Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2573516 A1 2008/07/08

(21) 2 573 516

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2007/01/08

(41) Mise à la disp. pub./Open to Public Insp.: 2008/07/08

(51) Cl.Int./Int.Cl. *G06Q 30/00* (2006.01)

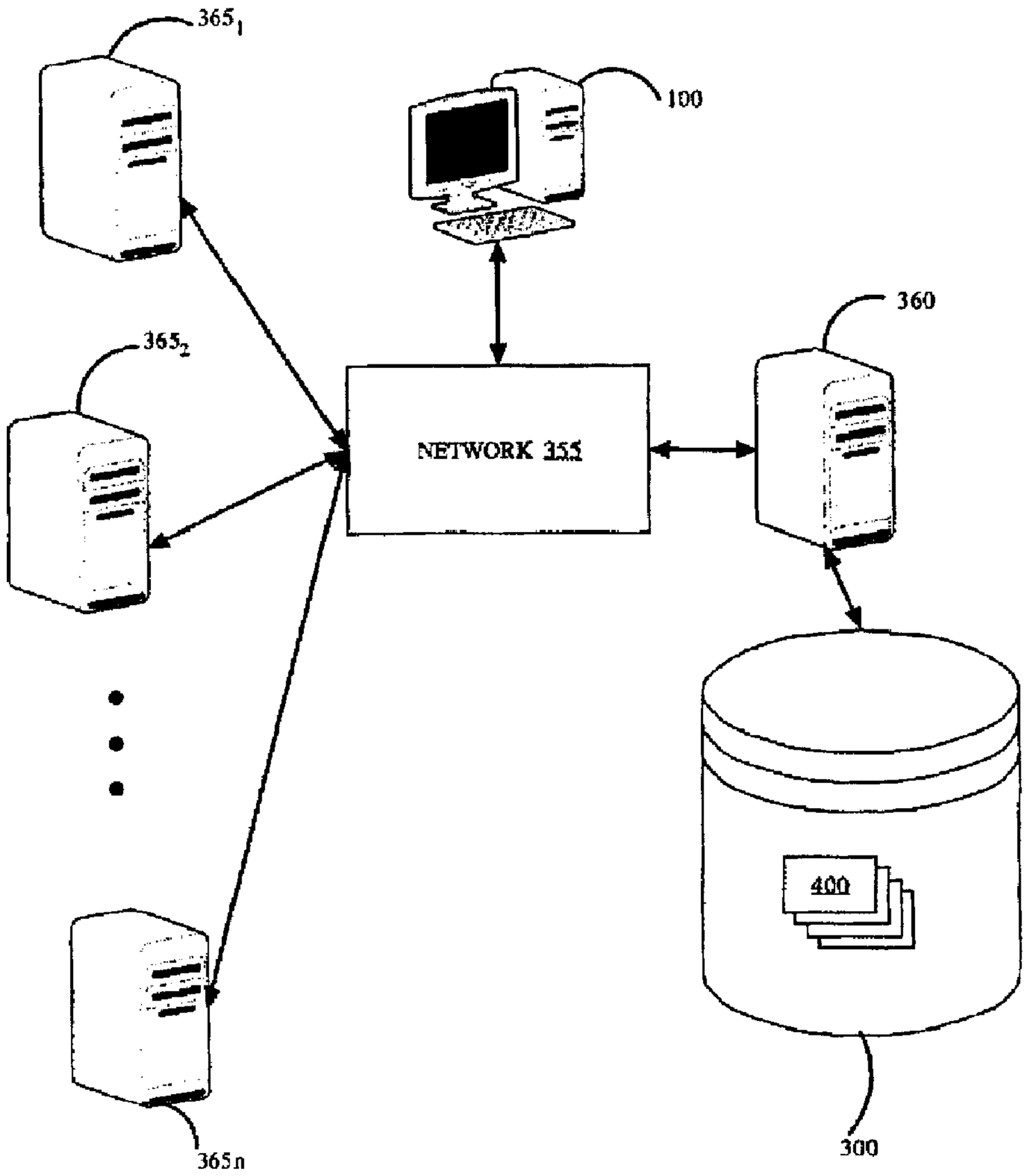
(71) Demandeur/Applicant:
MCCANN, DANIEL, CA

(72) Inventeur/Inventor:
MCCANN, DANIEL, CA

(74) Agent: FURMAN & KALLIO

(54) Titre : AUTOMATISATION DES OPERATIONS ET SAISIE COTE CLIENT DE L'INFORMATION PORTANT SUR LE SCHEMA DE FORMULAIRES

(54) Title: TRANSACTION AUTOMATION AND CLIENT-SIDE CAPTURE OF FORM SCHEMA INFORMATION



TRANSACTION AUTOMATION AND CLIENT-SIDE CAPTURE OF FORM SCHEMA INFORMATION

This invention is in the field of uniform interfaces for automating the submission of information through a series of web pages in a transaction sequence.

BACKGROUND

Many users now use the Internet to shop online. A user navigates to an online shopping website and can browse items for sale using a web browser. When the user finds an item he or she would like to purchase, they identify the item to the website, (typically by adding to their "virtual shopping cart") and when they are ready to purchase it they go through a transaction sequence in order to finalize the sale. This transaction sequence is commonly a series of web pages where the user is prompted for information the seller needs in order to complete the sale. At each step (web page) in the transaction sequence, a user is prompted for specific information, which the user must enter in the input fields provided on the web page and then the user must submit the information in one of a number of different way (clicking a next, a submit button, etc.) in order to move on to the next web page in the transaction sequence. To complete the transaction sequence successfully, the user provides the requested information and navigates through the series

of web pages until the sale is made and the owner of the website prepares to ship the purchased articles to the user.

Much of the information requested by a website during a transaction sequence will be the same across most if not all online shopping websites. For example, for most, if not all transaction sequences on any number of online shopping websites, the website requests a name of the user, shipping address, billing information, etc. While many of these websites request the same information from a user, each website is free to request this information in any order they would like and the steps and web pages that must be navigated for a successful transaction can be as varied as the websites themselves. There is no standard or uniform way to implement a transaction sequence for a website that all online shopping websites use but transactions sequences will vary from website to website.

Websites can vary on what information is asked for in what stages of a transaction sequence and also by the text label the website assigns to requested information. For example, one website may ask for a shipping address on a web page early in the transaction sequence, while another website may leave it to a web page near the very end of the transaction sequence. Additionally, websites might vary in the labels they assign input. What a website labels an input of information is completely up to the programmers of the website and often information will be called by different names by

different websites. For example, one website might label the first name of a user as FIRST_NAME, while another might label it CUSTOMER_FNAME. Each website is free to establish their own proprietary system for requesting information from a user as part of a transaction sequence. Websites do not have to ask for the necessary information to complete a sale or transaction in a uniform manner, but can do it in almost any sequence or way.

Users or other programs interacting with web sites will not be presented with the same sequence of requests for information from different web sites and transaction sequences might be very different from website to website. While this may be an inconvenience for a user, sometimes making it hard for a user to navigate through transactions sequences on different websites, it makes it incredibly hard for other programs that are unable to asses the information being prompted for and supply their own. When a user is manually navigating through the web pages in a transaction sequence, with each page in the transaction sequence prompting the user for the information it requires at that stage and then having the user enter the information entered in the provided fields, it may be inconvenient and confusing but it is not overly difficult or in some cases impossible for a user to use the textual identification of the various input fields to determine what information should be entered on a webpage and where. However, this becomes a real problem when trying to automate the sequence transaction so that it is done with either minimal or no user input. While for a user it may not seem complex; for a computer, this

is not a simple operation and there are numerous variations between websites on how they want to request the information.

There are prior programs that allow the automatic completion of forms on a specific webpage. However, they require a user to first navigate to the web page and then to manually submit the web page or navigate to the web page. The information requested in the form is then mapped to information known by the program and the necessary information inserted in the correct forms. While these provide some automation, they merely allow the inputting of information on a specific web page, they do not allow a whole transaction sequence consisting of a series of web pages to be fully automated on a number of different websites. A user still sees the original web page and must manually navigate through the transaction sequence itself.

SUMMARY OF THE INVENTION

A method and apparatus for providing a uniform interface to a website in order to perform a transaction sequence over a series of web pages is provided. A database containing a plurality of page objects is provided where each page object corresponds to a web page in a transaction on one of a number of different websites.

To fully or partially automate a transaction sequence, an interface application operates in conjunction with a web browser. As a user views a web page that is the start of a transaction sequence on a website, the interface application uses the URL of the web page to locate a page object in the database corresponding to the web page. The page object is then verified against the current web page before it is used by the interface application to complete the necessary input requests on the web page. Once the input requests on the web page have been provided with input, the page object is used to determine how to navigate to the next step in the transaction sequence and this next step (web page) in the transaction sequence is navigated to. In this manner, the interface application keeps matching page objects to each web page it encounters and navigating through a series of web pages in the transaction sequence until the transaction sequence has been successfully completed.

The interface application can be used to fully automate the transaction sequence or it can be used to provide a uniform interface between the webpage and either a user or another application program.

The database of page objects can be populated manually, by passively monitoring a user as he or she manually completes a transaction sequence on a website or by dynamically creating the page objects using an intelligent agent.

DESCRIPTION OF THE DRAWINGS

While the invention is claimed in the concluding portions hereof, preferred embodiments are provided in the accompanying detailed description which may be best understood in conjunction with the accompanying diagrams where like parts in each of the several diagrams are labeled with like numbers, and where:

Fig. 1 is schematic illustration of a data processing system;

Fig. 2 is a schematic illustration of a program module in a memory storage device of the data processing system of Fig. 1 containing a web browser application and an interface application;

Fig. 3 is a schematic illustration of a network configuration in accordance with the present invention;

Fig. 4 is a schematic illustration of a data structure of a page object;

Fig. 5 is a state diagram of an interface application as it navigates through a transaction sequence;

Fig. 6 is a schematic illustration of a program module in a memory storage device of the data processing system of Fig. 1 containing a web browser application, an interface application and an overlaying application;

Fig. 7 is a schematic illustration of another aspect of a network configuration comprising a card reader operatively connected to a data processing system;

Fig. 8 is a flowchart of a method for populating a database with page objects by passively monitoring a user manually completing a transaction sequence; and

Fig. 9 is a flowchart of a method for dynamically populating a database with page objects.

DETAILED DESCRIPTION OF THE ILLUSTRATED EMBODIMENTS

Fig. 100 illustrates a data processing system 100 suitable for supporting the operation of methods in accordance with the present invention. The data processing system 100 typically comprises: at least one processing unit 103; a memory storage device 104; at least one input device 105; a display device 106 and a program module 108.

The processing unit 103 can be any processor that is typically known in the art with the capacity to run the program and is operatively coupled to the memory storage device 4

through a system bus. In some circumstances the data processing system 100 may contain more than one processing unit 103. The memory storage device 104 is operative to store data and can be any storage device that is known in the art, such as a local hard-disk, etc. and can include local memory employed during actual execution of the program code, bulk storage, and cache memories for providing temporary storage. Additionally, the memory storage device 104 can be a database that is external to the data processing system 100 but operatively coupled to the data processing system 100.

The input device 105 can be any suitable device suitable for inputting data into the data processing system 100, such as a keyboard, mouse or data port such as a network connection and is operatively coupled to the processing unit 103 and operative to allow the processing unit 103 to receive information from the input device 105. The display device 106 is a CRT, LCD monitor, etc. operatively coupled to the data processing system 100 and operative to display information. The display device 106 could be a stand-alone screen or if the data processing system 100 is a mobile device, the display device 106 could be integrated into a casing containing the processing unit 103 and the memory storage device 104.

The program module 108 is stored in the memory storage device 104 and operative to provide instructions to processing unit 103 and the processing unit 103 is responsive to the instructions from the program module 108.

Although other internal components of the data processing system 100 are not illustrated, it will be understood by those of ordinary skill in the art that only the components of the data processing system 1 necessary for an understanding of the present invention are illustrated and that many more components and interconnections between them are well known and can be used.

Furthermore, the invention can take the form of a computer readable medium having recorded thereon statements and instructions for execution by a data processing system 1. For the purposes of this description, a computer readable medium can be any apparatus that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device.

Fig. 2 illustrates a schematic illustration of a program module 108 in a memory storage device 104 of the data processing system 100. The data processing system 100 is operative to implement and run a browser application 220 over top of an operating system 210 and an interface application 250 runs over top of and in conjunction with the browser program 210. The browser application 220 could be any known in the art, such as Microsoft's Internet Explorer™, Mozilla Firefox™, Apple Safari™, Netscape Navigator™ or Opera™.

The interface application 250 has access to a user database 260 containing information related to a particular user such as a name of the user, a shipping address, preferences,

etc. The user database 260 is shown in Fig. 2 as being located on the memory storage device 104, but a person skilled in the art will understand that it can be located in another memory storage device operably connected to the data processing system 100.

Fig. 3 illustrates a network configuration wherein the data processing system 100 is connected over a network 355, such as the internet, to a database 300 and a plurality of content servers 365_1 to 365_N.

A plurality of content servers 365_1 to 365_N are configured to act as web servers and provide data and media content, generally although not necessarily in the form of websites containing web pages, to the data processing system 100. The data processing system 100 can access any of the content servers 365 to view web pages contained on the content servers 365. The data processing system 100 uses the browser application 220 to access any of the content servers 365_1 to 365_N, which are web servers and the content accessed on any of the content servers 365 are generally files in a markup language which the browser application 220 displays as a web site and web pages on the data processing system 100.

The database 300 contains metadata about a number of different websites and is operatively connected the data processing system 100 through the network 355. By using this stored metadata data, processing system 100 can automate a transaction sequence

comprising a series of web pages on a website and in a further aspect automatically inserting the proper information to complete the transaction successfully.

The database 300 contains a plurality of pages objects 400, wherein each page object 400 corresponds to a particular web page on a website. Fig. 4 illustrates a data model of a page object 400 in accordance with one aspect. Each stored page object 400 contains information about field names and corresponding information types, form architecture, error conditions, and other useful information required to be able to successfully automate through a transaction sequence of web pages on a website. Each page object 400 contains metadata both about the web page itself and the inputs of information requested on the web page. Typically each page object 400 comprises: a base URL field 410; a transaction sequence identifier 420; a unique identifier 430; a path identifier 440; information 450; optionally validation formatting information 470; and a navigation field 470

The base URL field 410 contains a URL of the page a specific page object 400 corresponds to but typically stripped of all URL parameters (i.e. as per the HTTP spec). The URL field 410 is used as a first means of identifying the web page that a specific page object 400 corresponds to.

The unique ID field 420, the transaction sequence ID field 430 and the path ID field 440 are all used as a secondary means to verify that a particular page object 400 corresponds

to a specific web page that is currently being viewed as a step in a transaction sequence. The unique identifier field 420 stores a value that identifies what inputs are on the web page, in order to provide an additional criteria with which the proper corresponding web page can be verified. The value stored in the unique identifier field 420 is used as a way to differentiate web pages that share the same URL. In a DHTML environment, the contents of a web page may be generated dynamically using the same URL. This means that although the URL might be the same, different inputs might be requested depending on what step of a transaction sequence a web page is being displayed at. By using a unique identifier in addition to the URL of the web page being viewed, web pages using the same URL, yet generated dynamically and asking for different inputs, can be differentiated.

In one aspect, the value stored in the unique ID field 420 is a hash value that is generated using the input names on the web page being viewed. A hash value for the input names of the web page being viewed is generated and this hash value is compared to a hash value stored in the unique ID field 420. By generating the hash value using the input names of the web page, only another web pages with those same input names will result in the same hash value. If the hash value determined for a web page being viewed does not match the hash value stored in the unique ID field 420, the page object 400 does not correspond with the web page being viewed even though the URL may be the same.

The transaction sequence ID field 430 stores an identifier of the location of the page in a transaction sequence. Certain websites require that a user visit the same web page in the website multiple times during a transaction sequence. For example, some websites use same-page postbacks, validation or simply a "state" web page. Sometimes these web pages can not be differentiates using the unique identifier stored in the unique identifier field 420 of the page object 400 because the web page may have the same input names, however, the user is required to insert different information or form submissions than the previous time and failure to differentiate those events will not result in a successful transaction. Alternatively, the input fields may be shown on the page with information previously provided with the user as a way of confirming the information and allowing the user to change the information if necessary, however, the web page must be exited in a different manner than previously in order to carry on to the next web page in the transaction sequence.

Optionally, a path ID field 440 holds a path identifier that can be used to determine one of three potential paths: new account; previous account logged in; and previous account not logged in. Often a user will have to navigate a series of web pages and enter certain information on these web pages based on whether they have an account with the website. This might involve the user having providing more information or less information depending on which of the paths the user must follow. For example, a new user that has never used a particular website may have to provide more information than a user that is

logged in as a repeat user of a website. The value in the path identifier field 440 is used to determine whether the metadata is appropriate for a user.

Information fields 450 provide information about the user-interactive form elements on the web page and contains metadata both about the page itself and the inputs on the page and may contain a plurality of different entries. The information fields 450 provide all of the information necessary for the interface application 250 to input the necessary information into the web page to successfully fill out the form on the web page corresponding to the page object 400. The information fields 450 can take a number of forms. The information fields 450 can provide an indication of the mandatory inputs on the web page that must be completed in order for the web page to be successfully completed. Often, a web page will request inputs that are not strictly required for the page to be successfully completed and the transaction will be successful even if certain inputs are not provided. The information fields 450 can identify which inputs are mandatory and which are optional. Also the information fields 450 can provide mapping information that allows the interface application 250 to recognize what the information being requested by the web page is and to match the requested information to information the interface application 250 has access to in the user database 260. For example, the information fields 450 may contain a mapping of the name of the user name called USER_FNAME by the web page input tag to how it is identified in the user database 260, FIRST_NAME. This information tells the interface application 250 that the input being requested by the web page USER_FNAME corresponds to FIRST_NAME (or how

it is identified by the interface application 250). Additionally, this mapping might also include further information specifying how the input should be formatted in cases the web page requires the input information to be stored in a different way from how it is stored in the user database. In some cases an input on the website might map to a field in the user database 260, however the user database 260 might provide the information in a different format than the input expects it. For example, an input on a web page might map to a field in the user database 260 that contains a salutation identifier and the value stored in this field may be a string of "Mr". However, the webpage requires an integer between 1 and 4 to represent the type of salutation that is appropriate, with the number 3 corresponding to "Mr". The information fields 450 would therefore identify the values that the string "Mr" corresponds to so that an acceptable value (in this case 3) for the input can be provided to the web page.

This information fields 450 also contain any information on formatting that is needed to properly enter the requested input. For example, the web page may ask for a date to be inserted in three fields, one for day, month and year, however, the interface application 250 stores the date as an eight (8) digit number.

During a transaction, the interface application 250 may encounter an input which may be required for a transaction, but does not correspond to any information contained in user database 260 (for example an input prompting a user to adjust the quantity of items in their shopping cart). The information fields 450 may contain information on how to

handle such an input in order to complete the transaction. For example, the information field 450 may contain information for this input that the input should be populated by the interface application 250 with a default value, or ignored.

In this manner, the information fields 450 provide all the instructions and information necessary for the interface application 250 to provide all the requested inputs of the web page corresponding to the page object 400, allowing the interface application 250 to successfully complete the form on the web page.

Optionally, the page object 400 and/or information fields 450 may contain validation formatting information 470 that allows the input submitted to the page to be validated or formatted correctly before the web page is submitted.

Finally, the page object 400 comprises a navigation field 470 that indicates the next action that has to be carried out in order to navigate to the next web page in the transaction sequence. This navigation field 470 comprises at least one of: the next page URI/URL; or what is necessary to submit the form on the current web page. For example, the name of the input that causes the form submission and subsequent navigation to the next page, and what kind of event triggers the form submission (user click, key press, etc.) The interface application 250 uses the navigation field 470 to determine how to move to the next web page in the transaction sequence after it has

provided all the inputs necessary to successfully complete the form on the current web page.

USING THE PAGE OBJECTS TO NAVIGATE A TRANSACTION SEQUENCE

By using the page objects 400 stored in the database 300 the automation of a transaction sequence consisting of a series of web pages on a website can be automatically navigated by the interface application 250. The interface application 250 can provide a generic interface for a user or other program to any number of different websites. Instead of merely providing a means to automatically complete forms on a single web page, the interface application 250 is able to automatically navigate a series of web pages in a transaction sequence without a user manually intervening to navigate between web pages in the transaction sequence.

Fig. 5 illustrates a state diagram of the interface application 250 as it navigates through a transaction sequence. As a user navigates through web pages on the content servers 255, interface application 250 is in state 510 and is intercepting the URL of each web page viewed by the browser application 210. The interface application 250 then tries to match the intercepted URL to a page object 400 stored in the database 300.

Referring to Fig. 4, the interface application 250 could match the base URL of a page being viewed directly to the URL field 410 of a page object 400 or the interface could download a file periodically from the database 300 that can be stored on a data processing system 100 that identifies page objects 400 in the database 300 without requiring the data processing system 100 to communicate with the database 300 each time the user views a web page using the data processing 100.

The interface application 250 continues to intercept the URL of the web pages being viewed by the web browser 210 and when the interface application 250 finds a URL that matches a page object 400 in the database 300, the interface application 250 enters state 520. In state 520, the interface application 250 communicates with database 300 and obtains a page object 400 having a URL in the URL field 410 that matches the current page being viewed.

Once the interface application 250 obtains the page object 400, the interface application 250 enters state 530 where it checks the page object 400 against the web page being viewed to verify whether or not the page object 400 truly reflects the web page that is currently being viewed. This process involves a number of checks. The unique identifier in the unique ID field 420 of the page object 400 is checked against the web page being viewed to verify that the web page being viewed is the web page represented by the page object 400. In one aspect a hash of the input names on the web page is made and compared to the hash value stored in the unique identifier field 420 of the page object

400. The transaction sequence ID in the transaction sequence ID field 430 is also checked to verify the web page being viewed matches the web page represented by the page object 400. If the path ID field 430 is used, the path ID stored in the path ID field 430 is also checked to make sure the information 450 in the page object 400 is for the proper path the interface application 250 is following.

If the interface application 250 cannot verify the page object 400 against the web page being viewed at state 530, the interface application 250 moves back into state 520 and tries to obtain another page object 400 with a URL contained in the URL field 410 matching the base URL of the web page being viewed. When the interface application 250 finds another page object 400 that contains the same base URL in the URL field 410 of the page object, the interface application 250 will once again move into state 530 and once again attempt to verify the newly found page object 400 against the web page being viewed.

Once the interface application 250 finds a page object 400 that it can verify matches the web page being viewed. the interface application 250 enters state 540 where it inserts that information required for the web page, as instructed by the information fields 450 in the page object 400. For example, the interface application 250 will map the inputs requested by the web page, using the information fields 450, to the information the interface application 250 is aware of in the user database 260 to allow the interface application 250 to submit the requested inputs to the web page.

Once the inputs have been provided to the web page as specified by the information contained in the information fields 450 of the page object 400, application interface 250 enters state 550 if a validation information field 460 is provided. In state 550, the application interface 250 uses the information in this field to validate the information provided to the inputs before moving on to state 560.

Once the interface application 250 has provided the necessary requested inputs to the current web page by using the instructions provided in the information fields 450 of the page object 400 to insert the proper information in the proper fields of the web page, the interface application 250 moves into state 560 and the navigation field 460 of the page object 400 is used by the interface application 250 to navigate to the next web page in the transaction sequence of the website.

At this point the interface application 250 is again in original state 510 and simply intercepts the URL of the web page being viewed and again attempts to match it to a page object 400 in the database 300. The interface application 250 does not need to have pre-existing knowledge of all the steps in the transaction sequence, it merely matches each web page to a page object 400 as it encounters each web page in the transaction sequence; follows the instructions in the information fields 450 to provide the requested input for the web page; and then navigates to the next web page, using the navigation field 470 to instruct it how to navigate to the next step (web page) in the transaction

sequence. Because the previous web page was identified as a step in a transaction sequence and the present web page being viewed was arrived at by following the instructions in the navigation field 460 of the page object 400 corresponding with the previous web page that was viewed, the next web page is almost certainly the next step in a transaction sequence. The interface application 250 does not need to have comprehensive knowledge of the previous web page, it simply continues to intercepts the URL of the web pages being viewed by the web browser application 210 and matching them to their corresponding page object 400 in that database 300. In this manner, the interface application 250 can navigate a series of web pages in a transaction sequence one by one completing the form requested on each web page and moving to the next web page in the transaction sequence until the transaction sequence is complete.

The interface application 250 can be used to completely automate the completion of a transaction sequence for a website, providing all of the information as needed as it is requested on a web page by the website. However, the interface application 250 can also serve as a uniform interface between a user or another program application to allow a user or the other program application to provide uniform inputs to the interface application 250 which are then used to conduct a transaction sequence on a number of different websites, irregardless of the format and order the website is requesting the information.

In one aspect a uniform user interface is provided by the interface application 250 and displayed to a user so that, although the user is requested to provide information, the user interface is the same for any number of different websites the user is accessing. In this manner, a user can always see a familiar user interface rather than having to figure out each a different transaction sequence for each new website he or she visits.

In another aspect, the interface application 250 allows another program application to interact with the website. Fig. 6 illustrates a schematic illustration of a program module 108 in a memory storage device 104 of the data processing system 100 wherein the application interface 250 is acting as a generic interface to the browser application 220 for an overlaying application 600. Overlaying application 600 must interact with a transaction sequence on a number of website, but rather than requiring any specifics knowledge of the various websites, the overlaying application 600 has uniform inputs and outputs to the interface application 250 that allows the interface application 250 to navigate transaction sequences on a number of different websites.

Fig. 7 illustrates an aspect of the overlaying application 600 where it communicates with a card reader 700 that is operably connected to the data processing system 100 running the overlaying application 600. When a page object 400 is obtained by the interface application 250 wherein the information field 450 informs the interface application 250 that at least some of the inputs on the web page require the input of a credit card number and the corresponding information, the interface application 250 communicates with the

overlaying application 600. The overlaying application 700 in turn communicates with the card reader 700 and prompts the user to swipe his or her credit card using the card reader 700. The overlaying application 700 receives the credit card information (either unencrypted or encrypted) from the card reader 700 that was generated from the user swiping his or her credit card in the card reader 700. This information is then passed to the interface application 250 to be submitted to the website in the proper input field.

Without the interface application 250 operating using the above disclosed methods, the overlaying application 600 would have no idea when the card reader 600 should be used, nor how the information from the card reader 600 should be used.

DATABASE POPULATION

The page objects 400 in the database 300 can be populated in a number of ways. The easiest method to implement is to have a person manually go through the website and enter the correct information needed for each page object 400. This could be done by a person manually editing the database 300, but in a further aspect involves a specially designed user interface that a user will look at while they navigate a website and enter in the information as they go along. When a user navigates to a web page that does not have a corresponding page object 400 in the database 300, the user will be presented with an interface and the user will classify: which inputs are required for a successful completion of the web page; the appropriate values to go into the inputs or the class of information that should go into the inputs; any special formatting required for a given

input; the path they are using (e.g. new user, etc.); and the method of navigating to the next step (i.e. typically form submission).

With this information, a page object 400 for the web page can be created and stored in the database 300.

PASSIVE TRANSACTION MONITORING

The database can also be populated by passively monitoring a user as he or she manually navigates a series of web pages on a website to complete a transaction sequence. By passively monitoring and examining the interaction of a user with a specific website, the necessary information need to create the corresponding page objects 400 can be obtained.

Fig. 8 illustrates a flowchart of method 800 for creating a page object 400 by passively monitoring a user as he or she completes a web page that is a step in a transaction sequence on a website. The method 800 is typically implemented by the interface application 250 running on the data processing device 100 operated by the user and comprises the steps of: determining web page identifiers 805; determining a context for each input 810; applying the determined context to create metadata for the input 820; examining user interaction with web pages 830; resolving ambiguities 840; comparing context data to examined data 850; checking to see if the context data and examined data

are in accordance 860; resolving the inaccordance 870; determining an exit method 880 and saving the meta data as a page object 890.

Method 800 begins when a user navigates to a web page on a website that is part of a transaction sequence. At step 805, the method 800 determines a set of web page identifiers for the web page currently being viewed by the user. These page identifiers will include the URL, the unique identifier, transaction sequence identifier and optionally the path identifier, which will be stored in the URL field 410, unique ID field 420, transaction sequence ID field 430, and optionally the path ID field 440 of the eventual page object 400 created for the web page, respectively.

At step 810, the method 800 determines the context for each requested input on the web page. This context could include the unique identifiers that the web page uses to tag the input, the label text for the input field, the adjacent text, location of the input field relative to other elements on the pages, etc.

At step 820, the context for each input on the web page is used to create metadata about the input. By analyzing the metadata created from the determined context, information about they requested inputs can be determined. For example, certain information requests are typically grouped with other information, such as last name and first name in the same area and street address is typically followed by the city that the user lives in. This metadata is used to determine a number of characteristics about the inputs such as:

what class of information the field should contain (labeled as an address, etc.); if any value needs to be inserted into the input to successfully execute a transaction (whether the context marks it as a required input); if the value is unique to the transaction or consistent across all transactions; any formatting required by the input field (e.g. a phone number) or if the field contains multipart data (i.e. a phone number with an area code).

At step 830, the method 800 then examines how the user interacts with the web page. Each input that is entered or modified by the user is recorded and this user input is used to create a mapping between an element the user interacted with and the type of information required by that input. The interface application 250 will examine the information input by the user into an element in the web page and try to match it to information available about the user in the user database 260.

The interface application 250 with its access to the user database 260 containing user information is used to map the input fields in step 830. This user information will be information about a user that is commonly requested by a website and typically comprises a first name of the user, a shipping address, card number, etc., however, there could be a multitude of information in the user database 360, e.g. there may be more than one shipping address, etc. Each field containing information in the user database 260 will have a textual identifier for the file (i.e. FIRST_NAME to identify a field that will contain the first name of a user).

If the interface application 250 is able to match the input information to a value stored in the user database 260 about the user, the interface application 250 can map the input to the field in the user database 260 and associate the textual identifier for the field in the user database 260 to the input field on the web page. For example, if the interface application 250 examines the input of a user to a field on the web page and determines that it matches the information contained in a field called FIRST_NAME in the user database 260, the interface application 250 can associate the input field as asking for the first name of a user, which is stored in the user database 260 corresponding to the user under FIRST_NAME.

Next, method 800 tries to resolve any ambiguities in the information input to the web page by the user at step 840. For example, for a date such as 10/10/1982, a user might enter a "10" into a single input field on the web page. This could refer to either the day of the month. In order to resolve this ambiguity, the application interface 250 can either: use the context to see if it can determine what the information being input is (i.e. the label text could include the text string "MONTH" and the interface application 250 would then know that the 10 input by the user refers to the month portion of the date); or prompt the user to manually resolve the ambiguities (i.e. prompt them to identify from a list what the input is related to).

At step 850, the method 800 verifies the classification it has given the input fields on the web page being viewed by examining the input of the user against the context it has

determined for the input field. The method 800 compares the context it has determined for each input on the web page at step 820 to the information entered by the user and examined in step 830. For example, the application interface 250 may have determined from the context that a set of input fields relates to a shipping address and therefore certain types of information are expected to be mapped to a filed, such as street address, city, country, etc. Therefore, if the method 800 has classified one of these input fields as something other than information related to a shipping address, this classification will not be verified at step 850.

If the context agrees with how the information was classified by the interface application 250, the method 800 moves on to step 880. However, if the context does not agree with how the interface application 250 has classified the information, the method 800 will determine whether to accept the manually inserted information, the calculated context or require additional training data to classify the field, at step 870.

At step 880, the exit method for submitting the information requested by the web page and moving to the next web page in the transaction sequence is recorded. The interface application 250 record the action a user performs to initiate navigation from the current web page to the next web page in the transaction sequence, whether this action is a direct action or indirect action.

Finally, at step 890, the metadata from the above process is used to create a page object 400 corresponding to the web page and this page object 400 is saved to the database 300.

The created page object 400 may be proofed and tested at a later time.

AUTOMATIC POPULATION OF THE DATABASE VIA AN INTELLIGENT AGENT

In a further aspect, the database 300 can be populated with page objects 400 using an intelligent agent that navigates through a website and extracts metadata from the web page in the website to create page objects 400 corresponding to a series of web pages making up a transaction sequence. The intelligent agent does not have to operate in conjunction with a web browser or wait for a user to access and navigate through a website, rather the intelligent agent works autonomously from a user and can crawl through websites without user intervention.

Fig. 9 illustrates a flowchart of method 900 used by an intelligent agent for automatically populating the database 300 with page objects 400. Method 900 comprises the steps of: determining whether a website is a shopping website 905; adding an item to the shopping cart 910; navigating to a shopping cart page on the website 915; attempting to navigate to a next page in the transaction sequence 920; checking if it was successful 925; crawling through the DOM of the web page and calculating context for each input 930;

determining remaining inputs 935; analyzing each method of transferring to another web page 940; creating an M-tree of web pages 945; checking more web pages occur in the M-tree 950; selecting a web page in the M-tree 955; checking if a linear path through the transaction sequence has been located 960; selecting a next M-tree 965; creating a page objects with the collected metadata 970; and saving the page objects to a database 975.

The method 900 begins by navigating to the home page of a website. At step 905, the method 900 attempts to determine whether the website is a shopping website. If the website is not a shopping website, the method 900 ends.

If the method 900 at step 905 determines that website is a shopping website, the method 900 proceeds to step 910 where it attempts to add an item to the shopping cart and then navigates to the shopping cart page at step 915. From step 915, the method 900 attempts to automatically navigate through the transaction sequence in order to create a page object 400 for each web page in the transaction sequence. An item is attempted to be purchased so that the method 900 can try to determine how to complete a successful transaction sequence for the website.

From the shopping chart page, the method 900 attempts to navigate to the next web page in the transaction sequence at step 920. This will typically be either a checkout web page or a login/registration page. From step 920, the method 900 will being recording information related to the transaction sequence to attempt to generate the necessary page

objects 400. If the method 900 fails to navigate to the first page in a transaction sequence at step 925, the method 900 ends.

However, if at step 925 the method 900 is successful at navigating to the first web page in the transaction sequence, the method 900 moves to step 930 and crawls through the DOM of the current web page. The document object model (or DOM) of the current web page calculating the context each requested input on the web page and using the context to determine what the information that is necessary to input to the webpage in order to successfully complete the current web page. In this manner, step 930 determines which inputs are requested by the current web page.

At step 935, the method 900 examines the context determined at step 930 and determines which inputs remain out of the set of basic inputs required for a successful completion of the transaction sequence, e.g. shipping address, billing address, billing information etc. In this way the method 900 can keep track of the information the transaction input has already requested so that it can be taken into account when using the context of later web pages to determine what information later pages may requests as input. For example, if the billing address is already requested on the first web page in the transaction sequence, the method 900 may determine from the context of a web page that an address is being requested, however since the billing address was already requested in an earlier web page in the transaction sequence, the method 900 can use it to decide the address being asked for is the shipping address.

At step 940, the method 900 analyzes each method of transfer to another web page and at step 945 creates an M-tree of web pages that can be visited from the current web page. Each web page that can be navigated to from the current web page is represented in the M-tree.

Steps 950 and 955 cause each web page in the M-tree to be navigated to by the method 900 with steps 930, 935, 940 and 945 repeated for each web page in the M-tree.

Once all the web pages in the M-tree created for the first page has been examined by the method 900, the method 900 checks at step 960 to see if a linear path has been found through the transaction sequence. The M-tree of the next web page is selected and steps 930, 935, 940, 945, 950 and 955 are repeated for the M-tree created from the next webpage.

In this manner, the method 900 crawls through the web pages moving from web page to web page until a path is found at step 960 that allows a successful transaction. This path is then used as the sequence transaction.

The information gathered in the method 900 related to the web pages is then used to create page objects 400 for each web page identified in the transaction sequence and saved to the database 300 at step 975.

The foregoing is considered as illustrative only of the principles of the invention. Further, since numerous changes and modifications will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation shown and described, and accordingly, all such suitable changes or modifications in structure or operation which may be resorted to are intended to fall within the scope of the claimed invention.

CLAIMS

What is claimed is:

1. A ...

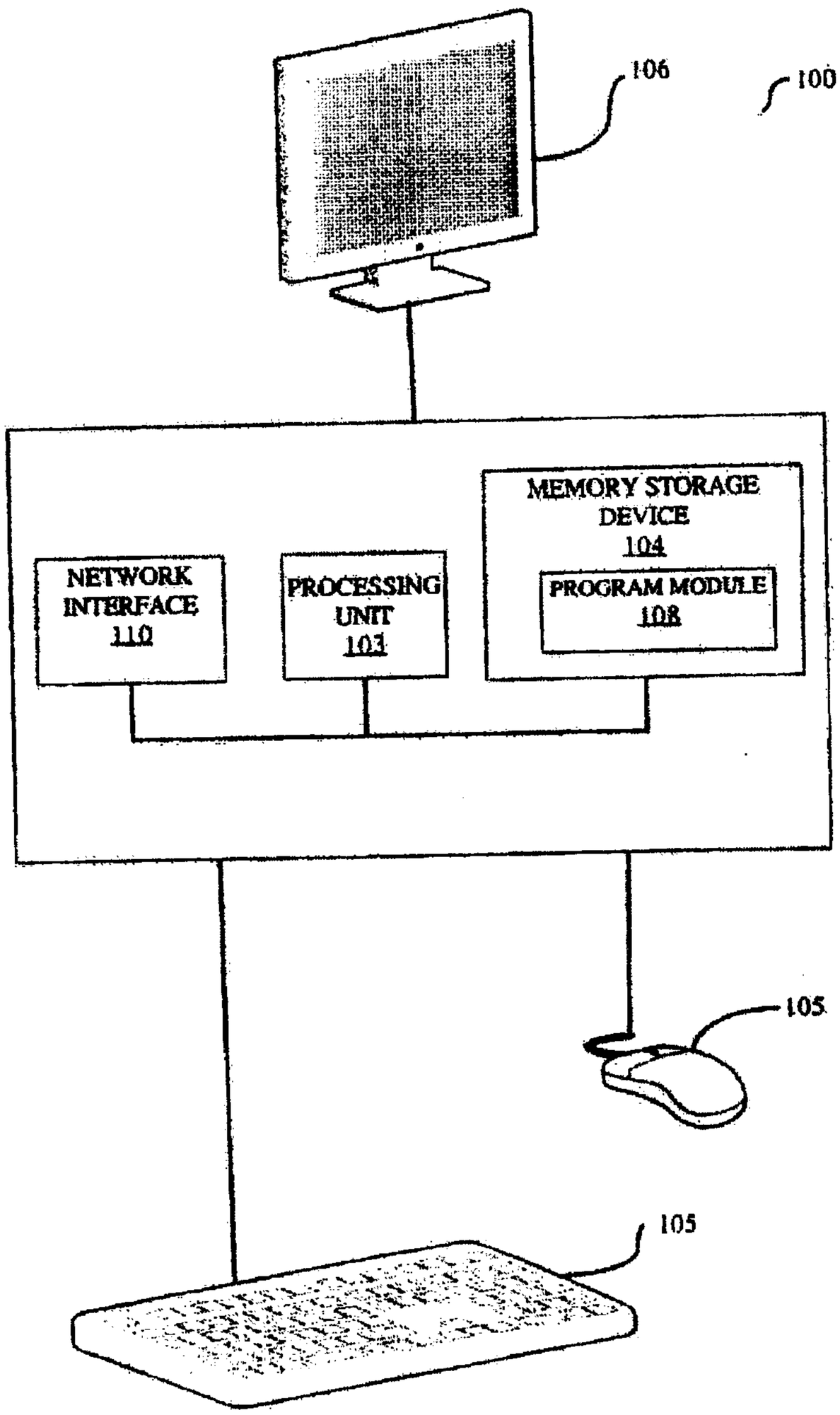

106
100
NETWORK INTERFACE 110
PROCESSING UNIT 103
MEMORY STORAGE DEVICE 104
PROGRAM MODULE 108
105
105

FIG. 1

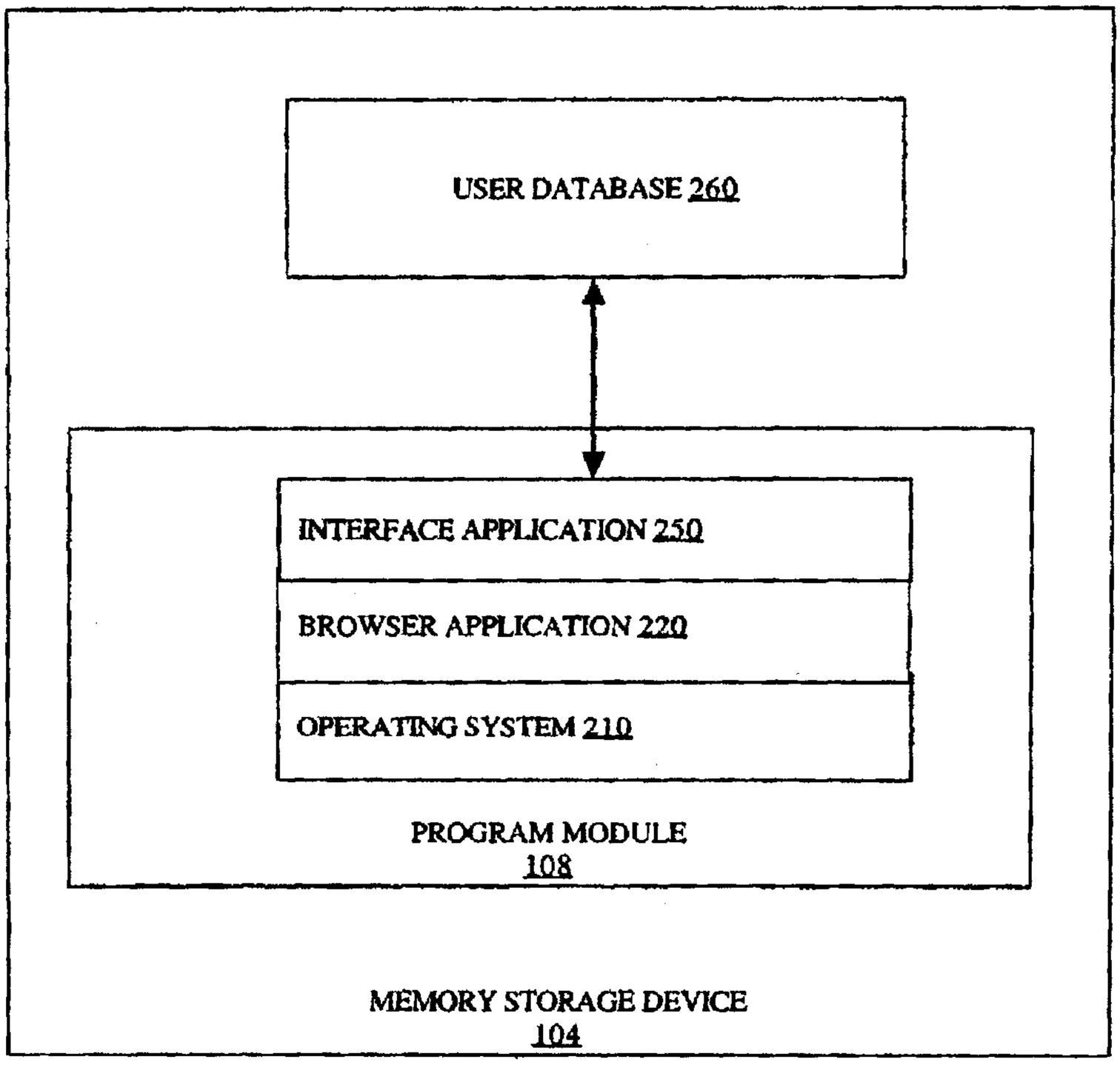

USER DATABASE 260
INTERFACE APPLICATION 250
BROWSER APPLICATION 220
OPERATING SYSTEM 210
PROGRAM MODULE 108
MEMORY STORAGE DEVICE 104

FIG. 2

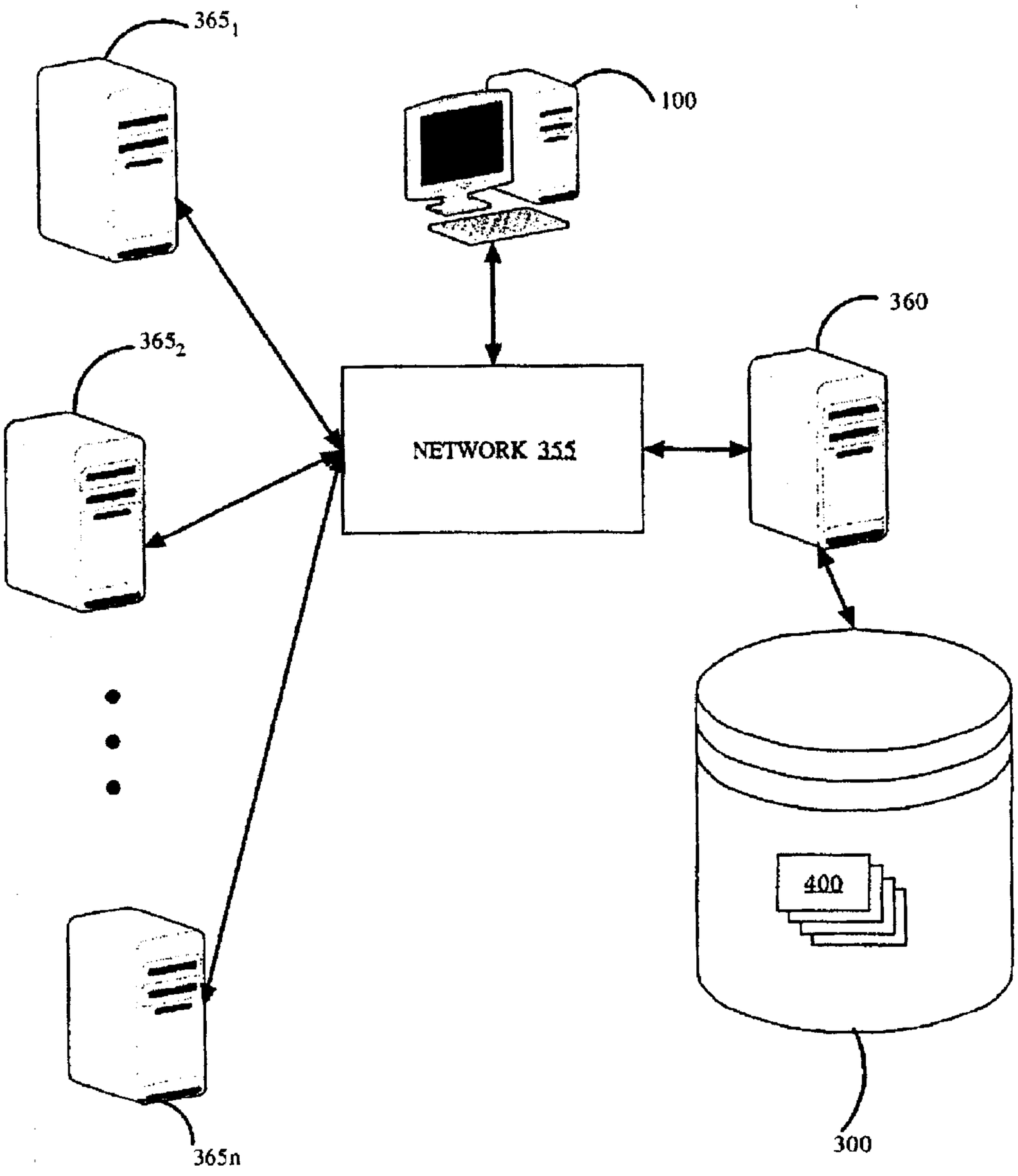


FIG. 3

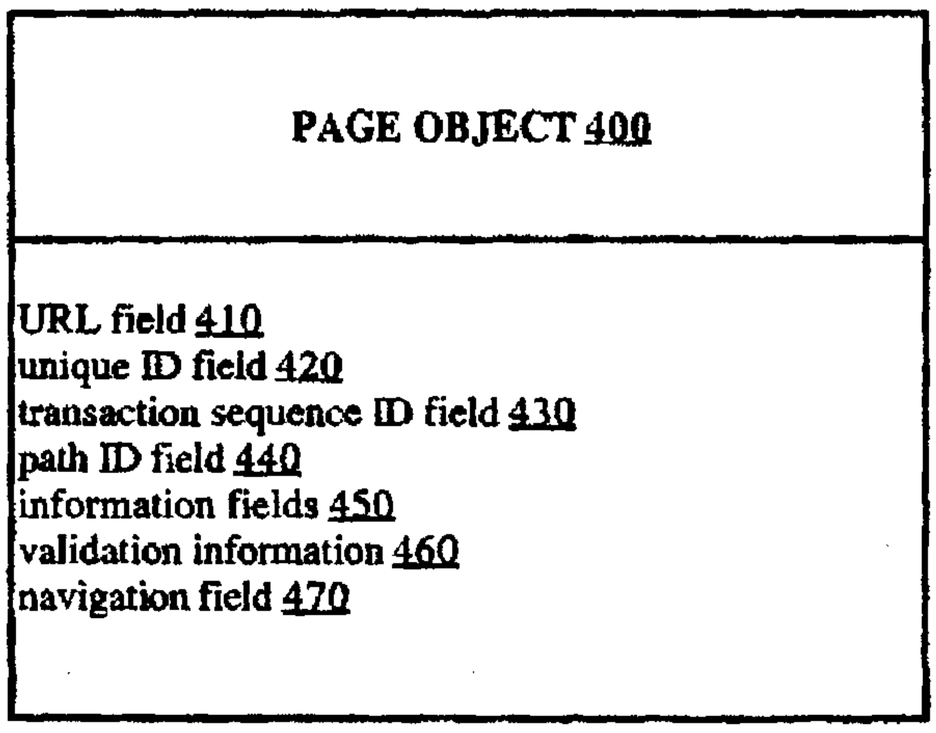

PAGE OBJECT 400
URL field 410
unique ID field 420
transaction sequence ID field 430
path ID field 440
information fields 450
validation information 460
navigation field 470

FIG. 4

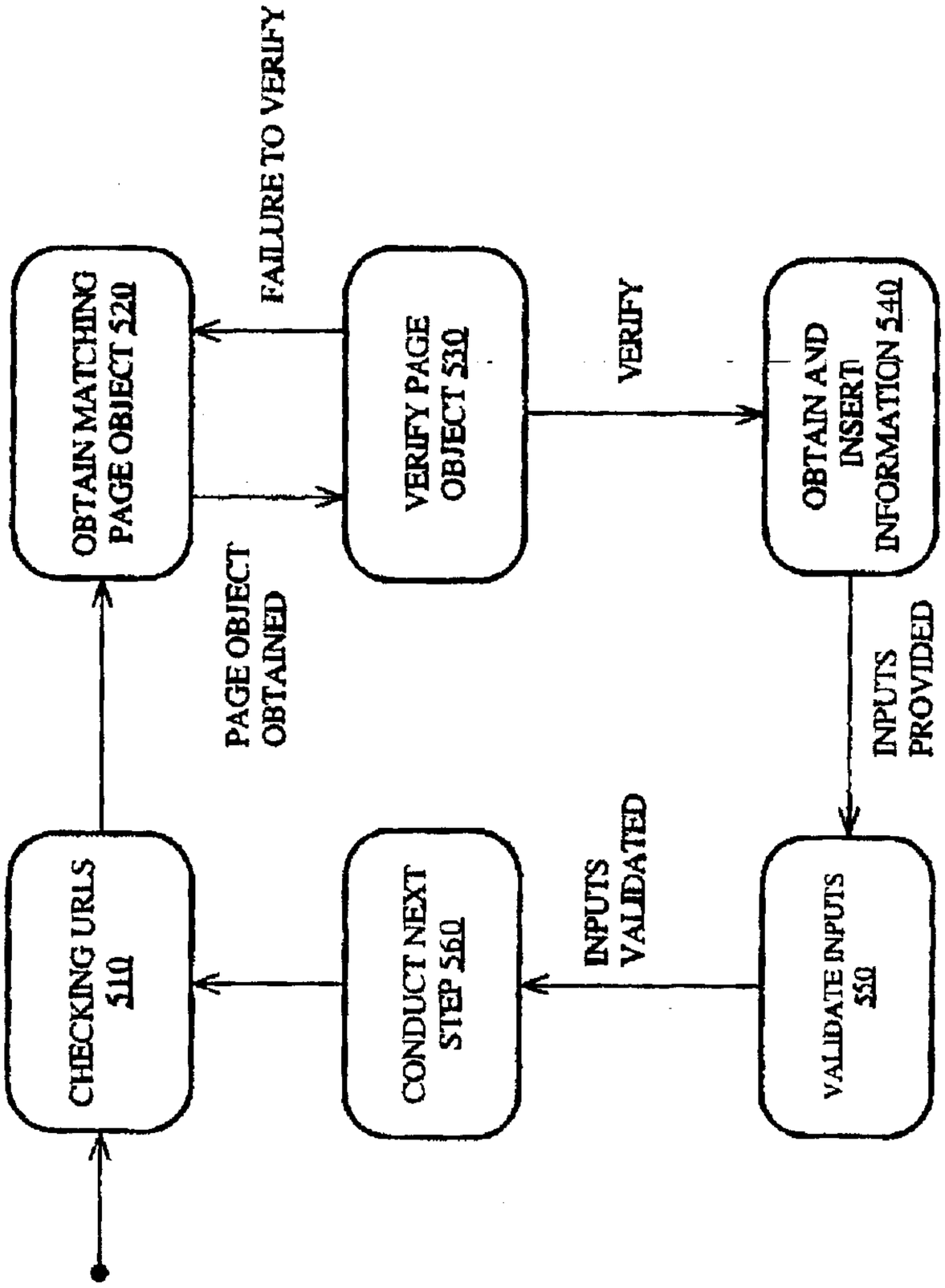

CHECKING URLS 510
CONDUCT NEXT STEP 560
VALIDATE INPUTS 550
OBTAIN MATCHING PAGE OBJECT 520
VERIFY PAGE OBJECT 530
OBTAIN AND INSERT INFORMATION 540
PAGE OBJECT OBTAINED
FAILURE TO VERIFY
VERIFY
INPUTS VALIDATED
INPUTS PROVIDED

FIG. 5

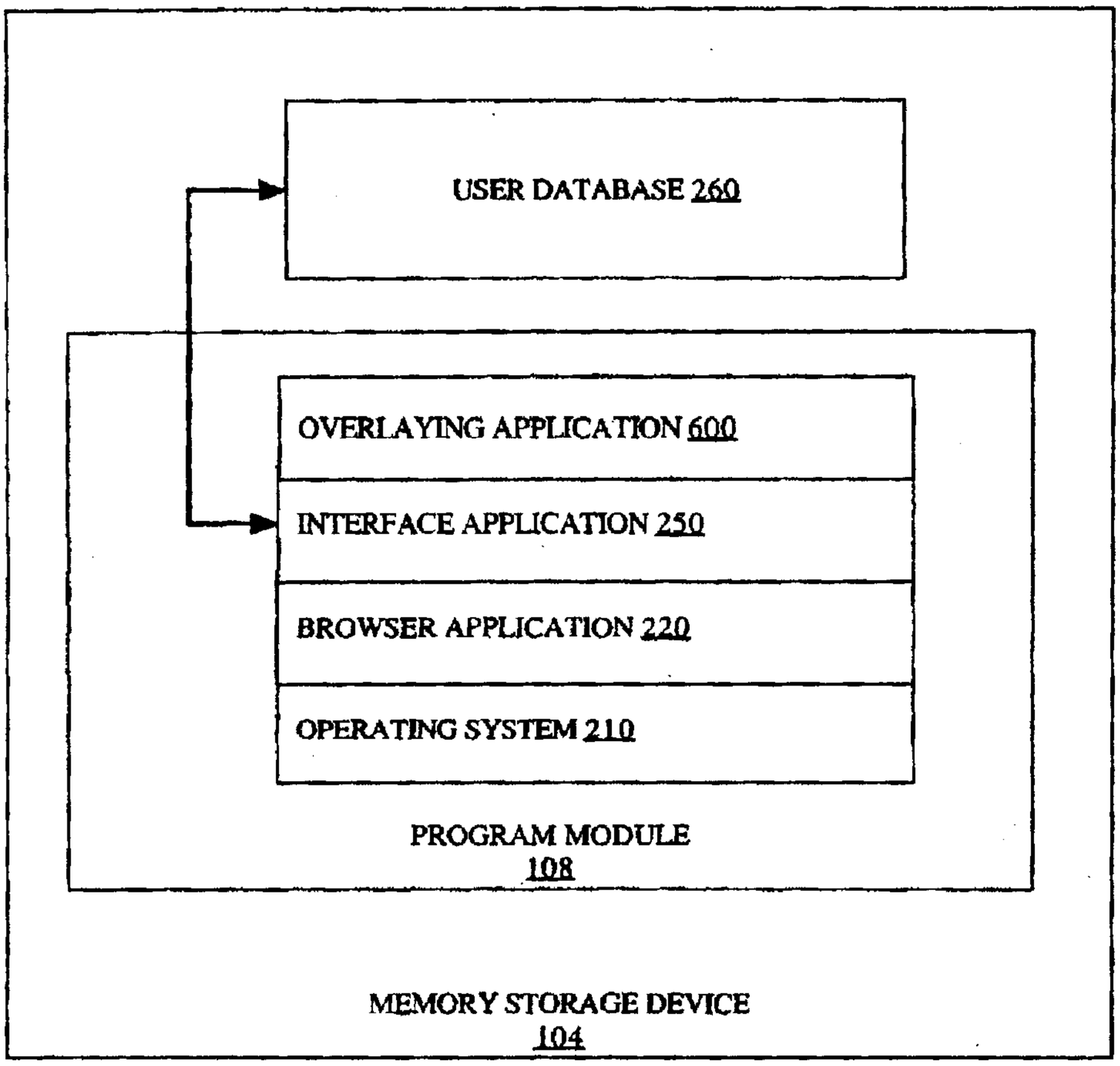

USER DATABASE 260
OVERLAYING APPLICATION 600
INTERFACE APPLICATION 250
BROWSER APPLICATION 220
OPERATING SYSTEM 210
PROGRAM MODULE
108
MEMORY STORAGE DEVICE
104

FIG. 6

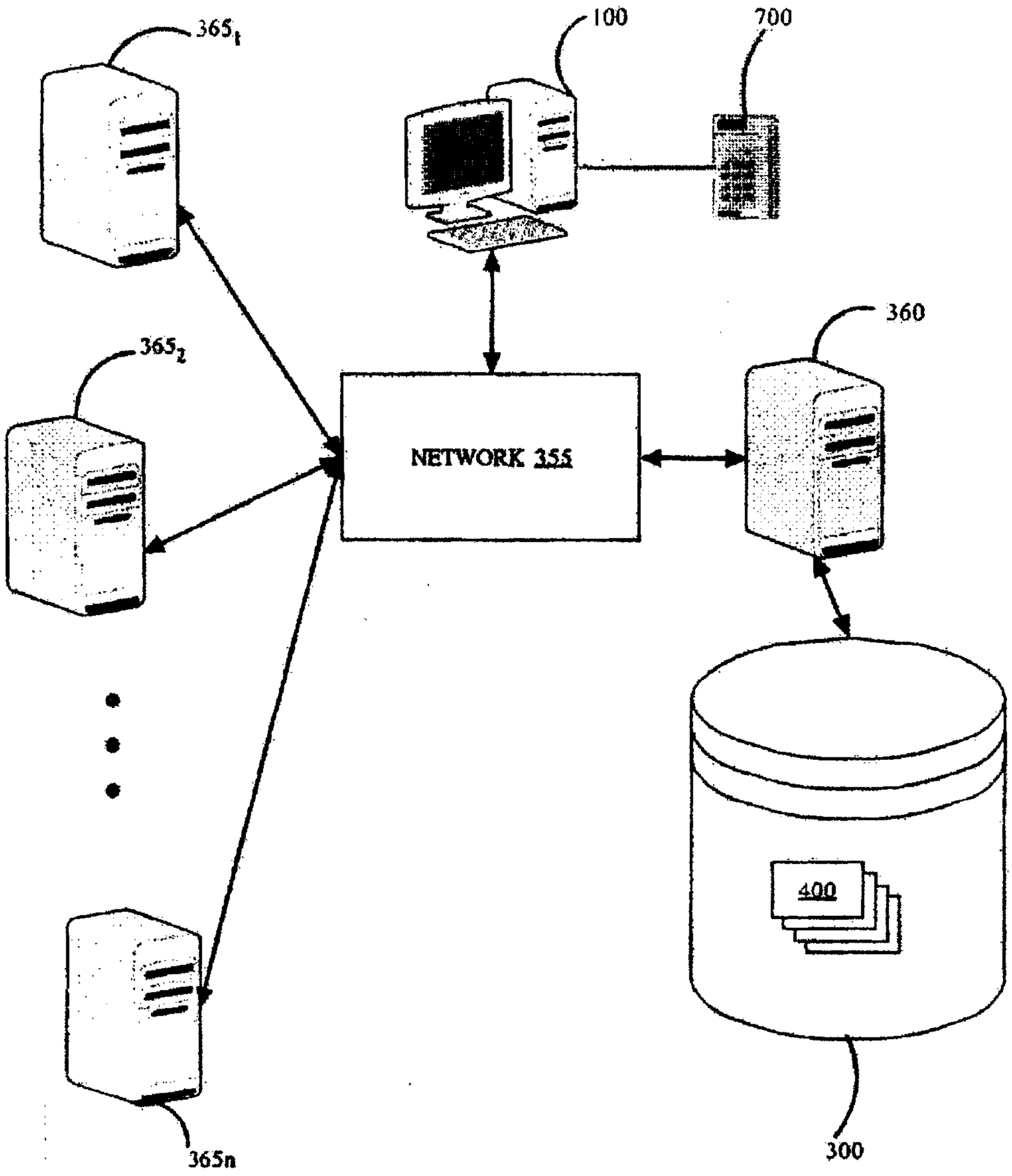


FIG. 7

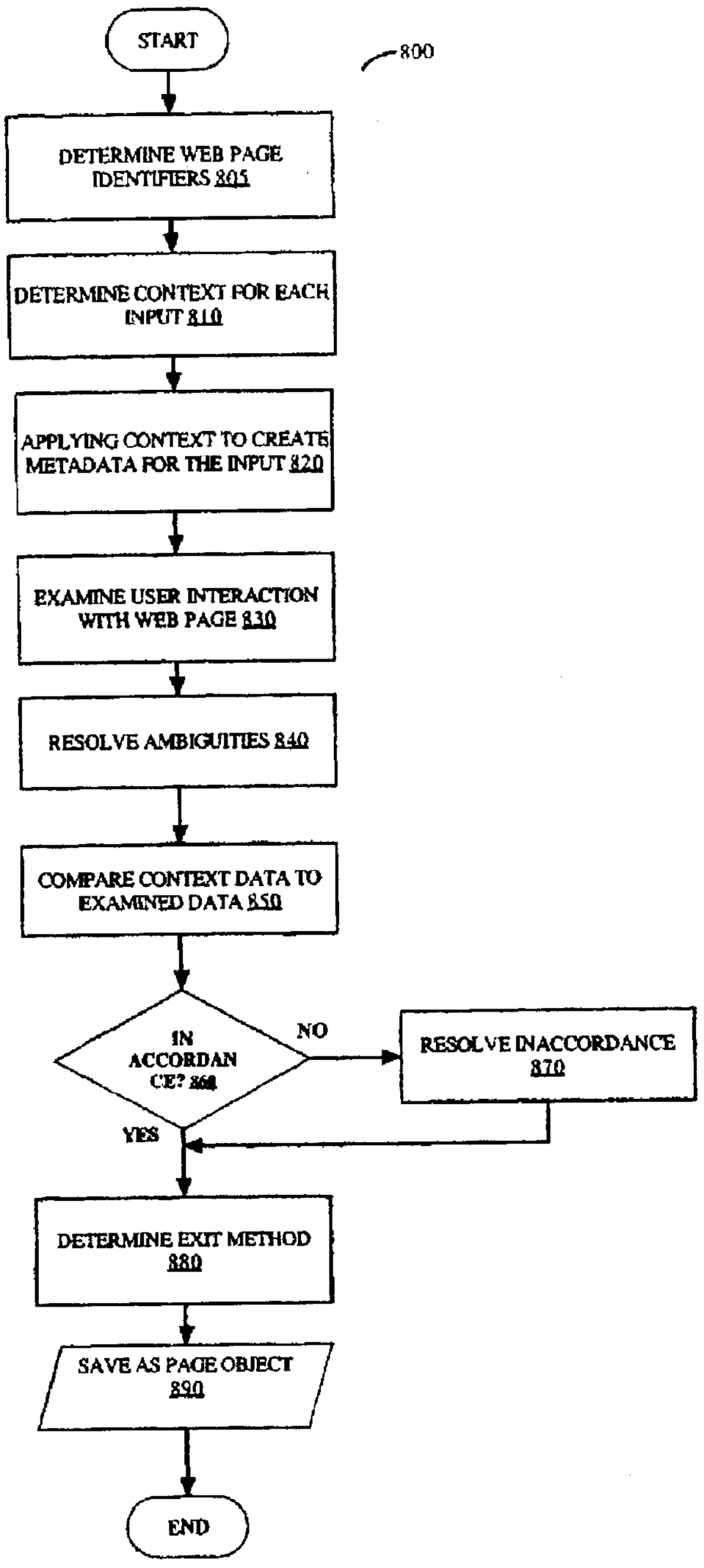

START
800
DETERMINE WEB PAGE IDENTIFIERS 805
DETERMINE CONTEXT FOR EACH INPUT 810
APPLYING CONTEXT TO CREATE METADATA FOR THE INPUT 820
EXAMINE USER INTERACTION WITH WEB PAGE 830
RESOLVE AMBIGUITIES 840
COMPARE CONTEXT DATA TO EXAMINED DATA 850
IN ACCORDAN CE? 860
NO
RESOLVE INACCORDANCE 870
YES
DETERMINE EXIT METHOD 880
SAVE AS PAGE OBJECT 890
END

FIG. 8

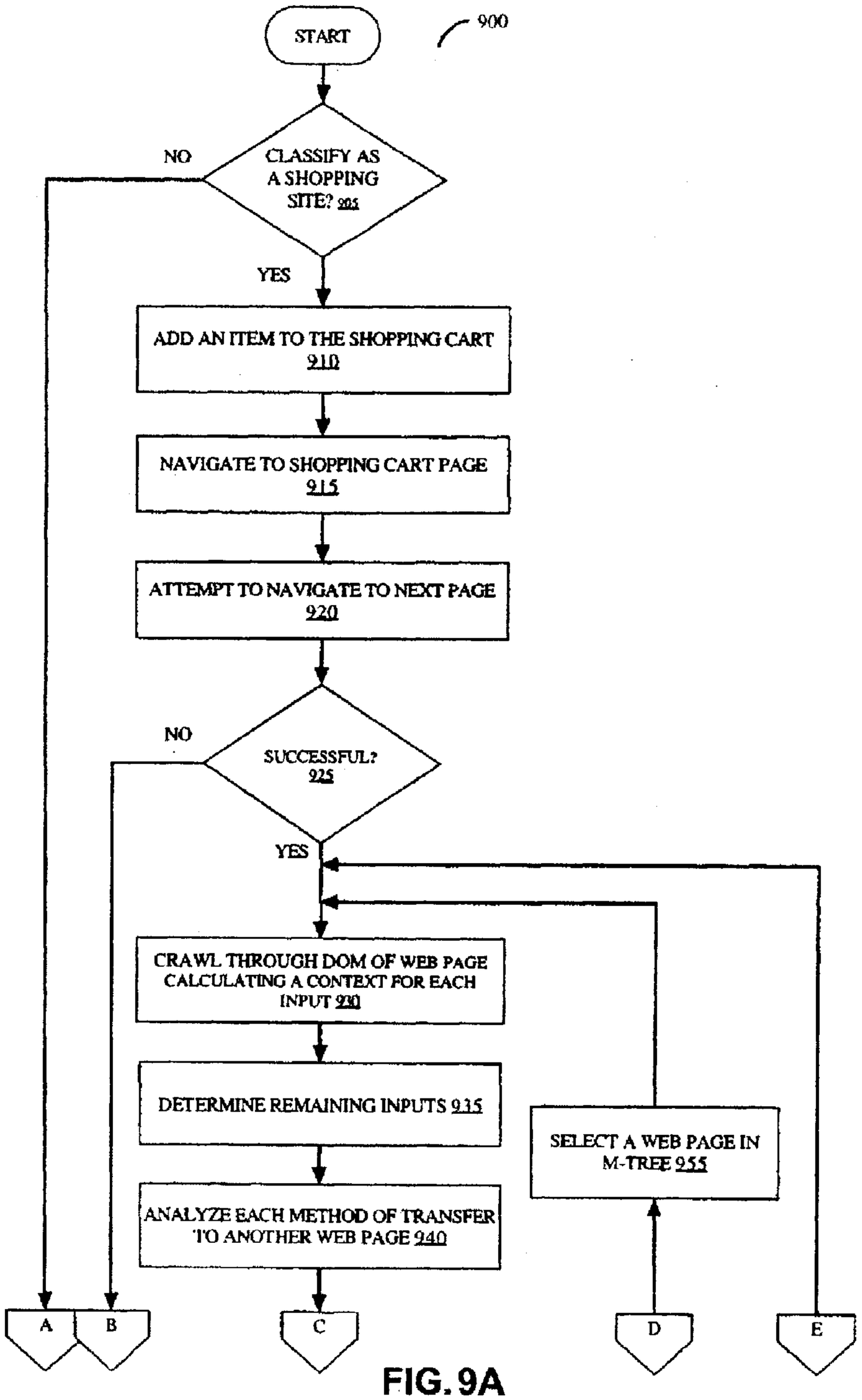

START
900
NO
CLASSIFY AS A SHOPPING SITE? 905
YES
ADD AN ITEM TO THE SHOPPING CART 910
NAVIGATE TO SHOPPING CART PAGE 915
ATTEMPT TO NAVIGATE TO NEXT PAGE 920
NO
SUCCESSFUL? 925
YES
CRAWL THROUGH DOM OF WEB PAGE CALCULATING A CONTEXT FOR EACH INPUT 930
DETERMINE REMAINING INPUTS 935
ANALYZE EACH METHOD OF TRANSFER TO ANOTHER WEB PAGE 940
SELECT A WEB PAGE IN M-TREE 955
A
B
C
D
E

FIG. 9A

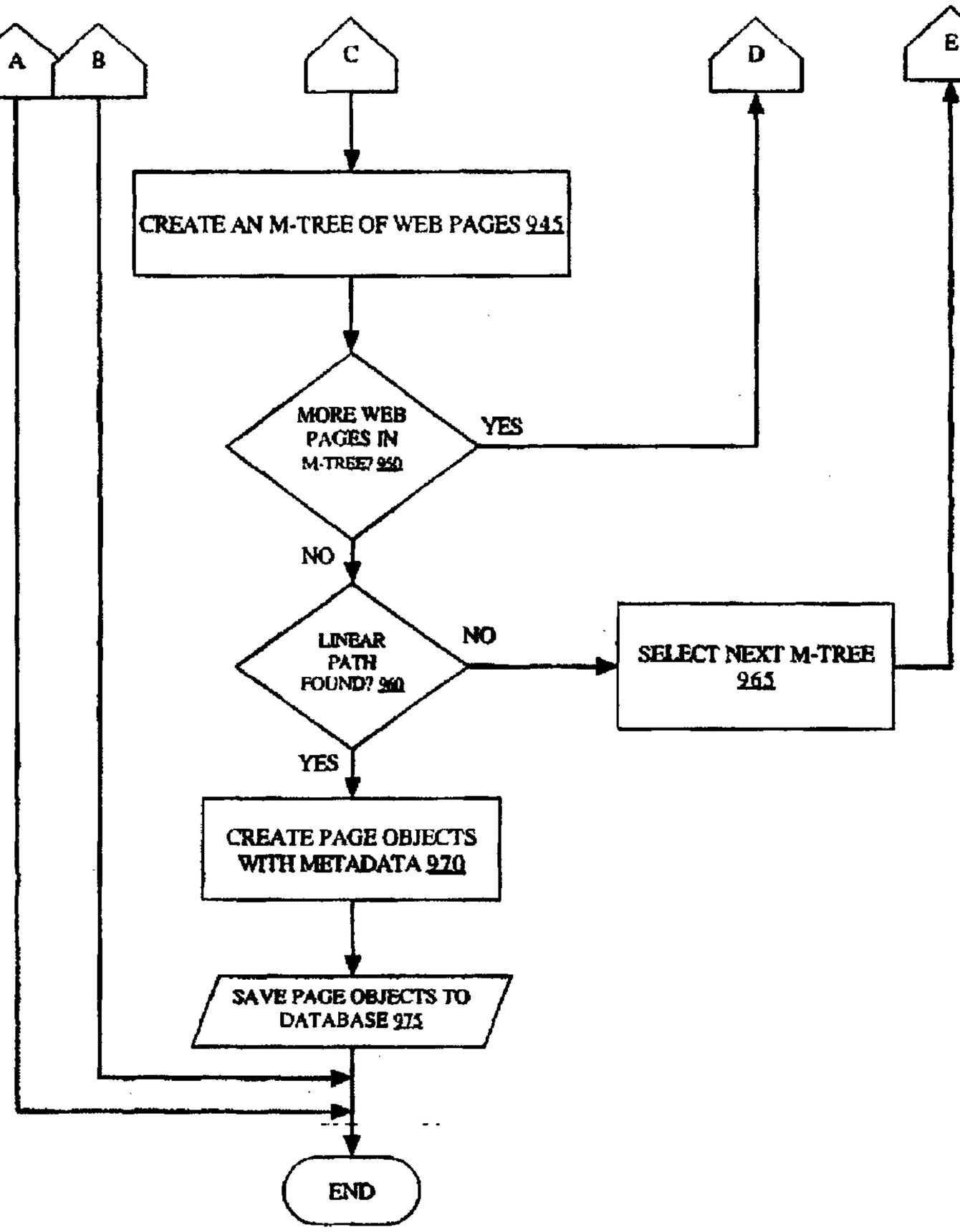
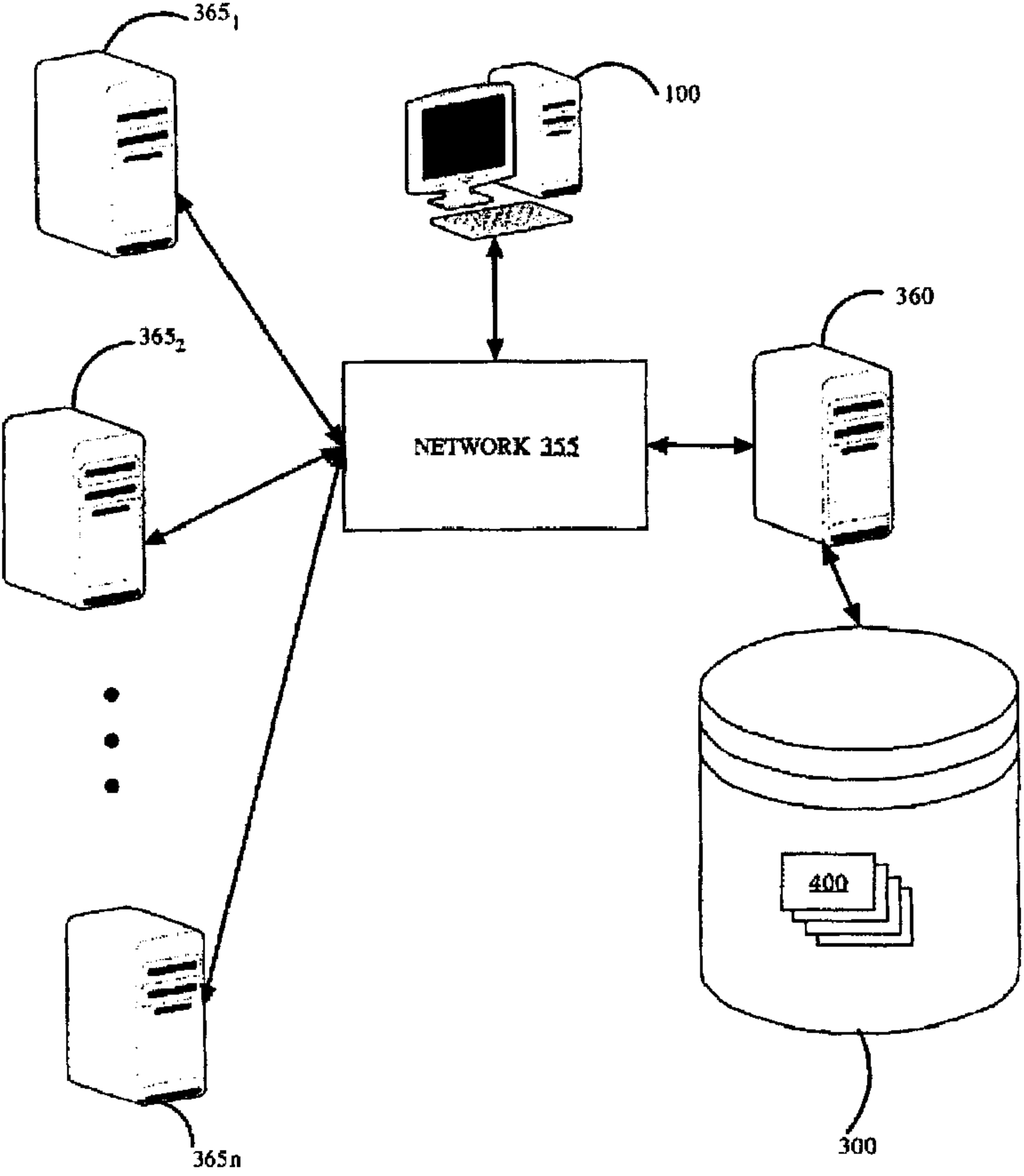

A
B
C
D
E
CREATE AN M-TREE OF WEB PAGES 945
MORE WEB PAGES IN M-TREE? 950
YES
NO
LINEAR PATH FOUND? 960
NO
SELECT NEXT M-TREE 965
YES
CREATE PAGE OBJECTS WITH METADATA 970
SAVE PAGE OBJECTS TO DATABASE 975
END

FIG. 9B


365 1
100
360
365 2
NETWORK 355
400
300
365n