

Microsoft Corporation

v.

X1 Discovery, Inc.

IPR2025-00253 | U.S. Patent No. 7,370,035

IPR2025-00254 | U.S. Patent No. 10,552,490

IPR2025-00255 | U.S. Patent No. 9,633,139

Demonstratives of Petitioner Microsoft Corporation

April 27, 2026

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Table Of Contents

Topic	Slide
Ground 1	8
Motivation To Combine	11
Reasonable Expectation Of Success	15
Terveen Explained REOS	15
Known To Incrementally Search Across Files	25
Lotus Notes' DB Not Too Large	33
Patents Provide No Detail	36
Ground 2	41
Entourage Searches Across Files	44
Motivation To Combine	47
Reasonable Expectation Of Success	53
Terveen Explained REOS	53
PO's Large DB Argument Fails	57

Topic	Slide
Appendix	63
Highlighting Claims	63
Lotus Notes' Single Search Field	68
Incremental Search Benefits Known	72
Email Databases Not Too Large	76
Implementing Incremental Search UI	78
Searching Partial Search Strings	80
Contact Records Are Files	87
Entourage's Search Is A Search	90
True Is Analogous Art	95
True Incrementally Searches For Multiple Search Strings	99

Overview Of Grounds

Ground 1

Feature	Lotus Notes	Raskin	Wu
Multi-String / Single Field	✓		
Incremental Results		✓	
Multiple Files	✓		
Indexed Files ('490 Only)	✓		
Intersect Results			✓

Ground 2

Feature	Entourage	True	Baeza-Yates
Multi-String / Single Field		✓	
Incremental Results	✓	✓	
Multiple Files	✓		
Indexed Files ('490 Only)			✓
Intersect Results			✓

Overview Of Grounds

Ground 1

Feature	Lotus Notes	Raskin	Wu
Multi-String / Single Field	✓		
Incremental Results		✓	
Multiple Files	✓		
Indexed Files ('490 Only)	✓		
Intersect Results			✓

Ground 2

Feature	Entourage	True	Baeza-Yates
Multi-String / Single Field		✓	
Incremental Results	✓	✓	
Multiple Files	✓		
Indexed Files ('490 Only)			✓
Intersect Results			✓

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Pertinent Claim Language

035 Patent, claim 1 (excerpted)

receiving a first string in a first search field;
in response to receiving the first string, incrementally locating a first group of documents that has at least a first word that begins with the first string;
receiving a second string in the first search field, wherein the first and second strings are separated by a string separator character in the first search field;
in response to receiving the second string, incrementally locating a second group of documents that has at least a second word that begins with the second string and identifying one or more documents that are included in each of the first and second groups of documents; and
displaying an indication of one or more of the identified one or more documents that are included in each of the first and second groups of documents.

'035 EX1001, 29:40-54.*

*Unless otherwise indicated, highlighting in excerpted text has been added by Petitioner.

490 Patent, claim 1 (excerpted)

receive a first partial search string in the first search field;
receive a second partial search string in the first search field;
wherein the first and second partial search strings are separated by a string separator character in the first search field;
searching the index data structure for a first one or more of the plurality of files each including at least one string that corresponds to the first partial search string;
searching the index data structure for a second one or more of the plurality of files each including at least one string that corresponds to the second partial search string; and
updating a set of matching files of the plurality of files as each successive character of the first partial search string and the second partial search string is received respectively in the first search field by identifying any of the plurality of files included in both the first one or more of the plurality of files and the second one or more of the plurality of files, wherein the first partial search string and the second partial search string are non-adjacent in at least one of the matching files.

'490 EX1001, 30:4-28.

139 Patent, claim 1 (excerpted)

receiving a first partial search string in a first search field;
receiving a second partial search string in the first search field;
wherein the first and second partial search strings are separated by a string separator character in the first search field;
incrementally searching a set of documents in order to identify search results, wherein the incrementally searching comprises:
searching for a first one or more documents each including at least one string that corresponds to the first partial search string;
searching for a second one or more documents each including at least one string that corresponds to the second partial search string; and
updating the search results as each successive character of the first partial search string and the second partial search string is received respectively in the first search field by identifying one or more documents included in both the first one or more documents and the second one or more documents, wherein the first partial search string and the second partial search string are non-adjacent in at least one of the identified documents.

'139 EX1001, 29:44-67.

The Disputed Features Were Known

191; EX2020, ¶9.) Alternatively, a search could utilize an index to avoid having to look at every word of every file upon every search. (See EX1023, 191-92; EX2020, ¶9.) Search user experiences could be either delimited or incremental. (EX2001, ¶35.) Delimited search was most common and required a user to take an action such as hitting a search button before the search was initiated, whereas a search that was incremental would be conducted as the user typed their query. (EX2001, ¶35.) Searches could be characterized as occurring in a single open file, or across a large distributed collection of files. (EX2020, ¶9.) Searches could allow for single string (search term) queries where a string might be a single word or multiple words treated together as a single string, or multiple string queries where each string is treated as separate in the search. (EX1021, 125; EX1019, 6:54-62.) Finally, a search system might implement the ability to perform multiple string queries using multiple search field which each accept a single string, or by allowing for multiple strings in a single search fields separated by a string separator character or, alternatively, by a Boolean operator. (EX2020, ¶9.)

← Index

← Incremental Search

Multiple Files →

Multiple Strings in Single Field →

'490 POR (Paper 21), 3;
'035 POR (Paper 21), 2-3;
'139 POR (Paper 21), 2-3.

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

The Disputed Features Had Known Benefits

Incremental Search: Responsive, Efficient, Easy To Use

made. From the point of view of interface engineering, the advantages of incremental searching are so numerous and the advantages of delimited searches so few that I can see almost no occasions when a delimited search would be preferred.⁷ In spite of near universal agreement about the desir-

EX1021, 126 (cited by '490 Petition (Paper 1), 16, 27; '490 EX1003, ¶¶69-70, 93-94; '035 Petition (Paper 1) 21, 31; '035 EX1003, ¶¶66-67, 84-85; '139 Petition (Paper 1) 19, 29-30; '139 EX1003, ¶¶66-67, 85-86); *see also* EX1003, ¶38.

Multiple Strings: Targeted Results

Searching for two or more keywords—both of which *must* appear in the results—is an excellent strategy for doing precise searches and greatly reducing the number of search results you have to consider. Some search

EX1057, 27 (quoted by '490 Petition (Paper 1), 53, 61, 67; '490 EX1003, ¶¶42, 138, 155, 162, 166, 202; '035 Petition (Paper 1), 50, 54, 60; '035 EX1003, ¶¶42, 121, 128, 135, 139; '139 Petition (Paper 1), 56, 60, 66-67; '139 EX1003, ¶¶42, 136, 143, 150, 154).

Indexes: Speed Up Search

specialized data structures to ensure faster access to the stored information. An old and popular data structure for faster information retrieval is a collection of selected words or concepts with which are associated pointers to the related information (or documents) — the *index*. In one form or another, indexes are at the core of every modern information retrieval system. They provide faster access to the data and allow the query processing task to be speeded up. A detailed

EX1023, 6 (quoted by '490 Petition (Paper 1), 56; '490 EX1003, ¶147); '035 EX1102, ¶147; '139 EX1102, ¶147); *see also* '035 Reply (Paper 27), 13-14; '035 EX1100, ¶49; '139 Reply (Paper 27), 13-14; '139 EX1100, ¶49.

Ground 1

Ground 1 Combination

Ground 1: Lotus Notes (EX1059, EX1061) + Raskin (EX1021) + Wu (EX1019)

A POSITA would have been motivated with a reasonable expectation of success to combine the teachings from the Lotus Notes references on locating indexed files containing multiple strings entered in a single search field with Raskin's teachings to provide search results incrementally (i.e., as the user types their query) and Wu's teachings to locate the files containing the multiple strings by searching each string separately and then intersecting the results. EX1003, ¶75.

'490 Petition (Paper 1), 18; '035 Petition (Paper 1), 23; '139 Petition (Paper 21), 21.

Ground 1 Combination: Disputed MTC + REOS

Feature	Lotus Notes	Raskin
Multi-String / Single Field	✓	
Incremental Results		✓
Multiple Files	✓	
Indexed Files	✓	

Ground 1 – Motivation To Combine (MTC)

MTC - Ground 1: Raskin Explains Problems With Delimited Searches

face strategies. The most common strategy is the **delimited search**, found in most word processors. In a typical delimited search, the user enters a mode in

This traditional method **is rather punishing** to the user, although most computer aficionados have become so accustomed to it that they no longer feel the pain. For example, an individual can type in a search string, make a typo—unnoticed until too late, of course, because he pressed Return by habit—and sit there, waiting for a search that he knows will fail. Most searches are not interruptible: a serious design error. Because the computer waits until the user has finished the pattern before beginning its search, delimited searches often keep him waiting unnecessarily.

EX1021, 125 (cited by '490 Petition (Paper 1), 15-16, 19, 26-28; '035 Petition (Paper 1), 20-22, 25, 30-31; '139 Petition (Paper 1), 18-20, 23, 29).

MTC - Ground 1: Raskin Explains Benefits Of Incremental Searches

Incremental searching has a number of other advantages over delimited searching. Incremental searching wastes less of a user's time: The search begins when the first character of the pattern is typed; the system does not wait until the pattern is complete. With a delimited search, the computer waits for the user to type the pattern and delimit it, after which it is the user who waits while the computer does the search. When using a delimited search, the user must guess, beforehand, at how much of a pattern the computer needs to distinguish the desired target from other, similar targets; with an incremental search, he can tell when he has typed enough to disambiguate the desired instance, because the target has appeared on the display. Thus, as soon as he sees that he has reached the point desired, he can stop entering the pattern. If he types too many letters of the pattern—if the hand is quicker than the search—the pattern will still match, and the cursor will stay approximately where he wanted it. If he mistypes a pattern in a delimited search, he must wait until the search for the incorrect pattern is complete—or, at best, he can operate a mechanism for stopping the search—before he can correct his error. In a large text, the search can take a considerable period of time. In

EX1021, 126 (cited by '490 Petition (Paper 1), 16, 27; '490 EX1003, ¶¶69-70, 93-94; '035 Petition (Paper 1) 21, 31; '035 EX1003, ¶¶66-67, 84-85; '139 Petition (Paper 1) 19, 29-30; '139 EX1003, ¶¶66-67, 85-86); *see also* EX1003, ¶38*.

*Unless otherwise stated, citations apply to all three IPRs (IPR2025-00253 ('035 patent), IPR2025-00254 ('490 patent), and IPR2025-00255 ('139 patent)).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

MTC - Ground 1: Raskin Explains Benefits Of Incremental Searches

his error. In a large text, the search can take a considerable period of time. In a well-implemented incremental search, the user can backspace at any time and be returned to the last match found. Because backspacing after an error is habitual, the process of correcting an error is very fast, and the search stops immediately. He can also resume the search by typing the correct character.

Another advantage of incremental search is that, as the user enters the pattern, it provides constant feedback as to the results of the search. With a delimited search, he does not find out whether his pattern is suitable or even if it was typed correctly until it is fully entered and a search attempt has been made. From the point of view of interface engineering, the advantages of incremental searching are so numerous and the advantages of delimited searches so few that I can see almost no occasions when a delimited search would be preferred.⁷ In spite of near universal agreement about the desir-

EX1021, 126 (cited by '490 Petition (Paper 1), 16, 27; '490 EX1003, ¶¶69-70, 93-94; '035 Petition (Paper 1) 21, 31; '035 EX1003, ¶¶66-67, 84-85; '139 Petition (Paper 1) 19, 29-30; '139 EX1003, ¶¶66-67, 85-86); *see also* EX1003, ¶38.

Ground 1 – Reasonable Expectation Of Success
(REOS): Dr. Terveen Explained The REOS

Ground 1 – REOS: Dr. Terveen Explained The REOS

96. As also explained above in Section VIII.D, a POSITA would have reasonably expected to succeed in combining these teachings from Raskin with Lotus Notes' teachings because it would have required only simple and straightforward software modifications.

'490 EX1003, ¶96 (cited by '490 Petition (Paper 1), 28); '035 EX1003, ¶77 (cited by '035 Petition (Paper 1), 25); '139 EX1003, ¶88 (cited by '139 Petition (Paper 1), 30).

Ground 1 – REOS: Dr. Terveen Explained The REOS

41. Specifically, a POSITA would have reasonably expected success in implementing the grounds' prior art combinations using a suitable programming language such as C or C++, and/or the language(s) that the Lotus Notes and Entourage programs were written in (likely C or C++). As acknowledged by Mr. Haeberli during his deposition, POSITAs were familiar with C, C++, and other programming languages needed to implement an incremental search tool, as evidenced by the existence of several prior art incremental search tools. EX1098, 21:9-15, 24:17-25:10. A POSITA would have reasonably expected success in using one or more of these programming languages to implement/modify the front-end UI for accepting user's search strings, as well as the back-end logic for building indexes and executing the incremental search using data structures such as those I mentioned in paragraph 16 above.

EX1100, ¶41 (cited by '490 Reply (Paper 28), 13; '035 Reply (Paper 27), 14; '139 Reply (Paper 27), 14).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Ground 1 – REOS: Dr. Terveen Explained The REOS (Front-End UI)

42. For Ground 2, Entourage's UI already supports an incremental search. For Ground 1, a POSITA would have reasonably expected success implementing Lotus Notes' user interface (UI) to support incremental searching using an appropriate programming language such as C or C++, or whichever programming language Lotus Notes was built on at the time (likely C or C++). Mr. Haeberli's

EX1100, ¶42 (cited by '490 Reply (Paper 28), 13-14; '035 Reply (Paper 27), 14-15; '139 Reply (Paper 27), 14-15).

Ground 1 – REOS: Dr. Terveen Explained The REOS (Front-End UI)

44. Implementing such functionality would have required little effort from a programmer. Much of the input-handling logic needed to trigger the computer to update search results was already provided by the operating system and user-interface frameworks; the programmer did not need to write the code from scratch. For example, Windows supplied predefined logic for detecting character-by-character input through the WM_CHAR message, which an application could handle to trigger incremental search and updating of search results. EX1097, 232 (WM_CHAR and WM_SYSCHAR are messages “that deliver text to the window procedure.”), 234 (“If you need to read keyboard character input in your window, you process the WM_CHAR message.”).

EX1100, ¶44 (cited by '490 Reply (Paper 28), 13-14; '035 Reply (Paper 27), 14-15; '139 Reply (Paper 27), 14-15).

Ground 1 – REOS: Dr. Terveen Explained The REOS (Front-End UI)

43. Robust programming languages such as C and C++, together with well-developed operating system and UI frameworks, however, allowed an ordinary programmer to build a UI that supported incremental searching, as evidenced by the numerous prior art systems that implemented such functionality, including Entourage, Eudora, Emacs, Canon Cat, SwyftWare, and others. *E.g.*, EX1004, 8-9, 161; EX1021 at 125, 127, 129; EX1039, 102-103; EX1043, 385-389. Indeed, Mr. Haeberli acknowledged as much during his deposition. EX1098, 21:9-15, 24:17-25:10 (agreeing that a POSITA would have been familiar with C and C++ in 2002 and that a POSITA would have been able to implement a dynamic text box in C or C++, as evidenced by at least Emacs).

EX1100, ¶43 (cited by '490 Reply (Paper 28), 13-14; '035 Reply (Paper 27), 14-15; '139 Reply (Paper 27), 14-15).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Ground 1 – REOS: Dr. Terveen Explained The REOS (Back-End Logic)

46. A POSITA also would have reasonably expected success implementing the back-end logic for building indexes and executing the incremental search using data structures such as those I mentioned in paragraph 16 above and programming in a suitable programming language such as C or C++.

EX1100, ¶46 (cited by '490 Reply (Paper 28), 13-14; '035 Reply (Paper 27), 14-15; '139 Reply (Paper 27), 14-15).

Ground 1 – REOS: Dr. Terveen Explained The REOS (Back-End Logic)

Specifically, the programmer would reasonably expect to use a suitable programming language to look up a set of matches for the first search string in the index data structure, look up a set of matches for the second search string in the index data structure, and then intersect the results to find the list of files that contain both search strings, as taught by Wu and explained in my Initial Declaration. EX1019, 6:51-66, 8:66-11:32, FIG. 6; EX1003, ¶¶74, 79, 92, 97. When the user adds a character to one of the search strings, the tool would only need to narrow the already-found results for that search string (i.e., locates a set of matches within the first set of matches for the search string), and then intersect the results again with the already-found matches for the other search string. For example, when implementing

EX1100, ¶47 (cited by '490 Reply (Paper 28), 13; '035 Reply (Paper 27), 14; '139 Reply (Paper 27), 14).

Ground 1 – REOS: Dr. Terveen Explained The REOS (Back-End Logic)

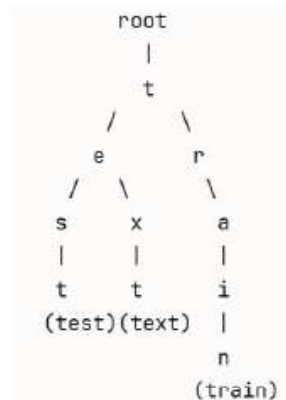
already-found matches for the other search string. For example, when implementing Lotus Notes' full-text index to perform an incremental search, the index data structure could include associations for each prefix or only whole words. If the data structure indexes only whole words, the search tool would simply join the records of the words that have the partial search string. For example, if Lotus Notes' full-text index is represented as a trie data structure (as discussed in Baeza-Yates and an example of which is shown below), each leaf node would include an indication of the files containing the word. *See, e.g.,* EX1023, 195, 199-200. Thus, in the below exemplary trie, the leaf node for "test" would include a list of the files containing the word "test," the leaf node for "text" would include the list of files containing the word "text," and the leaf node for "train" would include the list of files containing the word "train." If a user enters the single character "t," the computer simply joins

EX1100, ¶47 (cited by '490
Reply (Paper 28), 13; '035
Reply (Paper 27), 14; '139
Reply (Paper 27), 14).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Ground 1 – REOS: Dr. Terveen Explained The REOS (Back-End Logic)

the lists of files associated with these three words. When an “e” is received after the “t” (such that the search string is “te”), the computer joins the lists of files associated with “test” and “text,” and so on and so forth. Alternatively, each node/position in the trie could include a list of the associated files containing the prefix so that the computer does not have to join any lists.



EX1100, ¶47 (cited by '490
Reply (Paper 28), 13; '035
Reply (Paper 27), 14; '139
Reply (Paper 27), 14).

Ground 1 – REOS: Incrementally Searching Across Files Known

Ground 1 – REOS: Raskin Already Combined Incremental Search And Multiple Files

Incremental Search Feasible At Scale Using Techniques Such As Lookahead And Indexing

The speed of incremental searches is increased when a few implementation tricks are used; for example, when you type the first character of fast. Indexing all local mass-storage devices can make searching over a local system or network take only milliseconds. Interaction at high speeds, with

EX1021, 129 (quoted by '490 Reply (Paper 28), 6; '035 Reply (Paper 27), 6; '139 Reply (Paper 27), 6; EX1100, ¶21); *see also* '490 Petition (Paper 1), 16; '490 EX1003, ¶68; '035 Petition (Paper 1), 21; '035 EX1003, ¶65; '139 Petition (Paper 1), 19; '139 EX1003, ¶65).

*In larger texts, a search can proceed circularly not only through the local document but also afterward in automatically expanding domains, up to and including the entire Internet.*¹⁰ After the local document has been completely searched,

10. It is entirely possible, by such techniques as lookahead and indexing, to maintain the 250 msec response time.

EX1021, 132, n.10 (quoted by '490 Reply (Paper 28), 6; '035 Reply (Paper 27), 6; '139 Reply (Paper 27), 6; EX1100, ¶21); *see also* '490 Petition (Paper 1), 16; '490 EX1003, ¶68; '035 Petition (Paper 1), 21; '035 EX1003, ¶65; '139 Petition (Paper 1), 19; '139 EX1003, ¶65).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Ground 1 – REOS: Raskin Already Combined Incremental Search And Multiple Files

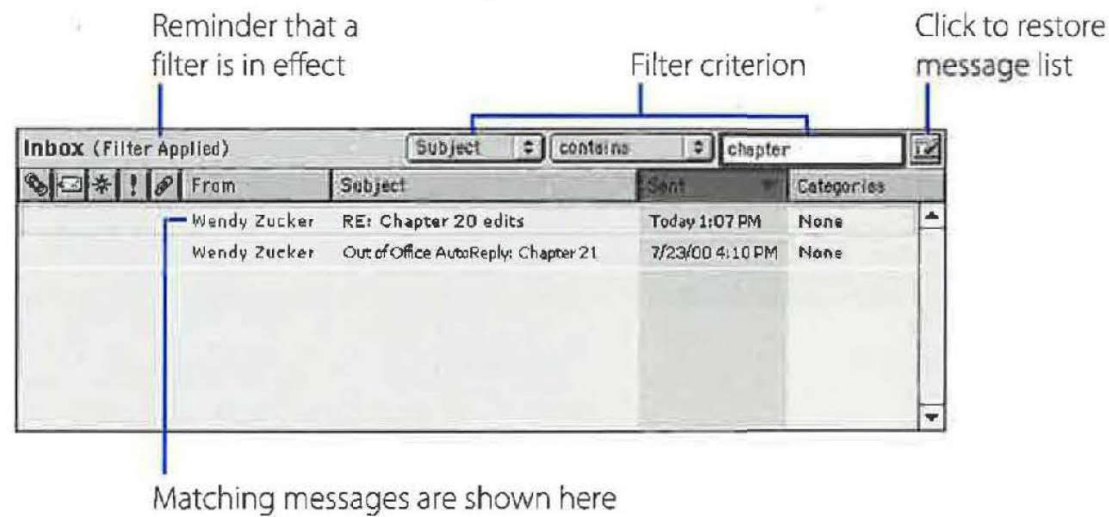
Generally, people will use more efficient strategies than simply relying on this expanding sweep of the hierarchy. For example, if you are looking for a particular document in the current folder, you would be likely to do a repeated search for document characters and would thus quickly be able to review the beginning of, and any heading on, each document. When the desired document is found, an incremental search for the target is initiated. This guarantees that the chosen document is searched first; the advantage of this is that you can typically use a shorter pattern if the area to be searched is smaller. If you did not know in what document the target resided or if you did not want to look for the document, you could perform a specific search from the beginning, which would find the target anyway.

EX1021, 132 (cited by '490 Sur-Reply (Paper 38), 16; '035 Sur-Reply (Paper 37), 17-18; '139 Sur-Reply (Paper 37), 16-17); *see also* '490 Reply (Paper 28), 6; '035 Reply (Paper 27), 6; '139 Reply (Paper 27), 6.

Ground 1 – REOS: Incremental Search Across Files/Documents Known

Entourage (State Of The Art)

- 3** When searching within the Subject, From, or To fields, enter a search string in the filter criteria text box. The search is conducted as you type, and the matching headers are displayed in the message list.



EX1043, 385 (cited by '490 Petition (Paper 1), 64; '490 EX1003, ¶160; '035 Petition (Paper 1), 57-58; '035 EX1003, ¶133; '139 Petition (Paper 1), 63-64; '139 EX1003, ¶148); EX1100, ¶31); *see also* EX1004, 8-9, 38, 161; EX1005, 173; EX1006, 384 (cited by '490 Petition (Paper 1), 64-66; '490 EX1003, ¶160; '035 Petition (Paper 1), 57-59; '035 EX1003, ¶133; '139 Petition (Paper 1), 63-65; '139 EX1003, ¶148; EX1100, ¶31).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Ground 1 – REOS: Incremental Search Across Files/Documents Known

Eudora (State Of The Art)

If you type something while a mailbox is the active window, Eudora shows the message with the Who or Subject entry that is alphabetically closest to what you typed. (Eudora uses the Who column unless you have sorted by Subject.) For example, if you sort by subject and type t, the mailbox will change to highlight a message whose Subject starts with T. If no messages start with T, Eudora will

EX1039, 102 (cited by EX1003, ¶¶33, 37; EX1100, ¶24); *see also* '490 Petition (Paper 1), 6-7 (citing EX1003, ¶¶33, 37); '490 Reply (Paper 28), 7-8 (citing EX1100, ¶24); '035 Petition (Paper 1), 9-10 (citing EX1003, ¶¶33, 37); '035 Reply (Paper 27), 8 (citing EX1100, ¶24); '139 Petition (Paper 1), 8 (citing EX1003, ¶¶33, 37); '139 Reply (Paper 28), 8 (citing EX1100, ¶24).

If you type j in your inbox, Eudora will highlight a message from Jeff Chee. If you type ja (quickly), Eudora will highlight a message from Andy Jain.

EX1039, 103 (quoted by EX1100, ¶24) (cited by EX1003, ¶¶33, 37); *see also* '490 Petition (Paper 1), 6-7 (citing EX1003, ¶¶33, 37); '490 Reply (Paper 28), 7-8 (citing EX1100, ¶24); '035 Petition (Paper 1), 9-10 (citing EX1003, ¶¶33, 37); '035 Reply (Paper 27), 8 (citing EX1100, ¶24); '139 Petition (Paper 1), 8 (citing EX1003, ¶¶33, 37); '139 Reply (Paper 28), 8 (citing EX1100, ¶24).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Ground 1 – REOS: Incremental Search Across Files/Documents Known

Spaey (State Of The Art)

31. Spaey discloses performing an incremental search across Internet-based files, such as files containing information about CDs, which Mr. Haerberli does not dispute. *E.g.*, EX1012, [0026], [0051-53], [0167] (“Incremental search, i.e. the search process on the server 14 is triggered as soon as a character 68 is typed and narrows down synchronously with each additional character 70(see FIGS. 8A, 8B & 8C).”), [0247], FIGS. 3, 8A-9. For example, typing in the letter “r” yields a first set of search results (*id.*, FIG. 8A), “re” yields a smaller second set of search results (*id.*, FIG. 8B), and “ree” yields an even smaller third set of results (*id.*, FIG. 8C).

EX1100, ¶31 (cited by '490 Reply (Paper 28), 7; '035 Reply (Paper 27), 8; '139 Reply (Paper 27), 8); *see also* EX1003, ¶¶ 33, 37 (cited by '490 Petition (Paper 1), 6-7; '035 Petition (Paper 1), 9-10; '139 Petition (Paper 1), 8).

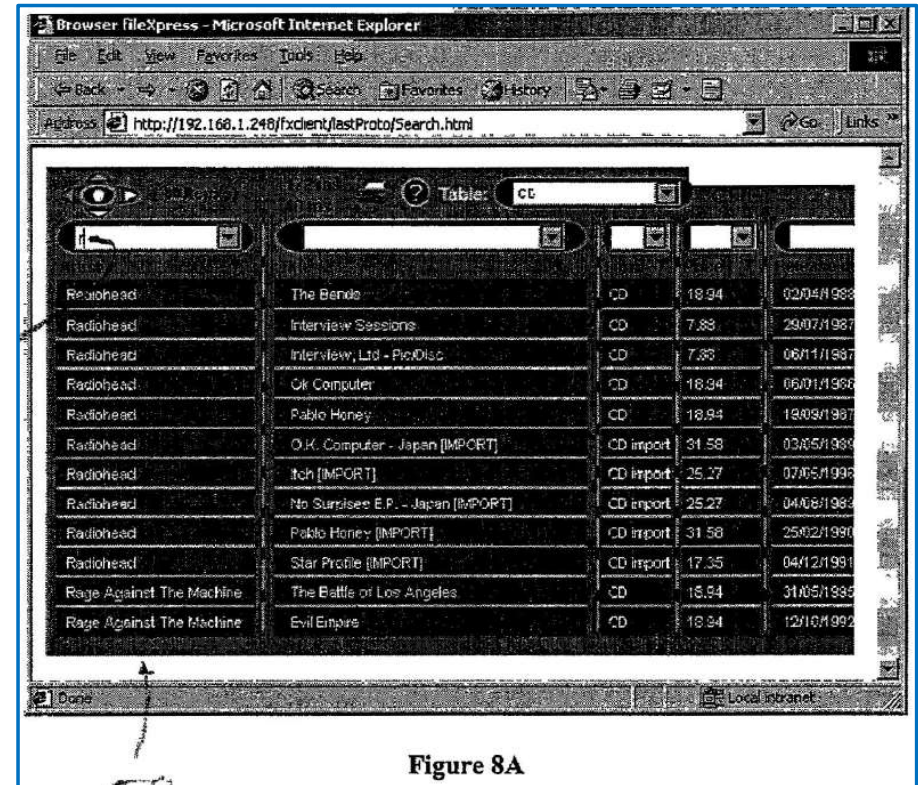


Figure 8A

EX1012, FIG. 8A (cited by EX1100, ¶31; EX1003, ¶¶33, 37); *see also* '490 Petition (Paper 1), 6-7 (citing EX1003, ¶¶33, 37); '490 Reply (Paper 28), 7-8 (citing EX1100, ¶31); '035 Petition (Paper 1), 9-10 (citing EX1003, ¶¶33, 37); '035 Reply (Paper 27), 8 (citing EX1100, ¶31); '139 Petition (Paper 1), 8 (citing EX1003, ¶¶33, 37); '139 Reply (Paper 28), 8 (citing EX1100, ¶31).

Ground 1 – REOS: Incremental Search Across Files/Documents Known

Dr. Terveen Discussing Nguyen (State Of The Art)

28. Nguyen discloses incrementally updating search results for “discrete cases” (i.e., “discrete record entities”) “each relating to a separate topic,” such as “solution[s] to [] problem[s] commonly occurring in network printers.” EX1010, 1:60-61, 5:39-47; *see also id.*, 7:14-8:66, FIGS. 2-4B. Nguyen “initiates a search” “if a keystroke ... is detected” (*id.*, 7:33-36) and “[i]f additional keystrokes are detected, the method again conducts ... searches ... as modified by the additional keystroke” (*id.*, 8:34-39). For example, “[t]he first search would be performed on the keystroke ‘p’ and would produce a voluminous number of matches ... [T]he next keystroke [] modify[ing] the search field to ‘pr’ ... would significantly lower the ranking of the POST procedure topic” and a “third keystroke [] modify[ing] the search field to ‘pri’ [wou]ld lower[] the ranking of the various ‘processor’ topics.” *Id.*, 7:68-8:47. “In this manner, by continuing to input additional keystrokes ... the results of each search would be narrowed for each keystroke” *Id.*, 8:51-57. Mr.

EX1100, ¶28 (cited by ’490 Reply (Paper 28), 7; ’035 Reply (Paper 27), 8; ’139 Reply (Paper 27), 8); *see also* EX1003, ¶¶ 33, 37 (cited by ’490 Petition (Paper 1), 6-7; ’035 Petition (Paper 1), 9-10; ’139 Petition (Paper 1), 8).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Ground 1 – REOS: Incremental Search Across Files/Documents Known

Siitonen (State Of The Art)

As the user types in the search key, the PDA virtually instantly displays the items matching the search found in the contact data base. The user can refine the search by adding additional search criteria until finally producing for viewing a minimum number of data base records matching the search criteria. For example, if the user types the letter “j” all records having names beginning with the letter “j” appear. The user may continue to type additional letters defining a name, for example, the pair of letters “on” chooses records such as “Jones” but not records such as “Jackson”. Further typing the letter “a” would eliminate “Jones” as a possibility and display any names having as their first four letters “jona” such as “Jonathan”. This method of searching is referred to

EX1016, 2:51-63 (quoted by EX1100, ¶29) (cited by EX1003, ¶¶33, 37) *see also* '490 Petition (Paper 1), 6-7 (citing EX1003, ¶¶33, 37); '490 Reply (Paper 28), 7-8 (citing EX1100, ¶29); '035 Petition (Paper 1), 9-10 (citing EX1003, ¶¶33, 37); '035 Reply (Paper 27), 8 (citing EX1100, ¶29); '139 Petition (Paper 1), 8 (citing EX1003, ¶¶33, 37); '139 Reply (Paper 28), 8 (citing EX1100, ¶29).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Ground 1 – REOS: Lotus Notes' Database Suitable For Incremental Search

Ground 1 – REOS: Lotus Notes' Index = 5 MB

◆ **Disk space.** Indexing a database takes up more disk space — maybe an additional 50 percent (or more) of the database's size. In other words, if your e-mail database takes up, say, 10MB of disk space, indexing could increase the size to 15MB. Predicting exactly how big the index is going to be is hard, because the size depends on the type of database you're indexing. For a database that contains a lot of pictures and a relatively small amount of text, the index won't be as big (percentage-wise) as it would be for a database that's almost all text. Of course, if the database you want to index is on a server, rather than on your own computer, you don't need to be as conscious of the disk-space requirements, because you're not going to use any of your hard-disk space — the index is stored with the Notes database on the server.

EX1061, 136 (cited by '490 Petition (Paper 1), 21; '490 Reply (Paper 28), 7; '139 Reply (Paper 27), 7; '035 Reply (Paper 27), 7); *see also* '139 Petition (Paper 1), 15, 25; '035 Petition (Paper 1), 17, 27.

Ground 1 – REOS: Index For 4 MB Database = 200 KB (Dr. Terveen, EX1100)

14. Re-running Mr. Haeberli's same analysis (which assumes the index is 5% the size of the database), an index for a four-megabyte database would have been only 200 kilobytes—0.16% the 128-megabyte RAM capacity of 2002-era computers. Thus, 2002-era computers would not have had any issues performing an incremental search across files.

EX1100, ¶14 (cited by '490 Reply (Paper 28), 5, 7; '035 Reply (Paper 27), 5-7; '139 Reply (Paper 27), 5-7).

Ground 1 – REOS: The Patents Do Not
Explain How To Solve PO's Alleged
Scalability Problem

Ground 1 – REOS: Challenged Patents Describe A Standard Inverted Index

Challenged Patents

The index engine **103** can utilize one or more indexing algorithms to create an index, such as a reverse or **inverted index**. The index includes a data structure that associates character strings with files, documents, and the like. In one example embodiment, for each word or character string found with a file or document, the index stores which fields of which documents or files contain that word or character string.

'490 EX1001, 5:29-36 (cited by Sur-Reply (Paper 38), 20); '035 EX1001, 5:16-23 (cited by Sur-Reply (Paper 37), 22); '139 EX1001, 5:25-32 (cited by Sur-Reply (Paper 37), 21).

1994-Era Prior Art

Document Index 201	
Keyword 202	(Document Identifier, Occurrence) 203 204
Areas	(B, 12)
Cases	(A, 6)
Congress	(B, 1)
Court	(A, 10)
Courts	(A, 3, 16) (B, 6)
Created	(B, 3)
Disputes	(B, 9)
District	(A, 15)
Federal	(A, 9, 14) (B, 5)
Handle	(B, 8)
Law	(B, 14)
Most	(A, 5)
Particular	(B, 11)
Specialized	(B, 4)
System	(A, 11)
Trial	(A, 2)

EX1054, FIG. 2 (cited by '490 EX1003, ¶46); *see also* '035 EX1102, ¶46 (cited by '035 EX1100, ¶15); '139 EX1102, ¶46 (cited by '139 EX1100, ¶15).

Ground 1 – REOS: Challenged Patents Describe Standard Use Of The Index

Challenged Patents

FIG. 2B illustrates an example search lookup operation. At state 202B one or more search characters or strings are received. At state 204B the character or character strings are located in the words file. At state 206B the pointer or pointers for the character or character strings in the occur-offsets file are located. At state 208B document-occurrence information is read from the occur file using the located pointers. At state 210B the search results are presented to the user. The search results are displayed in the list pane or area.

'490 EX1001, 7:1-9 (cited by Sur-Reply (Paper 38), 20; '035 EX1001, 6:57-65 (cited by Sur-Reply (Paper 37), 22; '139 EX1001, 6:64-7:5 (cited by Sur-Reply (Paper 37), 21).

Terveen's Reply Declaration

51. Moreover, I see nothing in the challenged patent that describes how to scale up its incremental search to the large volumes that Mr. Haeberli seems to allege are required. EX2001, ¶105; EX2020, ¶52. For example, the patent describes how the initial search is in the 'words' file, which stores all the words that occur in the target files. EX1001, 7:1-4. There is no discussion as to how to prevent that file from becoming too large to provide results fast enough to perform the incremental search when searching across a large volume of files.

EX1100, ¶51 (cited by '490 Reply (Paper 28), 13; '035 Reply (Paper 27), 14; '139 Reply (Paper 27), 14).

Ground 1 – REOS: Challenged Patents Describe Standard Prefix Indexing

Challenged Patents

In one optional embodiment, the index stores “prefix” entries, which further enhances the speed of incremental filtering or searching. For example, in addition to bare or complete words, such as “dog,” common prefixes, such as “d” or “do”, are stored as well. The “d” prefix entry contains information indicating which documents or files contain a word or character string starting with “d”. By preparing this information during indexing, the response for the first letter or two of a filtering or search operation is greatly enhanced as compared to a standard inverted index, which would have to locate thousands of matches at the time of receiving the user query, thereby resulting in slow filtering operations.

’490 EX1001, 7:37-48 (cited by ’490 Sur-Reply (Paper 38), 20);
’035 EX1001, 7:26-37 (cited by ’035 Sur-Reply (Paper 37), 22);
’139 EX1001, 7:33-44 (cited by ’139 Sur-Reply (Paper 37), 21).

Terveen’s Reply Declaration

52. The patent description of storing prefixes suffers from the same problem. EX1001, 7:37-48. This would result in an even larger index, there is no discussion of how exactly that index would be stored, and there certainly is no explanation as to how that index would provide results quick enough to perform an incremental search when searching across a large volume of files. Further, searching the index incrementally would result in a massive set of initial results. For example, because many common words start with ‘a’ (e.g., an, and, etc.) a search of ‘a’ would result in a massive set of results when searching across a large volume of files (i.e., every or nearly every file in the database would be a match), and the challenged patent does not include any discussion of how the system would handle that initial massive set of results.

EX1100, ¶52 (cited by ’490 Reply (Paper 28), 13; ’035 Reply (Paper 27), 14; ’139 Reply (Paper 27), 14).

Ground 1: MTC + REOS

Feature	Lotus Notes	Raskin
Multi-String / Single Field	✓	
Incremental Results		✓
Multiple Files	✓	
Indexed Files	✓	

Ground 2

Ground 2 Combination

Ground 2: Entourage (EX1004-EX1006) + True (EX1009) + Baeza-Yates (EX1023)

The Entourage+True+Baeza-Yates combination combines the Entourage references' teachings to incrementally locate files, including emails, containing at least one search string entered in a search field with True's teachings to do so for multiple search strings entered in the search field and Baeza-Yates teachings to do so by searching each string separately in an index data structure and then intersecting the results. EX1003, ¶137.

'490 Petition (Paper 1), 53; '035 Petition (Paper 1), 50; '139 Petition (Paper 1), 56.

Ground 2 Combination: Disputed MTC + REOS

Feature	Entourage	True	Baeza-Yates
Multi-String Search		✓	
Incremental Search	✓	✓	
Inter-file Search	✓		
Index			✓

Ground 2 – Entourage Teaches Incrementally Searching Across Files/Documents

Ground 2: Entourage Teaches Incrementally Searching Across Files/Documents

Although it is not as sophisticated as the Find and Advanced Find commands discussed later in this section, filtering is the fastest way to perform a simple search.

EX1004, 8 (cited by '490 Petition (Paper 1), 64-66; '490 EX1003, ¶160; '035 Petition (Paper 1), 57-59; '035 EX1003, ¶133; '139 Petition (Paper 1), 63-65; '139 EX1003, ¶148; EX1100, ¶31).

EX1004, 9 (cited by '490 Petition (Paper 1), 64-66; '490 EX1003, ¶160; '035 Petition (Paper 1), 57-59; '035 EX1003, ¶133; '139 Petition (Paper 1), 63-65; '139 EX1003, ¶148; EX1100, ¶31).

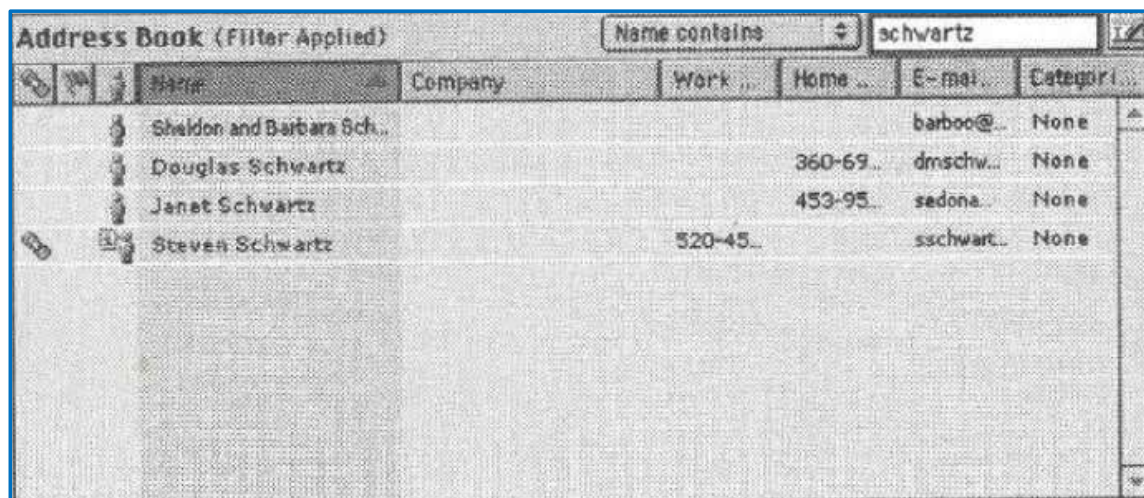
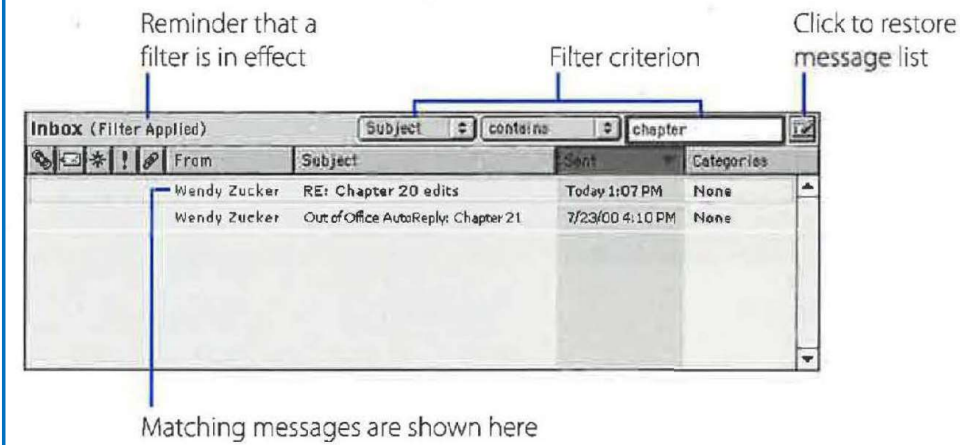


Figure 1.12 As you type the filter text, only matches are shown in the list.

Ground 2: Entourage Teaches Incrementally Searching Across Files/Documents

- 3** When searching within the Subject, From, or To fields, enter a search string in the filter criteria text box. The search is conducted as you type, and the matching headers are displayed in the message list.



EX1043, 385 (cited by '490 Petition (Paper 1), 64; '490 EX1003, ¶160; '035 Petition (Paper 1), 57-58; '035 EX1003, ¶133; '139 Petition (Paper 1), 63-64; '139 EX1003, ¶148); EX1100, ¶31); *see also* EX1004, 8-9, 38, 161; EX1005, 173; EX1006, 384 (cited by '490 Petition (Paper 1), 64-66; '490 EX1003, ¶160; '035 Petition (Paper 1), 57-59; '035 EX1003, ¶133; '139 Petition (Paper 1), 63-65; '139 EX1003, ¶148; EX1100, ¶31).

Ground 2 – Motivation To Combine (MTC)

Ground 2 – MTC: Multi-String Search Provides More Targeted Results

Dr. Terveen's Initial Declaration

166. Thus, it would have been obvious to a POSITA to incrementally locate files containing two partial search strings (as opposed to just one search string) to provide the user with more relevant results, as is suggested by the Entourage references and was known to a POSITA. *Supra* Section VI.B.2;

'490 EX1003, ¶166 (cited by '490 Petition (Paper 1), 69); '035 EX1003, ¶139 (cited by '035 Petition (Paper 1), 63); '139 EX1003, ¶154 (cited by '139 Petition (Paper 1), 69).

162. Moreover a POSITA would have been motivated to incorporate True's teachings to incrementally locate (i.e., locate as the user types or modifies their search query) files containing multiple search strings entered in the same search field (e.g., files containing the strings "queen" and "mother") to provide more targeted searching, as was common sense and well known in the art.

'490 EX1003, ¶162 (cited by '490 Petition (Paper 1), 67); '035 EX1003, ¶135 (cited by '035 Petition (Paper 1), 60); '139 EX1003, ¶150 (cited by '139 Petition (Paper 1), 66-67).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Ground 2 – MTC: Multi-String Search Provides More Targeted Results

Glossbrenner (State Of The Art)

Searching for two or more keywords—both of which *must* appear in the results—is an excellent strategy for doing precise searches and greatly reducing the number of search results you have to consider. Some search

EX1057, 27 (quoted by '490 Petition (Paper 1), 53, 61, 67; '490 EX1003, ¶¶42, 138, 155, 162, 166, 202; '035 Petition (Paper 1), 50, 54, 60; '035 EX1003, ¶¶42, 121, 128, 135, 139; '139 Petition (Paper 1), 56, 60, 66-67; '139 EX1003, ¶¶42, 136, 143, 150, 154).

Ground 2 – MTC: Benefits Of Multi-String Queries Were Known

Raskin (State Of The Art): Multi-String Searches Benefit From Being Incremental

strategies through the feedback received. Even building a Boolean search pattern is made more effective when the early results of the pattern appear as the user adds to the pattern's specificity. The found instance should appear in

EX1021, 127 (quoted by '490 EX1003, ¶166 (cited by '490 Petition (Paper 1), 69); '035 EX1003, ¶139 (cited by '035 Petition (Paper 1), 63); '139 EX1003, ¶154 (cited by '490 Petition (Paper 1), 69)).

Ground 2 – MTC: Multi-String Search Provides More Targeted Results

Reply

43-48, 54-91. As also explained in the Petition, the highlighting functionality in the Find tool and multi-string search capability of the Advanced Find tool had known benefits: multi-string queries provide more targeted search results and search term highlighting directs the user to the relevant portion(s) of the results and gives the user more confidence in the results. Paper 1, 8, 59-61, 80-84; EX1003, ¶¶48-52, 138, 155, 188-194. Thus, a POSITA would have been motivated with a reasonable

expectation of success to combine these well-known functionalities from Entourage's search tools to achieve their known benefits. EX1100, ¶¶59-61.

'490 Reply (Paper 28), 18-19; '035 Reply (Paper 27), 19-20; '139 Reply (Paper 27), 19-20.

Dr. Terveen's Supplemental Declaration

describes this functionality in the Advanced Find tool. If anything, a POSITA would understand that such multi-string functionality would be inherent and/or easy to include in the Quick Find tool because such functionality utilizes the same or similar approach for performing a single-string search and would not expect it to require significant extra computing power or time. For example, if the user types "c," the Quick Find tool returns results containing the words that start with "c." If the user then types a second string "h" (so that the query is "c h"), the Quick Find tool would locate emails containing words that start with "h" using the same approach that it did to find emails containing words that start with "c." Then, as was well known in

the art, the tool would simply intersect the results (e.g., perform a Boolean AND operation). EX1003, ¶¶42-43, 71, 74-75, 79, 81, 92, 97-98, 139, 167-171.

EX1100, ¶60 (cited by '490 Reply (Paper 28), 19; '035 Reply (Paper 27), 20; '139 Reply (Paper 27), 20).

Ground 2 – MTC: Baeza-Yates' Indexing Improves Search Speed

Dr. Terveen's Initial Declaration

191-228. A POSITA would have been motivated to incorporate Baeza-Yates' teachings to implement an index data structure comprising associations between each of a plurality of character strings to associated files in which the character strings are included to expedite the search process, as taught by Baeza-Yates and was well known in the art. *Supra* Section VI.B.3.

'490 EX1003, ¶146 (cited by '490 Petition (Paper 1), 55); *see also* '490 EX1003, ¶¶45-47 (cited by '490 Petition (Paper 1), 8; '490 Reply (Paper 28), 6), ¶148 (cited by Petition (Paper 1), 56).

Baeza-Yates

specialized data structures to ensure faster access to the stored information. An old and popular data structure for faster information retrieval is a collection of selected words or concepts with which are associated pointers to the related information (or documents) — the *index*. In one form or another, indexes are at the core of every modern information retrieval system. They provide faster access to the data and allow the query processing task to be speeded up. A detailed

EX1023, 6 (quoted by '490 Petition (Paper 1), 56; '490 EX1003, ¶147).

Ground 2 – Reasonable Expectation Of Success (REOS): Dr. Terveen Explained The REOS

Ground 2 – REOS: Dr. Terveen Explained The REOS

167. When implementing True's teachings to search two search strings, a POSITA would have been motivated with a reasonable expectation of success to incorporate Baeza-Yates' teachings to locate files containing the two search strings by independently searching each string and then taking the intersection of those results (i.e., locating a first set of files that each includes at least one string that corresponds to the first partial search string, locating a second set of files that each includes at least one string that corresponds to the second partial search string, and identifying one or more files that are included in both the first and second sets of files) at least because this was one of only a limited number (a few at most) of known ways to identify files containing more than one search string, it was simple, predictable and easy to implement, and was especially well suited for Entourage's incremental search. EX1023, 101-103, 195-196, 207-208; *supra* Sections VI.B.2-VI.B.3.

'490 EX1003, ¶167 (cited by '490 Petition (Paper 1), 70); '035 EX1003, ¶140 (cited by '035 Petition (Paper 1), 64); '139 EX1003, ¶155 (cited by '139 Petition (Paper 1), 70); *see also* '490 EX1003, ¶168-172 (cited by '490 Petition (Paper 1), 71-72); '035 EX1003, ¶141-144 (cited by '035 Petition (Paper 1), 65-66); '139 EX1003, ¶156-159 (cited by '139 Petition (Paper 1), 71-72).

Ground 2 – REOS: Dr. Terveen Explained The REOS

48. For Ground 2, a POSITA would have reasonably expected success in using a suitable programming language (e.g., the language Entourage was written in) to implement an index data structure, as taught by Baeza-Yates, to locate files (e.g., emails) containing two search strings. As I explained above in paragraphs 15-17, a POSITA would have reasonably expected an inverted index or other well-known index data structure (e.g., tries, binary trees) described by Baeza-Yates to store prefixes and/or allow for quick prefix results to be able to deliver character-at-a-time results (to the extent the claims require this), especially when searching across smaller file sizes and smaller sets of files.

EX1100, ¶48 (cited by '490 Reply (Paper 28), 12-13; '035 Reply (Paper 27), 13; '139 Reply (Paper 27), 13).

Ground 2 – REOS: Dr. Terveen Explained The REOS

49. Specifically, the programmer would reasonably expect to use a suitable programming language to look up a set of matches for the first search string in the index data structure, look up a set of matches for the second search string in the index data structure, and then intersect the results to find the list of files that contain both search strings, as taught by Baeza-Yates and explained in my Initial Declaration. EX1023, 5-27, 101-105, 165-166, 191-228; EX1003, ¶¶136, 139, 146-148, 167-168. See my analysis above (paragraph 47) for Ground 1, which applies equally here for Ground 2.

EX1100, ¶49 (cited by '490 Reply (Paper 28), 12-13; '035 Reply (Paper 27), 13; '139 Reply (Paper 27), 13).

Ground 2 – REOS: PO's Large Database Argument Is Misplaced

Ground 2 – REOS: Median Email Size = 4.4 KB (EX1099)

Table 2 Size of emails 2002 and 2003

Year	Email size statistics (bytes)				
	Min	Max	Mean	Median	Standard deviation
2002	177	6 751 428	49 997	2228	194 912
2003	172	9 989 173	128 276	4435	549 506

EX1099, 190 (cited by '490 Reply (Paper 28), 4; '035 Reply (Paper 27), 4; '139 Reply (Paper 27), 4); EX1100, ¶13).

13. *Third*, 2002-era computers were entirely capable of incrementally searching across a plurality of (i.e., two or more) files, especially with the help of indexes. For example, the median size of two emails in 2002 was approximately 4.4 kilobytes—0.0044% of the alleged 128-megabyte RAM capacity of 2002-era computers. EX1099, 190; EX2020, ¶48. Mr. Haeberli's assertion that 2002-era

EX1100, ¶13 (cited by '490 Reply (Paper 28), 5, 7; '035 Reply (Paper 27), 6-7; '139 Reply (Paper 27), 6-7).

Ground 2 – REOS: Index For 4 MB Database = 200 KB (Dr. Terveen, EX1100)

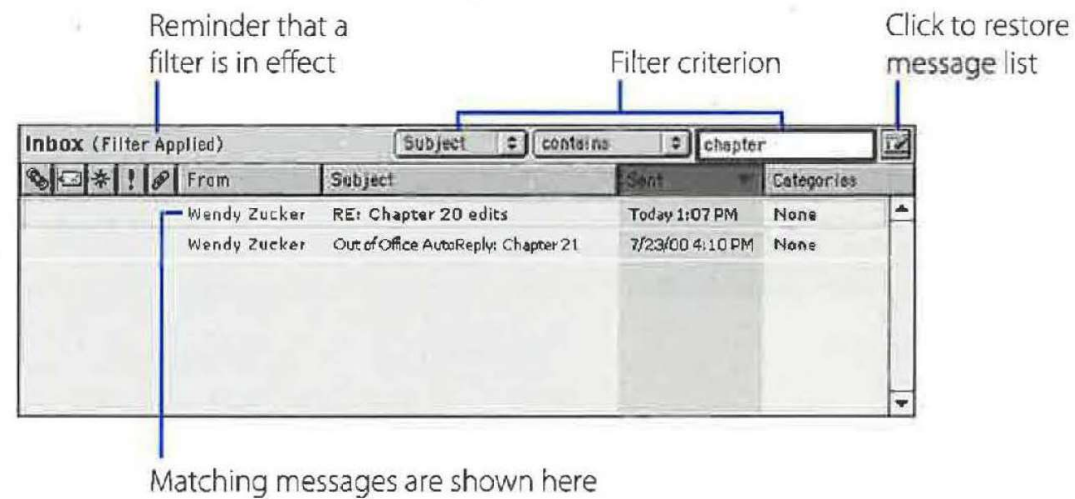
14. Re-running Mr. Haeberli's same analysis (which assumes the index is 5% the size of the database), an index for a four-megabyte database would have been only 200 kilobytes—0.16% the 128-megabyte RAM capacity of 2002-era computers. Thus, 2002-era computers would not have had any issues performing an incremental search across files.

EX1100, ¶14 (cited by '490 Reply (Paper 28), 5, 7; '035 Reply (Paper 27), 6-7; '139 Reply (Paper 27), 6-7).

Ground 2 – REOS: Entourage Already Teaches Incrementally Searching Across Files/Documents

Entourage

- 3** When searching within the Subject, From, or To fields, enter a search string in the filter criteria text box. The search is conducted as you type, and the matching headers are displayed in the message list.



EX1043, 385 (cited by '490 Petition (Paper 1), 64; '490 EX1003, ¶160; '035 Petition (Paper 1), 57-58; '035 EX1003, ¶133; '139 Petition (Paper 1), 63-64; '139 EX1003, ¶148); EX1100, ¶31); *see also* EX1004, 8-9, 38, 161; EX1005, 173; EX1006, 384 (cited by '490 Petition (Paper 1), 64-66; '490 EX1003, ¶160; '035 Petition (Paper 1), 57-59; '035 EX1003, ¶133; '139 Petition (Paper 1), 63-65; '139 EX1003, ¶148; EX1100, ¶31).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Ground 2 – REOS: Indexing Supports Incremental Searches Across Files

Raskin (State Of The Art)

10. It is entirely possible, by such techniques as lookahead and indexing, to maintain the 250 msec response time.

EX1021, 132 n.10 (quoted by '490 Reply (Paper 28), 6; '035 Reply (Paper 27), 6; '139 Reply (Paper 27), 6; EX1100, ¶21); *see also* '490 Petition (Paper 1), 16; '490 EX1003, ¶68; '035 Petition (Paper 1), 21; '035 EX1003, ¶65; '139 Petition (Paper 1), 19; '139 EX1003, ¶65).

Ground 2: MTC + REOS

Feature	Entourage	True	Baeza-Yates
Multi-String / Single Field		✓	
Incremental Results	✓	✓	
Multiple Files	✓		
Indexed Files			✓

Appendix

Highlighting Claims Are Obvious

A User Interface Can Include Multiple Windows, As Acknowledged By PO's Expert

19	Q. Would a POSITA in 2002 know what a user
20	interface is?
21	A. Yes.

EX1098, 18:19-21 (cited by '490 Reply (Paper 28), 24; '035 Reply (Paper 27), 24; '139 Reply (Paper 27), 24).

13	Is a graphical user interface limited to only
14	one window?
15	A. No.

EX1098, 19:13-15 (cited by '490 Reply (Paper 28), 24; '035 Reply (Paper 27), 24; '139 Reply (Paper 27), 24).

19	Q. So it's your opinion that a -- a user
20	interface can include multiple windows?
21	MS. DAUGHTREY: Object to form.
22	THE WITNESS: Yes, although even the notion of

EX1098, 19:19-22 (cited by '490 Reply (Paper 28), 24; '035 Reply (Paper 27), 24; '139 Reply (Paper 27), 24).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Entourage Displays Selected File/Document In Same Window

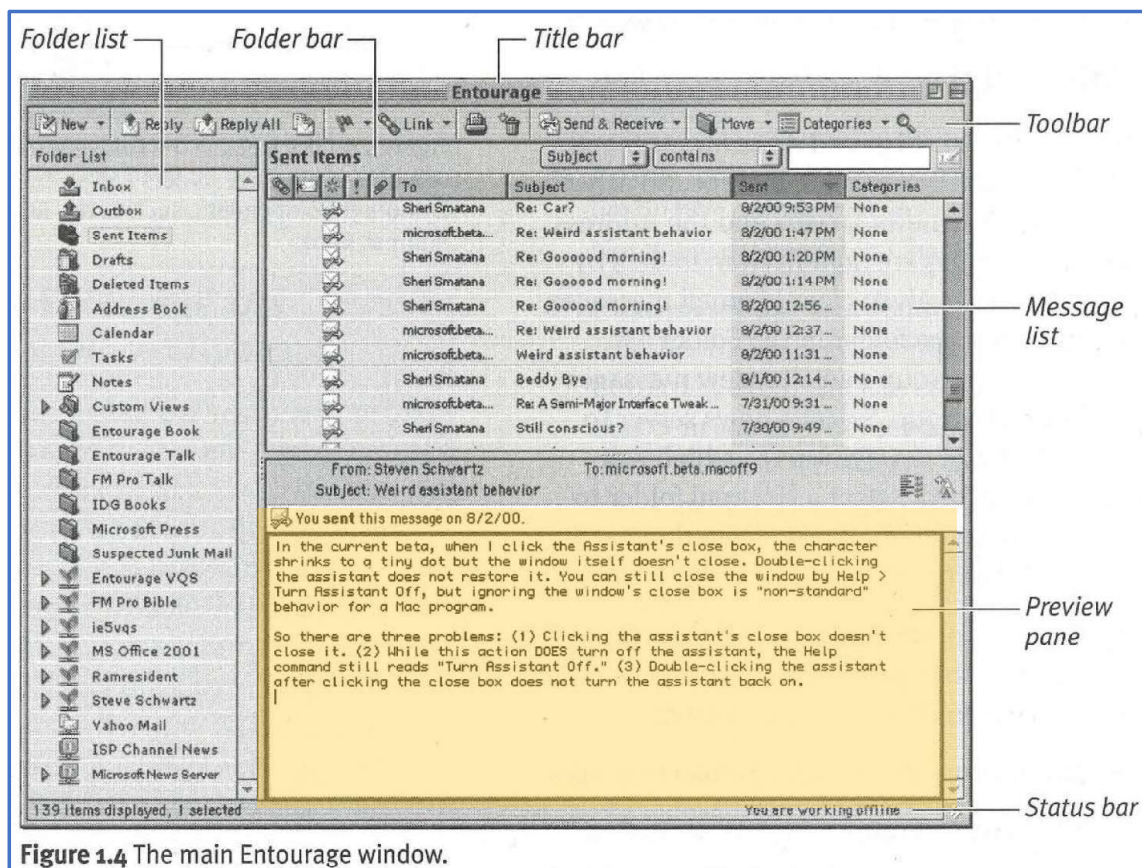


Figure 1.4 The main Entourage window.

EX1004, 3 (cited by '490 Petition (Paper 1), 80-81; '490 EX1003, ¶¶188-190; '035 Petition (Paper 1), 72-73; '035 EX1003, ¶¶150-151; '139 Petition (Paper 1), 75-76; '139 EX1003, ¶¶169-170).

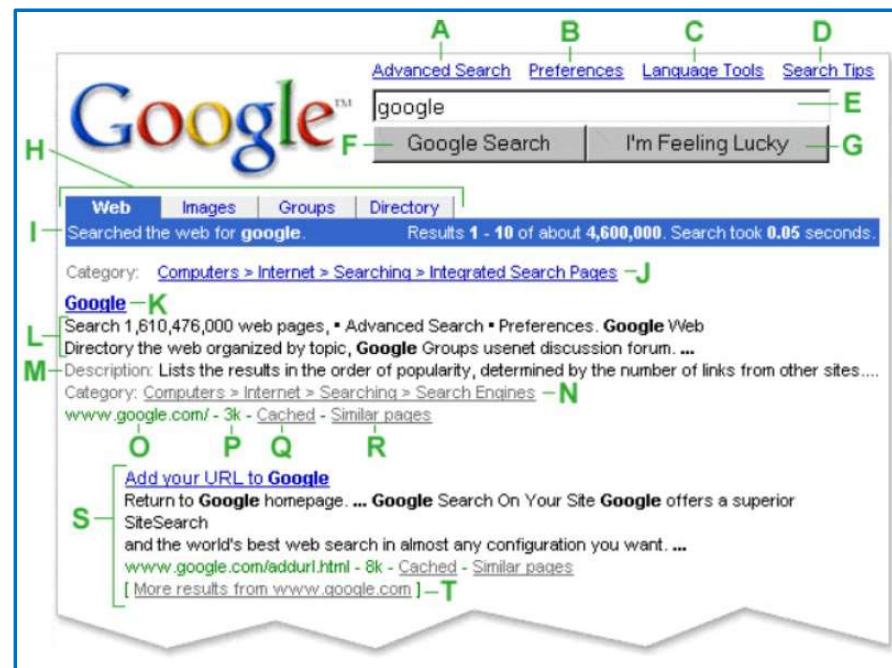
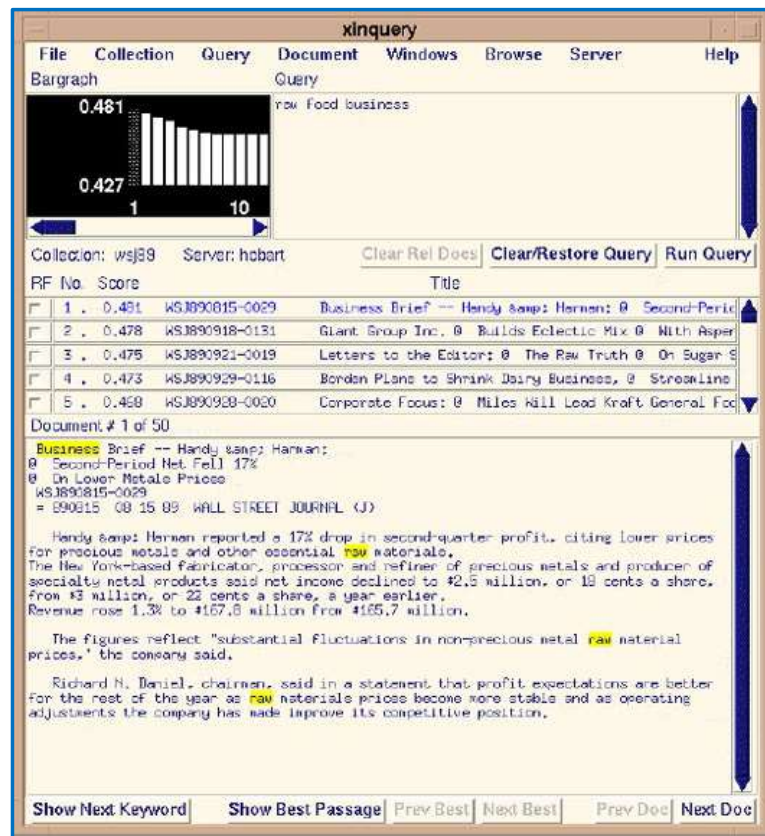
DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Obvious To Highlight Search Strings In Same Window As Search Field

18:19-20:5; EX1101, 162-163, 358. Further, even if the claims did recite highlighting search strings in the same window as the search results, this too would have been obvious at least because it was known to do so, was one of only a limited number of known options (either the highlighting is shown in a separate window or the same window as the search results), and was nothing more than a simple design choice. EX1003, ¶¶48-51, 193.

'490 EX1100, ¶66 (cited by '490 Reply (Paper 28), 24);
'035 EX1100, ¶63 (cited by '035 Reply (Paper 27), 24);
'139 EX1100, ¶63 (cited by '139 Reply (Paper 27), 24).

Known To Highlight Search Strings In Same Window As Search Field



EX1038, 56 (annotations in original) (cited by '490 EX1003, ¶¶49, 193; '035 EX1003, ¶¶46, 97, 157; '139 EX1003, ¶46); *see also* '490 Petition (Paper 1), 8, 84 (citing '490 EX1003, ¶¶49, 193); '035 Petition (Paper 1), 11, 36, 77 (citing '035 EX1003, ¶¶46, 97, 157); '139 Petition (Paper 1), 9 (citing '139 EX1003, ¶46).

EX1044, 5 (highlighting in original) (cited by '490 EX1003, ¶¶50, 193; '035 EX1003, ¶¶47, 97, 148; '139 EX1003, ¶¶47, 168) *see also* '490 Petition (Paper 1), 8, 84 (citing '490 EX1003, ¶¶50, 193); '035 Petition (Paper 1), 11, 36, 71 (citing '035 EX1003, ¶¶47, 97, 148); '139 Petition (Paper 1), 9, 75 (citing '139 EX1003, ¶¶47, 168).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Appendix

Ground 1: Lotus Notes Discloses A Single Search
Field

Dr. Terveen Explained Why A Single Search Field Would Have Been Obvious

EX1003, ¶¶40-41; EX1059, 161-164; EX1061, 140, 142; EX2020, ¶9. As taught by the Lotus Notes references, the single search field is preferred when searching for just a few search strings, while multiple search fields are preferred when searching

for many strings. EX1061, 140 (explaining that the single-field Search Bar tool allows “you to find all the occurrences of a word (or a couple of words) very, very quickly”), 142 (explaining that the multi-field Search Builder is a better “option when you want to look for more than just a couple of words or phrases”); *see also* EX1003, ¶¶40-44. Thus, when performing a search on only two search strings, as recited in the claims, a POSITA would have been motivated to use a single search field, as in the Search Bar and Query Builder.

EX1100, ¶56 (cited by '490 Reply (Paper 28), 16-17; '035 Reply (Paper 27), 17-18; '139 Reply (Paper 27), 17-18).

Lotus Notes Explains The Benefit Of A Single Search Field

The Search icon allows a *full-text search*, which is a way for you to find all the occurrences of a word (or a couple of words) very, very quickly. You can search a huge database in no time at all. Most searches don't take more than two or three *seconds*. And when you ask Notes to search for a word, Notes finds *all* the documents for you, no matter where (or how many times) the documents contain the word. Notes calls this full-text search not because it finds full text (no such thing exists), but because full-text search is more powerful — it enables you to find words near each other or in the same sentence or paragraph, and it allows you to locate variations of the search text.

EX1061, 140 (cited by '490 Reply (Paper 28), 16-17; '035 Reply (Paper 27), 17-18; '139 Reply (Paper 27), 17-18).

The Board Rejected PO's Arguments As To The Search Builder

Resp. 16. The issue, however, is not whether “over the years Lotus Notes moved away from a unified search field[,]” as Patent Owner suggests. *Id.* at 15. The issue is what “inferences and creative steps” a person of ordinary skill in the art would have employed, based on Wilcox and Londergan, when designing a new method and system for search indexing. *KSR*, 550 U.S. at 418. Petitioner’s argument is that one of ordinary skill in the art would

’490 Institution Decision (Paper 14), 21;
’035 Institution Decision (Paper 14), 20;
’139 Institution Decision (Paper 14), 20.

Appendix

MTC – Ground 1: The Benefits Of Incremental
Searches Were Known

MTC - Ground 1: The Benefits Of Incremental Searches Were Known

True

The immediate visual feedback provided to the user can increase the speed with which the user achieves satisfactory search results. Because query results are displayed as the query is being input, the user need not wait until the full query is entered before viewing the results. This allows the user to modify a partially-entered query in response to unsatisfactory or unexpected results. Immediate display of segments of interest in response to modification of the interest level threshold increases the speed with which the user can identify segments of interest.

EX1009, 2:39-48 (cited by EX1003, ¶¶33, 38); *see also* '490 Petition (Paper 1), 6-7 (citing EX1003, ¶¶33, 38); '035 Petition (Paper 1), 10 (citing EX1003, ¶¶33, 38); '139 Petition (Paper 1), 8 (citing EX1003, ¶¶33, 38).

MTC - Ground 1: The Benefits Of Incremental Searches Were Known

Schneiderman (EX1044)

2. **Action:** Searches may be started explicitly or implicitly. The typical usage process in current systems is to have users click on a Search button to initiate the search and then wait for the results. But a very appealing alternative is that of "dynamic queries": there is no Search button but the result set is continuously displayed and updated as phases of the search are changed. See, for example, the commercial system [Folio Viewer](#), or research prototypes like Ahlberg and Shneiderman (1994) or Shneiderman (1994). This approach requires adequate screen space and high bandwidth, plus, for a large database, very rapid processing: whether it is feasible depends very much on the situation. However, when it is practical, the advantages are great: users can broaden, narrow, or refocus their search several times in as many seconds. Designers may also allow users to choose between approaches.

EX1044, 3 (cited by EX1003, ¶¶33, 38) *see also* '490 Petition (Paper 1), 6-7 (citing EX1003, ¶¶33, 38); '035 Petition (Paper 1), 10 (citing EX1003, ¶¶33, 38); '139 Petition (Paper 1), 8 (citing EX1003, ¶¶33, 38).

MTC - Ground 1: The Benefits Of Incremental Searches Were Known

Schneiderman (EX1052)

phase immediately produces a new set of search results. *Dynamic queries*—in which users adjust query widgets to produce continuous updates—have proved to be effective and satisfying. They require adequate screen space and rapid processing, but their advantages are great.

EX1052, 518 (cited by EX1003, ¶¶33, 38) *see also* '490 Petition (Paper 1), 6-7 (citing EX1003, ¶¶33, 38); '035 Petition (Paper 1), 10 (citing EX1003, ¶¶33, 38); '139 Petition (Paper 1), 8 (citing EX1003, ¶¶33, 38).

Appendix

REOS: Email Databases Were Not Too Large

Typical Email Database Size < 10 MB: EX1095, EX1096

Yahoo has been offering 4 megabytes of free e-mail storage, although some people with accounts opened several years ago have 6 megabytes of free storage. MICROSOFT EXHIBIT 1095

EX1095, 1 (cited by '490 Reply (Paper 28), 7; ' 035 Reply (Paper 27), 7; '139 Reply (Paper 27), 7; EX1100, ¶13).

Sign up for Yahoo Mail (you'll need to give your name, birth date, and gender), select a username and a password, and Yahoo hands over a mailbox that holds 4MB of messages and attachments. This figure is down from 6MB just several months ago, alas, but it's still more than Hotmail's measly 2MB. Once you've

EX1096, 1 (cited by '490 Reply (Paper 28), 7; ' 035 Reply (Paper 27), 7; '139 Reply (Paper 27), 7; EX1100, ¶13).

Appendix

REOS: Incremental Search User Interface (UI)

PO's Expert: Developer Could Implement Dynamic Text Box

17 Q. Okay. But again, I'm not asking about Visual
18 Basic.
19 I'm saying that, if a developer in 2002 was
20 familiar with C+, would they have been able to implement
21 a dynamic text box?
22 A. If you --
23 MS. DAUGHTREY: Object to form.
24 THE WITNESS: You phrased the question in the
25 context of a -- of C+ -- I assume you mean a C++, or

1 C-native implementation.
2 BY MR. THACKER:
3 Q. Correct.
4 A. And I would say it -- manifestly, yes,
5 because, among other things, it was -- was implemented
6 in Emacs by then.
7 Q. And do you agree that a POSITA would have been
8 familiar with C+ in 2002?
9 A. A POSITA would have been familiar with C++ and
10 C in 2002.

EX1098, 24:17-25:10 (cited by '490
Reply (Paper 28), 15; '035 Reply (Paper
27), 16; '139 Reply (Paper 27), 16).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Appendix

'139/'490 Patents: Partial Search Strings

PO's Expert: Claims Do Not Require Indexing Prefixes

25	So again, I'm going to ask: Does the index	
1	data structure of Claim 1 of the '490 patent require	
2	indexing every two-character string?	
3	MS. DAUGHTREY: Object to form.	
4	THE WITNESS: Based on my understanding of	
5	this patent's disclosure and the literal claim language,	
6	I would say there's not a specific architectural	
7	requirement about how or what exactly is indexed except	
8	that there be associations between pluralities of	
9	character strings to associated files.	

EX1098, 42:25-43:9 (cited by '490 Reply (Paper 28), 21).

Prefix Searches Known: Baeza-Yates (EX1023)

- **Prefixes** A string which must form the beginning of a text word. For instance, given the prefix 'comput' all the documents containing words such as 'computer,' 'computation,' 'computing,' etc. are retrieved.

EX1023, 105 (cited by EX1100, ¶16); *see also* '490 Reply (Paper 28), 12-13 (citing EX1100, ¶16); '035 Reply (Paper 27), 13-14 (citing EX1100, ¶16); '139 Reply (Paper 27), 13-14 (citing EX1100, ¶16).

- (4) Selection of index terms to determine which words/stems (or groups of words) will be used as an indexing elements. Usually, the decision on whether a particular word will be used as an index term is related to the syntactic nature of the word. In fact, noun words frequently carry more semantics than adjectives, adverbs, and verbs.

EX1023, 165 (cited by EX1100, ¶16); *see also* '490 Reply (Paper 28), 12-13 (citing EX1100, ¶16); '035 Reply (Paper 27), 13-14 (citing EX1100, ¶16); '139 Reply (Paper 27), 13-14 (citing EX1100, ¶16).

Prefix Searches Known: Baeza-Yates (EX1023)

Single-word queries can be searched using any suitable data structure to speed up the search, such as hashing, tries, or B-trees. The first two give $O(m)$ search cost (independent of the text size). However, simply storing the words in lexicographical order is cheaper in space and very competitive in performance, since the word can be binary searched at $O(\log n)$ cost. Prefix and range queries can also be solved with binary search, tries, or B-trees, but not with hashing. If the query is formed by single words, then the process ends by delivering the list of occurrences (we may need to make a union of many lists if the pattern matches many words).

EX1023, 195 (cited by EX1100, ¶16); *see also* '490 Reply (Paper 28), 12-13 (citing EX1100, ¶16); '035 Reply (Paper 27), 13-14 (citing EX1100, ¶16); '139 Reply (Paper 27), 13-14 (citing EX1100, ¶16).

If a suffix tree on the text can be afforded, many basic patterns such as words, prefixes, and phrases can be searched in $O(m)$ time by a simple trie search. However, suffix trees are not practical for large texts, as explained. Suffix arrays, on the other hand, can perform the same search operations in $O(\log n)$ time by doing a binary search instead of a trie search.

EX1023, 202 (cited by EX1100, ¶16); *see also* '490 Reply (Paper 28), 12-13 (citing EX1100, ¶16); '035 Reply (Paper 27), 13-14 (citing EX1100, ¶16); '139 Reply (Paper 27), 13-14 (citing EX1100, ¶16).

Prefix Searches Known: Lotus Notes (EX1061)

When you have a full-text-indexed database, the following options are available to you:

- ◆ **Use word variants:** This option tells Notes that you want to look for not only the base word you typed, but also the base word with any possible prefixes or suffixes. For example, searching for *sneeze* also finds *sneezed* and *sneezing*. Bless you.

EX1061, 144 (cited by EX1100, ¶16); *see also* '490 Reply (Paper 28), 12-13 (citing EX1100, ¶16); '035 Reply (Paper 27), 13-14 (citing EX1100, ¶16); '139 Reply (Paper 27), 13-14 (citing EX1100, ¶16).

Prefix Searches Known: EX1018 (Green)

is very time consuming. A complete indexing of characters in each word permits very fast exact match and prefix searches but places considerable demand on disk space. Numerous techniques are known for partial indexing of word lists on prefix values (starting characters) or word length with a view to reducing the number of words that must actually be compared with a search string. A very well

EX1018, 1:38-43 (cited by EX1100, ¶16); *see also* '490 Reply (Paper 28), 12-13 (citing EX1100, ¶16); '035 Reply (Paper 27), 13-14 (citing EX1100, ¶16); '139 Reply (Paper 27), 13-14 (citing EX1100, ¶16).

Prefix Searches Known: EX1018 (Green)

The word look-up table **24** is sorted on the character-length values of column one, primarily on starting character and secondarily on length. The sorting expedites location of 55
any specified character-length value in a conventional manner. More significantly, however, such sorting allows quick identification of a set of word numbers, specifically a set of consecutive word numbers, that represent words with the same starting character but different lengths, as might be 60
required, for example, in a prefix search. A prefix search is of course matched by any word that begins with a specified search string. The minimum length of any matching word is consequently the string length, and the maximum length is bounded by the longest word in the lexicon **22** (whose length 65
can be identified and stored during parsing **40**). To select a starting word number, column one of the table is scanned

EX1018, 5:53-5:67 (cited by EX1100, ¶16); *see also* '490 Reply (Paper 28), 12-13 (citing EX1100, ¶16); '035 Reply (Paper 27), 13-14 (citing EX1100, ¶16); '139 Reply (Paper 27), 13-14 (citing EX1100, ¶16).

Appendix

Siitonen (Background Reference) Discloses
Incrementally Searching Across Files Because Its
Contact Records Are Files

PO's Expert Admitted It Was Known To Store Contact Records As Files

16 Q. So is it still your opinion that contact
17 records are not files?
18 MS. DAUGHTREY: Object to form.
19 THE WITNESS: I think that's a
20 mischaracterization of my answer, by the way.
21 That -- if you could have your previous
22 question reread, what I was trying to allow for is that
23 there may be many different -- if you'd like to have
24 that reread -- there might be many different ways of
25 structuring contact information.

EX1098, 12:16-25 (cited by EX1100, ¶30); *see also* '490 Reply (Paper 28), 7-8 (citing EX1100, ¶30); '035 Reply (Paper 27), 8 (citing EX1100, ¶30); '139 Reply (Paper 27), 8 (citing EX1100, ¶30).

EX1098, 13:1-21 (cited by EX1100, ¶30); *see also* '490 Reply (Paper 28), 7-8 (citing EX1100, ¶30); '035 Reply (Paper 27), 8 (citing EX1100, ¶30); '139 Reply (Paper 27), 8 (citing EX1100, ¶30).

1 For example, one way might be to have one file
2 per contact. Another might be to have a set of database
3 records, which themselves might be represented in a
4 single file or a number of different files. Each record
5 might then correspond to a contact -- or might add --
6 another scenario in which multiple contacts -- one or
7 more contacts are represented in a file and a contact
8 collection or -- or aggregate for a user might then be
9 represented in multiple different files. So those are
10 all potential examples of -- there might be many others.
11 BY MR. THACKER:
12 Q. So your -- your testimony, then, is that
13 contact records -- a contact record could be implemented
14 in its -- as its own file or is in -- in a file with
15 other contact records?
16 A. Or there might be other ways of representing
17 them. For example, I mentioned a database.
18 Q. Would a POSITA in 2002 have been aware of
19 these various ways of storing or representing a contact
20 record?
21 A. Yes.

Patents' Claims: Files/Documents Comprise Contact Information

13. The computing system of claim 1, wherein the files comprise emails, text documents, contact information, appointment information, task information, music files, system files, photographs, videos, and/or images.

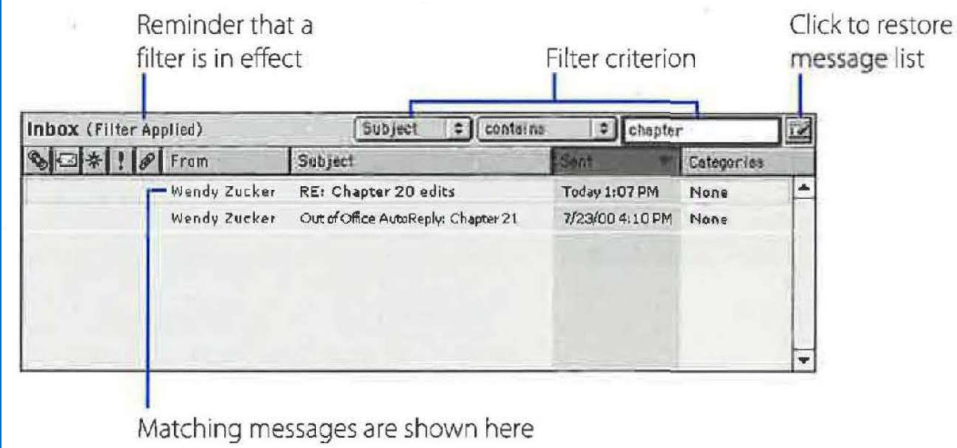
'490 EX1001, 31:18-21 (cited by '490 EX1100, ¶30); '035 EX1075, 30:23-26 (cited by '035 EX1100, ¶30); '139 EX1001, 30:23-26 (cited by '139 EX1100, ¶30); *see also* '490 Reply (Paper 28), 7-8 (citing EX1100, ¶30); '035 Reply (Paper 27), 8 (citing EX1100, ¶30); '139 Reply (Paper 27), 8 (citing EX1100, ¶30).

Appendix

Ground 2: Entourage's Incremental Search Is A
Search

Entourage Refers To Its Incremental Search As A Search

- 3** When searching within the Subject, From, or To fields, enter a search string in the filter criteria text box. The search is conducted as you type, and the matching headers are displayed in the message list.



EX1043, 385 (cited by '490 Petition (Paper 1), 64; '490 EX1003, ¶160; '035 Petition (Paper 1), 57-58; '035 EX1003, ¶133; '139 Petition (Paper 1), 63-64; '139 EX1003, ¶148; EX1100, ¶31); *see also* EX1004, 8 ("filtering is the fastest way to perform a simple search") (cited by '490 Petition (Paper 1), 64-66; '490 EX1003, ¶160; '035 Petition (Paper 1), 57-59; '035 EX1003, ¶133; '139 Petition (Paper 1), 63-65; '139 EX1003, ¶148; EX1100, ¶31).

Challenged Patents Refer To Searching And Filtering Synonymously

(57)

ABSTRACT

The present invention provides for quick and efficient searching. One embodiment includes a first instruction configured to read at least a first search string and a second search string entered into a same first search field, and a second instruction configured to incrementally filter search results to locate at least a first document that has a least a first word that begins with the first string and a second word that begins with the second string.

EX1001, Abstract (cited by EX1100, ¶27); *see also* '490 Reply (Paper 28), 7-8 (citing EX1100, ¶27); '035 Reply (Paper 27), 8 (citing EX1100, ¶27); '139 Reply (Paper 27), 8 (citing EX1100, ¶27).

enters search characters or terms. To further refine or filter the search results, the user can enter other search terms into other fields. In one embodiment, the filtering process is performed substantially immediately upon the user entering addition search characters or terms.

'490 EX1001, 3:54-58 (cited by EX1100, ¶27); '035 EX1001, 3:42-46 (cited by EX1100, ¶27); '139 EX1001, 3:50-54 (cited by EX1100, ¶27); *see also* '490 Reply (Paper 28), 7-8 (citing EX1100, ¶27); '035 Reply (Paper 27), 8 (citing EX1100, ¶27); '139 Reply (Paper 27), 8 (citing EX1100, ¶27).

engine 510 will further refine the search. Similarly, with each deletion of a search string character, the search engine 510 will reduce the amount of search filtering to broaden the search. For example, when the user types "fast", four separate searches are optionally sent to the server, one for "f", one for "fa", one for "fas", and one for "fast". Each time, the server finds all documents containing words starting with that sequence of letters, and returns a list of the top matches. At state 614 the server system 508 transmits a list of search results to the browser 504.

'490 EX1001, 28:36-45 (cited by EX1100, ¶27); '035 EX1001, 28:20-29 (cited by EX1100, ¶27); '139 EX1001, 28:24-33 (cited by EX1100, ¶27); *see also* '490 Reply (Paper 28), 7-8 (citing EX1100, ¶27); '035 Reply (Paper 27), 8 (citing EX1100, ¶27); '139 Reply (Paper 27), 8 (citing EX1100, ¶27).

Challenged Patents' Search Filters A List

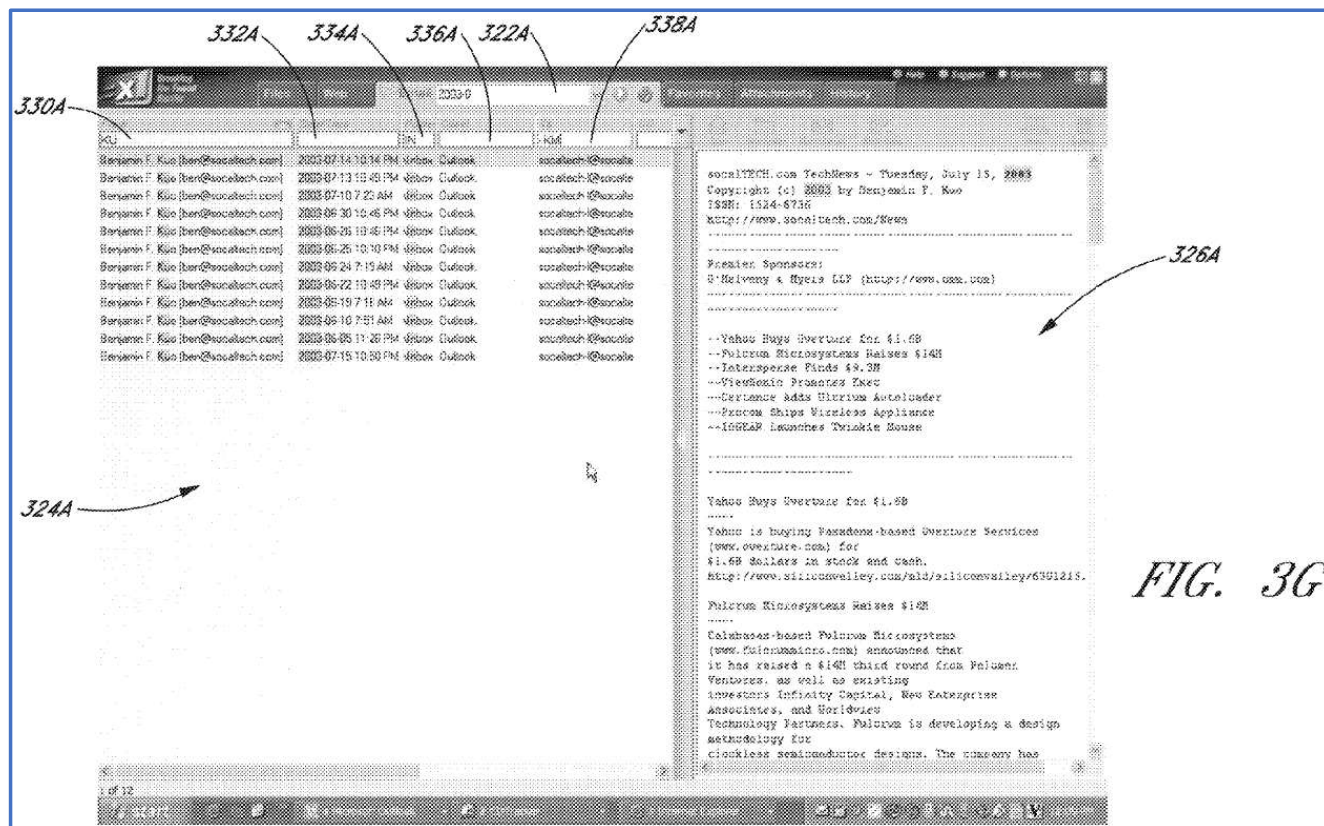


FIG. 3G

EX1001, FIG. 3G (cited by EX1100, ¶27); see also '490 Reply (Paper 28), 7-8 (citing EX1100, ¶27); '035 Reply (Paper 27), 8 (citing EX1100, ¶27); '139 Reply (Paper 27), 8 (citing EX1100, ¶27).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Challenged Patents' Search Filters A List

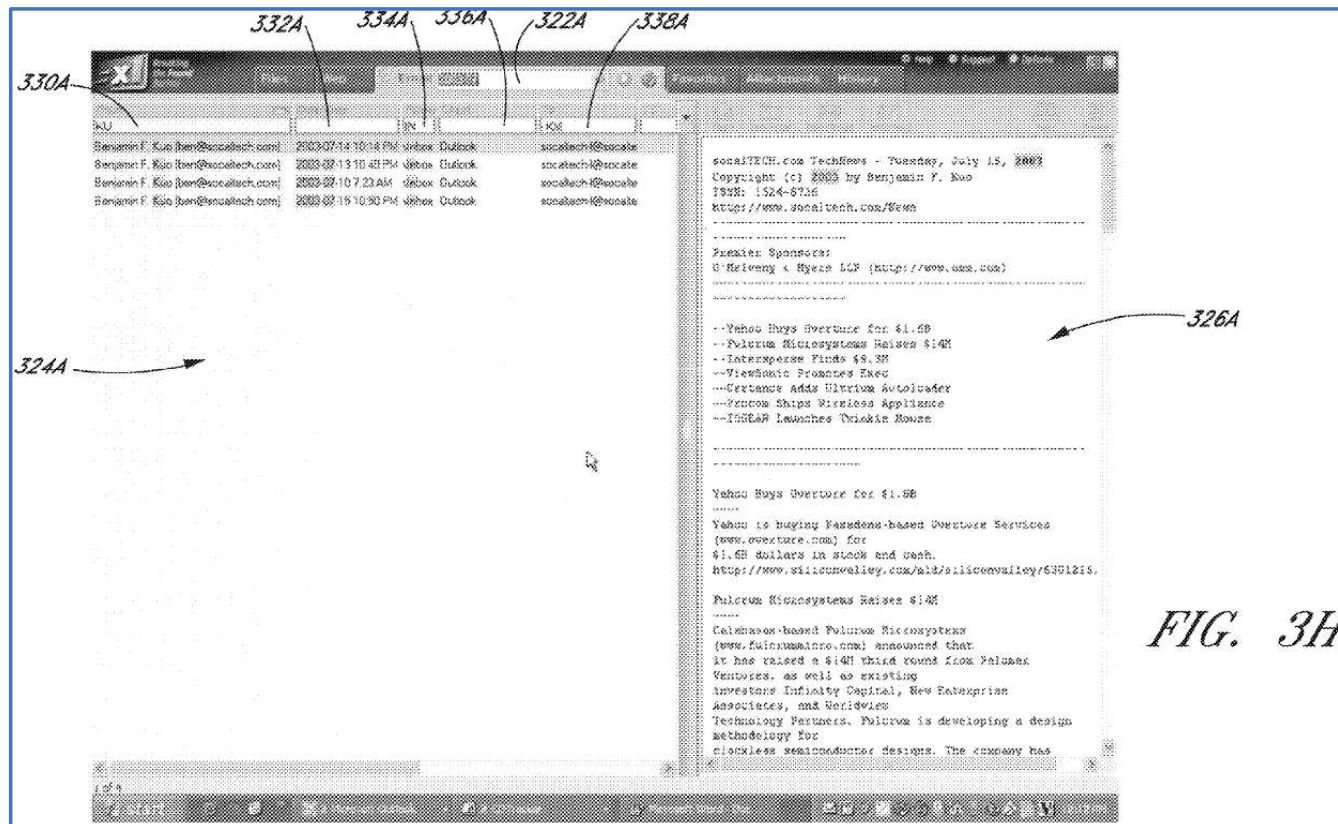


FIG. 3H

EX1001, FIG. 3H (cited by EX1100, ¶27); see also '490 Reply (Paper 28), 7-8 (citing EX1100, ¶27); '035 Reply (Paper 27), 8 (citing EX1100, ¶27); '139 Reply (Paper 27), 8 (citing EX1100, ¶27).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Appendix

Ground 2: True Is Analogous Art

True Describes Its Field Broadly

[57]

ABSTRACT

A user searches searchable information representing underlying information using an interactive user interface, which indicates to the user which parts of the searchable information represent information in the underlying information that is of interest to the user.

EX1009, Abstract (cited by '490 Reply (Paper 28), 10; '035 Reply (Paper 27), 11; '139 Reply (Paper 28), 11).

Referring to FIG. 1a, when a computer user needs to search a file of searchable information 1, such as a file containing the text of issued patents, he typically enters a query 2 containing keywords interconnected by boolean operators. A search engine 3 then searches the file of searchable information 1 for records (e.g., patents) which satisfy the query 2. Each record either satisfies or does not satisfy the query 2. The search engine 3 produces a set of hits 4 representing the records that satisfy the query. The hits 4 may be presented to the user in various ways. One common approach is to present the user with a list of the hits 4 and to allow the user to examine selected hits.

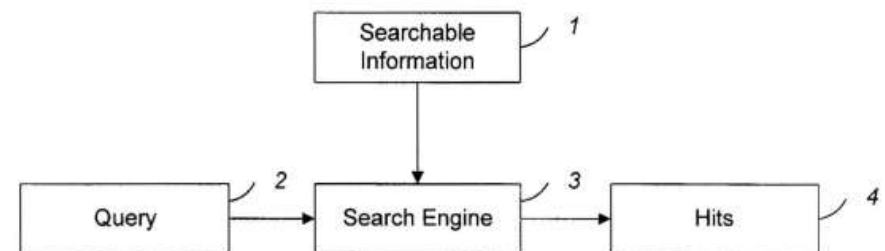
EX1009, 1:6-17 (cited by '490 Reply (Paper 28), 10; '035 Reply (Paper 27), 11; '139 Reply (Paper 28), 11).

True Solves The Same Problem As The Challenged Patents

40 The immediate visual feedback provided to the user can increase the speed with which the user achieves satisfactory search results. Because query results are displayed as the query is being input, the user need not wait until the full query is entered before viewing the results. This allows the user to modify a partially-entered query in response to unsatisfactory or unexpected results. Immediate display of segments of interest in response to modification of the interest level threshold increases the speed with which the user can identify segments of interest.

EX1009, 2:26-48 (cited by '490 Reply (Paper 28), 10; '035 Reply (Paper 27), 11; '139 Reply (Paper 28), 11).

FIG. 1a



EX1009, FIG. 1A (cited by '490 Reply (Paper 28), 10; '035 Reply (Paper 27), 11; '139 Reply (Paper 28), 11).

PO's Expert Admitted True Discloses Searching Text That Is 100% Accurate

1 BY MR. THACKER:

2 Q. Right. I know what True says. What I'm
3 asking is that: Does that disclosure to you mean that
4 the confidence levels could all be one?

5 A. Yes.

6 Q. Does a confidence level of one mean
7 100 percent accuracy?

8 A. In the context of True, that's -- would appear
9 to be what that means, although further discussion in
10 Column 8 says it could mean something else, see line 10
11 of Column 8.

12 Q. Right. But -- of course, it -- it could be
13 something else, but True does disclose that the -- the
14 con- -- in one embodiment, that the confidence levels
15 correspond to accuracy, correct?

16 A. Yes.

17 Q. Okay. So in that embodiment, a confidence
18 level of one would mean 100 percent accuracy.

19 A. Yes.

EX1098, 49:1-49:19 (cited by '490 Reply (Paper 28), 10; '035 Reply (Paper 27), 10; '139 Reply (Paper 28), 10).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Appendix

Ground 2: True Discloses Incrementally Searching
For Multiple Search Strings

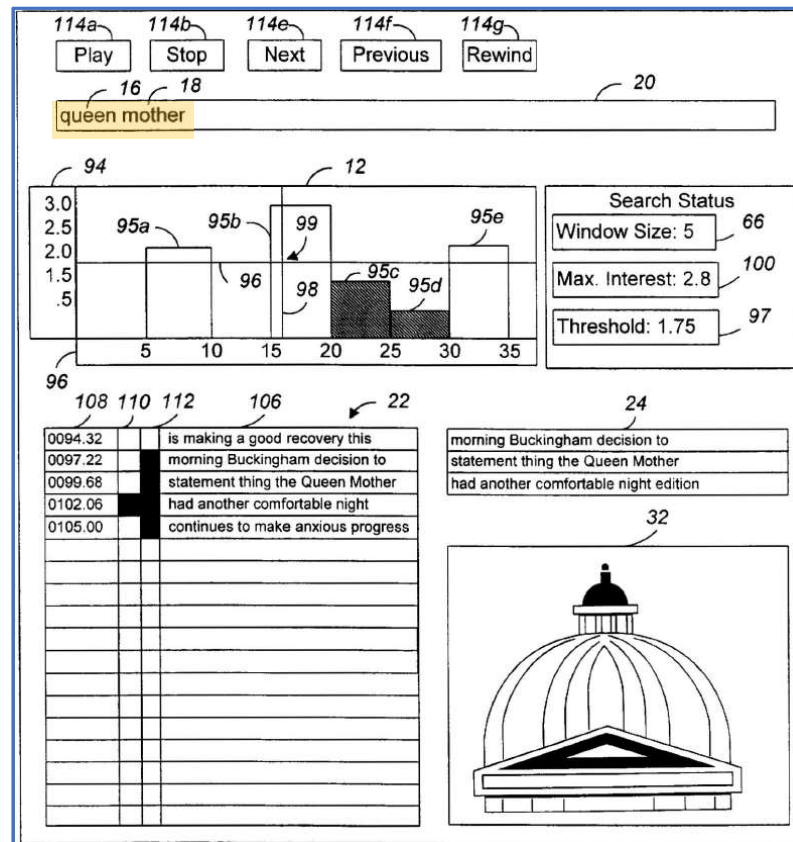
Ground 2: True Discloses Incrementally Searching For Multiple Search Strings

12. Modifying an existing query can include adding a single character to or deleting a single character from the query. As a result, the user can view the results of a query as the query is being modified. To make efficient use of system resources,

EX1009, 5:3-6 (cited by '490 Petition (Paper 1), 67, 69; '490 EX1003, ¶¶162, 165; '035 Petition (Paper 1), 60, 63; '035 EX1003, ¶¶135, 138; '139 Petition (Paper 1), 66, 69; '139 EX1003, ¶¶150, 153).

165. The search tool separately searches each search string (e.g., “queen” and “mother”) and displays “query results ... as the query is being input” (EX1009, 2:41-42) according to the number of occurrences of the partial search strings in each file (transcript window) and the system’s confidence that the strings were correctly transcribed by the speech recognizer. EX1009, 4:21-5:15. 5:41-50

'490 EX1003, ¶165 (cited by '490 Petition (Paper 1), 69);
'035 EX1003, ¶138 (cited by '035 Petition (Paper 1), 63);
'139 EX1003, ¶153 (cited by '139 Petition (Paper 1), 69).



EX1009, FIG. 2 (cited by '490 Petition (Paper 1), 67-69; '490 EX1003, ¶¶162, 164, 165; '035 Petition (Paper 1), 60-63; '035 EX1003, ¶¶135, 137, 138; '139 Petition (Paper 1), 66-69; '139 EX1003, ¶¶150, 152, 153).

DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

CERTIFICATE OF SERVICE
IN COMPLIANCE WITH 37 C.F.R. § 42.6(e)(4)

The undersigned certifies that on April 22, 2026, a complete copy of
PETITIONER’S DEMONSTRATIVES were served on counsel for the Patent
Owner via electronic mail as follows:

Lori Gordon GOODWIN PROCTER LLP 1900 N St. N.W. Washington, D.C. 20036 Gordon-ptab@goodwinlaw.com DG-X1-IPR@goodwinlaw.com	Christie L. Larochelle GOODWIN PROCTER LLP 100 Northern Avenue Boston, MA 02210 clarochelle@goodwinlaw.com
	Natasha E. Daughtrey (pro hac vice) GOODWIN PROCTER LLP 601 S. Figueroa Street, Ste. 4100 Los Angeles, CA 90017 ndaughtrey@goodwinlaw.com
	Emma L. Murray GOODWIN PROCTER LLP 100 Northern Avenue Boston, MA 02210 emurray@goodwinlaw.com

By: /Todd M. Siegel/
Todd M. Siegel, Reg. No. 73,232
todd.siegel@klarquist.com
KLARQUIST SPARKMAN, LLP
One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Tel: 503-595-5300
Fax: 503-595-5301

Counsel for Petitioner