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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 December 2003 (18.12.2003)

PCT

(10) International Publication Number
WO 03/105028 A2

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US03/18379

(22) International Filing Date: 9 June 2003 (09.06.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/387,040 7 June 2002 (07.06.2002) US

(71) Applicant *(for all designated States except US)*: **WEST PUBLISHING COMPANY, DBA WEST GROUP** [US/US]; 610 Opperman Drive, Eagan, MN 55123 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **MORTON, Gerald, J.** [US/US]; 224 15th Avenue North, South St. Paul, MN 55075 (US). **LUND, Elizabeth, S.** [US/US]; 4339 Lyndale Avenue South, Minneapolis, MN 55409 (US).

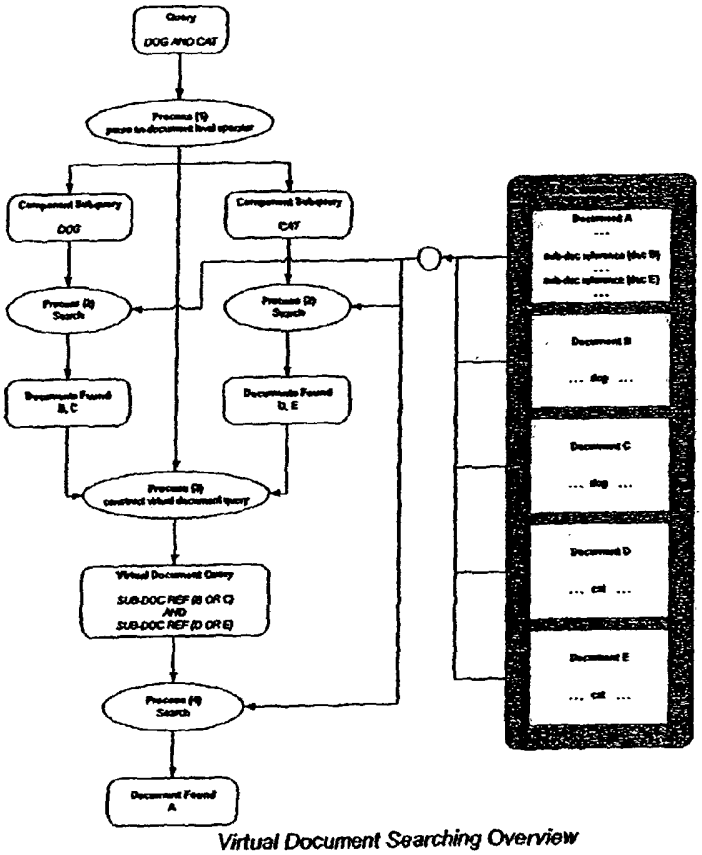
(74) Agents: **STEFFEY, Charles, E.** et al.; Schwegman, Lunberg, Woessner & Kluth, P.A., P.O. Box 2938, Minneapolis, MN 55402 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: VIRTUAL DOCUMENT SEARCHING



Virtual Document Searching Overview

(57) Abstract: An exemplary search system parses a received query into one or more sub-queries, with each parsed sub-query including one or more search operators. The system then processes the sub-queries against one or more databases to produce corresponding sub-query result sets, with each set including a list of documents. The sub-query results sets are then combined based on the one or more search operators to produce a query result set for the received query, with the query result set identifying at least one virtual document, the virtual document comprising one or more sub-documents. The one virtual document is presented to a query generator as a unified document.

WO 03/105028 A2

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

VIRTUAL DOCUMENT SEARCHING

Background

The Goal

Many documents contain component parts that are in fact shared with many other documents. For example, West Headnotes, are shared in Case documents, Statute documents, and Analytical documents. Due to the current limitations of West's search engine, these headnotes must be physically replicated in every document in which they occur. This results in increased complexities and costs in maintenance as well as storage.

Virtual Document Searching will allow documents and their individual component parts to be stored separately and assembled as virtual documents for search purposes[1]. This will allow shared component parts to be stored only once, reducing the maintenance complexities and storage costs associated with storing these parts multiple times.

The Problem

Document searching utilizes a variety of operations that allow a researcher to define what he or she is looking for. These operations can be generally classified into two distinct categories: document level operations and word level operations.

Document level operations are those that qualify documents strictly by the presence of absence of query components within a documents. AND and OR are examples of document level operations. The query *dog and cat* qualifies documents as long as both *dog* and *cat* are present in the document.

Word level operations are those that qualify documents based on the actual word positions of query components within a document. Phrases and word proximities are examples of document level operations. The query "*dog house*" qualifies documents as long as *dog* and *house* are not only in the same document, but occur within one word position of each other in the order specified.

The basic problem with searching virtual documents is that since the document is not physically assembled at the time of indexing, word positions

[1] Virtual documents for display purposes is not a difficult technical problem.

cannot be assigned that provide a single continuous sequencing of words across the entire virtual document. Therefore, word level search operations would not work.

A fundamental premise of this invention is that in many if not all circumstances, there isn't truly a need to have word level search operations work across components of a virtual document. Word level operations are clearly necessary within components of a virtual document, but when it comes to spanning components of a virtual document, all that is needed are document level operations.

An example of this situation is West's Caselaw documents. They consist of the actual text of the case along with West created headnotes (as well as other things). If these documents were physically stored as virtual documents, the caselaw document would not actually contain the individual headnotes, but would instead reference them. When a user is searching these cases, they may use word level operations to identify concepts within individual components (individual headnotes or the main text), but when combining these concepts, they would use document level operations. For example, take the query "*medical malpractice*" *AND sports/3 injury*. This query is searching for a document with two concepts. The word level operations clearly make sense within an individual component, but don't necessarily make sense across components. The document level operation makes sense across components and will result in finding cases that have two different headnotes that each contain one of the concepts.

Given this behavior, it is not necessary to solve the word position assignment problem described earlier in order to support virtual documents.

Overview

Virtual documents are created by encoding a reference to a sub-document into a document as opposed to physically copying the sub-document as is done today. The virtual document search engine then operates as follows.

1) A query is parsed into component sub-queries based on the document level operators.

2) The component sub-queries are processed using the standard search engine to produce lists of (component) documents that satisfy each sub-query.

3) A virtual document query is constructed using the results from the component sub-queries together the operators by which original query was parsed in step 1.

4) The virtual document query is processed using the standard search engine. The final result identifies virtual documents that matched the original query even though parts of the query were satisfied by physically separate sub-documents of the virtual document. Please see Figure 1.

Process

Query Parsing

The first step in the virtual document search process is to parse the input query into sub-queries. Search queries are generally represented as tree structures where the intermediate nodes in the tree represent operations, the branches in the tree represent sub-trees, and the leaves of the tree represent actual searchable entities (e.g. words).

The goal of parsing is to create sub-queries out of the lowest possible sub-trees of the query. Since document level operations are supported at the virtual document level when searching across components, parsing can continue down through these operators. However, parsing must stop when the first word level operators are reached. This is because word level operators are only supported within a single component document.

This results in sub-queries that are either searchable entities, or sub-queries where the root of the sub-query tree is a word level operator. Please see Figure 2.

Component Searching

Once the original query is parsed and the individual sub-queries created, the standard search engine is used to process each sub-query against the set of candidate documents. The results from each sub-query search are a set of (component) documents that satisfy the criteria for that sub-query.

Since the sub-queries were created by parsing only down to the first word level operator, at the completion of this step, all word level operations specified in this initial query have been satisfied. All that remains is to combine these results based on the document level operations that are above the sub-query trees in the original query tree.

Virtual Document Query Construction

When the individual sub-query searches have been completed, the virtual document query must be constructed. Understanding of this process begins with a description of how virtual documents are actually coded.

Virtual Document Coding

The coding of a virtual document involves placing a special type of reference in the virtual document that points to the component document that is to be considered part of this virtual document. Assuming documents are coded in XML, a simple example of such a reference is the following:

<sub-doc_ref>IF6432EB85AD042359640046C07BB9543</sub-doc_ref>

In this example, a component document identifier (GUID) is encoded within a specific element identifying it as a sub-document reference. This type of reference could occur any number of times and in any number of places within a virtual document.

Query Construction

The results from the individual sub-query searches are used to replace the corresponding sub-trees in the original query to form the virtual document query. An element restriction search is also used to properly focus the search on occurrences of the sub documents when used as component documents within a virtual document.

Figures 3, 4 and 5 show examples of this construction process. The queries used in these examples are the same ones used earlier to show the initial query parsing process.

Virtual Document Searching

Once the virtual document query is constructed, the standard search engine is again used to process this query against the set of candidate documents. The results from this search are the virtual documents who either by themselves or in conjunction with their component parts, satisfy the original query.

Note that it is possible for the component documents to reside in a separate document collection from the virtual documents that contain these component documents. As long as the appropriate control information is maintained to indicate which document collections are to be used at each step, virtual documents and component documents can be maintained in the same or separate document collections.

Other Details

Nested Virtual Documents

It is possible to define virtual documents that contain other virtual documents, thus creating nested virtual documents. The basic process can be extended to support any desired level of nesting. To implement this feature, the search results from each virtual document query are used to reconstruct the virtual document query which is then used to search again. This process continues until either the desired level of nesting is reached, or until no new documents are retrieved by the latest virtual document query. Please see Figure 6.

Field Searching

Another searching feature is to be able to restrict a search to within a particular field or logical part of a document such as a title or summary section. In XML this can translate into restricting a search to within a particular element. An example query is: *summary(dog)* which is requesting to find occurrences of *dog* within the *summary* field or XML element.

At the simplest level, searching for individual component parts that satisfy the entire query is straight forward requiring no special processing. However, in order to find virtual documents where the field is defined within the virtual document while the term is actually within the sub-document requires additional processing by the Virtual Document Search Engine.

Since it is possible for the field restriction to be satisfied at either the sub-document or virtual document level, the restriction must be first checked at the sub-document level, and if not satisfied, it must be checked again at the virtual document level. This is accomplished by formulating the sub-query with and without the field restriction, then crafting a virtual document query that combines the two results and applies the field restriction to the result from the sub-query that did not have the field restriction. Please see Figure 7.

Date Searching

Date searching (as used in Westlaw) is strictly a document level operation and would therefore only be applied to the virtual document query. It would have no impact on the sub-queries created during the initial parsing phase.

Exemplary Application(s)

Ability to search for documents that appear to be a single unit when in fact they are physically fragmented into multiple "sub-documents."

Exemplary Purpose

West Headnotes are currently incorporated into cases, statutes, and analytical materials. In order to facilitate searching in today's world, headnotes are physically instantiated in each document within which they reside. This results in duplication of headnotes which has a direct impact on storage costs as well as update costs. The exemplary software would allow headnotes (or other sub-documents) to be stored once while still supporting the search functionality that makes them appear as part of another document.

Brief Summary

Virtual documents are created by encoding a reference to a sub-document into a document as opposed to physically copying the sub-document as is done today. The exemplary virtual document search engine then operates as follows. 1) Queries are parsed into sub-queries at the lowest level of "document level" operators (e.g. &). 2) Sub-queries are processed using the standard search engine to produce lists of documents that satisfy each sub-query. 3) The sub-queries results are merged together through the original operators by which the original query was parsed in step 1. 4) Processes 2 and 3 may be repeated recursively if virtual document structures may be nested many levels deep. The

final result would identify virtual documents that matched the original query even though parts of the query were satisfied by physically separate sub-documents of the virtual document.

Claims

1. A system comprising:

one or more client computer systems, with each including means for allowing users to generate queries and access query results; and

at least one server computer system for receiving queries from one or more of the client computer systems, the server computer system including at least one processor coupled to a volatile or non-volatile memory, with the memory including instructions for:

parsing a received query into one or more sub-queries, with each parsed sub-query including one or more search operators;

processing the one or more sub-queries to produce corresponding sub-query result sets, with each set including a list of documents; and

combining one or more of the sub-queries result sets based on the one or more search operators to produce a query result set for the received query, with the query result set identifying at least one virtual document, the virtual document comprising one or more sub-documents.

2. The system of claim 1, wherein parsing a received query into one or more sub-queries, comprises processing the received query as a natural-language or a Boolean query.

3. The system of claim 1, wherein at least one of the sub-documents includes a headnote regarding a particular point of law made in a judicial opinion.

4. An electrical, magnetic, or optical machine-readable medium comprising the instructions of claim 1.

5. A computer-implemented method comprising:

parsing a received query into one or more sub-queries, with each parsed sub-query including one or more search operators.

6. The method of claim 5, further comprising:

processing the one or more sub-queries to produce corresponding sub-query result sets, with each set including a list of documents.

7. The method of claim 6, further comprising:

combining one or more of the sub-queries result sets based on the one or more search operators to produce a query result set for the received query, with the query result set identifying at least one virtual document, the virtual document comprising one or more sub-documents.

8. A computer-readable medium comprising instructions for performing the methods of claim 5, 6, or 7.

9. A computer system comprising a processor operatively coupled to the computer-readable medium of claim 8.

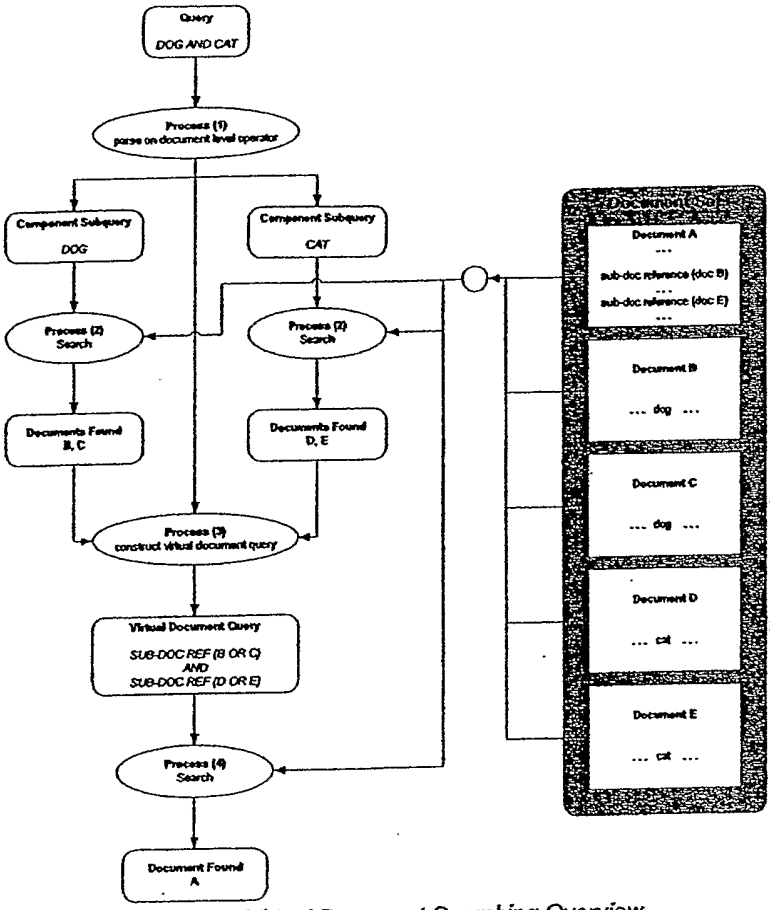


Figure 1 – Virtual Document Searching Overview

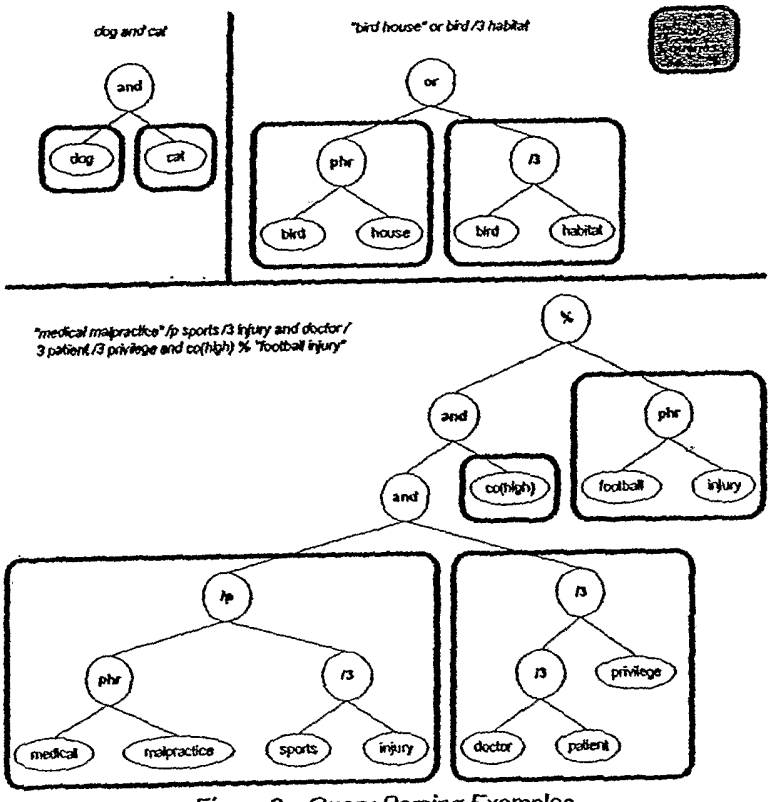


Figure 2 – Query Parsing Examples

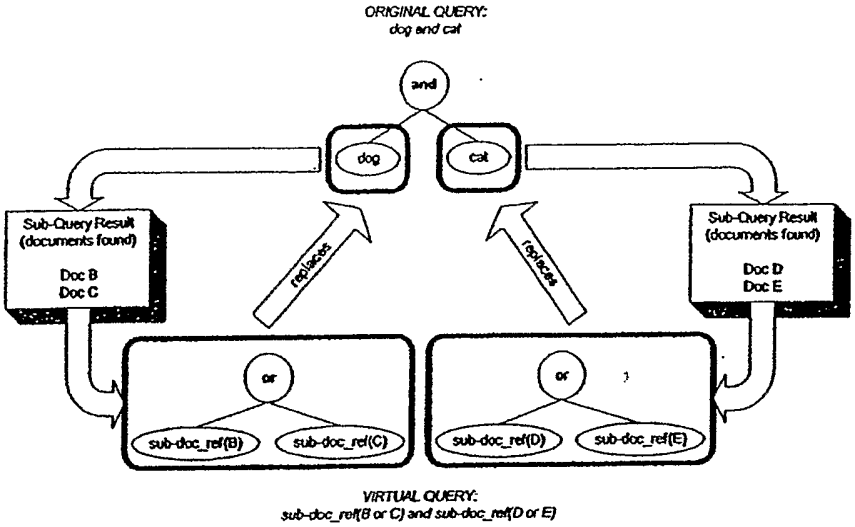


Figure 3 – Virtual Document Query Formation – Detailed Example 1

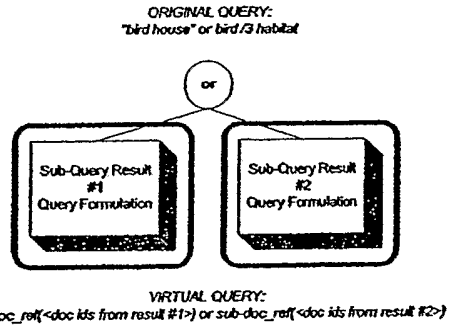


Figure 4 – Virtual Document Query Formation – Example 2

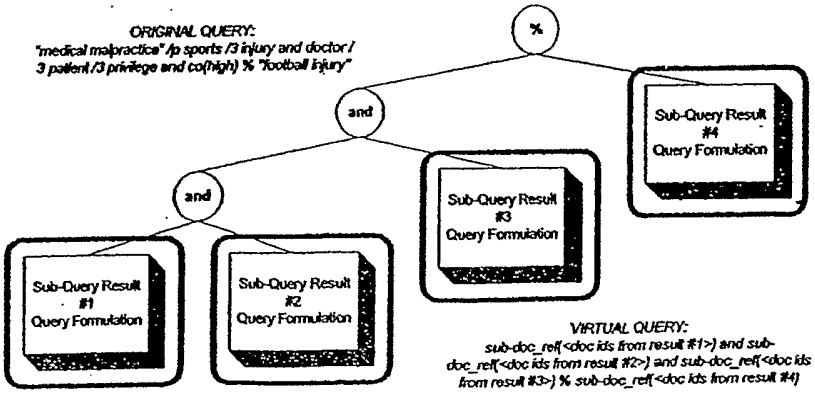


Figure 5 – Virtual Document Query Formation – Example 3

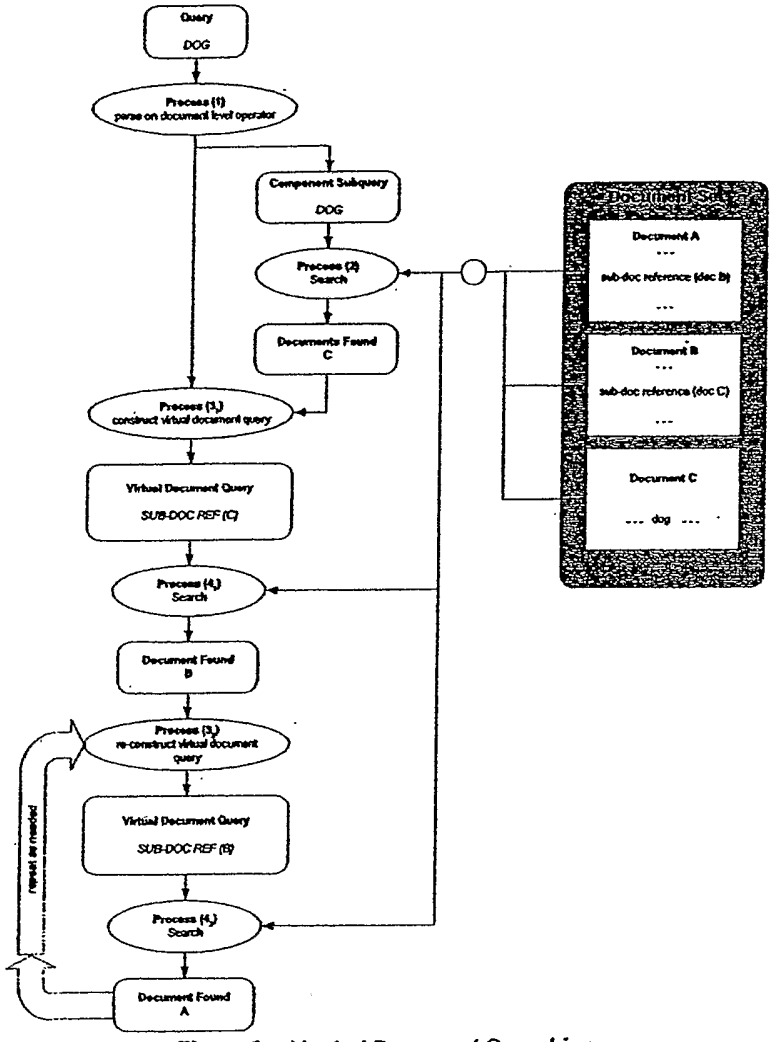


Figure 6 – Nested Document Searching

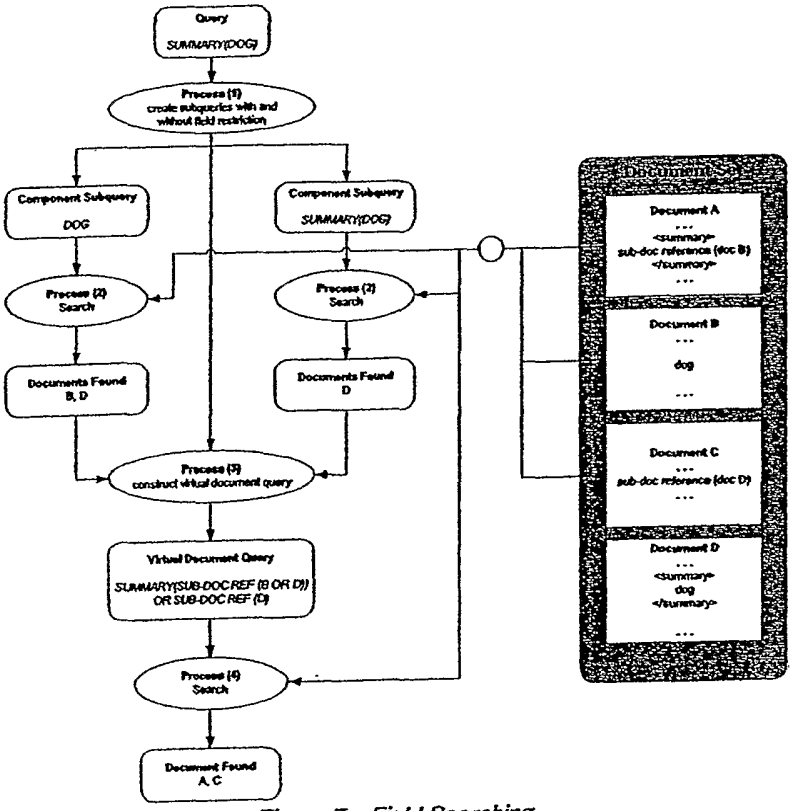


Figure 7 – Field Searching