
Pointing Device 202
Microphone 204
102A

Figure 2

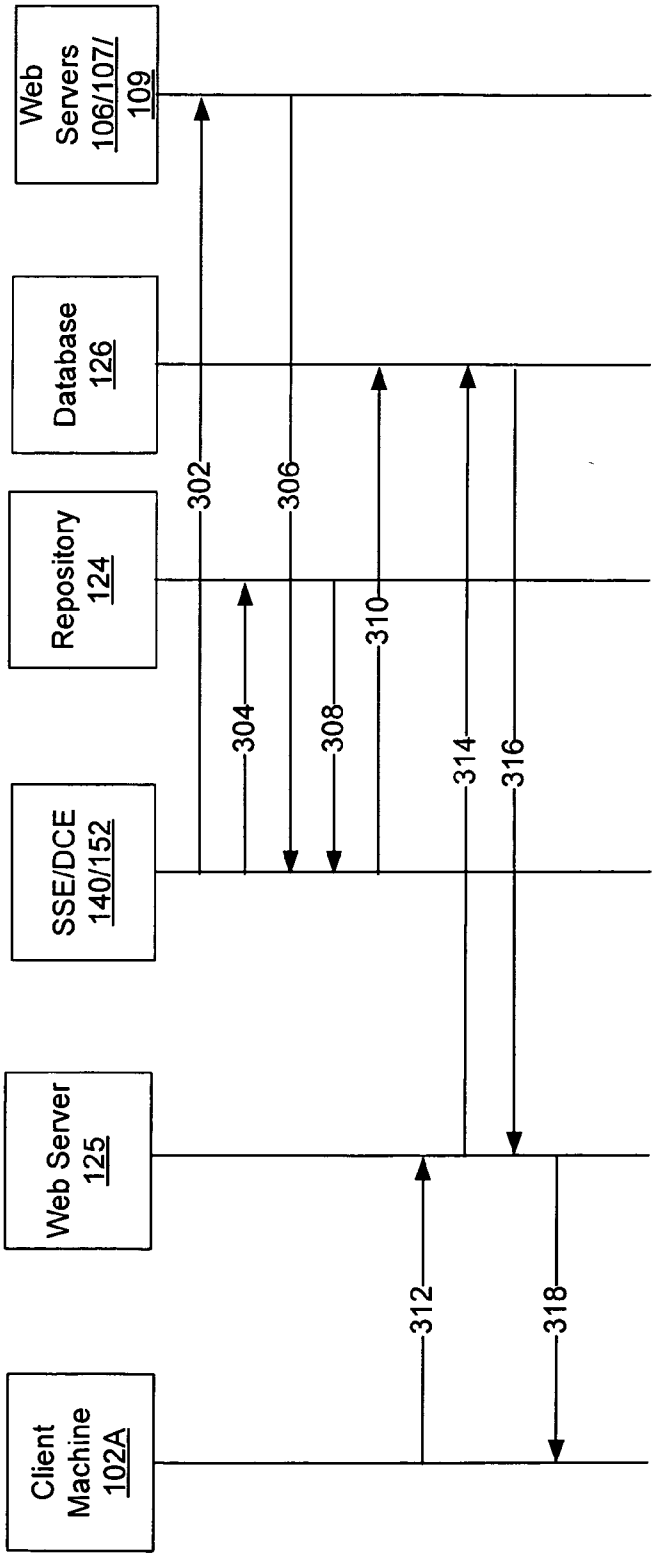

Web Servers 106/107/ 109
Database 126
Repository 124
SSE/DCE 140/152
Web Server 125
Client Machine 102A
302
304
306
308
310
312
314
316
318

Figure 3

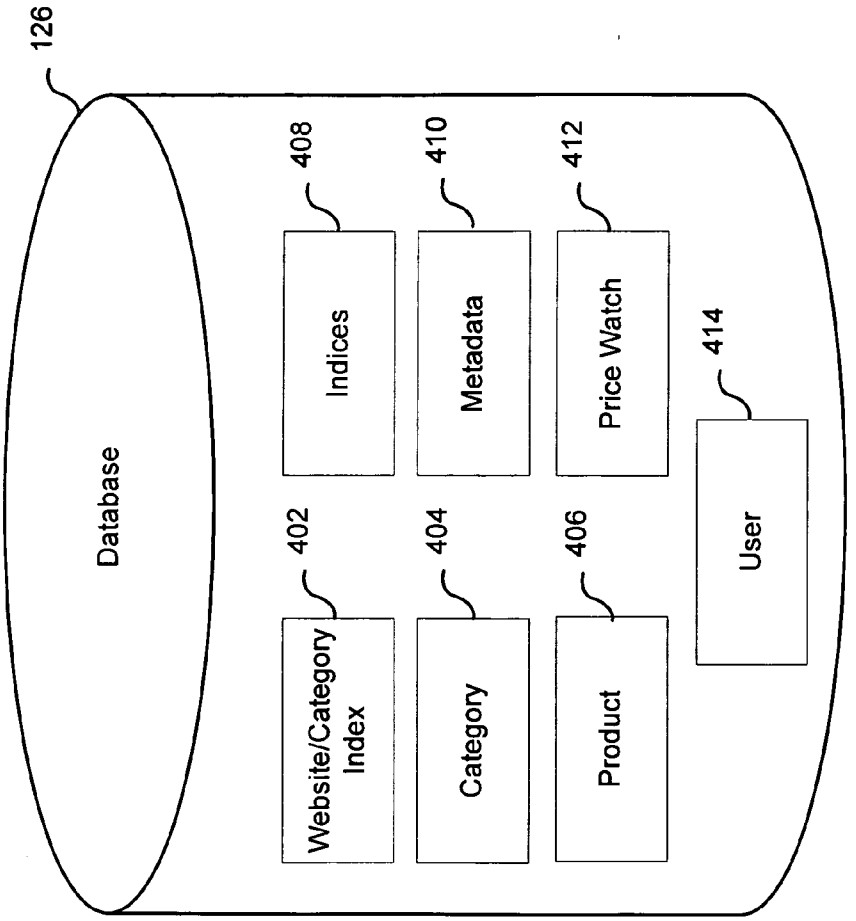

126
Database
408
Indices
410
Metadata
412
Price Watch
414
User
402
Website/Category Index
404
Category
406
Product

Figure 4

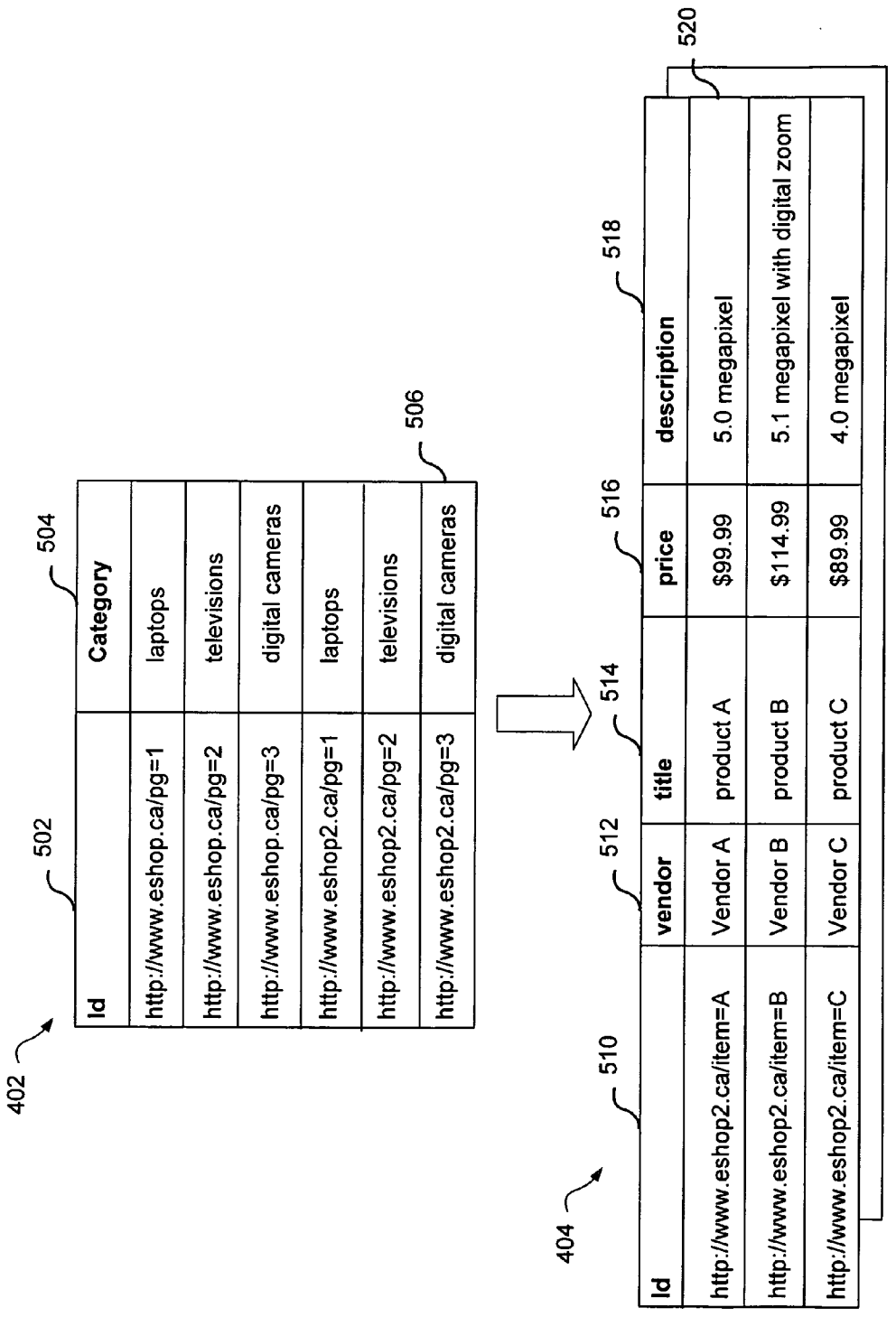

402
502
504
506
Id
Category
http://www.eshop.ca/pg=1
laptops
http://www.eshop.ca/pg=2
televisions
http://www.eshop.ca/pg=3
digital cameras
http://www.eshop2.ca/pg=1
laptops
http://www.eshop2.ca/pg=2
televisions
http://www.eshop2.ca/pg=3
digital cameras
404
510
512
514
516
518
520
Id
vendor
title
price
description
http://www.eshop2.ca/item=A
Vendor A
product A
$99.99
5.0 megapixel
http://www.eshop2.ca/item=B
Vendor B
product B
$114.99
5.1 megapixel with digital zoom
http://www.eshop2.ca/item=C
Vendor C
product C
$89.99
4.0 megapixel

Figure 5

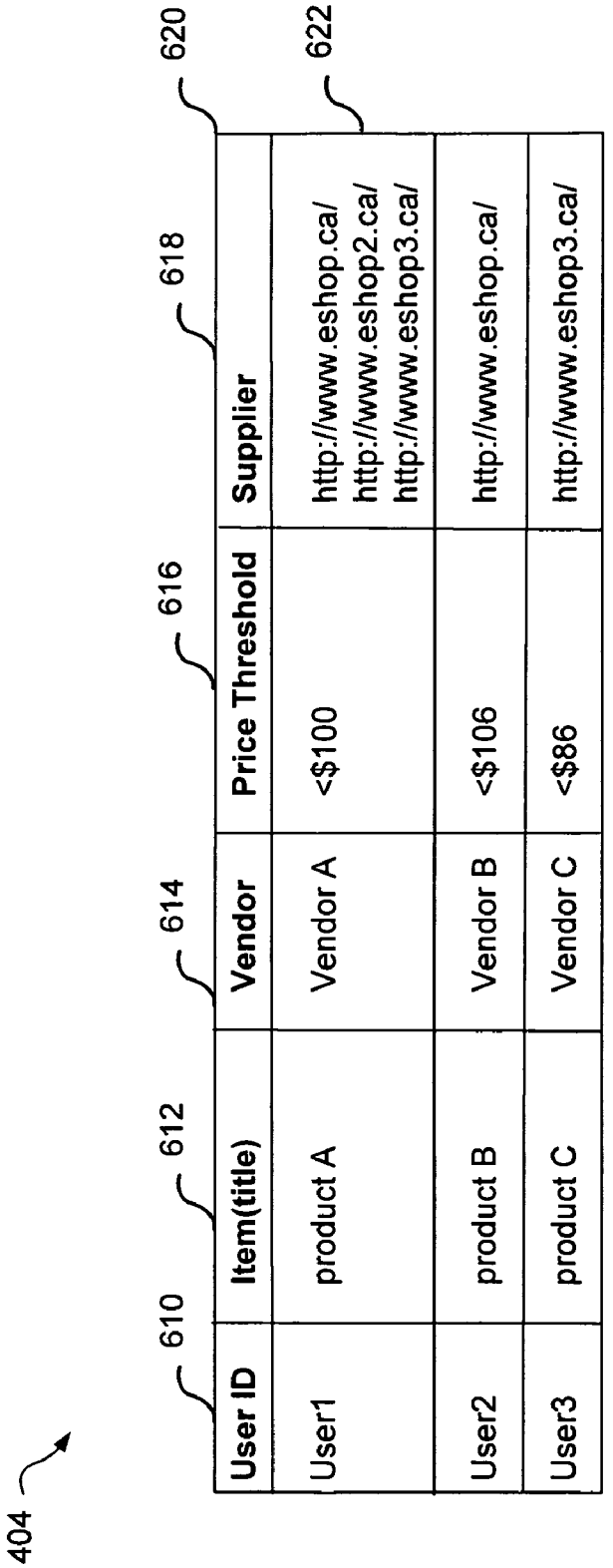

404
610
612
614
616
618
620
622
User ID
Item(title)
Vendor
Price Threshold
Supplier
User1
product A
Vendor A
<$100
http://www.eshop.ca/
http://www.eshop2.ca/
http://www.eshop3.ca/
User2
product B
Vendor B
<$106
http://www.eshop.ca/
User3
product C
Vendor C
<$86
http://www.eshop3.ca/

Figure 6

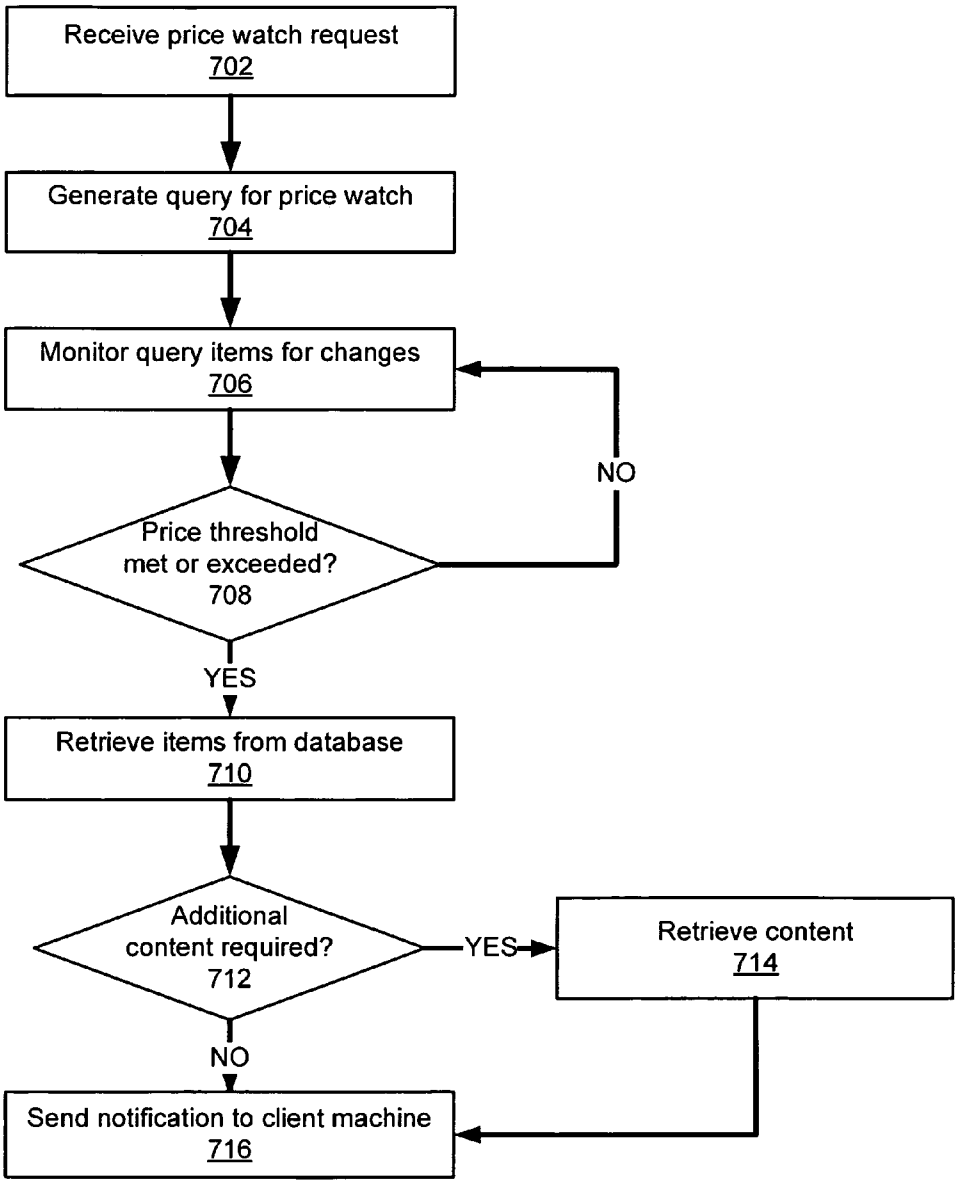

Receive price watch request
702
Generate query for price watch
704
Monitor query items for changes
706
Price threshold met or exceeded?
708
NO
YES
Retrieve items from database
710
Additional content required?
712
YES
Retrieve content
714
NO
Send notification to client machine
716

Figure 7

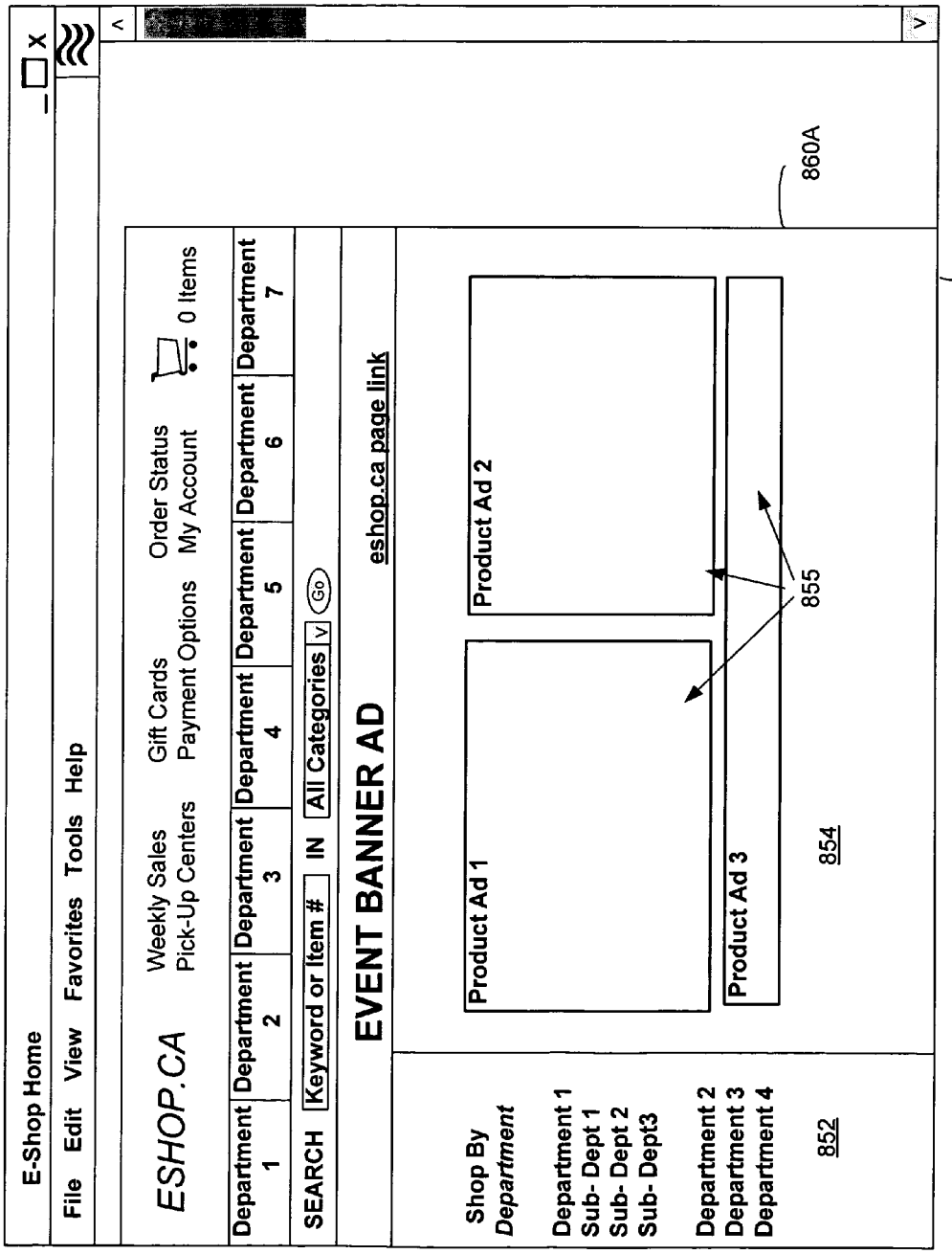

E-Shop Home
File Edit View Favorites Tools Help
ESHOP.CA
Weekly Sales
Pick-Up Centers
Gift Cards
Payment Options
Order Status
My Account
0 Items
Department 1
Department 2
Department 3
Department 4
Department 5
Department 6
Department 7
SEARCH
Keyword or Item #
IN
All Categories
Go
EVENT BANNER AD
eshop.ca page link
Shop By Department
Department 1
Sub- Dept 1
Sub- Dept 2
Sub- Dept3
Department 2
Department 3
Department 4
852
Product Ad 1
Product Ad 2
Product Ad 3
854
855
860A
850

Figure 8A

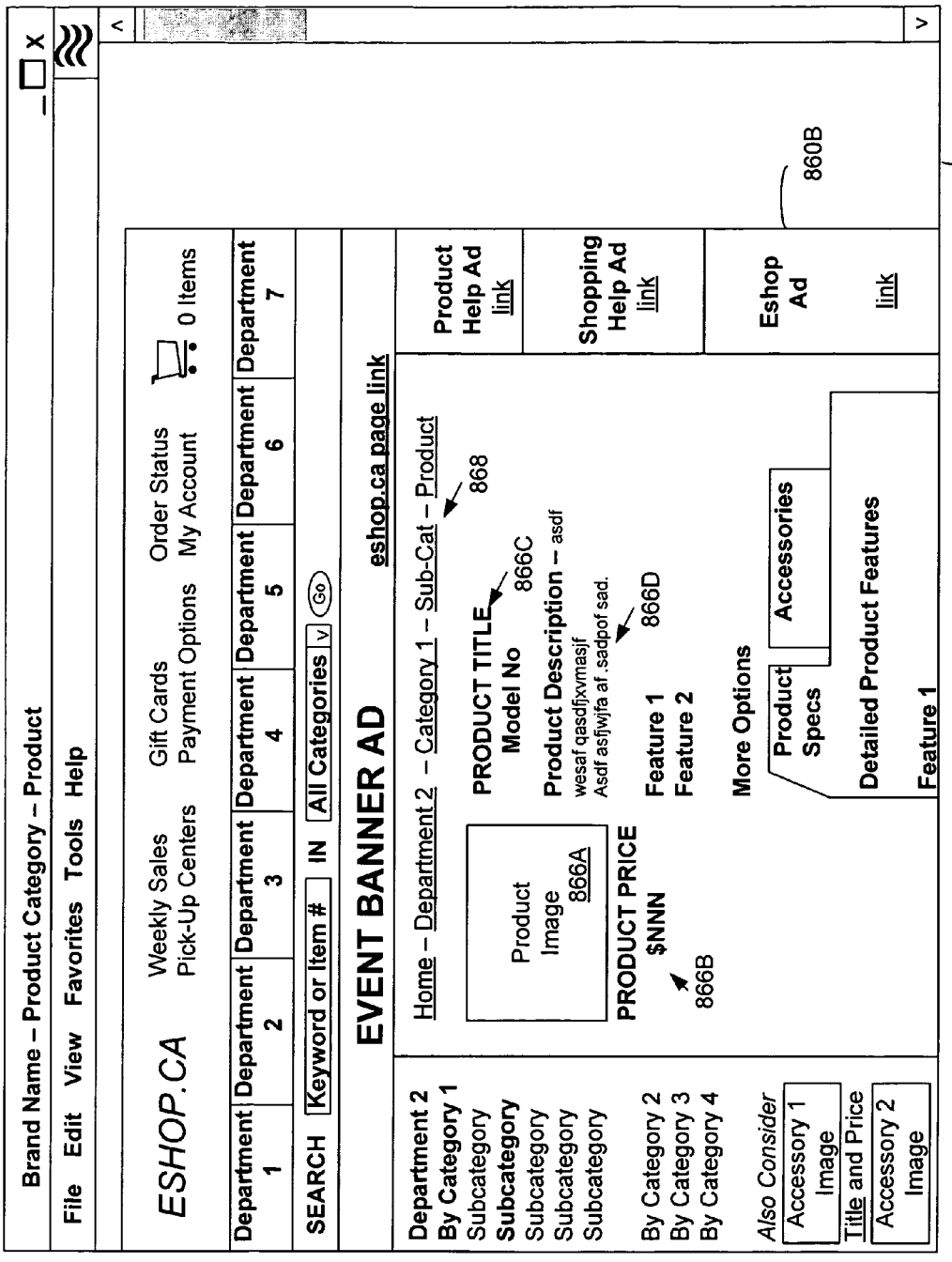

Brand Name – Product Category – Product
File Edit View Favorites Tools Help
ESHOP.CA
Weekly Sales
Pick-Up Centers
Gift Cards
Payment Options
Order Status
My Account
0 Items
Department 1
Department 2
Department 3
Department 4
Department 5
Department 6
Department 7
SEARCH Keyword or Item # IN All Categories Go
EVENT BANNER AD
eshop.ca page link
Department 2
By Category 1
Subcategory
Subcategory
Subcategory
Subcategory
By Category 2
By Category 3
By Category 4
Also Consider
Accessory 1 Image
Title and Price
Accessory 2 Image
Home – Department 2 – Category 1 – Sub-Cat – Product
868
Product Image
866A
PRODUCT TITLE
Model No
866C
Product Description – asdf
wesaf qasdfjxvmasjf
Asdf asfjwjfa af .sadpof sad.
PRODUCT PRICE
$NNN
866B
Feature 1
866D
Feature 2
More Options
Product Specs
Accessories
Detailed Product Features
Feature 1
Product Help Ad link
Shopping Help Ad link
Eshop Ad
860B
link
850

Figure 8B

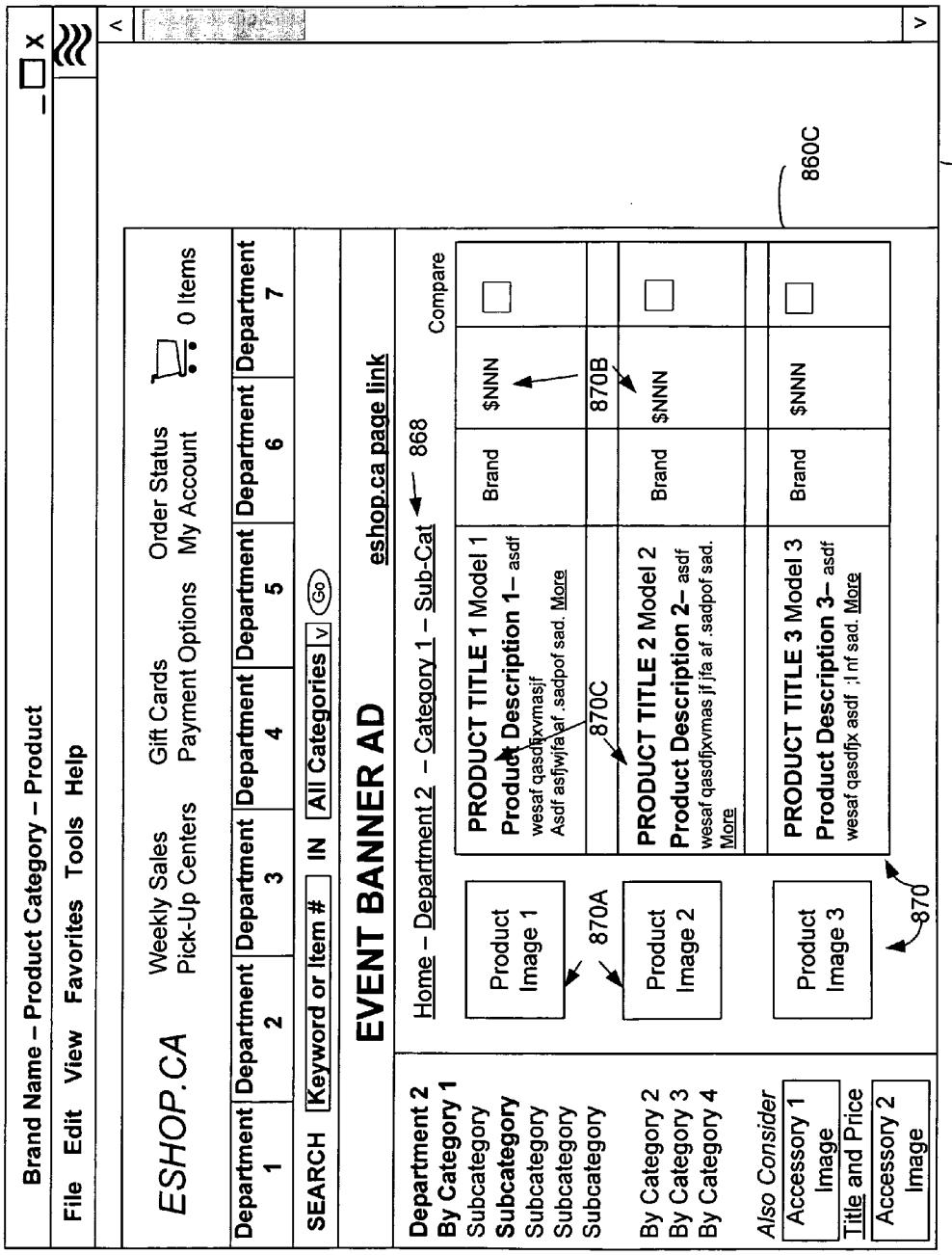

Brand Name – Product Category – Product
File Edit View Favorites Tools Help
ESHOP.CA
Weekly Sales
Pick-Up Centers
Gift Cards
Payment Options
Order Status
My Account
0 Items
Department 1
Department 2
Department 3
Department 4
Department 5
Department 6
Department 7
SEARCH Keyword or Item # IN All Categories Go
EVENT BANNER AD
eshop.ca page link
Home – Department 2 – Category 1 – Sub-Cat
868
Compare
Department 2
By Category 1
Subcategory
Subcategory
Subcategory
Subcategory
By Category 2
By Category 3
By Category 4
Also Consider
Accessory 1 Image
Title and Price
Accessory 2 Image
Product Image 1
Product Image 2
Product Image 3
870A
870
PRODUCT TITLE 1 Model 1
Product Description 1– asdf
wesaf qasdfjxvmasjf
Asdf asfjwjfa af .sadpof sad. More
Brand
$NNN
870C
870B
PRODUCT TITLE 2 Model 2
Product Description 2– asdf
wesaf qasdfjxvmas jf jfa af .sadpof sad.
More
Brand
$NNN
PRODUCT TITLE 3 Model 3
Product Descrip 3– asdf
wesaf qasdfjx asdf ;l nf sad. More
Brand
$NNN
860C
850

Figure 8C

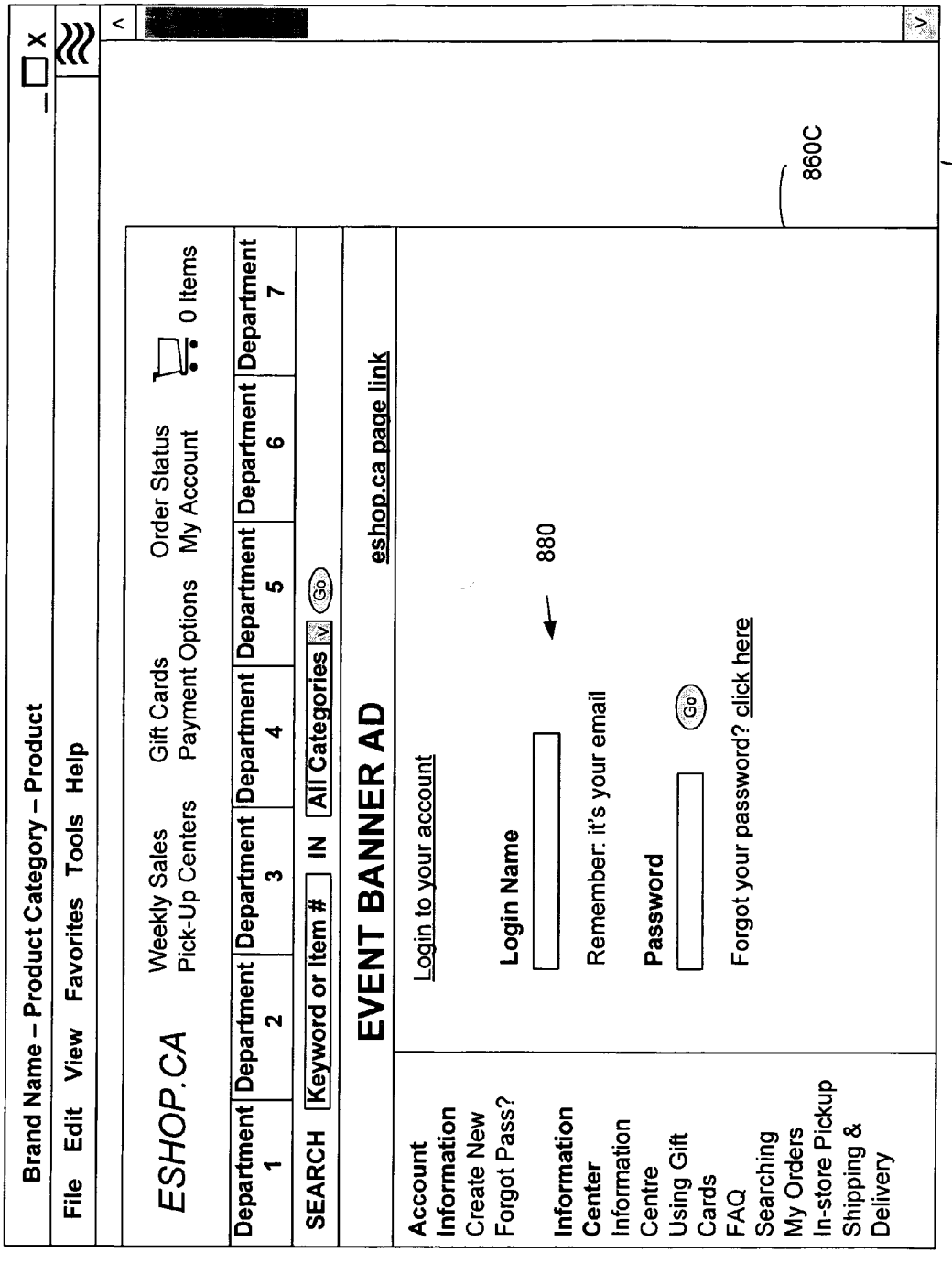

Brand Name – Product Category – Product
File Edit View Favorites Tools Help
ESHOP.CA
Weekly Sales
Pick-Up Centers
Gift Cards
Payment Options
Order Status
My Account
0 Items
Department 1
Department 2
Department 3
Department 4
Department 5
Department 6
Department 7
SEARCH
Keyword or Item #
IN
All Categories
Go
EVENT BANNER AD
eshop.ca page link
Account Information
Create New
Forgot Pass?
Information Center
Information Centre
Using Gift Cards
FAQ
Searching
My Orders
In-store Pickup
Shipping & Delivery
Login to your account
Login Name
880
Remember: it's your email
Password
Go
Forgot your password? click here
860C
850

Figure 8D

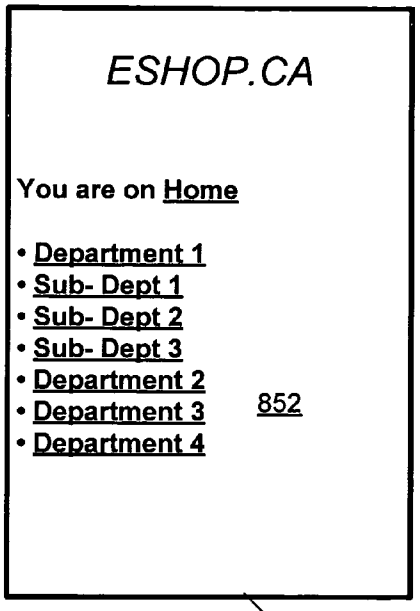


Figure 9A

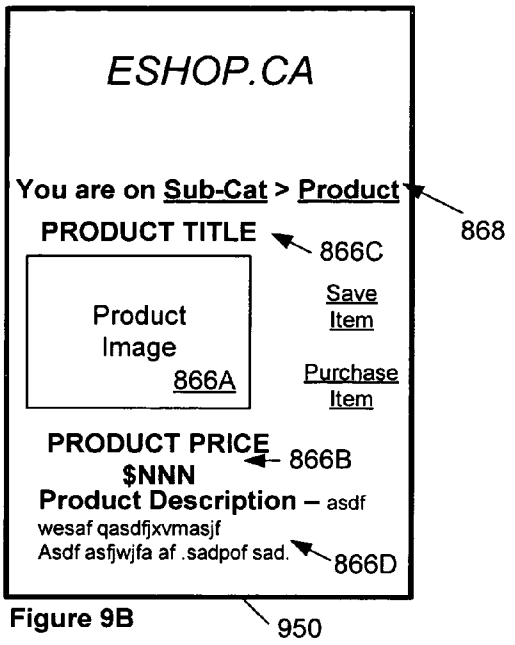


Figure 9B

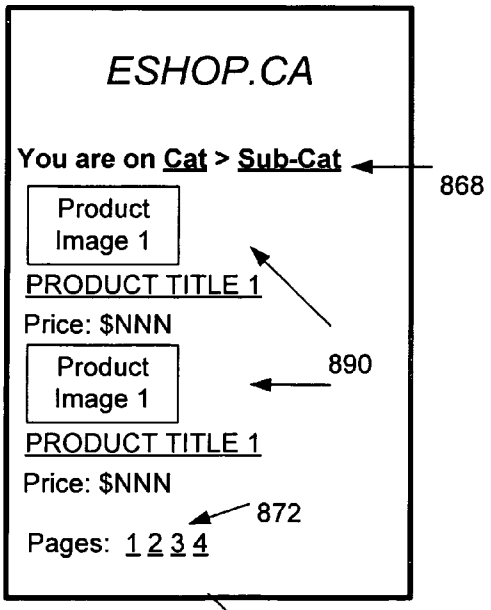


Figure 9C

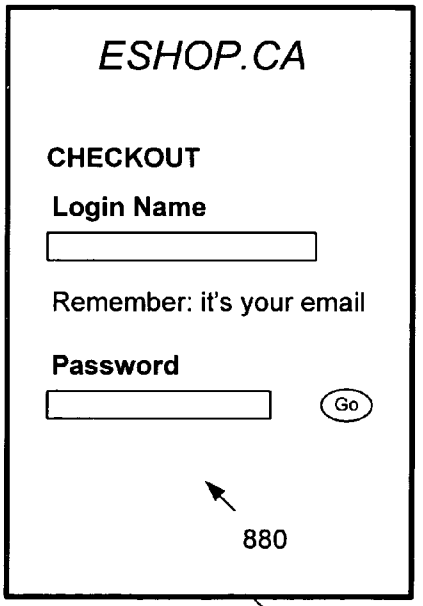


Figure 9D

METHOD AND SYSTEM FOR AGGREGATE WEB SITE DATABASE PRICE WATCH FEATURE

CROSS REFERENCE

[0001] This application claims the benefit of the prior filing of U.S. Provisional Patent Application Ser. No. 60/924503 filed May 17, 2007, the disclosure of which is incorporated herein by reference.

COPYRIGHT

[0002] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights.

FIELD

[0003] The present application relates generally to telecommunications and more particularly to a system and method for generating an aggregate web site search database and price watch feature.

BACKGROUND

[0004] Web sites host and provide information using web pages that are communicated electronically via a telecommunications network. Accessing this information by some client computing devices can be challenging. Computing devices are becoming smaller and increasingly utilize wireless connectivity. Examples of such computing devices include portable computing devices that include wireless network browsing capability as well as telephony and personal information management capabilities.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] FIG. **1** is schematic representation of a system for content navigation.

[0006] FIG. **2** is a schematic representation of a wireless communication device from FIG. **1**.

[0007] FIG. **3** illustrates a flow of interactions among components of the system of FIG. **1**.

[0008] FIG. **4** is a schematic representation of an aggregate web site search database for e-commerce.

[0009] FIG. **5** is a schematic representation of tables in an aggregate web site search database for e-commerce.

[0010] FIG. **6** is a schematic representation of a price watch table.

[0011] FIG. **7** illustrates a method of providing an aggregate web site price watch.

[0012] FIGS. **8**A-**8**D and **9**A-**9**D respectively illustrate representative web pages rendered on a first browser window and portions of said representative web pages transcoded and rendered on a second browser window in accordance with an embodiment.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0013] The smaller size of such client devices necessarily limits their display capabilities. Furthermore the wireless connections to such devices typically have less or more expensive bandwidth than corresponding wired connections. The Wireless Application Protocol ("WAP") was designed to address such issues, but WAP can still provide a very unsatisfactory experience or even completely ineffective experience, particularly where the small client device needs to effect a connection with web sites that host web pages that are directed to traditional full desktop browsers. Ability to monitor prices of products on e-commerce sites is dependent on a Really Simple Syndication (RSS) feed or other push mechanism from the merchant site to pricing web sites. These price monitoring mechanisms are dependent on the merchant site conveying the information to the appropriate pricing site.

[0014] Signature schema documents may be pre-defined using a query language to provide instructions for application by an engine to extract data from web pages of respective web sites for storage to an aggregate database. For a particular web page, signature schema instructions identify a web page family for the web page and extract desired data from the web page in accordance with its web page family. The instructions use signatures previously identified within web pages of the same family to distinguish the web page family (e.g. in accordance with a shared template for each family) from others of the web site and to distinguish the desired data from other data for the web page family. A gateway server may receive data (e.g. web page) from a web site and apply signature schema instructions maintained in a repository coupled to the engine. Extracted data can be cached to a database coupled to the engine to facilitate querying of the data to enable aggregate results to be presented to a client machine (e.g. a wireless communication device). The aggregate database can be queried based upon defined user criteria such as price changes of an item and provide appropriate notification to the client machine when changes occur.

[0015] In accordance with the present disclosure there is provided a method of providing aggregate web site price watch feature, the method comprising: receiving a price watch request for a product from a client machine comprising a threshold price for the product; generating a query to an aggregate database for the product; retrieving data from the aggregate database containing information provided from one or more web sites, extracted by parsing received web site data by an associated signature schema; determining when the threshold price has been satisfied; retrieving one or more product details from the aggregate database when the threshold price has been satisfied; and notifying the client machine that the price threshold was satisfied.

[0016] In accordance with the present disclosure there is provided a system for providing aggregate web site database price watch feature comprising: at least one computing device comprising a processor and a memory coupled thereto, said memory storing instructions and data for configuring the processor to provide an engine to: receive a price watch request for a product from a client machine comprising a threshold price; generate a query to an aggregate web site database for the requested product; retrieve data from the aggregate database containing information provided from one or more web sites extracted by parsing received web site data by and associate signature schema; determine if the threshold price is satisfied; retrieve one ore more product details from the aggregate database when the product price threshold has been satisfied; and notify the client machine that the price threshold was satisfied.

[0017] In accordance with the present disclosure there is provided a computer program product storing computer readable instructions which when executed by a computer proces-

sor configure the processor for: receiving a price watch request for a product from a client machine comprising a threshold price for the product; generating a query to an aggregate database for the product; retrieving data from the aggregate database containing information provided from one or more web sites, extracted by parsing received web site data by an associated signature schema; determining when the threshold price has been satisfied; retrieving one or more product details from the aggregate database when the threshold price has been satisfied; and notifying the client machine that the price threshold was satisfied.

[0018] In accordance with the present disclosure there is provided a method of providing aggregate web site price watch feature for a wireless device, the method comprising: receiving a price watch request for a product from a client machine comprising a threshold price for the product; generating a query to an aggregate database for the requested product; retrieving data from the aggregate database containing information provided from one or more web sites, extracted by parsing received web site data by an associated signature wherein the schema comprises an eXtensible Markup Language (XML) documents comprising query language for extracting data from the web page; determining when the threshold price has been satisfied; periodically monitoring the one or more websites for changes in the product price; retrieving one ore more product details from the aggregate database when the product threshold price has been satisfied; and notifying the wireless device that the price threshold was satisfied.

[0019] Referring now to FIG. **1**, there is illustrated a system **100** for content navigation via a telecommunications network. In a present embodiment system **100** comprises one or more of client computing devices in the form of client machines **102**A and **102**B (collectively **102**), web site servers **106**, **107** and **109** respectively host web sites **104**, **103** and **105** and a gateway and schema server **120**. Machines **102** are respectively coupled to communicate with gateway and schema server **120** to obtain web pages (e.g. **110**) transcoded from web sites **103**, **104** and **105** and to access aggregate data from the web sites through web server **125** hosting web site **150**.

[0020] In the present embodiment, web sites **103**, **104** and **105** host web sites which contain data that is to be aggregated into database **126**. For example, web site **104** comprises a web server **106** serving web pages (e.g. **110**) defined from one or more web page family templates **108**A-**108**D (collectively **108**) and web page content (described further herein below) from data store **112**. In the present embodiment of system **100**, gateway and schema server **120** is coupled to a schema repository **124** from which to obtain a signature schema **122** for a particular web site. Signature schema documents (e.g. **122**) provide instructions and data with which an engine **140** of server **120** can extract data from web pages (e.g. **110**) and transcode same to a target format to provide transcoded web page data (e.g. **130** and **132**) to the respective requesting client machines **102**A and **102**B as described more fully below. Gateway and schema server **120** may also be coupled to a database **126** for retrieving/storing data extracted from web sites in accordance with its operations. The database **126** may be a relational database for storing extracted data objects and elements and their relationships from web sites in relation to the defined signature schema. The stored data can be accessed by a Structured Query Language (SQL) to retrieve desired data from database **126**. Signature schemas for respective web sites may be defined (e.g. coded) using a computing device **128** as described herein below. A web server **125** is coupled to the aggregate web site database **126** to enable access to the aggregated web site database **126** data by a web site **150**. The web server **125** can also provide a data collection engine **150**, or web crawler, for sending requests to web sites **103**, **104** and **105** for desired pages and provide content to schema engine **140** for processing.

[0021] Representative client machines **102** include any type of computing or electronic device that can be used to communicate and interact with content available via web sites. Each of the client machines **102** may be operated by a respective user U (not shown). Interaction with a particular user includes presenting information on a client machine (e.g. by rendering on a display screen) as well as receiving input at a client machine (e.g. such as via a keyboard for transmitting to a web site). In the present embodiment, client machine **102**A comprises a mobile electronic device with the combined functionality of a personal digital assistant, cell phone, email paging device, and a web-browser. Such a mobile electronic device may comprise a keyboard (or other input device (s)), a display screen, a speaker, (and other output device(s) (e.g. LEDs)) and a chassis for housing such components. The chassis may further houses one or more central processing units, volatile memory (e.g. random access memory), persistent memory (e.g. Flash read only memory) and network interfaces to allow client machine **102**A to communicate over the telecommunication network.

[0022] Referring now to FIG. **2**, a schematic block diagram shows an exemplary client machine **102**A in greater detail. It should be emphasized that the structure in FIG. **2** is purely exemplary, and contemplates a device that may be used for both wireless voice (e.g. telephony) and wireless data (e.g. email, web browsing, text) communications. Client machine **102**A includes one or more input devices which in a present embodiment includes a keyboard and, typically, additional input buttons, collectively **200**, an optional pointing device **202** (e.g. a trackball or trackwheel) and a microphone **204**. Other input devices, such as a touch screen, and camera lens are also contemplated. Input from keyboard/buttons **200**, pointing device **202** and microphone **204** may be received at a processor **208**. Processor **208** may be further operatively coupled with a non-volatile storage unit **212** (e.g. read only memory ("ROM"), Erasable Electronic Programmable Read Only Memory ("EEPROM"), or Flash Memory) and a volatile storage unit **216** (e.g. random access memory ("RAM") speaker **220**, display screen **224** and one ore more lights (LEDs **222**). Processor **208** may be operatively coupled for network communications via a subsystem **226**. Wireless communications are effective via at least one radio (e.g. **228**) such as for Wi-Fi or cellular wireless communications. Client machine **102**A also may be configured for wired communications such as via a USB or other port and for short range wireless communications such as via a Bluetooth® radio (all not shown).

[0023] Programming instructions that implement the functional teachings of client machine **102**A as described herein are typically maintained, persistently, in non-volatile storage unit **212** and used by processor **208** which makes appropriate utilization of volatile storage **216** during the execution of such programming instructions. Of particular note is that non-volatile storage unit **212** persistently maintains a web browser application **86** and, in the present embodiment, a native menu application **82**, each of which can be executed on processor

208 making use of non-volatile storage 216 as appropriate. An operating system and various other applications (not shown) are maintained in non-volatile storage unit 212 according to the desired configuration and functioning of client machine 102A, one specific non-limiting example of which is a contact manager application (also known as an address book, not shown) which stores a list of contacts, addresses and phone numbers of interest to user U and allows user U to view, update, and delete those contacts, as well as providing user U an option to initiate telecommunications (e.g. telephone, email, instant message (IM), short message service (SMS)) directly from that contact manager application.

[0024] Native menu application 82 may be configured to provide menu choices to user U according to the particular application (or other context) that is being accessed. By way of example, while user U is activating the contact manager application, user U can activate menu application 82 to access one or more menu choices available that are respective to contact manger application 90. For example, menu choices may include options to invoke other applications (e.g. a mapping application to map a contact's address) or communication functions (e.g. call, SMS, IM, email, etc.) on the client machine 102A for a particular contact. Menu application 82 may be associated to a particular input button (e.g. one of buttons 200) and invoked to provide a contextual menu comprised of one or more menu choices that are reflective of the context in which the button 200 was selected. Note that the options in a contextual menu are stored within non-volatile storage 212 as being specifically associated with a respective application. Menu application 82 may be therefore configured to generate one or more different contextual menus that are reflective of the particular context in which the menu application 82 is invoked. For example, in an email application where an email is being composed, invoking menu application 82 would generate a contextual menu that included the options of sending the email, cancelling the email, adding addresses to the email, adding attachments, and the like. The contents for such a contextual menu would also be maintained in non-volatile storage 212. Other examples of contextual menus will occur to those of ordinary skill in the art.

[0025] FIGS. 8A-8D and 9A-9D respectively illustrate representative web pages rendered on a first browser window and portions of a subset of data from said representative web pages transcoded and rendered on a second browser window in accordance with an embodiment. FIG. 8A illustrates a representative home web page 860A of an e-commerce web site (e.g. 104) in a browser window 850. Window 850 is illustrative of a rendering to a large size display device (e.g. desktop monitor). Web page 860A comprises, among other things, a menu portion 852 and a primary content display portion 854, in the example, showing various advertisements 855 for products. FIG. 9A illustrates the menu portion 852 extracted and transcoded and rendered as a web page on a second browser window 950. Window 950 is illustrative of a rendering to a small size display device (e.g. of a wireless mobile device). In addition to transcoding as a web page, menu portion 852 may be transcoded for menu application 82 e.g. for invocation when browsing the site 104 as referenced further herein.

[0026] FIG. 8B illustrates an exemplary product web page 860B in window 850 showing various product data (collectively 866) including image 866A, price 866, title 866C and description 866D data that is transcoded and shown in window 950 of FIG. 9B. Also transcoded is the web page hierarchy list 868 showing where the page is on the web site.

[0027] FIG. 8C illustrates an exemplary product list web page 860C in window 850 showing a list of products (collectively 870). A subset of the product data such as image 870A, price 870B, and title 870C is transcoded and shown in window 950 of FIG. 9C. Note that multiple pages 872 may be provided for the list 870.

[0028] FIG. 8D illustrates an exemplary account checkout web page 860D in window 850 showing a login form 880 for receiving account login and password, which form is transcoded and shown in window 950 of FIG. 9D. Though not shown, other checkout pages (e.g. for payment or order confirmation, etc.), search pages, product and information pages may be similarly transcoded.

[0029] Returning now to FIG. 1, web servers 106, 107 or 109 and gateway and schema server 120 (which can, if desired, be implemented on a single server) can be based on any commonly available server environments or platforms including a module that houses one or more central processing units, volatile memory (e.g. random access memory), persistent memory (e.g. hard disk devices) and network interfaces to allow servers 106, 107, 109 and 120 to communicate over the telecommunications network. Web servers hosts software applications comprising instructions and data for generating and serving web pages dynamically from the template families 108 and current informational content therefore from data store 112. Load balancing, security/firewall, billing, account and other applications may also be present as is well-known in the art.

[0030] Gateway and schema server 120 hosts software applications comprising instructions and data for proxying requests and responses between the client machines 102 and web sites 103, 104 and 105. In addition to software for maintaining HTTP communications, performing requests, maintaining sessions, handling cookies, etc., engine 140 may be implemented in software to apply the signature schemas to web pages from web sites. There may be provided an interpreter that interprets the signature schema document and applies the actions against the web page code (as an ASCII (plain text) document) to extract desired data to produce a result set. A renderer may be provided to express the desired data result set (i.e. transcode to a target format such as cHTML (Compact HTML) for a mobile device browser) for transmitting to the client machines also in accordance with the signature schema.

[0031] The web server 125 provides web pages to the requesting client machine through a browser or application on the client for rendering. The web data may be directly pushed to client machines 102A by e-mail or by other push based applications, or the data may be accessed by queries to web site 150 directly. The web site 150 may also extract content from the aggregate database 126 and apply signature schema 122 to the extracted database data, which schema may be configured to transcode the data in accordance with the target client machine 102A to tailor the output result.

[0032] Machines 102, schema server 120 and web site 104 are coupled via a telecommunication network (not shown) typically comprising one or more interconnected networks that may include wired and (at least for machine 102A) wireless networks. It should now be understood that the nature of the network is not particularly limited and is, in general, based on any combination of architectures that will support interactions between client machines 102 and servers 106, 107,

109, **125** and **120**. In a present embodiment the network includes the Internet as well as appropriate gateways and backhauls.

[0033] More specifically, in the present embodiment, a wireless network for client machine **102**A may be based on core mobile network infrastructure (e.g. Global System for Mobile communications ("GSM"); Code Division Multiple Access ("CDMA"), Enhanced Data rates for GSM Evolution ("EDGE"), Evolution Data-Optimized ("EV-DO"), High Speed Downlink Packet Access ("HSPDA"), Universal Mobile Telecommunications System ("UMTS"), etc.) or on wireless local area network ("WLAN") infrastructures such as the Institute for Electrical and Electronic Engineers ("IEEE") 802.11 Standard (and its variants) or Bluetooth or the like or hybrids thereof. In the present embodiment of system **100** it is contemplated that client machine **102**B may be another type of client machine such as a PC (desktop or laptop) configured to include a full desktop computer or as a "thin-client". Typically such have larger display monitors/screens than portable machines like **102**A. A wired network for system **100** and machine **102**B can be based on a T1, T3 or any other suitable wired connection.

[0034] As previously stated in relation to FIGS. **1** and **2**, each of the client machines **102** is configured to interact with content available over the network, including web pages on web site **104**. In a present embodiment, client machines **102**A and **102**B may navigate for content using a browser application (e.g. **86**). As will be explained further below, on client machine **102**A, browser application **86** may be a mini-browser in the sense that it may be configured to render web pages on the relatively small display **224** of client machine **102**A. Often, during such rendering, those pages are presented in a format that may be different from how those pages are rendered on a traditional desktop browser application (e.g. browser **86** of client machine **102**B). Mini-browsers typically attempt to convey substantially the same information as if the web pages had been rendered on a full browser such as Internet Explorer®, Safari® or Firefox® on a traditional desktop or laptop computer like client machine **102**B.

[0035] FIG. **3** is a flowchart illustrating operations/interactions for generating and maintaining an aggregate database from web sites **103**-**105** for populating and updating database **126**. The flowchart provides an example of the interaction among the gateway and schema server **120**/**140** and data collection engine **152** with the web servers **106**, **107** and **109** hosting web sites **103**, **104** and **105** to generate and maintain the aggregate database **126**. The data collection engine **152** (DCE), makes a request **302** to the web site's web server (for example web server **106**) for the specified pages based upon the type of data to be aggregated. The web page code (e.g. **110**) is generated by the server **106** and sent **306** to DCE **152**. The web page code received is a text file. It typically does not include objects referenced by the code such as images, video, audio, further web pages, etc. that are typically subsequently retrieved and inserted at the time of rendering a web page by a browser. The schema server engine **140** (SSE) (for example, in parallel or without waiting for a response from server **106**) makes a request **304** to the signature repository **124** for the signature schema document **122** for the web site, which request may use the domain in the URL as an identifier for obtaining the document **122**. The schema server engine **140** receives **308** the schema and does not render the web page **110** per se but instead uses the instructions in the signature schema document **122** to extract the desired data from the web page **110**. The signature schema **122** is configured to extract data from web page **110** in accordance with the specific desired content characteristics for the database **126** which is based upon the target client machines **102**, and having knowledge of display **224** capabilities—such as screen size, resolution, and other parameters—useful in determining the way in which the transcoded data is to be displayed on the machine **102**A. The web page **110** or extracted data is stored **310** in database **126** in a relational database structure, where data related to the defined signature schema from the web site is stored.

[0036] Client machine **102**A can then make a request **312** web site **150** on server **125**. The request may be to perform a search for information such as product information provided by web sites **103**, **104** and **105**, to establish price watches such as product price watch notifications, or request status of an existing price watch. Web site **150** requests **314** relevant data from the database **126**, or in the case of a price watch populates the appropriate tables to enable monitoring and notification. The results are extracted **316** and are sent **318** to the client machine as aggregate results. For providing a price watch function, a confirmation or status can be sent in accordance with the schema **122**, to the requesting client machine **102**A processed by the signature schema engine **140** before presentation by web server **125**.

[0037] Alternatively, the client machine **102**A may send a request **312** for configuration of a push based notification service to pull data from the database **126** and display the data on the client machine **102**A through a push based application. The web server would provide confirmation **318** of configuration of the push web service. As noted above, transcoded data **130** may comprise transcoded navigational data for menu application **82** and informational content data (e.g. a list of products and related information from a web page) for displaying by browser application **86**.

[0038] Signature schemas are pre-defined documents, and may be eXtensible Markup Language (XML) documents utilizing an SQL-like query language, to incorporate instructions and data with which to intelligently extract the data from web pages (which web pages are typically coded in HTML, DHTML, XHTML, XML, RSS, Javascript, etc). This extracted data may be transcoded and provided to client machines **102**, used to dynamically generate a relational database (e.g. **126**) or both. Each signature schema incorporates an understanding of a particular web site's data including relationships among the various data (e.g. among its primary informational content found in the body of its web pages as well as among such content and associated navigational data (e.g. web page links) that govern the data in the page. As described further herein below, prior knowledge of the web page code including specific identifiers, tags and text (i.e. strings) used within the code (sometimes referred to as "signatures" herein), may be used to define instructions to identify portions of the code of interest and to extract specific desired data.

[0039] A signature schema document may be defined for all the pages of a particular web site. Large data-driven web sites (e.g. **104**) don't maintain thousands of individual web pages per se. The sites typically adopt a few page family templates **108** and dynamically populate these with pertinent content from database **112** comprising information (e.g. weather, stock data, news, shopping/product data, patent data, trademark data etc.) as applicable when a client requests a particular page. Each template represents a family of pages having objects and attributes. Below are representative example page

family templates and their objects and attributes for a web site offering news and an e-commerce web site offering products for sale electronically:

EXAMPLE 1

News Site

[0040] Family: List Page
[0041] Objects: lists a selection of news stories
[0042] Attributes: Title, abstract and date
[0043] Family: Detail page
[0044] Objects: lists a single news story (and optionally other related stories)
[0045] Attributes: Journalist, City, Date, Title, Full Story, Image

EXAMPLE 2

E-Commerce Site

[0046] Family: List Page
[0047] Objects: lists a selection of products
[0048] Attributes: Image, Item Name, Price, Sale Price
[0049] Family: Search Page (a specific kind of list page)
[0050] Objects: Similar to a list page
[0051] Attributes: Similar to a list page
[0052] Each family of pages (the family template) can be identified by a "signature" or unique set of one or more features that automatically identifies a given page on a web site as part of the family and differentiates that family from another family of pages. Similarly each object and attribute field of interest can be identified with its respective unique signature within a family of pages. A signature schema document typically comprise numerous pieces of information (commands), for example, information that instructs the engine **140** for:

[0053] identifying all page families;
[0054] identifying and extracting data (i.e. desired objects and attributes) for each page family;
[0055] capturing the (implicit or explicit) relationships between the objects and attributes; and
[0056] transcoding the data.

[0057] A signature schema document may also be configured to enable special functionality for the target web site including searching, logging in a user, purchasing items, etc.

[0058] In accordance with a present embodiment, the structure and syntax of a representative signature schema document for a representative e-commerce site eshop.ca is shown and described. Engine **140** may be configured to receive web page code comprising text data and search through the text in accordance with the schema document instructions that provide SQL-query like language instructions. Engine **140** maintains a pointer within the text as it moves through the web page code performing various actions, as described below, in accordance with the schema instructions. Table 1 illustrates a snippet of a representative signature schema:

TABLE 1

XML Signature Schema Snippet for E-Shop.ca

```
1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <site>
3       <version major="1" minor="2"/>
4       <url location="http://www.eshop.ca" key="eshop.ca" name="E-Shop" />
5       <advanced>
6
7             <index_link value="http://www.eshop.ca/home.asp" />
8       </advanced>
9       <page_type>
10            <lookup type="pex" action="locate_string" name="list_elements"
id="mylist_1"
                  ref="Compare products" alt1="Sort products" />
11            <lookup type="pex" action="locate_string" name="item_elements"
id="myitem_1"                 ref="&amp;quot;product-details&amp;quot;" />
12            <lookup type="pex" action="locate_string" name="menu_elements"
id="mymenu_2"                 ref="anc-lhsnav-subItem" />
13            <lookup type="pex" action="locate_string" name="menu_elements"
id="mymenu_1"                 ref="product-table" />
14            <lookup type="pex" action="locate_string" name="item_elements"
id="myitem_1"                 ref="*" />
15      </page_type>
16      <list_elements id="mylist_1">
...
17      </list_elements>
...
18      <item_elements id="myitem_1">
19            <actions>
20                  <lookup type="pex" action="move_ptr" ref="&amp;lt;/head&amp;gt;" />
21            </actions>
22            <element>
23                  <lookup type="pex" action="get_string" name="image"
ref="largeimageref"                 location="after" start="&amp;lt;img
src=&amp;quot;" end="&amp;quot;" />
24                  <lookup type="pex" action="get_string" name="title" ref="product-
details-prd-title"                 location="after" start="&amp;lt;span"
end="&amp;lt;/span&amp;gt;" include_sz="1"             strip_tags="1" />
25                  <lookup type="pex" action="get_string" name="price" ref="our
price:"                 location="after" start="&amp;lt;td"
```

TABLE 1-continued

XML Signature Schema Snippet for E-Shop.ca

```
end="&lt;/td&gt;" include_sz="1"  strip_tags="1" />
26  <lookup type="pex" action="get_string" name="sale_price"
ref="sale price:"  location="after" start="&lt;td"
end="&lt;/td&gt;" include_sz="1"  strip_tags="1"
tolerance="1" />
27  <lookup type="pex" action="get_string" name="description"
ref="detailbox-text"  location ="middle" start="&lt;p"
end="&lt;/p&gt;" include_sz="1"  strip_tags="1" />
28  </element>
29  </item_elements>
...
```

[0059] In the XML code snippet of Table 1, instructions at line 4 are for verifying that the web page under consideration and the signature schema relate to the same web site/domain—eshop.ca. Instructions at lines 9-15 are for determining the particular page family to which the web page under consideration belongs. A respective signature that defines the particular page family has been previously identified for use to distinguish the page. The engine **140** processes the <page type> tag by registering the identification strings for each page family. When a web page is obtained by the engine as input, the engine may be able to identify the page family by its unique string ref=" and the command provides the related tag within the signature schema document where further instructions for the particular web pages are found:

[0060] action="locate-string": command to check for the existence of a string.

[0061] name=": identifies the type of page family for each identified family.

[0062] id=": assigns an id to the page family that is used across the signature schema document.

[0063] For example, at line **10**, the instructions identify a web page using the signatures "Compare products" or "Sort Products". Web pages with these strings are of the same family type. The instructions at line 10 provide a reference tag to further instructions for this family, providing a link to instructions for the list_elements page family with and ID of mylist_1 (see lines 16-17). Similarly the other lookup instructions provide references to the specific instructions within the signature schema document for handling a web page of each web page family. Representative instructions for some of the web page families are provided in Table 1, for example, at lines 16-17 and 18-29 with others omitted for brevity.

[0064] With reference to the extraction instructions for one of the web page families (e.g. item_elements id="myitem_1") at lines 18-29, the instruction at line 20 advances the scan pointer within the text file of the web page code to a beginning limit of a region of interest indicated by a signature reference. This establishes an upper limit for review within the text file. Though not shown in this table, an end limit may be defined as well (See Table 4). Further such instructions at lines 22-28 may comprise commands to locate desired data using "signatures" such as string identifiers that uniquely identify the data within the region of interest. In the present example the instructions locate and extract one or more elements, namely, product image, title, price, sale price and description for a product of the item web page family. For example, instructions at line 23 extract a string in between the first "&It;img src="" and """ that appears after next appearance of "largeimageref". The string returned is the path (relative URL at web site eshop.ca) to the product image. By advancing a search scan pointer within the web code to a desired location, references before that location can be skipped when searching. Any prior instances of a signature string such as "largeimageref" may be ignored. In this way, otherwise ambiguous signature references can be avoided.

[0065] The example in Table 1 shows at least some of the instructions (e.g. lines 23-27) including one or more directional references relative to the signatures to locate and extract the desired data. For example, directional references such as "before" or "after" command the engine to extract desired data that is in a relative position in the web page before or after the signature string (i.e. ref=). Moreover, such instructions may further include at least one of a start reference or an end reference further pinpointing the location of the desired data in accordance with that direction. Additional directional reference information is discussed herein with reference to code snippets in other Tables and the discussion of an embodiment of signature transcoding engine syntax presented below.

[0066] The example within Table 1 demonstrates the extraction of data and the establishment of relationships between objects and elements within a same page of a web site. However, signature schema documents may further capture relevant attributes of an object across pages. For example, a user of client machine **102**A may click through a number of web pages in eshop.ca to get to a specific product page (e.g. Department→Product Category→Product Sub-Category→Specific Product, such as TV & Video>19"-21" TVs>LCD TVs>BrandX Product. The navigational hierarchy representing a categorization may be captured and associated to the extracted objects and there elements.

[0067] For brevity, certain instructions were omitted from Table 1. Tables 2-4 provide representative instructions for further web page families for e-shop.ca that may be read with Table 1. Table 2 below provides representative instructions, e.g. for lines 16 and 17 of Table 1, including instructions for a web page family related to a list of items/products for sale. Whereas instructions at lines 22-28 provided product data extraction instructions for a web page family showing a single item (i.e. product), the instructions of Table 2 provide additional instructions that repeat product data extractions for each product in the list.

TABLE 2

XML Signature Schema Snippet for Product List Web Page Family of E-Shop.ca

```
1  <list_elements id="mylist_1">
2    <paging>
3      <page_variable value="page" />
4      <page_start value="0" />
5      <lookup type="pex" action="get_string" name="link"
ref="Next&amp;nbsp;" location="before" start="&lt;a
class=" end="&lt;/a&gt;" include_sz="1" strip_tags="1" />
6    </paging>
7    <actions>
8      <lookup type="pex" action="move_ptr" ref="Sort or compare
products" ref_alt_1="Sort products" />
9    </actions>
10   <element>
11     <lookup type="pex" action="get_string" name="link"
ref="thumbnail" location="before"
start="&lt;ahref=&quot;"
end="&quot;&gt;" />
12     <lookup type="pex" action="get_string" name="image"
ref="thumbnail" location="middle" start="&quot;"
end="&quot;" />
13     <lookup type="pex" action="get_string" name="title"
ref="class=&quot;tx-strong-dgrey&amp;quot;"
location="after" start="&lt;a href="
end="&lt;/a&gt;" include_sz="1" strip_tags="1" />
14     <lookup type="pex" action="get_string" name="price"
ref="pricepill/" location="after"
start="/" repeat_start="1"
end=".gif" tolerance="1" />
15     <lookup type="pex" action="move_ptr" ref="pricepill/" />
16   </element>
17 </list_elements>
```

[0068] If the engine **140** identifies that the page is of the "mylist-1" family, the engine determines the location in the signature schema document that contains the signature for the objects and elements of that family and applies the instructions therefor. A product list at e-shop.ca may span multiple web pages. Instructions at lines 2-6 of Table 2 find the number of pages and generate the links for each of the pages. Instructions at lines 7-9 (action tag) advance the search scan pointer to the region of web page code that may be of interest (i.e. in this case, the start of the list). In this way, a local signature reference can be used and any earlier ambiguous references skipped. Skipping to the local region of interest may also make the specification of the signature reference less complicated.

[0069] Taking advantage of inherent repeated patterns in the web page code, instructions at lines 10-16 (elements tag) of Table 2 provide product data extraction instructions that may be repeated for each product in the list. The engine **140** may be provided with commands to scan for each data element of interest using a signature reference e.g. ref='', an action, one or more positional instruction(s) to further identify the data within the text of the web page code, and any additional text data manipulation instructions to extract the desired data (e.g. to remove HTML formatting characters or add characters). The instruction at line 15 moves the scan pointer to the end of the object (in this example a product in a list of products) to ready the instructions for application against the next object (product) in the list.

[0070] More particularly:

[0071] lookup type="pex": string lookup

[0072] action="get-string": returns a value back that is the desired element of the object.

[0073] name="link": the object element, in this case the link to the product page

[0074] ref="thumbnail": the reference string that identifies where to find the value of the link

[0075] location="before": the value of the link is before the ref string

[0076] start="<a href="": look for the ref string after this value

[0077] end="">": look for the ref string before this value.

TABLE 3

E-Shop Search Family Signature Schema Snippet

```
1 <search_elements id="mysearch_1">
2   <settings>
3     <search_path value="http://www.eshop.ca/search/search.asp" />
4     <search_variable value="keyword" />
5   </settings>
6   <paging>
7     <page_variable value="page" />
8     <page_start value="0" />
9     <lookup type="pex" action="get_string" name="link"
        ref="Next&amp;nbsp" location="before" start="&lt;a href="
        repeat_start="1"
end="&lt;/a&gt;"
        include_sz="1" strip_tags="1" />
10  </paging>
11  <actions>
12    <lookup type="pex" action="move_ptr" ref="bg-compare-hero" />
13  </actions>
14  <element>
15    <lookup type="pex" action="get_string" name="link" ref="&gt;"
location="after"
        start="&lt;a href=&quot;" end="&quot;&gt;" />
16    <lookup type="pex" action="get_string" name="image"
ref="&lt;a href" location="after"
        start="&lt;img src=&quot;" end="&quot;" />
17    <lookup type="pex" action="get_string" name="title"
        ref="class=&quot;tx-strong-dgrey&amp;quot;" location="after"
start="&lt;a
        href=" end="&lt;/a&gt;" include_sz="1" strip_tags="1" />
18    <lookup type="pex" action="move_ptr" ref="bg-compare-hero" />
19  </element>
20 </search_elements>
```

[0078] If the engine **140** has identified that the page is of the "mysearch_1" family the engine applies the portion of the signature schema document that contains the signature for the objects and elements of that family, shown above in Table 3.

[0079] <settings> . . . </settings>: Contains any web page specific manual overrides such as excluding certain menu items, customization, modification of a menu that may be desired. In this example, as per line 3 a value of form variable "keyword" will be posted to "http://www.eshop.ca/search/search.asp".

[0080] <paging> . . . </paging>: Manages paging for the search pages.

[0081] <actions> . . . </actions>: Instruct the engine to move the scan pointer to the string "bg-compare-hero" (line 12 of Table 3) and start looking for elements from there.

[0082] <element> . . . </element>: Contains lookup instructions for each object element as previously described.

TABLE 4

E-shop Menu Family Signature Schema Snippet

```
1 <menu_elements id="mymenu_1">
2    <settings>
3       <black_list value="Site Index##External Link" />
4    </settings>
5    <actions>
6       <lookup type="pex" action="move_ptr" ref="bg-lhsnav-title" />
7       <lookup type="pex" action="end_ptr" ref="&lt;/table&gt;" />
8    </actions>
9    <element>
10      <lookup type="pex" action="get_string" name="link"
ref="&lt;li&gt;" location="after"
            start="&lt;a href=&quot;" end="&quot;" />
11      <lookup type="pex" action="get_string" name="title"
ref="&lt;li&gt;" location="after"
          start="&lt;a href=&quot;" end="&lt;/a&gt;" include_sz="1"
strip_tags="1" />
12      <lookup type="pex" action="move_ptr" ref="&lt;/li&gt;"/>
13   </element>
14 </menu_elements>
```

[0083] If the engine **140** has identified that it is looking for a menu on a page that contains the menu style of the "mymenu_1" family, the engine applies the portion of the signature schema document that contains the signature for the objects and elements of that family, shown above in Table 4.

[0084] <settings> . . . </settings>: Contains any page specific manual overrides such as exclude list, customization, modification, personalization, etc. In this example, as per line 3, any result that matches "Site Index", "External Link" are excluded but partial matches are also possible by using wild card strings.

[0085] <action> . . . </action>: Lines 6-7 of Table 4 sets the start and end limits to instruct the engine **140** where to look for menu items.

[0086] <element> . . . </element>: Contains lookup instructions for each object element as previously described. In this example, lines 10 and 11 of Table 4, an element in 'mymenu_1' (each individual menu entry of web page) contains link and title as its properties. Line 12 instructs the engine to move the pointer to "&It;/li>" to get ready to loop through and extract the next menu item with the same elements, taking advantage of the repeated patterns within the text of the web page code.

[0087] Though the example described relates to extracting informational content for an e-commerce oriented site, no limitation should be applied. Similar instructions may be defined for other types of sites, for pages which permit a user to input information and for navigational data extraction.

[0088] Signature schema document **122** may further comprise transcoding instructions (not shown) for use by engine **140** to express the extracted desired data (e.g. which may be retrieved from database **126**) in a target format (e.g. a format of HTML, XML, script etc.) for use by the requesting client machine **102**. For example, the transcoding instructions may define a web page for displaying the extracted data in browser application **86** that is suitable for display on the client machine **102**. The formatting rules can be system and/or user defined and can include parameters such as but not limited to: object positioning, object colour, object size, object shape, object font/image characteristics, background style, and navigational item display (e.g. in a menu as described above) or for display with the content in the generated page on the client screen. Browser application **86** (e.g. of machine **102**A) may be configured for using a different markup language (e.g. cHTML) or other code format that is not identical to the code provided by web page **110**. Alternatively, transcoding instructions may be defined to express the extracted desired data in XML or another code format such as for use by a different client application or plug-in to a client application such as menu application **82** or another application (not shown) on client machine **102**.

[0089] Signature schema documents may be prepared (i.e. coded) using a computing device such as computing device **128**. Computing device **128** may be any suitable desktop or laptop device capable of coding documents (which may be but need not be XML-type documents) and may be configured to automate or semi-automate coding of such documents.

[0090] Computing device **128** may be coupled to web site **104** to retrieve web pages from the site for reviewing to prepare the custom signature schema document for the site. Computing device **128** may be configured to automatically review the web page code and apply heuristics or other techniques (e.g. spatial analysis) to determine probable content of interest (i.e. desired data) and generate code to extract the desired data. For example, primary content of interest tends to be located toward the centre of the web page. In another embodiment, the computing device may facilitate a user coding signature schema to manually assist with the analysis of the web page and identification of desired data and the generation of the instructions. Computing device **128** may be further coupled to repository **124** to provide (e.g. up-load or publish) coded signature schema documents for use by server **120**.

[0091] It will be apparent to a person of ordinary skill in the art that as a web site may be re-designed or otherwise changed such that the code of one or more web page families may be changed or a family added, an existing signature schema may require re-coding to account for the change/addition, as applicable.

Signature (Transcoding) Engine Syntax

[0092] In accordance with a present embodiment, further details concerning the syntax of schema instructions is described.

Lookup Syntax

[0093] The lookup tag instructs the engine **140** to perform an insert, delete or query the document contents.

[0094] Type: Defines the data type of the lookup. Type may be "pex" for a string expression. Type may also support more advanced options such as regular expressions, API calls, and SQL queries.

Action:

[0095] Action="locate_string": Look for a string ("ref" identifier") value within the data. Return true iff the string exists in the data (i.e. the "ref" identifier index>=0).

[0096] Action="replace_string": Replace a string within the data with the "ref" identifier.

[0097] Action="move_ptr": Remove all characters in the data that exist before the location of the "ref" identifier.

[0098] Action="end_ptr": Remove all characters in the data that exist after the location of the "ref" identifier.

[0099] Action="get-string" Extract a string based on the location of the "ref", "start", and "end" identifiers.

[0100] ID: ID is an identifier of another section within the signature. It allows the result of a query to trigger another set of actions within the signature. This is primarily used when identifying page types. Once a match has been made, specific instructions are executed that are marked with this ID. Recursive data structures (e.g. lists within lists) may also be supported.

[0101] Ref: Ref defines the initial identifier that the lookup searches for. If an AND case is required multiple ref identifiers can be used (i.e. ref="string1" ref1="string2"). If an OR case is required ref_[ref identifier]_alt_1 can be used (i.e. ref="string1" ref_alt_1="string2"). To demonstrate (X="1" || Y="2") && (A="8" || B="9") would translate to ref="1" ref_alt_1="2" ref1="8" ref1_alt_1="9".

[0102] Repeat_[identifier]: Repeat executes the identifier query additional times. For example, if ref="hello" to set the identifier index at the second occurrence of hello the following tag would be added: repeat_ref="1".

Location:

[0103] Location="before": Search the data in a reverse direction, starting from the "ref" identifier. This implies that both the "start" and "end" identifier indexes must be less than the "ref" index.

[0104] Location="middle": Search the data in two directions, starting from the "ref" identifier. This implies that the "ref" identifier index is greater than the "start" identifier index and less than the "end" identifier index.

[0105] Location="after": Search the data in a forward direction, starting from the "ref" identifier. This implies that both the "start" and "end" identifier indexes must be greater than the "ref" index.

[0106] Start: Start is primarily used when action="get_string" and may also be used for replace/remove instructions. The start identifier index will be the start index of the string to extract. If an AND case is required multiple "start" identifiers can be used (i.e. start="string1" start1="string2"). If an OR case is required start_[start identifier]_alt_1 can be used (i.e. start="string1" start_alt_1="string2"). To demonstrate (X="1" || Y="2") && (A="8" || B="9") would translate to start="1" start_alt_1="2" start1="8" start1_alt_1="9". To find the n^{th} match see the repeat syntax.

[0107] End: End is primarily used when action="get_string" and may also be used for replace/remove instructions. The end identifier index will be the end index of the string to extract. If an AND case is required multiple "end" identifiers can be used (i.e. end="string1" end1="string2"). If an OR case is required end_[end identifier]_alt_1 can be used (i.e. end="string1" end_alt_1="string2"). To demonstrate (X="1" || Y="2") && (A="8" || B="9") would translate to end="1" end_alt_1="2" end1="8" end1_alt_1="9". To find the n^{th} match see the repeat syntax

[0108] Max_index: Max_index is used to limit the scope of a query by ensuring that no other identifier index is greater than the "max_index" . . . If an AND case is required multiple "max_index" identifiers can be used (i.e. max_index="string1" max_index1="string2"). If an OR case is required max_index_[max_index identifier]_alt_1 can be used (i.e. max_index="string1" max_index_alt_1="string2"). To demonstrate (X="1" || Y="2") && (A="8" || B="9") would translate to max_index="1" max_index alt_1="2" max_index="8" max_index_alt_1="9". To find the n^{th} match see the repeat syntax.

[0109] Max_Index_Use_Ref: Max_Index_Use_Ref is a Boolean value set to 0 or 1. It is used with Max_index. When set to 0, the "max_index" will begin querying at the beginning of the data. When set to 1, the "max_index" will begin querying from the "ref" identifier index.

[0110] Gbl_append_[identifier]: Gbl_append appends a string passed via the url to the identifiers query value

[0111] Gbl_Repeat_[identifier]: Gbl_Repeat executes the identifier query additional times. For example, if ref="hello" to set the identifier index at the second occurrence of hello the following tag would be added: gbl_repeat_ref="var" where var would be passed in the URL i.e. http://www.eshop.ca/mobile/fatfree.asp?site=...&url=...&var=1.

[0112] Tolerance: Tolerance is a Boolean value set to 0 or 1. It is used to return an empty string. By default tolerance is set to 0 which enforces that a property be found on a page, otherwise the page will be marked as "invalid" and an appropriate error message returned. When set to one, an empty value is returned for properties that can not be located.

[0113] Include_sz: Include_sz is a Boolean value set to 0 or 1 and used with get_string. It is by default set to 0. When set to 1 it includes the "start" value and the "end" value as part of the result.

[0114] Include_start: Include_start is a Boolean value set to 0 or 1 and used with get_string. It is by default set to 0. When set to 1 it includes the "start" value as part of the result.

[0115] Include_end: Include_end is a Boolean value set to 0 or 1 and used with get string. It is by default set to 0. When set to 1 it includes the "end" value as part of the result.

[0116] Closetag: Closetag is a Boolean value set to 0 or 1 and used when action="get_string". It appends />to the extracted value.

[0117] Strip_Tags: Strip_Tags removes HTML tags from the value and used when action="get_string".

[0118] Strip_tags="1": remove all tags.

[0119] Strip_tags="2": remove all br and script tags.

[0120] Strip_tags="3": remove all tags except replace </p> with
.

[0121] Strip_tags="4": remove all tags except replace
 with
.

[0122] Strip_tags="tag1,tag2 . . . tagN": remove all tag**1**, tag**2**, . . . tagN leaving any tag not listed.

[0123] Notrim: Notrim is a Boolean value set to 0 or 1 and used when action="get_string". By default all value have white spaced trimmed. When this property is set to 1, white space is not trimmed.

[0124] Append: Append is a string value and used when action="get_string". It appends a string to the extracted value.

[0125] Prepend: Prepend is a string value and used when action="get_string". It prepends a string to the extracted value.

[0126] Upper: Upper is a Boolean value set to 0 or 1 and used when action="get_string". It converts all characters to upper case.

[0127] Lower: Lower is a Boolean value set to 0 or 1 and used when action="get_string". It converts all characters to lower case.

Page Syntax

[0128] The page syntax extracts the paging information from the data. This allows the end user the ability to change pages just as on the desktop.

[0129] Page_variable: Defines unique key that defines a family's paging feature.

[0130] Page_start: Defines value of first page in a family's paging feature.

[0131] Page_post: Path where paging variable(s) must be transmitted to.

[0132] Page_start :Defines value of first page in a family's paging feature.

[0133] Page_increment: Defines value that paging increases by for each page in a family's paging feature.

[0134] Page_block: Defines unique key that defines a family's paging block feature.

[0135] Page_block_size: Defines the size of the family's page block. (i.e. 10 items per page)

[0136] Url_append: Append the unique key that defines a family's paging feature and the page number.

Search Syntax

[0137] Make a web site family's search feature functional by specifying details such as what variable to post.

[0138] Search_path: Search path where search variable must be transmitted to

[0139] Search_variable: Name of search variable which a web site's search feature is looking to read, request, post, etc.

[0140] Url_replace: Remove a portion of the url that is specific to posting search parameters

URL Syntax

[0141] The url tag defines global properties for a site, including the url, and name:

[0142] <url location="http://www.eshop.ca" key="eshop.ca" name="E-Shop" />

[0143] Name: Name is the name to display when browsing using the gateway **120**

[0144] Location: Location defines the fully qualified address of the site.

[0145] Key: Key is the site.

Advanced Syntax

[0146] The advanced tag defines global properties for the site. This at a minimum includes the path to the initial page of the site.

```
<advanced>
    <index_link value="http://www.eshop.ca" />
    <check_out value="1" />
</advanced>
```

[0147] Index_link: Index_link specifies the path to the initial page of the site. This is usually the same page as the location property from the URL syntax. This field is always required.

[0148] Append_link: Appends a string value to every URL requested for this site.

[0149] No_purchase: No_purchase is a Boolean value 0 or 1. The default value is 0 which implies that an item should contain a purchase link. When true, the purchase link is removed.

[0150] No_item: No_item is a Boolean value 0 or 1. The default value is 0 which implies that Item pages should show up in the breadcrumb. When true, the item is not added to the breadcrumb.

[0151] Check_out: Check_out is a Boolean value 0 or 1. The default value is 0 which implies that Item purchase link sends the request and control away from the gateway server **120**. When true, then a checkout process has been created for use with gateway server **120**.

[0152] Product_img_width: Product_img_width defines the width of all item images.

[0153] Use_cookies: Use_cookies a Boolean value 0 or 1. By default it is set to 0, and cookies are not passed to the site. When true, gateway **120** passes all cookies from client machine **102** to the site **104**, and from the site **104** to the client machine.

Page Type Syntax

[0154] The page type is a collection of lookup queries that have an id associated with them. Lookup queries may be processed in a top down fashion. The first successful lookup will trigger another section in the signature schema document. For example, if the following evaluates to true:

```
<page_type>
<lookup type="pex" action="locate_string" name="list_elements"
id="mylist_1" ref="&lt;!--" />
</page_type>
```

[0155] Then the tag element <list_elements id="mylist_1"> would be executed next.

General Element Syntax

[0156] Elements include list_elements, menu_elements, item_elements, search_elements, form_elements. Each element has an ID. For example a menu element:

[0157] <menu_element id="menu_id"/>

[0158] The element may contain the following sub containers (settings, actions, elements, paging) which scope resides only within the element. Each element is associated with a specific rendering function.

```
<menu_element id="menu_id"/>
    <settings> </settings>
    <paging> </ paging >
    <elements> </ elements >
        <actions> </ actions >
</menu_element>
```

Settings Syntax

[0159] Settings syntax varies based on the type of element it resides in. Settings allow customizations that only apply to a specific page family.

[0160] Black_list—menu_elements: Black_list removes menu items with names that reside in the black list. Each entry is separated delimited (e.g. using two pound characters

[0161] Pass_image—list_elements, search_elements: Pass_image adds the image path to the url when requesting an item. The image added to the url will be used as the item image.

[0162] Price[n]—item_elements: Price[n] where n is an integer renames the rendered item with name price[n].

[0163] Action—form_elements: Overrides the action of a form displayed to the end user.

[0164] Handle—form_elements

[0165] Handle="display"—display the form to the end user.

[0166] Handle="post"—post the form.

[0167] Handle="get"—get the form.

[0168] Cookie—form_elements: Send additional cookies when posting this form.

[0169] Input_[identifier]—form_elements: Input tag adds/modifies a form value with name [identifier] setting its value.

[0170] Rename_[identifier]—form_elements: Rename tag renames a form value with name [identifier].

Actions Syntax

[0171] The actions tag primary function is data manipulation. It contains lookup queries that modify data with actions of "move_ptr" or "end_ptr".

```
<actions>
    <lookup type="pex" action="move_ptr" ref="&lt;/head&gt;" />
</actions>
```

[0172] Persons of ordinary skill in the art will appreciate that alternative embodiments are contemplated. System **100** may be implemented so that one or more web sites **103**, **104**, and **105** are coupled to the telecommunication network (either alone by a server **106** or by one or more web servers like web-server **106**), and that a corresponding one or more schemas for each of those web sites (or each of the web pages therein, or both) can be maintained by gateway and schema server **120** and repository **124**. Client machines **102** can be configured for proxied connections through different servers **120** and for accessing aggregated web site data from database **126**. Those skilled in the art will now further recognize that servers **120** and web server **125** can be hosted by a variety of different parties, including, for example but without limitation: a manufacturer of client machine **102**, a service provider that provides access to the telecommunication network on behalf of user U of a client machine **102**; the entity that hosts web-site **104** or a third party intermediary. In web site host example it can even be desired to simply combine the web server **106** and schema server engine **120** on a single server to thereby obviate the need for separate servers. Alternatively the functionality of server **120** and web server **125** may be locally resident on the client machine providing.

[0173] FIG. **4** is a schematic representation of an aggregate web site search database **126**. The database **126** contains one or more tables for storing aggregate web page data extracted from target web sites. The database may be a relational database enabling structured database queries. For an e-commerce web site, the database **126** may contain tables defining a web site identification and category index **402** and category data **404** containing specific item details, as will be discussed in connection with FIG. **5**. The category data **404** may be further broken down into product data **406** table if further data is required. Indices **408** can be created to reference aggregate web site data to improve query responsiveness and results. Metadata **410** associated with the original web pages may be stored such as images, web page formatting on navigation data. A price watch table **412** is provided to identify the originating client machine and items that are of interest and should be monitored by the engine **140** for changes as specified. A user table **414** may also be defined to store user or device specific information such as email or device identification information.

[0174] FIG. **5** is a schematic representation of tables in an aggregate web site search database for e-commerce. The extracted data from selected e-commerce web sites can be formatted into category index **402** which identifies **502** the web site where the data was extracted from and the category **504** of interest. In this example entry **506** identifies http://www.eshop2.ca/pg=3 as the reference for where digital cameras can be indexed. For each category in the category index **402**, an individual category **404** table can be created. The category table provides location identifiers **510** for each product retrieved from the web site based upon the defined signature schema and populated at step **310** during the retrieval of data from the web site. The vendor **512**, title **514** of the product, price **516** and description **518** extracted can then be stored. It should be understood that the categories identified can be tailored to the application. The database then provides a means for querying the aggregate data from the web site to present meaningful information to the client machine **102**A. By storing the aggregate data, queries can be created for price watch monitoring to meet desired requirements and transcoded for presentation on the client machine **102**A. As the schema is defined to extract elements with which objects and their attributes on the web page can be defined or described and the schema incorporates knowledge of what these objects and attributes represent, an intelligent database **126** can be defined.

[0175] FIG. **6** is a schematic representation of a price watch table **412**. The table may be configured to define a user ID **610** or client machine ID to which the price watch notification is to be provided. The item to price watch is identified **612**. If the item is defined generically a product vendor **614** or manufacturer can be identified. A price **616** threshold may be defined on which notification will be provided. The suppliers **618** that are to be monitored may then be identified based upon their web site addresses, alternatively if no suppliers are identified, only suppliers' data that is part of the aggregate database will be monitored. For example, entry **622** defines for User **1**, the pricing of product A is to be monitored, the product is manufactured by vendor A and the user wished to be notified when the price change occurs based upon a define threshold, for example, the price drops below $100 at the 3 supplier web sites identified. In other embodiments the entry may identify that notification should occur if a price change or sale happens for the identified product. The engine **140** would use this information to generate queries to the database periodically to determine if the price threshold has been met at which time a notification would occur to the associated client machine **102**A. The price watch table may also only identify an item and a price threshold, enabling any vendor and supplier to be

identified when the threshold is met or exceeded. Base upon the application, data may stored to define how the user will be notified, such as by push application, a transcoded web page or email etc.

[0176] FIG. **7** illustrates a method of providing an aggregate web site price watch. A price watch request is received **702** and the price watch table **412** entry is created based upon the price watch information such as item, price, vendor and user or device id. A price watch entry may then be created to generate price watch queries **704**. The engine **140** then periodically queries the aggregate database tables for changes **706** to determine if the threshold as defined by the query is met. Periodic queries may be generated by any number of methods such as, for example, on a define time interval or by notification messages from the web site that changes have occurred. Engine **140** then requests the appropriate pages from the website to update the aggregate database **126**. As part of the monitoring process the engine **140** must request web page data from the web site at predetermined intervals to ensure data is up to date. If the price threshold has not been met or exceeded, NO at **708**, monitoring continues at **706**. If the price threshold has been met or exceeded, YES at **708**, pricing details are retrieved **710** from the aggregate database tables. If additional web site related content is required, YES at **712**, such as pictures or web site related navigation, metadata can be retrieved at step **714**, otherwise no addition content, NO at **712**, is required. A notification can then be sent **716** to the client providing the retrieved data by email message or push application. Alternatively, the retrieved data can be generated using html the client machine **102**A and sent as a web page to the client machine or may produce transcoded version of the web page for deliver to the mobile.

[0177] The ability to pull aggregate data from identified e-commerce sites improves the efficiency and accuracy of price watching reducing reliance on RSS feeds from web site provider. In an embodiment, the data collection engine **152**, signature schema engine **140** and database **126** may be implemented directly on the client machine **102**A. The client machine **140** would be directly query the relevant web sites and maintain an aggregate database locally. Notification of price watch notification would then be provided locally to the client machine **102**A.

[0178] Further, it is recognized that no network calls may be required when retrieving data from the aggregate database **126**. For example, pre-caching to the aggregate database **126** could happen at the beginning of day via the user desktop and then synched to the mobile via a wired connection (for example), so that user can surf pre-cached information offline. For example, using signature schema to grab content based on pre-caching criteria, this could be used to generate the pre-cache database. For providing a price watch or price comparison functionality, the engine **140** can crawl the selected web site and build a comparison pricing information to make available to the public or other subscribers through the aggregate database **126**.

1. A method of providing aggregate web site price watch feature, the method comprising:

receiving a price watch request for a product from a client machine comprising a threshold price for the product;

generating a query to an aggregate database for the product;

retrieving data from the aggregate database containing information provided from one or more web sites, extracted by parsing received web site data by an associated signature schema;

determining when the threshold price has been satisfied;

retrieving one or more product details from the aggregate database when the threshold price has been satisfied; and

notifying the client machine that the price threshold was satisfied.

2. The method of claim **1** wherein notifying the client machine further comprises:

generating hyper-text markup language (HTML) data message from the retrieved one or more product details based upon the signature schema associated with the particular web site from which the product details originated; and

sending the generated HTML data message to the client machine.

3. The method of claim **1** wherein notifying the client machine further comprises pushing the one or more product details to a push application on the client machine.

4. The method of claim **1** wherein notifying the client machine further comprises sending an email to the client machine comprising the one or more product details.

5. The method of claim **1** further comprising:

periodically monitoring the one or more websites for changes in a product price associated with the price threshold.

6. The method of claim **1** where in the threshold price is satisfied when a product price is equal to or less than the threshold price.

7. The method of claim **1** wherein the data is stored in a relational database.

8. The method of claim **7** wherein the query is a Structured Query Language (SQL) query.

9. The method of claim **8** wherein the signature schema comprises eXtensible Markup Language (XML) documents comprising query language for extracting data from web pages.

10. The method of claim **1** wherein the aggregate database is stored on the client machine.

11. The method of claim **10** wherein the client machine periodically queries the one or more web sites to update data in the aggregate database.

12. The method of claim **1** wherein the client machine is a wireless device.

13. A system for providing aggregate web site database price watch feature comprising:

at least one computing device comprising a processor and a memory coupled thereto, said memory storing instructions and data for configuring the processor to provide an engine to:

receive a price watch request for a product from a client machine comprising a threshold price;

generate a query to an aggregate web site database for the requested product;

retrieve data from the aggregate database containing information provided from one or more web sites extracted by parsing received web site data by and associate signature schema;

determine if the threshold price is satisfied;

retrieve one ore more product details from the aggregate database when the product price threshold has been satisfied; and

notify the client machine that the price threshold was satisfied.

14. The system of claim **13** wherein when the engine notifies the client machine by providing hyper-text markup language (HTML) data generated from the retrieved details based upon a signature schema associated with the particular web site from which the product details originated.

15. The system of claim **13** wherein the processor is further configured to provide an engine to periodically monitoring a product price associated with the price threshold until the threshold price is satisfied.

16. The system of claim **13** wherein the price threshold is satisfied when the threshold price is met or exceeded.

17. The system of claim **13** wherein the data is stored in a relational database.

18. The system of claim **17** wherein the query is a Structured Query Language (SQL) query.

19. The system of claim **13** wherein when the one ore more product details are retrieved, additional metadata associated with the product is retrieved from the aggregate database.

20. The system of claim **13** wherein the signature schema comprises eXtensible Markup Language (XML) documents comprising query language for extracting data from web pages.

21. The system of claim **12** wherein the database comprises a table identifying an hypertext transport protocol (HTTP) link and a product category and a table identifying an product specific HTTP link, product information and pricing for each identified product category.

22. The system of claim **1** wherein the aggregate database is stored on the client machine.

23. The system of claim **1** wherein the client machine is a wireless device.

24. A computer program product storing computer readable instructions which when executed by a computer processor configure the processor for:

receiving a price watch request for a product from a client machine comprising a threshold price for the product;

generating a query to an aggregate database for the product;

retrieving data from the aggregate database containing information provided from one or more web sites, extracted by parsing received web site data by an associated signature schema;

determining when the threshold price has been satisfied;

retrieving one or more product details from the aggregate database when the threshold price has been satisfied; and

notifying the client machine that the price threshold was satisfied.

25. A method of providing aggregate web site price watch feature for a wireless device, the method comprising:

receiving a price watch request for a product from a client machine comprising a threshold price for the product;

generating a query to an aggregate database for the requested product;

retrieving data from the aggregate database containing information provided from one or more web sites, extracted by parsing received web site data by an associated signature wherein the schema comprises an extensible Markup Language (XML) documents comprising query language for extracting data from the web page;

determining when the threshold price has been satisfied;

periodically monitoring the one or more websites for changes in the product price;

retrieving one ore more product details from the aggregate database when the product threshold price has been satisfied; and

notifying the wireless device that the price threshold was satisfied.

* * * * *