

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

ORACLE CORPORATION,
Petitioner

v.

JOSEPH L. DEBELLIS,
Patent Owner

Case No.: IPR2024-00221
U.S. Patent No. 6,760,720
Original Issue Date: July 6, 2004

Title: SEARCH-ON-THE-FLY/SORT-ON-THE-FLY SEARCH ENGINE FOR
SEARCHING DATABASES

**DECLARATION OF EREZ ZADOK, PH.D., IN SUPPORT OF PETITION
FOR *INTER PARTES* REVIEW OF
U.S. PATENT NO. 6,760,720**

	<u>TABLE OF CONTENTS</u>	<u>Page</u>
I. INTRODUCTION		12
II. QUALIFICATIONS		14
III. ASSIGNMENT AND MATERIALS CONSIDERED IN FORMING MY OPINIONS		25
IV. UNDERSTANDING OF GOVERNING LAW		29
A. Claim Construction		29
B. Obviousness.....		31
V. LEVEL OF SKILL IN THE ART		36
VI. THE 720 PATENT’S EFFECTIVE FILING DATE		38
VII. TECHNOLOGY OVERVIEW.....		39
A. General Computer Operations.....		39
B. Determining Display and Terminal Capabilities.....		41
C. Databases.....		43
D. Summarizing and Formatting Information in Limited Spaces		49
E. Web Technologies and Graphical User Interfaces.....		51
F. Accessing and Searching Databases Via GUIs and the Web...		58
VIII. THE 720 PATENT		63
A. Overview		63
1. Background		63
2. The Alleged Invention		64
B. Prosecution History of the 720 Patent.....		67
C. The 720 Patent’s Claims		68
IX. CLAIM CONSTRUCTION		74
X. UNPATENTABILITY ANALYSIS		75
XI. THE PRIOR ART		76
A. Overview of the Prior Art.....		76

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

1.	United States Patent No. 5,995,979 to Cochran, Filed February 10, 1998 (“Cochran”)	76
2.	Mark Newsome et al., <i>HyperSQL: Web-based Query Interfaces for Biological Databases</i> , IEEE Computer Society, Proceedings of The Thirtieth Annual Hawaii International Conference on System Sciences (1997) (“HyperSQL”).....	78
3.	United States Patent No. 6,433,797 to Zellweger, Filed July 15, 1999, Claiming Priority to Provisional Application No. 60/095,811, Filed August 4, 1998 (“Zellweger”)	81
4.	Using Microsoft Access 97, Special Edition, by Roger Jennings (1997) (“Microsoft Access”)	85
5.	United States Patent No. 6,642,946 to Janes, Filed Aug. 12, 1999 (“Janes”).....	87
6.	United States Patent No. 6,629,079 to Spiegel, Filed June 25, 1998 (“Spiegel”)	89
XII.	GROUND 1: COCHRAN IN VIEW OF ZELLWEGER, AND THE KNOWLEDGE OF A POSITA RENDERS OBVIOUS CLAIMS 1-7, 15-20, AND 27-30	90
A.	Claim 1	90
1.	[1.PRE] “A method implemented on a computer for searching databases, comprising:”	90
2.	[1.A] “determining a database schema for a database, wherein the database includes database fields;”	91
3.	[1.B] “providing a list of the database fields, wherein the list includes a descriptor indicating a data category;” ...	92
4.	[1.C] “receiving a search selection for a database field on the provided list of the database fields;”	94
5.	[1.D] “determining a quantity of entries in the selected database field;”	95

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

6.	[1.E] “if the quantity exceed a specified amount, truncating data, and displaying the truncated data wherein the truncating reduces characters in one or more entries in the selected database field and the truncated data represents each of the entries in the selected database field; and”	102
7.	[1.F] “if the quantity does not exceed the specified amount, displaying contents of the database field.”	107
B.	Claim 2	110
1.	“The method of claim 1, further comprising providing a key word search.”	110
C.	Claim 3	110
1.	[3.PRE] “A method implemented on a computer for searching a database, comprising:”	110
2.	[3.A] “generating a list of data fields;”	110
3.	[3.B] “receiving a first data field selection from the list of data fields;”	111
4.	[3.C] “determining a first quantity indicative of a number of entries of the selected data field;”	111
5.	[3.D] “if the first quantity exceeds a specified limit, reducing a size of data to be displayed from the selected data field wherein the reducing reduces characters in one or more entries in the selected data field and the size-reduced data represents each of the entries in the selected field; and”	112
6.	[3.E] “displaying data from the selected data field.” ...	113
D.	Claim 4	116
1.	“The method of claim 3, wherein the specified limit is fixed.”	116
E.	Claim 5	118
1.	“The method of claim 3, wherein the specified limit is variable.”	118
F.	Claim 6	118

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

1.	“The method of claim 3, wherein the data are displayed on a terminal, and wherein the specified limit is determined dynamically, based on a characteristic of the terminal.”	118
G.	Claim 7	122
1.	“The method of claim 3, wherein the specified limit is a user-determined limit.”	122
H.	Claim 15	122
1.	[15.A] “The method of claim 3, further comprising: receiving a first constraint, wherein the first constrain[t] is related to a data element in a data field; and”	122
2.	[15.B] “receiving one or more subsequent constraints, wherein search results are generated based on a combination of the first and the one or more subsequent constraints.”	124
I.	Claim 16	126
1.	“The method [of] claim 3, wherein each display of data comprises a search cycle, and further comprising receiving a refresh command, the refresh command resetting all search cycles.”	126
J.	Claim 17	135
1.	“The method [of] claim 3, further comprising displaying the list of data fields and a representation of the selected data field, wherein the representation of the selected data field comprises one or more elements from the selected data field.”	135
K.	Claim 18	138
1.	“The method of claim 3, wherein the method comprises executing a plurality of search cycles, each search cycle of the plurality of search cycles providing a search result list comprising selected data elements from the data base.”	138
L.	Claim 19	146

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

1.	“The method of claim 18, further comprising resetting one or more search cycles of the plurality of search cycles.”	146
M.	Claim 20	147
1.	[20.A] “The method of claim 3 further comprising: determining a schema of the database;”	147
2.	[20.B] “comparing the determined schema to a capability of a terminal for displaying the data; and”	147
3.	[20.C] “formatting a search request from the terminal to correspond to the determined schema.”	152
N.	Claim 27	153
1.	“The method of claim 3, further comprising displaying the data on a terminal, the terminal including one of a handheld device, a cellular phone, a geosynchronous positioning satellite (GPS) device, a wrist-worn device, an interactive phone device, a household appliance, a television, a television set top box, a handheld computer, a main frame computer and a personal computer.”	153
O.	Claim 28	155
1.	“The method of claim 3, wherein the database is coupled to an Internet website, the method further comprising accessing the Internet website.”	155
P.	Claim 29	157
1.	“The method of claim 3, wherein the database is coupled to a node in a telecommunications network, the method further comprising accessing the node in the telecommunications network.”	157
Q.	Claim 30	158
1.	“The method of claim 3, wherein the database is coupled to a terminal for displaying the data.”	158
XIII.	GROUND 2: HYPERSQL IN VIEW OF ZELLWEGER AND THE KNOWLEDGE OF A POSITA RENDERS OBVIOUS CLAIMS 1-5, 7, 15-19, AND 27-30	162

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

A.	Claim 1	162
1.	[1.PRE] “A method implemented on a computer for searching databases, comprising:”	162
2.	[1.A] “determining a database schema for a database, wherein the database includes database fields;”	164
3.	[1.B] “providing a list of the database fields, wherein the list includes a descriptor indicating a data category;” .	167
4.	[1.C] “receiving a search selection for a database field on the provided list of the database fields;”	169
5.	[1.D] “determining a quantity of entries in the selected database field;”	171
6.	[1.E] “if the quantity exceed a specified amount, truncating data, and displaying the truncated data wherein the truncating reduces characters in one or more entries in the selected database field and the truncated data represents each of the entries in the selected database field; and”	175
7.	[1.F] “if the quantity does not exceed the specified amount, displaying contents of the database field.”	182
B.	Claim 2	183
1.	“The method of claim 1, further comprising providing a key word search.”	183
C.	Claim 3	185
1.	[3.PRE] “A method implemented on a computer for searching a database, comprising:”	185
2.	[3.A] “generating a list of data fields;”	185
3.	[3.B] “receiving a first data field selection from the list of data fields;”	185
4.	[3.C] “determining a first quantity indicative of a number of entries of the selected data field;”	186

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

5.	[3.D] “if the first quantity exceeds a specified limit, reducing a size of data to be displayed from the selected data field wherein the reducing reduces characters in one or more entries in the selected data field and the size-reduced date represents each of the entries in the selected field; and”	186
6.	[3.E] “displaying data from the selected data field.” ...	187
D.	Claim 4	191
1.	“The method of claim 3, wherein the specified limit is fixed.”	191
E.	Claim 5	193
1.	“The method of claim 3, wherein the specified limit is variable.”	193
F.	Claim 7	193
1.	“The method of claim 3, wherein the specified limit is a user-determined limit.”	193
G.	Claim 15	194
1.	[15.A] “The method of claim 3, further comprising: receiving a first constraint, wherein the first constrain[t] is related to a data element in a data field; and”	194
2.	[15.B] “receiving one or more subsequent constraints, wherein search results are generated based on a combination of the first and the one or more subsequent constraints.”	196
H.	Claim 16	199
1.	“The method [of] claim 3, wherein each display of data comprises a search cycle, and further comprising receiving a refresh command, the refresh command resetting all search cycles.”	199
I.	Claim 17	203

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

1.	“The method [of] claim 3, further comprising displaying the list of data fields and a representation of the selected data field, wherein the representation of the selected data field comprises one or more elements from the selected data field.”	203
J.	Claim 18	206
1.	“The method of claim 3, wherein the method comprises executing a plurality of search cycles, each search cycle of the plurality of search cycles providing a search result list comprising selected data elements from the data base.”	206
K.	Claim 19	207
1.	“The method of claim 18, further comprising resetting one or more search cycles of the plurality of search cycles.”	207
L.	Claim 27	208
1.	“The method of claim 3, further comprising displaying the data on a terminal, the terminal including one of a handheld device, a cellular phone, a geosynchronous positioning satellite (GPS) device, a wrist-worn device, an interactive phone device, a household appliance, a television, a television set top box, a handheld computer, a main frame computer and a personal computer.”	208
M.	Claim 28	210
1.	“The method of claim 3, wherein the database is coupled to an Internet website, the method further comprising accessing the Internet website.”	210
N.	Claim 29	212
1.	“The method of claim 3, wherein the database is coupled to a node in a telecommunications network, the method further comprising accessing the node in the telecommunications network.”	212
O.	Claim 30	215

1.	“The method of claim 3, wherein the database is coupled to a terminal for displaying the data.”.....	215
XIV.	GROUND 3: HYPERSQL IN VIEW OF ZELLWEGER, MICROSOFT ACCESS, AND THE KNOWLEDGE OF A POSITA RENDERS OBVIOUS CLAIMS 21-23.....	219
A.	Claim 21	219
1.	[21.A] “The method of claim 3, wherein the method for searching the database comprises searching a first database and a second database, and further comprising:”	219
2.	[21.B] “receiving a first search result from the search of the first database;”.....	221
3.	[21.C] “receiving a second search result from the search of the second database; and”.....	221
4.	[21.D] “combining the first and the second search results.”	222
B.	Claim 22	225
1.	“The method of claim 21, wherein the combining step comprises a Boolean AND operation.”.....	225
C.	Claim 23	227
1.	“The method of claim 21, wherein the combining step comprises a Boolean OR operation.”.....	227
XV.	GROUND 4: HYPERSQL IN VIEW OF ZELLWEGER, MICROSOFT ACCESS, JANES, AND THE KNOWLEDGE OF A POSITA RENDERS OBVIOUS CLAIM 25	230
A.	Claim 25	230
1.	“The method of claim 21, wherein the first search result is represented by a first icon and the second search result is represented by a second icon, and wherein the combining step comprises overlaying the first and the second icons.”	230

XVI. GROUND 5: COCHRAN IN VIEW OF ZELLWEGER, JANES, SPIEGEL, AND THE KNOWLEDGE OF A POSITA RENDERS OBVIOUS CLAIM 26.....	235
A. Claim 26	235
1. [26.A] “The method of claim 3, wherein the displayed data includes one or more data records,”	235
2. [26.B] “each of the one or more data records represented by one or more of text title and an icon, and further comprising overlaying a text title or an icon on a symbol representing an online purchase,”	237
3. [26.C] “wherein the step of overlaying initiates an online purchase of an item represented by the text title or icon.”	239
XVII. MOTIVATIONS TO COMBINE.....	241
A. Motivations to Combine Cochran and Zellweger	241
B. Motivations to Combine HyperSQL and Zellweger	246
C. Motivations to Combine HyperSQL, Zellweger, and Microsoft Access.....	251
D. Motivations to Combine HyperSQL, Zellweger, Microsoft Access, and Janes	256
E. Motivations to Combine Cochran, Zellweger, Janes, and Spiegel	260
XVIII. SECONDARY CONSIDERATIONS OF NON-OBVIOUSNESS.....	264
XIX. DECLARATION	265

I, Erez Zadok, Ph.D., hereby declare as follows:

I. INTRODUCTION

1. I have been retained on behalf of Oracle Corporation (“Oracle” or “Petitioner”) as an independent expert consultant in this proceeding before the United States Patent and Trademark Office (“USPTO”). I am not an employee of Oracle or any affiliate or subsidiary of Oracle.

2. I have been asked to consider whether various references teach or suggest the features recited in certain claims of U.S. Patent No. 6,760,720 to Joseph L. DeBellis, which I refer to herein as the 720 Patent, and whether those references render those claims obvious.

3. My opinions and the bases for my opinions are set forth below.

4. In formulating my opinions, I have relied on my knowledge, training, and experience in the relevant field, which I will summarize briefly below. A more detailed summary of my background, education, experience, and publications is set forth in my curriculum vitae (CV), which is submitted as Ex. 1004.

5. I have personal knowledge of the facts and opinions set forth in this declaration and believe them to be true. If called upon to do so, I would testify competently thereto. I have been warned that willful false statements and the like are punishable by fine or imprisonment, or both.

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

6. My consulting company, Zadoks Consulting, LLC, is being compensated for my time at my current standard consulting rate. I am also being reimbursed for expenses that I may incur during the course of this work. My compensation is not contingent upon the results of my study and analysis, the substance of my opinions, or the outcome of any proceeding involving the claims that Oracle is challenging in its Petition for *Inter Partes* Review (the “Challenged Claims”). I have no financial interest in the outcome of this matter or in any litigation involving the 720 Patent.

II. QUALIFICATIONS

7. I am a Professor in the Computer Science Department at Stony Brook University (part of the State University of New York (“SUNY”) system). I direct the File-systems and Storage Lab (FSL) at Stony Brook’s Computer Science Department. My research interests include file systems and storage systems, operating systems, transactional systems including database technologies, information technology and system administration, security and information assurance, networking, energy efficiency, performance and benchmarking, virtualization, cloud systems, compilers, applied machine learning, and software engineering.

8. I studied at a professional high school in Israel, focusing on electrical engineering (“EE”), and graduated in 1982. I spent one more year at the high school’s college division, receiving a special Certified Technician’s degree in EE. I then went on to serve in the Israeli Defense Forces for three years (1983-1986). I received my Bachelor of Science degree in computer science (“CS”) in 1991, my Master’s degree in CS in 1994, and my PhD in CS in 2001—all from Columbia University in New York.

9. When I began my undergraduate studies at Columbia University, I also started working as a student assistant in the various campus-wide computer labs, eventually becoming an assistant to the head labs manager, who was managing all

public computer labs on campus. During that time, I also became more involved with research within the CS Department at Columbia University, conducting research on operating systems, file and storage systems, distributed and networked systems, security, and other topics. I also assisted the CS department's computer administrators in managing the department's computers, which included storage, IT, networking, and cyber-security related duties.

10. In 1991, I joined Columbia University's CS department as a full-time systems administrator, studying towards my MS degree part-time. My MS thesis topic is related to file system reliability, fault tolerance, replication, and failover in mobile networked storage systems using file virtualization. My main duties as a systems administrator involved installing, configuring, and managing many networked servers, proxies, and desktops running several operating systems, as well as network devices setup; this included many software and hardware upgrades, device upgrades, and BIOS firmware/chipset updates/upgrades. My duties also included ensuring reliable, secure, authenticated access to networked systems/storage and licensed software, as well as software updates, security and bug fixes. Examples of servers and their protocols included email (SMTP), file transfer (FTP), domain names (DNS), network file systems (NFS), network news systems (NNTP), and Web (HTTP).

11. In 1994, I left my systems administrator position to pursue my doctoral studies at Columbia University. My PhD thesis topic was on versatile file system development using stackable (virtualized) file systems, with examples in the fields of security and encryption, efficiency, reliability, and failover. I continued to work part-time as a systems administrator at the CS department, and eventually I was asked to serve as manager to the entire information technology (“IT”) staff. From 1991 to 2001, I was also a member of the faculty-level Facilities Committee that oversaw all IT operations at the CS department.

12. As part of my PhD studies at Columbia, I collaborated on projects to develop advanced AI-like techniques to detect previously unknown viruses (a.k.a. “zero-day malware”), using data mining and rule-based detection. This work led to several highly cited papers (over 1,600 citations for one of the papers alone), and two patents. I also became a Teaching Assistant (TA) for a first-ever Computer Security course given at Columbia University’s CS department with Dr. Matt Blaze as instructor.

13. From 1990 to 1998, I consulted for SOS Corporation and HydraWEB Technologies, as a systems administrator and programmer, managing data storage use and backup/restore duties, databases, web servers, as well as information assurance and cyber-security (*e.g.*, malware protection, software licensing). From 1994 to 2000, I led projects at HydraWEB Technologies, and then became the

Director of Software Development-overseeing the development of several products and appliances such as stateful firewalls and HTTP load-balancers, utilizing network-virtualization and high-availability techniques. From 2009 to 2019, I have consulted for Packet General Networks, a startup specializing in secure, virtualized, network storage and applications' data security in the cloud.

14. In 2001, I joined the faculty of Stony Brook University, a position I have held since that time. In 2002, I joined the Operations Committee, which oversees the IT operations of the CS department at Stony Brook University. From 2006 to 2010, I was the Director of IT Operations of the CS department; my day-to-day duties included setting policies regarding computing, hiring and training new staff, assisting any staff with topics of my specialty, defining requirements for new software/hardware, and purchasing. From 2010 to 2015, I had served as the Co-Chair to the Operations Committee. From 2016 to 2019, I oversaw the IT Operations as the Chair of the Operations Committee. A significant component of these duties included defining and helping implement policies for data management, so as to ensure the security of users and their data, and data reliability and availability, while minimizing the inconvenience and performance impact to users. I personally helped setup and maintain an initial virtual-host infrastructure in the department. Since late 2019, I've been a member of the department's Executive Committee that also oversees all IT operations.

15. In 2017, I became the department's Graduate Academic Adviser, advising all Master students (over 400 annually on average) and many other graduate students on an assortment of academic matters.

16. Since 2001, I have personally configured and managed my own research lab's network. This includes setting up and configuring multiple storage systems (*e.g.*, NFS, CIFS/SMB, NAS), virtual and physical environments, applications such as database (*e.g.*, MySQL, Postgresql), Web servers (*e.g.*, Apache), and mail servers; user access control (*e.g.*, NIS, LDAP), backups and restores, snapshot policies, and more. I've personally installed, configured, changed, replaced parts, and upgraded components in numerous devices including mobile devices, laptops, desktops, and servers, both physical and virtual.

17. Since 1995, I have taught courses on operating systems, storage and file systems, advanced systems programming in Unix/C, systems administration, data structures, data/software security, and more. My courses often use storage, file systems, distributed systems, and system/network security as key teaching principles and practical examples for assignments and projects. I have taught these concepts and techniques to my students, both to my direct advisees as well as in my courses. For example, in my graduate Operating Systems course, I often cover Linux's kernel mechanisms to protect users, applications, and data files, virtual file systems, as well as distributed storage systems (*e.g.*, NFS). And in the System Administration

undergraduate course, I covered many topics such as networking, storage, backups, and configuring complex applications such as mail, web, and database servers.

18. My research often investigates computer systems from many angles: security, efficiency, energy use, scalability, reliability, portability, survivability, usability, ease-of-use, versatility, flexibility, and more. My research gives special attention to balancing five often-conflicting aspects of computer systems: performance, reliability, energy use, security, and ease-of-use.

19. Since joining Stony Brook University in 2001, my group in the File-systems and Storage Lab (FSL) has developed many file systems and operating system extensions; examples include a highly-secure cryptographic file system, a portable copy-on-write (COW) versioning file system, a tracing file system useful to detect intrusions, a replaying file system useful for forensics, a snapshotting and sandboxing file system, a namespace unification file system (that uses stackable, virtualized, file-based COW), an anti-virus file system, an integrity-checking file system, a load balancing and replication/mirroring file system, network file system extensions for security and performance, distributed secure cloud-based storage systems, transactional key-value stores and file systems, OS-level embedded databases, a compiler to convert user-level C code to in-kernel efficient yet safe code, GCC plugins, stackable file system templates, and a Web-based backup system. Many of these projects used one form of virtualization or another (storage, network,

host, etc.). I continue to maintain and release newer versions of some of these file systems and software.

20. I have published over 120 refereed publications (in ACM, IEEE, USENIX, and more). To date, my publications have been cited more than 9,800 times (as per Google Scholar as of November 14, 2023). My papers cover a wide range of related technologies such as file systems, storage systems, transactional systems, security, clouds and virtualization, performance benchmarking and optimization, energy efficiency, system administration, web systems, and more. I also published a book titled “Linux NFS and Automounter Administration” (Sybex, 2001), covering systems administration topics related to network storage and data security.

21. Some of my research has led to public software releases that have been used worldwide. I have publicly maintained the Amd Berkeley Automounter in a package called “am-utils” since 1992; this software helps administrators manage the multitude of file system mounts on dozens of different Unix systems, especially helping to automate access to multiple NFS/NAS storage volumes. Since 1997, I have maintained and released several stackable (virtualized) file system software projects for Linux, FreeBSD, and/or Solaris, in a package called FiST. One of my stackable file system encryption projects, called Cryptfs, became the basis for IBM’s public release of eCryptfs, now part of Linux. Packet General Networks, for whom

I have provided consulting services between 2009 and 2019, licensed another encryption file system called Ncryptfs. Another popular file system released in 2003, called Unionfs, offers virtual namespace unification, transparent shadow copying (a.k.a. copy-on-write or COW), file system snapshotting (*e.g.*, useful for forensics and disaster recovery), and the ability to save disk space by sharing a read-only copy of data among several computers, among other features.

22. My research and teaching make extensive use of data security features. For example, each time I taught the graduate operating system course, the first homework assignment includes the creation of a new system call that performs new or added functionality, often for encrypting a file or verifying its integrity; many of my other assignments cover topics of user/process access control, anti-virus filtering, and more. Since 2001, over 1,000 graduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

23. Moreover, in an undergraduate course titled “Advanced Systems Programming in Unix/C,” I cover many topics of system security and vulnerabilities, such as the structure of UNIX processes, and memory segments such as the heap and stack. This course covers details of several hundred Linux system calls. Often, the first assignment for this course is to develop a tool to encrypt/decrypt files using advanced ciphers, use digital signatures to certify the cipher keys used, and reliably recover files in case of failures. Since 2001, several hundred undergraduate students

were exposed to these principles directly through my teaching and research at Stony Brook University.

24. In another undergraduate course, System Administration, I taught network configuration, security, and storage configuration and reliability. In a special topics course on Storage Systems, I covered many topics such as data deduplication, RAID, transactional storage, storage hardware including modern Flash based ones, virtual storage, backup/restore, snapshots and continuous data protection (CDP), NAS and SAN, and NFS.

25. Overall, in addition to the aforementioned experience, my technical experience relevant to the challenged patent(s) at the time of the alleged invention(s) included the following: I've setup, configured, and used an assortment of databases including SQL (e.g., SunOS/Solaris Sybase, Postgresql) and non-SQL (e.g., Berkeley DB); written and configured stored SQL queries; written code to display information on terminal screens, typically 80x25 in size, using the UNIX "curses" library, which included fitting data into limited display sizes and fields, including truncating data, as well as creating hierarchical, text-based menus; written lots of code in C/C++ which formats and displays numbers, strings, and other data into limited spaces, using functions (e.g., printf) to truncate the strings as needed, reduce the number of significant digits (e.g., truncate, round, trim), and more; used spreadsheets (e.g., excel) to display data into limited-space cells, including

truncating strings and rounding numbers; configured and maintained Web servers; and developed and maintained Web-based input forms (e.g., using Common Gateway Interfaces, or CGIs) to process input data and store it in various databases.

26. My research has been supported by many federal and state grants as well as industry awards, including an NSF CAREER award, two IBM Faculty awards, two NetApp Faculty awards, a Western Digital award, a Facebook award, several Dell-EMC awards, and several equipment gifts. I received the 2008 SUNY Chancellor's Excellence in Teaching award, and the 2022 SUNY Chancellor's Award for Excellence in Scholarship and Creative Activities (both awards can be given only once in a lifetime). In 2021 I was named an ACM Distinguished Member for "Outstanding Scientific Contributions to Computing."

27. My service record to the community includes serving as the co-chair for the USENIX Annual Technical Conference in 2020 (ATC'20); serving as the co-chair for USENIX File and Storage Technologies (FAST'15) in 2015 and on the FAST Conference Steering Committee from 2015 to 2023; serving on the ACM HotStorage Steering Committee since 2021; and serving as the co-chair in 2012 and on the Steering Committee of the ACM SYSTOR conference since 2012. I have served as an Associate Editor to the ACM Transactions on Storage (TOS) journal from 2009 to 2022; in 2022, I was named the Editor-in-Chief for ACM's TOS journal.

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

28. I am a named inventor on four patents, two titled “Systems and Methods for Detection of New Malicious Executables” (U.S. Patent No. 7,487,544, issued February 3, 2009; and U.S. Patent No. 7,979,907, issued July 12, 2011); and two more titled “Multi-Tier Caching,” (U.S. Patent No. 9,355,109, issued May 31, 2016; and U.S. Patent 9,959,279, issued May 1, 2018).

29. I have been disclosed as a testifying expert in 14 cases (including *inter partes* review (IPR) proceedings) in the past four years. I have been deposed 13 times and testified in trial twice.

III. ASSIGNMENT AND MATERIALS CONSIDERED IN FORMING MY OPINIONS

30. I have been asked to provide analysis and explain the subject matter of the 720 Patent, including the state of the art when the 720 Patent application was filed. I have also been asked to consider, analyze, and explain certain prior art to the 720 Patent, including how that art relates to the challenged claims of the 720 Patent and to provide my opinions regarding whether that art renders the claimed subject matter unpatentable as obvious.

31. The opinions expressed in this declaration are not exhaustive of opinions that I may offer in the future regarding the unpatentability of the claims of the 720 Patent. Therefore, the fact that I do not address a particular point should not be understood to indicate an agreement on my part that any claim of the 720 Patent complies with the requirements of any applicable patent or other rule.

32. I reserve the right to amend and supplement this declaration in light of additional evidence, arguments, or testimony presented during this IPR or related proceedings on the 720 Patent.

33. In forming the opinions set forth in this declaration, I have considered and relied upon my education, knowledge of the relevant field, knowledge of scientific and engineering principles, and my experience. I have also reviewed and considered the following additional materials:

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

Exhibit	Description
1001	U.S. Patent No. 6,760,720 to DeBellis (the “720 Patent”)
1002	Patent Prosecution History for U.S. Patent No. 6,760,720 (the “720 FH”)
1004	Curriculum Vitae of Dr. Erez Zadok
1005	U.S. Patent No. 5,995,979 to Cochran, Filed Feb. 10, 1998 (“Cochran”)
1006	Mark Newsome et al., <i>HyperSQL: Web-based Query Interfaces for Biological Databases</i> , IEEE Computer Society, Proceedings of The Thirtieth Annual Hawaii International Conference on System Sciences (1997) (“HyperSQL”)
1007	U.S. Patent No. 6,433,797 to Zellweger, Filed July 15, 1999, Claiming Priority to Provisional Application No. 60/095,811, Filed Aug. 4, 1998 (“Zellweger”)
1009	<i>Using Microsoft Access 97</i> , Special Edition, by Roger Jennings (1997) (“Microsoft Access”)
1010	Library of Congress Catalog Entry for Using Microsoft Access 97, Special Edition, by Roger Jennings (1997) – Full Catalog Record
1011	Library of Congress Catalog Entry for Using Microsoft Access 97, Special Edition, by Roger Jennings (1997) – MARC Tags
1012	U.S. Patent No. 6,642,946 to Janes, Filed Aug. 12, 1999 (“Janes”)
1013	U.S. Patent No. 6,629,079 to Spiegel, Filed June 25, 1998 (“Spiegel”)

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

Exhibit	Description
1014	Declaration of Rachel J. Watters on Authentication of Publication: Mark Newsome et al., <i>HyperSQL: Web-based Query Interfaces for Biological Databases</i> , IEEE Computer Society, Proceedings of The Thirtieth Annual Hawaii International Conference on System Sciences (1997)
1015	Certification From the Library of Congress for <i>Using Microsoft Access 97</i> , Special Edition, by Roger Jennings (1997)
1019	Plaintiffs’ Amended Complaint For Patent Infringement (Corrected), <i>Vilox Technologies, LLC and Joseph DeBellis v. Oracle Corporation</i> , Case No. 1-23-cv-00302-MN (D. Del. Sept. 11, 2023)
1020	Plaintiff’s Original Complaint For Patent Infringement, <i>Vilox Technologies, LLC, et al. v. Salesforce, Inc.</i> , Case No. 6-22-cv-01255-FB (W.D. Tex. Dec. 5, 2022)
1021	Plaintiff’s Original Complaint For Patent Infringement, <i>Vilox Technologies, LLC et al. v. Oracle Corporation</i> , Case No. 6-22-cv-01254-FB (W.D. Tex. Dec. 5, 2022)
1025	<i>Microsoft Computer Dictionary</i> , 4 th ed., Microsoft Press, 1999 (“Microsoft Dictionary”)
1026	<i>4.BSD Programmer’s Reference Manual</i> , O’Reilly & Associates, Inc., 1994 (“BSD Programmer Manual”)
1027	<i>4.BSD Programmer’s Supplementary Documents</i> , O’Reilly & Associates, Inc., 1994 (“BSD Supplementary Docs”)
1028	ANSI/ISO/IEC International Standard (IS), Database Language SQL, Part 2: Foundation (SQL/Foundation), International Standards Organization (ISO), 1999 (“SQL Standard”)
1029	<i>Oracle SQL*Plus: The Definitive Guide</i> , by Jonathan Gennick, O’Reilly & Associates, Inc., 1999 (“SQL*Plus Guide”)

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

Exhibit	Description
1030	ODBC 3.5 Developer's Guide by Roger E. Sanders, McGraw-Hill, 1998 ("ODBC Guide")
1031	<i>Oracle Developer™, Guidelines for Building Applications, Release 6.0</i> , September 1998 ("Oracle Building Apps Guide")

IV. UNDERSTANDING OF GOVERNING LAW

34. I am not an attorney but have been instructed in and applied the law as described in this section.

A. Claim Construction

35. I understand that the first step in determining whether a patent claim would have been anticipated or obvious is to ascertain how a person of ordinary skill in the art (“POSITA”) would have understood the claim terms.

36. I have been instructed by counsel on the law regarding claim construction and patent claims, and I understand that a patent may include two types of claims: independent claims and dependent claims. An independent claim stands alone and includes only the limitations it recites. A dependent claim can depend from an independent claim, or it can further depend from another dependent claim. I understand that a dependent claim includes all the limitations that it recites, in addition to all the limitations recited in the claim(s) from which it depends.

37. It is my understanding that in proceedings before the USPTO, the claims of a patent are to be construed under what is referred to as the “Phillips standard.” I understand that this means that claim terms of a patent are given the ordinary and customary meaning they would have to a POSITA in view of the description provided in the patent itself and the patent’s file history.

38. I understand that to determine how a person of ordinary skill would understand a claim term, one should look to those sources available that show what a person of skill in the art would have understood the disputed claim language to mean. I understand that, in construing a claim term, one looks primarily to the intrinsic patent evidence, including the words of the claims themselves, the remainder of the patent specification, and the patent's prosecution history. I understand that extrinsic evidence, which is evidence external to the patent and the prosecution history, may also be useful in interpreting patent claims when the intrinsic evidence itself is insufficient.

39. I understand that words or terms should be given their ordinary and accepted meaning unless it clearly appears that the inventors were using them to mean something else. In making this determination, the claims, the remainder of the patent, and the prosecution history are of paramount importance. Additionally, the patent and its prosecution history must be consulted to confirm whether the patentee has acted as its own lexicographer (*i.e.*, provided its own special meaning to any disputed terms) or intentionally disclaimed, disavowed, or surrendered any claim scope.

40. I further understand that in proceedings before the PTAB, claim terms need only be construed to the extent necessary to resolve any controversy that may arise.

41. In comparing the claims of the Challenged Patent to the prior art, I have considered the 720 Patent and its file history in light of the understanding of a POSITA at the time of the alleged invention. I address further how a POSITA would have understood the claims of the alleged invention in §IX below.

B. Obviousness

42. I have been further instructed and understand that a patent claim is unpatentable and invalid as obvious if the subject matter of the claim as a whole would have been obvious under 35 U.S.C. §103(a) to a person of ordinary skill in the art of the claimed subject matter as of the time of the invention at issue. I understand that when assessing the obviousness of claimed subject matter, the following factors are evaluated: (1) the scope and content of the prior art; (2) the difference or differences between each claim of the patent and the prior art; and (3) the level of ordinary skill in the art at the time the patent was filed. I also understand that the Board considers certain objective factors if they are present, generally referred to as “secondary considerations” or “objective indicia” of nonobviousness, as further discussed below in this section.

43. I have been informed that there are two ways to determine whether prior art is analogous and thus can be considered prior art: (1) whether the art is from the same field of endeavor, regardless of the problem addressed, and (2) if the reference is not within the field of the patentee’s endeavor, whether the reference still is

reasonably pertinent to the particular problem with which the patentee is involved. I have also been informed that the field of endeavor of a patent is not limited to the specific point of novelty, the narrowest possible conception of the field, or the particular focus within a given field. I have also been informed that a reference is reasonably pertinent if, even though it may be in a different field from that of the patentee's endeavor, it is one which, because of the matter with which it deals, logically would have commended itself to a patentee's attention in considering his or her problem.

44. I understand that claimed subject matter may be obvious in view of more than one item of prior art. I understand, however, that it is not enough to show simply that all the limitations of the claimed subject matter are spread throughout the prior art. Instead, for claimed subject matter to be obvious over multiple references, there generally must have been some reason or motivation for one of ordinary skill in the art to combine the teachings of the prior art references to arrive at the claimed subject matter.

45. I have been informed that, in seeking to determine whether an invention that is a combination of known elements would have been obvious to a person of ordinary skill in the art at the time of the invention, one must consider the references in their entirety to ascertain whether the disclosures in those references render the combination obvious to such a person.

46. I have been informed and understand that, while not required, the prior art references themselves may provide a teaching, suggestion, motivation, or reason to combine, but that other times the motivation linking two or more prior art references is common sense to a POSITA at the time of the invention.

47. I understand that a particular combination may be proven obvious merely by showing that it was obvious to try the combination. I have been informed that, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill. I have further been informed and understand that a POSITA combines the teachings of the prior art references and does not necessarily combine the embodiments of those teachings physically.

48. I further understand that an obviousness analysis recognizes that market demand, rather than scientific literature, often drives innovation, and that a motivation to combine references also may be supplied by the direction of the marketplace. For example, when there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp because the result is likely the product not of innovation but of ordinary skill and common sense.

49. I have been informed that the combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results. Thus, where all of the elements of a claim are used in substantially the same manner in devices in the same field of endeavor, the claim is likely obvious.

50. Additionally, I understand that a patent is likely to be invalid for obviousness if a person of ordinary skill can implement a predictable variation or if there existed at the time of the invention a known problem for which there was an obvious solution encompassed by the patent's claims. Therefore, when a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one.

51. I further understand that combining embodiments related to each other in a single prior art reference would not ordinarily require a leap of inventiveness.

52. I also understand that one of ordinary skill in the art must have had a reasonable expectation of success when combining references for claimed subject matter to be obvious.

53. I have been informed and I understand that factors referred to as "objective indicia of non-obviousness" or "secondary considerations" are also to be considered when assessing obviousness when such evidence is available. I understand that these factors can include: (1) commercial success; (2) long-felt but

unresolved needs; (3) copying of the invention by others in the field; (4) initial expressions of disbelief by experts in the field; (5) failure of others to solve the problem the claimed subject matter solved; and (6) unexpected results.

54. I also understand that evidence of objective indicia of nonobviousness must be commensurate in scope with the claimed subject matter. I further understand that there must be a relationship, sometimes referred to as a “nexus,” between any such secondary indicia and the claimed invention. I also have been informed that even when they are present, secondary considerations may be unable to overcome primary evidence of obviousness (*e.g.*, motivation to combine with predictable results) that is sufficiently strong.

55. Finally, I have been informed that one cannot use hindsight to determine that an invention was obvious.

56. I provide my opinions in this declaration based on the guidelines set forth above.

V. LEVEL OF SKILL IN THE ART

57. I have been informed and understand that the level of ordinary skill in the relevant art at the time of the invention is relevant to inquiries such as the meaning of claim terms, the meaning of disclosures found in the prior art, and the reasons a POSITA may have for combining references.

58. I have been informed and understand that factors that may be considered in determining the level of ordinary skill include: (1) the education of the inventor; (2) the type of problems encountered in the art; (3) prior art solutions to those problems; (4) rapidity with which innovations are made; (5) sophistication of the technology; and (6) education level of active workers in the relevant field. I have been further informed and understand that a POSITA is also a person of ordinary creativity.

59. A POSITA at the time of the alleged invention would, in my opinion, have had at least a bachelor's degree in computer science, electrical engineering, or a related subject, and at least two years of experience with databases, computer networks, and related web technologies. A more advanced degree can substitute for less work experience and vice versa.

60. I also recognize that in some situations the inventor of a patent may not fit the description of a regular POSITA at the time of the alleged invention of their own patent. For example, where an inventor is sufficiently self-taught in some

subject matter, they may find themselves knowledgeable enough to file and be granted a patent over that subject matter. For example, I recognize that the inventor, Joseph L. De Bellis, around the time of the invention of the 720 Patent, was a medical doctor specializing in Plastic Surgery and “CEO and Founder” of an entity called “Virtual Logistix Inc.” See <https://www.linkedin.com/in/joseph-l-de-bellis-md-b610164/>. Despite this, my opinion about the qualifications and experience of the POSITA for this patent remains unchanged. My opinion is founded in the subject matter of the 720 Patent and my own experience working on technologies intimately related to—and applied in—the 720 Patent, including database operation and management, computer networking, and related web technologies.

61. In view of my educational background and work experience, I am a person of more than the ordinary level of skill in the art of the 720 Patent and was so by February 25, 2000 (when the first application relating to the 720 Patent was filed). In forming my opinions, I have drawn on my academic background and professional experiences. My opinions herein, however, were formed taking into account the perspective of a POSITA.

VI. THE 720 PATENT'S EFFECTIVE FILING DATE

62. I understand that the application leading to the 720 Patent was filed on February 25, 2000.

63. For purposes of this declaration, I have been instructed to use February 25, 2000, as the effective filing date of the 720 Patent. My opinions in this declaration were formed from the perspective of a POSITA as of February 25, 2000, including both the knowledge of a POSITA at that time as well as how a POSITA would have understood the prior art.

VII. TECHNOLOGY OVERVIEW

64. In this section, I provide a brief overview of the background art as understood by a POSITA before the February 25, 2000, filing date.

65. To reiterate, the following overview of the background information regarding the art in this section (and all subsections) explains what a POSITA would have understood before the February 25, 2000, filing date.

A. General Computer Operations

66. Computers are hardware and software devices that execute instructions. The hardware often includes a central processing unit (CPU), typically called “processor,” volatile memory (*e.g.*, RAM (random-access memory), or DRAM (dynamic random-access memory)), non-volatile (persistent) storage (*e.g.*, a hard disk or flash drive), and input/output (I/O) devices (*e.g.*, network interface card, mouse, keyboard, display). RAM, hard disks, and flash drives are considered, generally, “storage” or “memory” devices because they can store and hold bits of information (whether persistently or not). Computers can have multiple storage devices. Processors are coupled to main memory using a memory bus and to peripherals such as hard disks and network interface cards using one or more I/O busses.

67. Memory and storage devices have considerable differences in their speeds, capacities, and costs. Non-volatile storage devices such as hard disks have

the largest capacity, are least expensive per gigabyte, and are slowest (typically operating at millisecond speeds). Volatile DRAM devices are smaller, more expensive, and faster than hard disks (typically operating at microsecond speeds). CPUs' own memory caches and operations are the smallest, most expensive, and fastest (typically operating at nanosecond speeds). Overall, CPUs are about 1,000× faster than DRAM, which in turn is about 1,000× faster than hard disks.

68. Computer programs are often written in a high-level human-readable language (*e.g.*, C, C++, Java, Perl, PHP, Python), then translated using a compiler or script processor to machine instructions understood by the CPU (*e.g.*, Intel or AMD processor). Machine instructions—generally called “software”—are stored in files on persistent media, loaded into DRAM, and then loaded into the CPU where they can be executed. All modern computers operate in this manner, regardless of their purpose: a small laptop, a server in a data center, a mainframe computer, a personal workstation, a proxy device, a gateway, a firewall, a router, an appliance, a virtual machine, a handheld or mobile device, smart card, etc. Computer systems may run multiple software programs, from background servers and daemons to user applications.

69. Users (including administrators) can login to one or more computers (*e.g.*, desktop, laptop, server) and execute one or more applications (*e.g.*, word processing, web browsing, backup tool).

B. Determining Display and Terminal Capabilities

70. The operating system (OS) software manages the computer's hardware and user applications, ensuring safety, security, and fairness in resource management and scheduling. Every hardware device has certain capabilities and limitations. At boot time, the OS discovers all devices and peripherals, and queries each for their capabilities. For example, the OS finds out how many processors (CPUs) exist on the system and how many cores they have; how much RAM is available on the system; how large are storage devices such as hard disks; how fast are network interfaces; what is the screen resolution of any display including the number of vertical and horizontal pixels, color depth capabilities, and more.

71. It is critical for the OS to know the hardware limits to protect the system from inadvertent misuse: for example, preventing a rogue application from attempting to write beyond available memory, as that could result in a buffer-overflow bug and may even crash the system. The OS also exports certain properties to user applications: for example, many graphical user applications must know the screen's resolution so they can display information in chunks that are not too small to be hard to see, and yet not too large so the information can fit on the screen. Therefore, OSs provide various functions and mechanisms to control and retrieve parameters, often via I/O control (ioctl) system calls.

72. In Unix, “The `ioctl()` function manipulates the underlying device parameters of special files. In particular, many operating characteristics of character special files (e.g. terminals) may be controlled with `ioctl()` requests.” Ex. 1026 (BSD Programmer Manual), p. 3. In some operating systems, the “`stty`” and/or “`gtty`” commands were used to “retrieve[] the state of the terminal.” *Id.*, p. 8.

73. One specific I/O control command is the “`TIOCGWINSZ`” (which stands for “Terminal I/O Control—Get Window Size”). This command will “Put the window size information associated with the terminal in the *winsize* structure pointed to by *ws*. The window size structure contains the number of rows and columns (and pixels if appropriate) of the devices attached to the terminal. It is set by user software and is the means by which most full-screen oriented programs determine the screen size. The *winsize* structure is defined in `<sys/ioctl.h>`.” Ex. 1026 (BSD Programmer Manual), p. 12.

74. In more modern operating systems (closer but still prior to the February 25, 2000 filing date of the 720 Patent), the terminal capabilities are discovered by the operating system and available to query via the aforementioned `ioctl()` commands. In certain older operating systems, that information was available in a “terminal capability database” file, called the “`termcap`” file, that users and applications could query. Ex. 1026 (BSD Programmer Manual), p. 14. The “`co`”

capability defined the “Number of columns in a line.” *Id.*, p. 15. The “li” capability defined the “[n]umber of lines on screen or page.” *Id.*, p. 16.

75. To enable the development of full-screen applications, Unix developers designed the “curses” library as an “aid for writing screen-oriented, terminal-independent programs.” Ex. 1027 (BSD Supplementary Docs), p. 3. Any application using the “curses” library had to initialize it by calling the “`initscr()`” routine; this was critical for the application to know the precise terminal or display dimensions. “If the system supports the **TIOCGWINSZ** *ioctl*(2) call, it is used to get the number of lines and columns for the terminal, otherwise it is taken from the **termcap** description.” *Id.*, p. 4. (I have personally developed software using the curses library in the 1990s, as discussed in §II above.)

C. Databases

76. A *database* is “[a] file composed of records, each containing fields together with a set of operations for searching, sorting, recombining, and other functions.” Ex. 1025 (Microsoft Dictionary), p. 5. As such, even an Excel spreadsheet is a form of a database.

77. A *database management system* (DBMS) is “[a] software interface between the database and the user. A database management system handles user requests for database actions and allows for control of security and data integrity

requirements. *Acronym*: DBMS. *Also called* database manager. *See also* database engine.” Ex. 1025 (Microsoft Dictionary), p. 5.

78. A *database server* is “[a] network node, or station, dedicated to storing and providing access to a shared database. *Also called* database machine.” Ex. 1025 (Microsoft Dictionary), p. 5.

79. Large databases must be organized carefully to use resources efficiently and permit versatile access. That is why the world’s most complex databases are often designed as multiple tables, each with one or more index keys, and logical connections (or relations) across the tables.

80. In large databases, there could be a lot of data in complex structures. A *relational database* is “[a] database or database management system that stores information in tables—rows and columns of data—and conducts searches by using data in specified columns of one table to find additional data in another table. In a relational database, the rows of a table represent records (collections of information about separate items) and the columns represent fields (particular attributes of a record). In conducting searches, a relational database matches information from a field in one table with information in a corresponding field of another table to produce a third table that combines requested data from both tables. For example, if one table contains the fields EMPLOYEE-ID, LAST-NAME, FIRST-NAME, and HIRE-DATE, and another contains the fields DEPT, EMPLOYEE-ID, and

SALARY, a relational database can match the EMPLOYEE-ID fields in the two tables to find such information as the names of all employees earning a certain salary or the departments of all employees hired after a certain date. In other words, a relational database uses matching values in two tables to relate information in one to information in the other. []." Ex. 1025 (Microsoft Dictionary), p. 26.

81. In relational databases, a *table* is "a data structure characterized by rows and columns, with data occupying or potentially occupying each cell formed by a row-column intersection. The table is the underlying structure of a relation." Ex. 1025 (Microsoft Dictionary), p. 34.

82. Moreover, each *data field* is "[a] well-defined portion of a data record, such as a column in a database table." Ex. 1025 (Microsoft Dictionary), p. 6.

83. Databases index their information for easier access. An *index* is "[a] listing of keywords and associated data that point to the location of more comprehensive information, such as files and records on a disk or record keys in a database." Ex. 1025 (Microsoft Dictionary), p. 15 ("index¹" definition). To *index* or find an item "in a database [is] to find data by using keys such as words or field names to locate records." *Id.*, p. 5 ("index²" definition).

84. The overall structure of the database is called a *schema*; it defines the tables, indexes, relations, types, data categories, and more. A *schema* is "[a] description of a database to a database management system (DBMS) in the language

provided by the DBMS. A schema defines aspects of the database, such as attributes (fields) and domains and parameters of the attributes.” Ex. 1025 (Microsoft Dictionary), p. 28.

85. Any database, especially a large and complex one, would not be useful unless it could be accessed by users and applications. That is why databases include methods and interfaces for universal access to query the database. Because databases have been so important and used for many years, standards emerged to define a uniform query language.

86. The *structured query language* (SQL) is “[a] database sublanguage used in querying, updating, and managing relational databases—the de facto standard for database products. *Acronym: SQL.*” Ex. 1025 (Microsoft Dictionary), p. 33. In fact, “SQL is a language designed to be used only with databases.” *Id.*, p. 11 (“general purpose language” definition).

87. The SQL language first appeared in the 1970s and became a standard in the 1980s; multiple drafts and releases of the SQL standard were released over a period of nearly half a century. *See generally* <https://en.wikipedia.org/wiki/SQL> (last viewed November 20, 2023). SQL has grown in complexity over the years, adding many features that have gotten standardized by the International Standards Organization (ISO). For example, “ISO/IEC 9075-2,” a 1,147-pages long specification, published in 1999, “defines the data structures and basic operations on

SQL-data. It provides functional capabilities for creating, accessing, maintaining, controlling, and protecting SQL-data.” Ex. 1028 (SQL Standard), p. 27.

88. To *query* a database is “[t]o extract data from a database and present it for use.” Ex. 1025 (Microsoft Dictionary), p. 25 (“query²” definition).

89. Users can use SQL to query a database. And as with any programming language, queries can use an assortment of logical or Boolean operators to construct more refined queries. A *Boolean operator* is “[a]n operator designed to work with Boolean values. The four most common Boolean operators in programming use are AND (logical conjunction), OR (logical inclusion), XOR (exclusive OR), and NOT (logical negation). Boolean operators are often used as qualifiers in database searches—for example, *find all records where DEPARTMENT = ‘marketing’ OR DEPARTMENT = ‘sales’ AND SKILL = ‘word processing’*. Also called logical operator. See also AND, exclusive OR, NOT, OR.” Ex. 1025 (Microsoft Dictionary), p. 3.

90. With the basic logical operators of NOT, OR, and AND, one can trivially construct more complicated logical operators using basic Boolean math. For example, the non-disjunctive (“NOT OR”) is abbreviated as “NOR.” Continuing with the above example, a logical OR was used to search a database for records where “DEPARTMENT = ‘marketing’ OR DEPARTMENT = ‘sales’”. Suppose a POSITA wanted to search for all database records whose department was

the opposite—neither ‘marketing’ nor ‘sales’. The negation of that OR expression, using NOR, can be simply written as “DEPARTMENT = ‘marketing’ NOR DEPARTMENT = ‘sales’”. This is mathematically identical to “NOT (DEPARTMENT = ‘marketing’ OR DEPARTMENT = ‘sales’)” and also identical to “NOT (DEPARTMENT = ‘marketing’) AND NOT(DEPARTMENT = ‘sales’)”. In other words, even if a database query language does not support a NOR operation explicitly, an identical functionality can be achieved with just NOT and OR, or just NOT and AND. See generally basic NOR definition, at https://en.wikipedia.org/wiki/Logical_NOR (last viewed November 16, 2023).

91. Users can use logical AND operators to refine a search query: each AND expression further refines the query, typically narrower and narrower. In addition, databases support several types of operators that can combine, merge, or join information from several databases and tables.

92. A *join* is “[a] database table operation that creates a resultant entry in another table for each entry in the one table whose key field matches that of an entry in the other. See also inner join.” Ex. 1025 (Microsoft Dictionary), p. 17. Databases support other join variants. See also *id.*, p. 16 (“inner join” definition), p. 21 (“outer join” definition), p. 23 (“product” operator definition), and p. 26 (“relational algebra” definition).

D. Summarizing and Formatting Information in Limited Spaces

93. When retrieving information from a database, users often want to format the data to fit a specific size. Most programming languages include operations to truncate or round numbers and strings. See Ex. 1025 (Microsoft Dictionary), p. 35 (“**truncate** *vb.* To cut off the beginning or end of a series of characters or numbers; specifically, to eliminate one or more of the least significant (typically rightmost) digits. In truncation, numbers are simply eliminated, unlike rounding, in which the rightmost digit might be incremented to preserve accuracy. *Compare* round.”), p. 27 (“**round** *vb.* To shorten the fractional part of a number, usually increasing the last remaining (rightmost) digit or not, according to whether the deleted portion was over or under 5. For example, 0.3333 rounded to two decimal places is 0.33, and 0.6666 is 0.67. Computer programs often round numbers, sometimes causing confusion when the resulting values do not add up ‘correctly.’ Percentages in a spreadsheet can thus total 99 percent or 101 percent because of rounding.”).

94. Truncation or rounding of numbers is well known to a POSITA. For example, the C programming language, created in the early 1970s, is often taught to beginner computer science majors. In C, users can use the print command to fit and round numbers. For example, the statement “printf(“%5.3f\n”, 3.1415926535);” says to format the floating-point number PI into five spaces,

while truncating up to three decimal points; this statement would print “3.142” (note that here, the period counts towards the five spaces). *See* Ex. 1026 (BSD Programmer Manual), pp. 4-7.

95. SQL itself supports multiple ways in which numbers, strings, etc. could be formatted to fit into limited spaces, truncated, rounded, and more. As discussed in §VII.C above, SQL is a rich, powerful, and versatile language. Therefore, Oracle itself created “SQL*Plus” to make it easier for more people to use SQL.

96. “SQL*Plus is essentially an interactive query tool, with some scripting capabilities. It is a non-GUI, character-based tool that has been around since the dawn of the Oracle age.” Ex. 1029 (SQL*Plus Guide), p. 3.

97. In SQL*Plus, “[a] *format specification* is a string of characters that tells SQL*Plus exactly how to format a number, date, or text string when it is displayed. The most notable of these commands is the COLUMN command, which is used to format columns of output from a SELECT query. There are other commands as well. ... ACCEPT ... SET NUMBER Defines the default display format for numbers. ... TTITLE, BTITLE, REPHEADER, REPFOOTER ...”. Ex. 1029 (SQL*Plus Guide), p. 5.

98. Just as with the aforementioned “printf” function, when formatting numbers, SQL*Plus can control the number of digits displayed: “9s are used to control the number of significant digits to be displayed.” Ex. 1029 (SQL*Plus

Guide), p. 6. It can even format decimal numbers as roman numerals. *Id.*, pp. 7-8. Strings can be truncated to any desired length. *Id.*, pp. 8-9. Similarly, dates can be formatted in SQL*Plus to summarize or abbreviate month names to just three characters, years to omit the century, and remove hours, minutes, and/or seconds as needed. *Id.*, pp. 9-12 (*see especially* Table B-4, Date Format Examples).

E. Web Technologies and Graphical User Interfaces

99. A *Web site* is “[a] group of related HTML documents and associated files, scripts, and databases that is served up by an HTTP server on the World Wide Web. The HTML documents in a Web site generally cover one or more related topics and are interconnected through hyperlinks. Most Web sites have a home page as their starting point, which frequently functions as a table of contents for the site. Many large organizations, such as corporations, will have one or more HTTP servers dedicated to a single Web site. However, an HTTP server can also serve several small Web sites, such as those owned by individuals. Users need a Web browser and an Internet connection to access a Web site. *See also* home page, HTML, HTTP server (definition 1), Web browser.” Ex. 1025 (Microsoft Dictionary), pp. 37-38.

100. Web browsers are one example of a graphical user interface (GUI). A *GUI* is “[a] visual computer environment that represents programs, files, and options with graphical images, such as icons, menus, and dialog boxes on the screen. The user can select and activate these options by pointing and clicking with a mouse or,

often, with the keyboard. A particular item (such as a scroll bar) works the same way to the user in all applications, because the graphical user interface provides standard software routines to handle these elements and report the user's actions (such as a mouse click on a particular icon or at a particular location in text, or a key press); applications call these routines with specific parameters rather than attempting to reproduce them from scratch. *Acronym: GUI.*" Ex. 1025 (Microsoft Dictionary), p. 12.

101. *Windows* is another old example of a graphical system. It is "[a]n operating system introduced by Microsoft Corporation in 1983. Windows is a multitasking graphical user interface environment that runs on MS-DOS-based computers (Windows 3x and Windows for Workgroups), and as a self-contained operating system for desktop computers (Windows 9x), workstations, (Windows NT Workstation, Windows 2000 Professional), and network servers (Windows NT Server, Windows NT Enterprise Edition, Windows 2000 Server, and Windows 2000 Advanced Server). Windows provides a standard graphical interface based on drop-down menus, windowed regions on the screen, and a pointing device such as a mouse." Ex. 1025 (Microsoft Dictionary), p. 39.

102. The *X Window System* is "[a] nonproprietary standardized set of display-handling routines, developed at MIT. Most often encountered on UNIX workstations, the X Window System is independent of hardware and operating

system. An X Window System client calls on the server, which is located on the user's workstation, to provide a window in which the client can generate a display of text or graphics. *Also called X Windows. See also X Consortium.*" Ex. 1025 (Microsoft Dictionary), p. 40.

103. Application and Web content developers can develop graphical interfaces to access information. *Visual programming* is "[a] method of programming using a programming environment or language in which basic program components can be selected through menu choices, buttons, icons, and other predetermined methods." Ex. 1025 (Microsoft Dictionary), p. 36.

104. Graphical operations are often integrated with general purpose programming languages such as Java. One example is *Visual J++*, "Microsoft's Java visual programming environment, which can be used to create applets and applications in the Java language. *See also* applet, Java, Java applet, visual programming." Ex. 1025 (Microsoft Dictionary), p. 36.

105. Developers can design Web sites with many standard GUI features including menus, selectors, lists, and scrolling for efficient and convenient information presentation to users.

106. A *menu* is "[a] list of options from which a user can make a selection in order to perform a desired action, such as choosing a command or applying a particular format to part of a document. Many application programs, especially

those that offer a graphical interface, use menus as a means of providing the user with an easily learned, easy-to-use alternative to memorizing program commands and their appropriate usage.” Ex. 1025 (Microsoft Dictionary), p. 19.

107. A *pop-up menu* or *popup menu* is, “[i]n a graphical user interface, a menu that appears on-screen when a user selects a certain item. Pop-up menus can appear anywhere on the screen and generally disappear when the user selects an item in the menu. *Also called* popup. *Compare* drop-down menu, pull-down menu.” Ex. 1025 (Microsoft Dictionary), p. 22.

108. A *pull-down menu* is “[a] menu that is pulled down from the menu bar and that remains available as long as the user holds it open. *Compare* drop-down menu.” Ex. 1025 (Microsoft Dictionary), p. 24.

109. A *list box* is “[a] control in Windows that enables the user to choose one option from a list of possibilities. The list box appears as a box, displaying the currently selected option, next to a button marked with a down-arrow. When the user clicks on the button, the list appears. The list has a scroll bar if there are more options than the list has room to show.” Ex. 1025 (Microsoft Dictionary), p. 18.

110. Often, the information one has to process is larger than can be viewed in one display screen. Just as people can scroll up and down a newspaper article, and flip back and forth over pages, so do computer systems.

111. To *scroll* is “[t]o move a document or other data in a window in order to view a particular portion of the document. Scrolling may be controlled by the mouse, arrow keys, or other keys on the keyboard. *See also* scroll bar.” Ex. 1025 (Microsoft Dictionary), p. 29.

112. A *scroll bar* is, “[i]n some graphical user interfaces, a vertical or horizontal bar at the side or bottom of a display area that can be used with a mouse for moving around in that area. Scroll bars often have four active areas: two scroll arrows for moving line by line, a sliding scroll box for moving to an arbitrary location in the display area, and gray areas for moving in increments of one window at a time.” Ex. 1025 (Microsoft Dictionary), p. 30.

113. GUIs also offer *horizontal scrolling*, “[a] feature of programs such as word processors and spreadsheets that enables the user to scroll left and right to display information beyond the horizontal limits of the screen (or window, in a graphical user interface).” Ex. 1025 (Microsoft Dictionary), p. 14.

114. To make scrolling efficient, the amount of information presented should fit in the display in question: if it is too much, the information may “spill over” the edge of the screen and essentially be invisible to users. Conversely, if the information presented is too small (*e.g.*, just a few items out of many), the user will be inconvenienced with excessive scrolling to reach the desired information.

115. Sometimes, there is too much information to present in a single menu—and even using a single scroll bar can be cumbersome, having to scroll repeatedly through a long list. That is where organizing the displayed data records hierarchically helps.

116. A *hierarchical menu* is “[a] menu that has one or more submenus. Such a menu/submenu arrangement is hierarchical because each level subsumes the next.” Ex. 1025 (Microsoft Dictionary), p. 13. A submenu (or “child menu”) is “[a] menu that appears as the result of the selection of an item on another, higher-level menu.” *Id.*, p. 33.

117. Because of the ever-changing nature of the Web, content creators and users alike desired a more flexible way to browse information—such that it adapts more quickly to specific needs. *Dynamic HTML* is “[a] technology designed to add richness, interactivity, and graphical interest to Web pages by providing those pages with the ability to change and update themselves dynamically, that is, in response to user actions, without the need for repeated downloads from a server. Dynamic HTML enables client-side scripts programs that affect elements on a Web page produced with languages such as VBScript and JavaScript to control and manipulate elements, such as fonts and graphics, by means of HTML tags that describe how the page is to appear on the user’s screen. From the user’s point of view, dynamic HTML produces the type of interactivity and relatively rapid download times

associated with multimedia CD-ROM products. Examples of dynamic HTML actions include moving graphics on the page and displaying information, such as menus or tables, in response to mouse movements or clicks. A number of vendors, including Microsoft and Netscape, have developed their own versions of Dynamic HTML and have submitted them to the World Wide Web Consortium (W3C) for possible inclusion in the Document Object Model (DOM) specification being developed by W3C. *See also* Document Object Model.” Ex. 1025 (Microsoft Dictionary), pp. 9-10.

118. GUIs enable several convenient operations using the mouse alone, such as picking objects like icons from one place on the screen, dragging them to another location, and dropping them in the new location—to effectuate new functionality. To *drag-and-drop* is “to perform operations in a graphical user interface by dragging objects on the screen with the mouse. For example, to delete a document, a user can drag the document icon across the screen and drop it on the trashcan icon (Macintosh OS) or in the Recycle Bin (Windows). *See also* drag, graphical user interface.”. Ex. 1025 (Microsoft Dictionary), pp. 7-8 (definition 2). Drag-and-drop could be used not just to delete a file, but also to add items to a shopping cart, to reorganize a set of files on a desktop, and more.

F. Accessing and Searching Databases Via GUIs and the Web

119. Databases' main purpose is to enable searching and updating data in indexed tables and records, using an expression that denotes selection criteria, keywords, and more. Indeed, one of the most common and heavily used SQL command is SELECT. To select is, "[i]n database management, to choose records according to a specified set of criteria. *See also* sort." Ex. 1025 (Microsoft Dictionary), p. 32 (definition 2).

120. A *Web site* is "[a] group of related HTML documents and associated files, scripts, and databases that is served up by an HTTP server on the World Wide Web. ["]". Ex. 1025 (Microsoft Dictionary), pp. 479-480.

121. There are multiple ways to search for information (*e.g.*, stored in databases) using Web site UIs. See Ex. 1025 (Microsoft Dictionary), p. 4 ("CGI *n.* 1. Acronym for Common Gateway Interface. The specification that defines communications between information servers (such as HTTP servers) and resources on the server's host computer, such as databases and other programs. For example, when a user submits a form through a Web browser, the HTTP server executes a program (often called a CGI script) and passes the user's input information to that program via CGI. The program then returns information to the server via CGI. Use of CGI can make a Web page much more dynamic and add interactivity for the user. ["]").

122. A *search engine* is a special Web site designed explicitly for searching information using a Web browser. Ex. 1025 (Microsoft Dictionary), p. 31 (“**search engine** *n.* 1. A program that searches for keywords in documents or in a database. 2. On the Internet, a program that searches for keywords in files and documents found on the World Wide Web... [] Others search across many sites, using such agents as spiders to gather lists of available files and documents and store these lists in databases that users can search by keyword. []”). Often, Web-based search engines offer a much easier and user-friendlier method to search for information; conversely, using SQL queries directly, while having powerful capabilities, is more complex.

123. Multiple databases can be accessed over the network. Ex. 1025 (Microsoft Dictionary), p. 20 (“**ODBC** *n.* Acronym for **open database connectivity**. In the Microsoft WOSA (Windows Open System Architecture) structure, an interface providing a common language for Windows applications to gain access to a database on a network. *See also* WOSA.”).

124. A database server can manage one or more databases, and multiple servers can each be accessible over a network. ODBC “[d]ata sources consist of sets of data and their associated environments, which include operating system, DBMS, and network platform used (if any) to access the DBMS that an application wants to access.” Ex. 1030 (ODBC Guide), p. 3; *see also id.*, p. 4 (Figure 2-2 “How the four

components of ODBC architecture recognize and work with file-oriented and client/server DBMS data sources”).

125. In ODBC, “[a] *data source* is simply the source of the data. It can be a flat-file, a particular database in a DBMS, or even a spreadsheet file. For example, a data source might be an Oracle DBMS running on Windows NT, accessed by Novell Netware; an IBM DB2 DBMS running on MVS, accessed through a gateway; a collection of dBASE files in a server directory; or a local Microsoft Access database file. In ODBC, the term *data source* refers to the actual location of the data as well as all the technical information needed to access the data: the driver name, network address, network software, and so on.” Ex. 1030 (ODBC Guide), p. 5.

126. A program using the ODBC API can connect to one or more data sources, each a database. Ex. 1030 (ODBC Guide), p. 6, Figure 6-1 (“How ODBC connection handles are established and used.”). “Once a connection handle has been allocated and its attributes/options have been set, an application can use it to connect to a data source or driver. Three different ODBC API functions can be used to establish a connection. They are: * SQLConnect() * SQLDriverConnect() * SQLBrowseConnect().” *Id.*, p. 7.

127. ODBC permits an application to connect to a plurality of data sources (databases). ODBC APIs enable applications to submit SQL queries to any established data source connection. “Applications can use any combination of these

functions to connect to any number of data sources and drivers—at the same time if necessary. Additionally, an application’s connections can consist of a variety of drivers and data sources, a single driver and a variety of data sources, or even multiple connections to the same driver and data source.” Ex. 1030 (ODBC Guide), p. 7. Figure 7-1 below shows how ODBC can connect and access multiple data sources, including several databases, spreadsheets, and even text files:

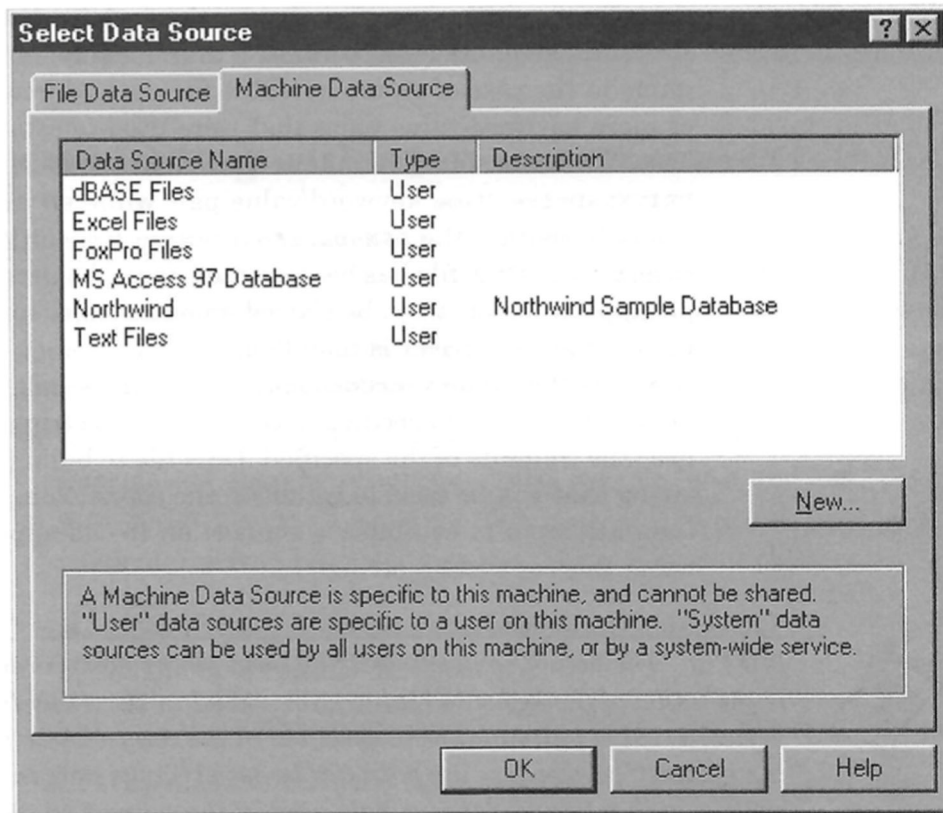


Figure 7-1 The ODBC 3.0 connection information dialog used by SQLDriverConnect() to obtain connection information from a user.

Id., p. 8. *See also id.*, p. 9 (SQLConnect function API).

128. Vendors have taken advantage of the ability to access multiple databases via ODBC. For example, Oracle describes that “[w]hen working with

multiple datasources” one can “[r]eferenc[e] tables from more than one datasource” with a special syntax. “Many datasources allow you to access tables that are located in other datasources if you specify the database, owner, and table (for example, database.owner.tablename).” Ex. 1031 (“Oracle Building Apps Guide”).

129. Finally, a POSITA would have known to issue one or more SQL SELECT queries to one or more different databases, using ODBC APIs or the “database.owner.tablename” syntax (*e.g.*, db1.owner1.table1, db2.owner2.table2, etc.). A POSITA would have also known that the results of querying multiple databases can be merged, joined, or combined in a variety of ways. *See* §VII.C above.

VIII. THE 720 PATENT

A. Overview

1. Background

130. The 720 Patent is in the context of searching computer databases. Ex. 1001 (720 Patent), Title, Abstract. Specifically, the 720 Patent's specification states that it is about a "search engine" which "provides an intuitive mechanisms [sic] for searching databases, allowing a user to access data in the database without having to know anything about the database structure." *Id.* The 720 Patent explains more about this "database structure:" "[a]ll databases require a consistent structure, termed a schema, to organize and manage the information." *Id.*, 1:38-40. For example, the "schema" may be a "collection of tables." *Id.* These tables define various columns, or sometimes referred to as database fields or, in the 720 Patent, "attributes." *Id.*, 1:25-37. The rows in a table store records with data filled in for one or many of the columns/database fields/attributes. For example, a database maintained by a business may have a Person table to store information about employees. The Person table would likely have fields / columns / attributes such as "employee ID," "first name," "last name," "hire date," and other fields descriptive of an employee as a person). A row in such a Person table may have values of "1," "John," "Doe," "5/1/2020," and so forth. As explained in §VII.C above, all of this was well understood by POSITAs at the time of the alleged invention of the 720 Patent.

131. To interact with the database, an administrator or user must send commands to the database in a syntax and format that the database understands. *See, e.g.,* Ex. 1001 (720 Patent), 1:49-65. One such language is Structured Query Language (SQL), which was described in more detail in §VII.C above. As explained there, SQL has a long and rich history dating back to well before the alleged invention of the 720 Patent.

132. The 720 Patent also discusses the use of tools used to interact with databases which are referred to in the 720 Patent as Database Management Systems (DBMS). Ex. 1001 (720 Patent), 1:49-65. These tools are generally designed to allow users and administrators alike a convenient way to manage or interact with the database, often at least offering the ability to submit to the database queries in a language understood by the database and to display any results of such queries. *Id.*

2. The Alleged Invention

133. The 720 Patent claims that there were problems with existing methods of searching databases. Ex. 1001 (720 Patent), 1:66-2:28. The 720 Patent states as a disadvantage of existing methods that “users still must have detailed knowledge of the schema to obtain needed information from a database since one can design many different schemas to represent the storage of a given collection of information.” *Id.*, 1:66-2:3. The other disadvantage identified in the 720 Patent’s specification is the

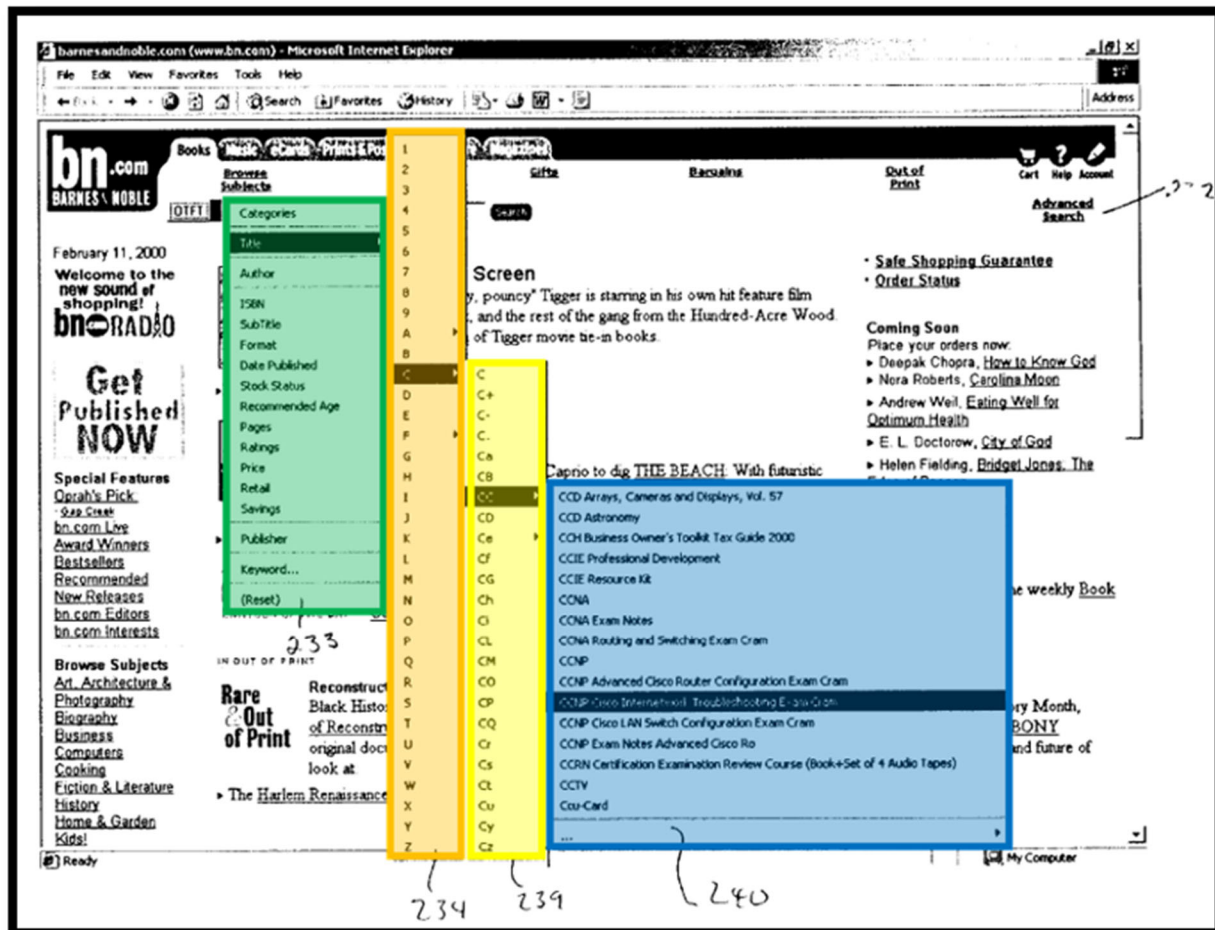
“tendency to return very large amounts of data” where the “time and expense involved in such a data review may be significant.” *Id.*, 2:23-28.

134. The 720 Patent sought to address these two issues using “an intuitive means for searching databases, allowing a user to access data in the database without having to know anything about the database structure.” Ex. 1001 (720 Patent), 2:30-35. The claimed methods involve determining the categories or “fields” that can be searched within a database (*i.e.*, the column headings) and providing the user with each of the potential search terms (*i.e.*, entries within the column) that could be used to search that field. *See, e.g., id.*, 5:42-57. The fields and search terms may be provided via a graphical user interface (“GUI”) with “pulldown” menus. *Id.*, 2:55-59.

135. The claims of the 720 Patent detail “truncating” the list of search terms when the list exceeds a specified number of entries (*i.e.*, when the list contains too many entries). *See* Ex. 1001 (720 Patent), Claims 1 & 3 and 5:63-67. The truncated data is shorter, but it still “represents each of the entries in the selected database field.” *Id.*, Claim 1; *see also id.*, 2:25-28.

136. Figure 13, for example, discloses how a database of books can be searched using the claimed methods. The categories or “fields” available for searching (in green) include book Categories, Title, Author, ISBN, Format, Recommended Age, etc. The user selects the field “title” and, because the number

of titles is voluminous, the list of titles is truncated into a representative list (orange). That representative list represents each title by the first character in the title (A, B, C, etc.). The user may continue to navigate using this truncated list (yellow and blue lists).



Ex. 1001 (720 Patent), Figure 13 (annotated).

137. The challenged dependent claims require additional steps that were well-known in the prior art.

B. Prosecution History of the 720 Patent

138. The application that resulted in the 720 Patent underwent several amendments during prosecution. The application met one final rejection and two non-final rejections before being allowed in January 2004. *See* Ex. 1002 (720 Patent FH), pp. 120-135 (April 2002 Rejection), pp. 159-173 (November 2002 Rejection), pp. 198-213 (May 2003 Rejection), p. 236 (Notice of Allowability). The applicant twice-amended the claims, once in supplemental response to the final rejection (April 2003) and once again in response to the second non-final rejection (October 2003). *See id.*, pp. 190-194 (April 2003 Amendment), pp. 220-228 (October 2003 Amendment). The first amendments were to amend Claim 1 while the second set of amendments were slightly more expansive and included amending Claims 1, 3, and 31 and cancelling one claim. *See id.*

139. Importantly, I note that none of the prior art that I rely upon in my declaration below was cited or considered by the examiner during prosecution. *See* Ex. 1002 (720 Patent FH), pp. 120-135, 159-173, 198-213; *see also* Ex. 1001 (720 Patent), Cover Page, Section 56 (“References Cited”).

140. Furthermore, the claims in the application were only allowed pursuant to the applicant’s amendment in October 2003 that added language that the claimed methods are “implemented on a computer,” and that the “truncation” in Claim 1 and “reducing” in Claim 3 “reduces characters in one or more entries in the selected

[database/data] field and the [truncated/size-reduced] [data/date] represents each of the entries in the selected [database] field.” *See* Ex. 1002 (720 Patent FH), pp. 220-228 (October 2003 Amendment). However, these limitations that were added to the claims to earn the Examiner’s allowance are disclosed, or at least rendered obvious, by the uncited prior art that I discuss in this declaration.

141. Specifically, I explain within the following sections how the prior art discloses “implemented on a computer:” §§XII.A.1 (Claim [1.PRE], Ground 1), XII.C.1 (Claim [3.PRE], Ground 1), XIII.A.1 (Claim [1.PRE], Ground 2), and XIII.C.1 below (Claim [3.PRE], Ground 2).

142. Furthermore, I explain within the following sections how the prior art discloses the limitations added in the final amendment related to “truncation” in Claim 1 and “reducing” in Claim 3: *see* §§XII.A.6 (Claim [1.E], Ground 1), XII.C.5 (Claim [3.D], Ground 1), XIII.A.6 (Claim [1.E], Ground 2), and XIII.C.5 below (Claim [3.D], Ground 2).

C. The 720 Patent’s Claims

143. For reference, claims 1-7, 15-23, and 25-30 of the 720 Patent are reproduced below:

1. A method implemented on a computer for searching databases, comprising:

determining a database schema for a database, wherein the database includes database fields;

providing a list of the database fields, wherein the list includes a descriptor indicating a data category;

receiving a search selection for a database field on the provided list of the database fields;

determining a quantity of entries in the selected database field;

if the quantity exceed a specified amount, truncating data, and displaying the truncated data wherein the truncating reduces characters in one or more entries in the selected database field and the truncated data represents each of the entries in the selected database field; and

if the quantity does not exceed the specified amount, displaying contents of the database field.

2. The method of claim 1, further comprising providing a key word search.

3. A method implemented on a computer for searching a database, comprising:

generating a list of data fields;

receiving a first data field selection from the list of data fields;

determining a first quantity indicative of a number of entries of the selected data field;

if the first quantity exceeds a specified limit, reducing a size of data to be displayed from the selected data field wherein the reducing reduces characters in one or more entries in the selected data field and the size-reduced data represents each of the entries in the selected field; and

displaying data from the selected data field.

4. The method of claim 3, wherein the specified limit is fixed.

5. The method of claim 3, wherein the specified limit is variable.

6. The method of claim 3, wherein the data are displayed on a terminal, and wherein the specified limit is determined dynamically, based on a characteristic of the terminal.

7. The method of claim 3, wherein the specified limit is a user-determined limit.

15. The method of claim 3, further comprising:

receiving a first constraint, wherein the first constraint is related to a data element in a data field; and

receiving one or more subsequent constraints, wherein search results are generated based on a combination of the first and the one or more subsequent constraints.

16. The method [of] claim 3, wherein each display of data comprises a search cycle, and further comprising receiving a refresh command, the refresh command resetting all search cycles.

17. The method [of] claim 3, further comprising displaying the list of data fields and a representation of the selected data field, wherein the representation of the selected data field comprises one or more elements from the selected data field.

18. The method of claim 3, wherein the method comprises executing a plurality of search cycles, each search cycle of the plurality of search cycles providing a search result list comprising selected data elements from the data base.

19. The method of claim 18, further comprising resetting one or more search cycles of the plurality of search cycles.

20. The method of claim 3 further comprising:

determining a schema of the database;

comparing the determined schema to a capability of a terminal for displaying the data; and

formatting a search request from the terminal to correspond to the determined schema.

21. The method of claim 3, wherein the method for searching the database comprises searching a first database and a second database, and further comprising:

receiving a first search result from the search of the first database;

receiving a second search result from the search of the second database; and

combining the first and the second search results.

22. The method of claim 21, wherein the combining step comprises a Boolean AND operation.

23. The method of claim 21, wherein the combining step comprises a Boolean OR operation.

25. The method of claim 21, wherein the first search result is represented by a first icon and the second search result is represented by a second icon, and wherein the combining step comprises overlaying the first and the second icons.

26. The method of claim 3, wherein the displayed data includes one or more data records, each of the one or more data records represented by one or more of text title and an icon, and further comprising overlaying a text title or an icon on a symbol representing an online purchase, wherein the step of overlaying initiates an online purchase of an item represented by the text title or icon.

27. The method of claim 3, further comprising displaying the data on a terminal, the terminal including one of a handheld device, a cellular phone, a geosynchronous positioning satellite (GPS) device, a wrist-worn device, an interactive phone device, a household appliance, a television, a television set top box, a handheld computer, a main frame computer and a personal computer.

28. The method of claim 3, wherein the database is coupled to an Internet website, the method further comprising accessing the Internet website.

29. The method of claim 3, wherein the database is coupled to a node in a telecommunications network, the method further comprising accessing the node in the telecommunications network.

30. The method of claim 3, wherein the database is coupled to a terminal for displaying the data.

IX. CLAIM CONSTRUCTION

144. I am not a legal expert or an attorney and offer no opinions on the law. I understand that claim construction is a matter of law. As discussed in §IV.A above, I have been informed by counsel of the legal standards that apply to claim construction in an *inter partes* review and I have applied them in forming my opinions.

145. In my opinion, for purposes of this proceeding, the terms of the 720 Patent's claims do not require further construction and should be afforded their plain and ordinary meaning. Should Patent Owner advance any different constructions, I reserve the right to address those in a reply declaration.

X. UNPATENTABILITY ANALYSIS

146. I have reviewed and analyzed the prior art references and materials listed in §XI below.

147. For the reasons I explain in greater detail below, it is my opinion that before the earliest possible filing date of the 720 Patent in 2000, a POSITA would have considered claims 1-7, 15-23, 25-30 of the 720 Patent to be unpatentable as obvious over the prior art discussed in this declaration. Below is the summary of the grounds detailed in this declaration:

Ground	Claim(s)	Unpatentability Ground
1	1-7, 15-20, 27-30	Obvious over Cochran in view of Zellweger and the knowledge of a POSITA
2	1-5, 7, 15-19, 27-30	Obvious over HyperSQL in view of Zellweger and the knowledge of a POSITA
3	21-23	Obvious over HyperSQL in view of Zellweger, Microsoft Access, and the knowledge of a POSITA
4	25	Obvious over HyperSQL in view of Zellweger, Microsoft Access, Janes, and the knowledge of a POSITA
5	26	Obvious over Cochran in view of Zellweger, Janes, Spiegel, and the knowledge of a POSITA

XI. THE PRIOR ART

A. Overview of the Prior Art

148. Next, I provide an overview of the main prior art references I discuss throughout this declaration.

1. United States Patent No. 5,995,979 to Cochran, Filed February 10, 1998 (“Cochran”)

149. Cochran is a patent that issued from an application filed on February 10, 1998. I understand that this means that Cochran is prior art under at least §102(e).

150. Cochran discloses a method for accessing a database using a graphical user interface (GUI) which displays a plurality of fields and available search terms. Ex. 1005 (Cochran), Abstract, 1:15-19. The fields shown in Cochran’s GUI correspond to the various fields (“also referred to as indexes or list identifiers”) in the database containing the data or information. *Id.*, 1:21-30. The search terms are the information that may be contained within the corresponding database field. *Id.*, 1:30-39. Cochran discloses that access is performed “via a World Wide Web page by a Web Browser.” *Id.*, 6:31-39. Thus, in such an embodiment, the GUI would be displayed on and interacted with using a Web Browser. The “search terms” are displayed and selected in a type of interactive input element that Cochran refers to as “pick boxes.” *Id.*, 10:13-18. Cochran explains that the term “pick box” “is used [] to refer to the actual displayed box used to present the search terms” and notes that “[p]ick boxes are standard elements of modern windowing technology.” *Id.*

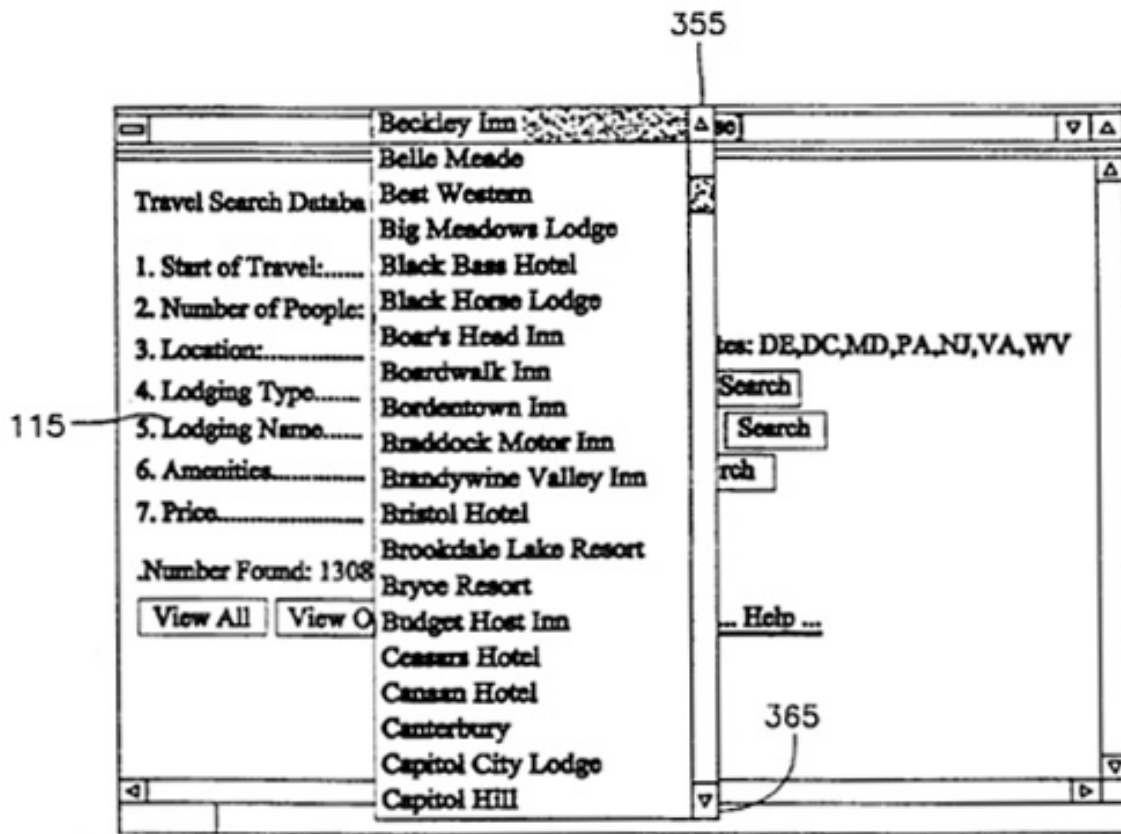
151. For example, Figure 4b below show seven data fields. In Figure 4e, the user expanded the Lodging Name “pick box” from Figure 4b, which lists the available entries for that field taking into consideration selections in other pick boxes. Ex. 1005 (Cochran), 12:26-13:67; *see also id.*, 4:29-50, 8:54-9:22.

The screenshot displays a window titled "Travel Search Database" with a subtitle "Travel Search Database - Selection Screen". It contains seven numbered fields, each with a dropdown menu and a "Search" button:

- 1. Start of Travel: January-March
- 2. Number of People: Disregard Occupancy 314
- 3. Location: Within a Region, Atlantic States: DE,DC,MD,PA,NJ,VA,WV
- 4. Lodging Type: Hotel/Motel 214
- 5. Lodging Name: Airport Inn 215
- 6. Amenities: Outdoor Swimming 315
- 7. Price: Economy 216

Below these fields, it shows "...Number Found: 1308" and a row of buttons: "View All" (121), "View One" (122), "Reset 4-7" (123), "Start Over" (124), and "... Help ...".

Id., Figure 4b.



Id., Figure 4e.

2. Mark Newsome et al., *HyperSQL: Web-based Query Interfaces for Biological Databases*, IEEE Computer Society, Proceedings of The Thirtieth Annual Hawaii International Conference on System Sciences (1997) (“HyperSQL”)

152. *HyperSQL: Web-based Query Interfaces for Biological Databases* (Ex. 1006, “HyperSQL”) is a conference paper authored by Mark Newsome et al. HyperSQL is included in the compilation of the Proceedings of the Thirtieth Hawaii international conference on system sciences. A declaration by Rachel J. Watters, (Ex. 1014), shows that HyperSQL “was received by the [Kurt F. Wendt Library, University of Wisconsin-Madison Libraries] on or before April 22, 1998, catalogued

and available to library patrons within a few days or at most 2 to 3 weeks after April 22, 1998.” I understand that this means HyperSQL is prior art under at least §102(b). I was at least a POSITA at the time this conference was held. I was well aware of it: I read and cited some of this conference’s papers. The fact that this conference ran thirty (30) times by 1998 alone indicates its popularity and longevity.

153. HyperSQL describes a browser-based interface that users use to search complex databases. Ex. 1006 (HyperSQL), 329 (“HyperSQL is an interoperability layer that enables database administrators to rapidly construct browser-based query interfaces to remote Sybase databases”). HyperSQL further describes using a website with pages called “query forms,” where the “query form” pages allow users to specify search criteria for database data fields and to conduct a search using those user-selected search criteria. *Id.*, 329-333, Figures 2-5. HyperSQL discloses that search terms can be selected from inputs called “querylists,” where the “querylists” are scrollable lists of “‘allowed’ values” from the database. *Id.*, 333, Figure 2.

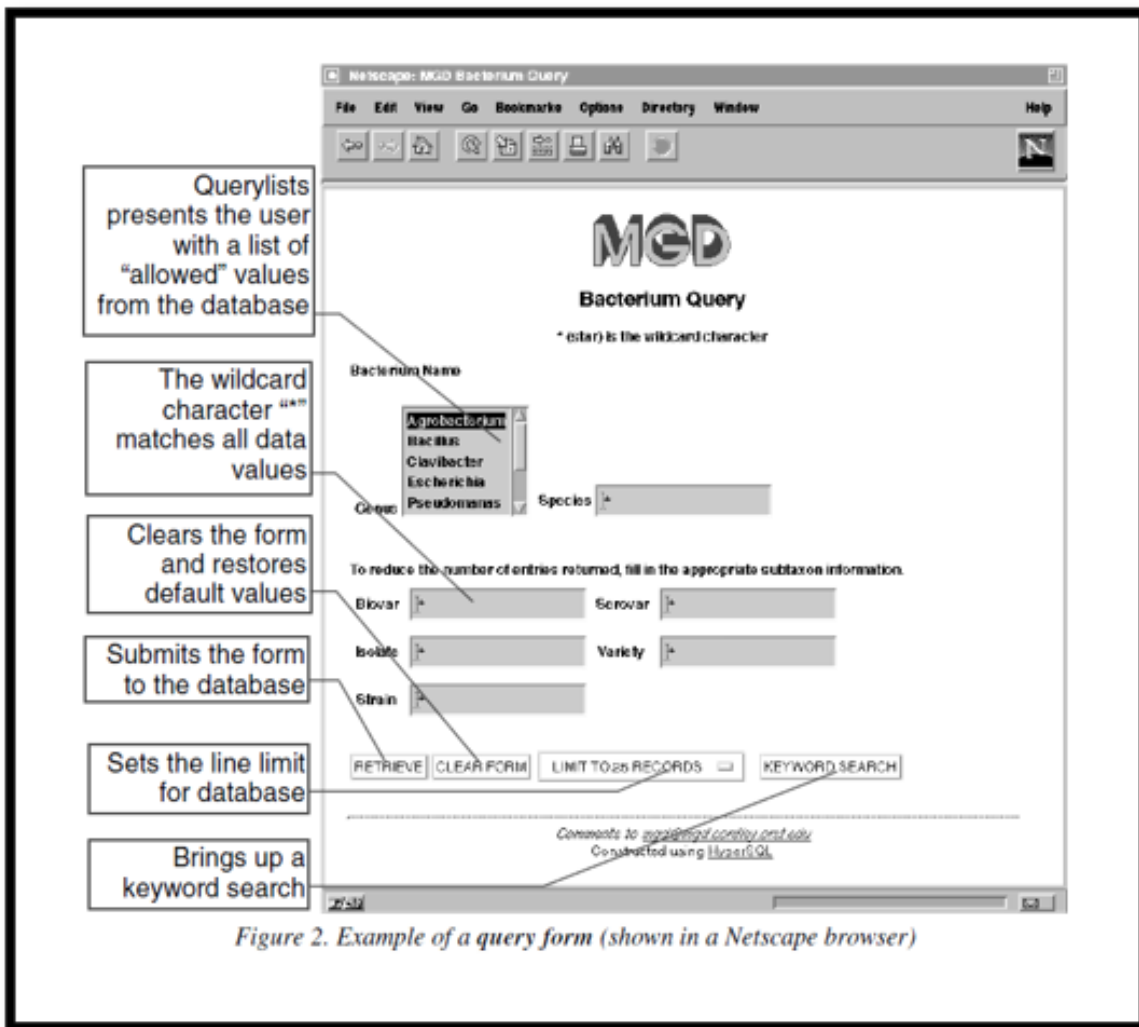


Figure 2. Example of a query form (shown in a Netscape browser)

Id., Figure 2.

154. Figure 2 shows a query form with eight data fields including “Genus” and “Species.” HyperSQL further explains that it supports a feature called “query refinement” which relies on the use of multiple querylists. Specifically query refinement refers to a capability “allowing the user to fill in fields incrementally by selecting from ‘querylists’ (lists of allowed values).” Ex. 1006 (HyperSQL), 331.

3. United States Patent No. 6,433,797 to Zellweger, Filed July 15, 1999, Claiming Priority to Provisional Application No. 60/095,811, Filed August 4, 1998 (“Zellweger”)

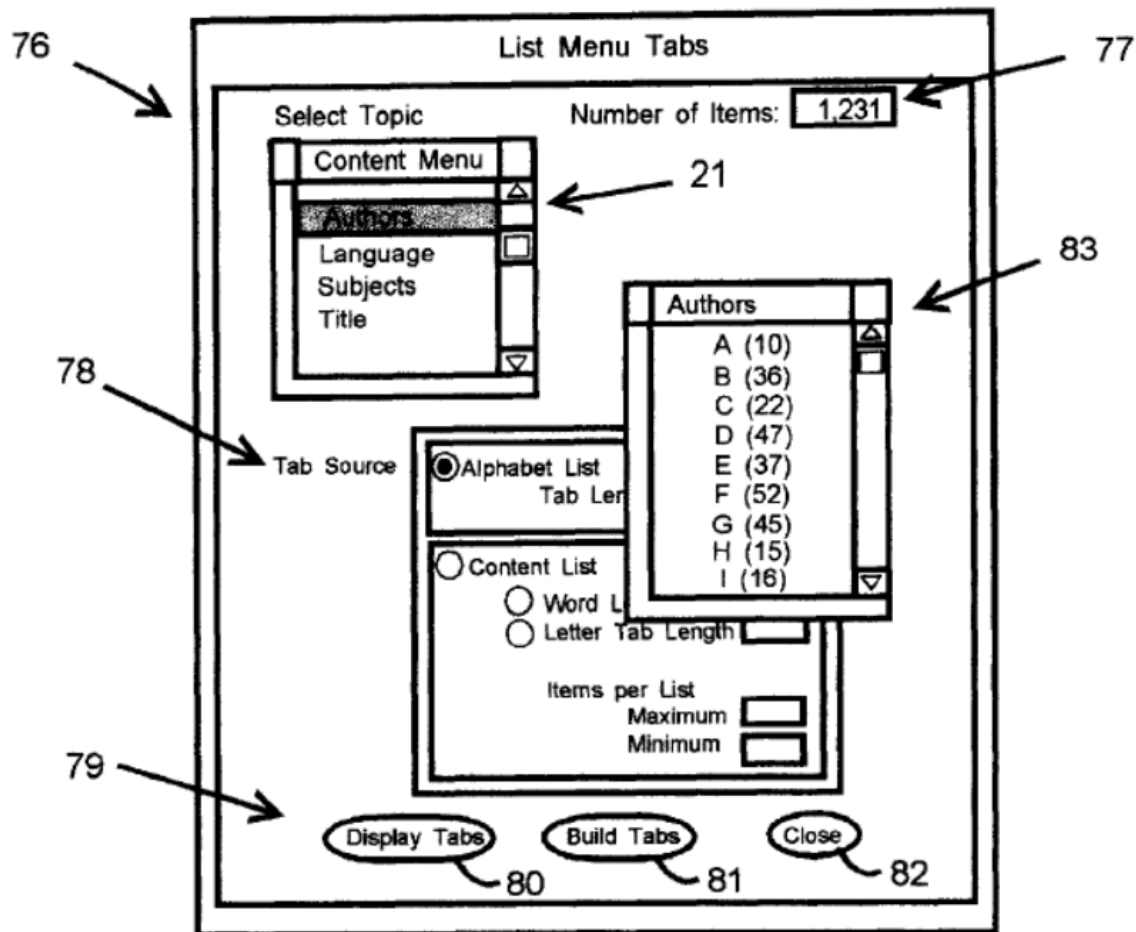
155. Zellweger is a patent that issued from an application filed on July 15, 1999. I understand that this means Zellweger is prior art under §102(e).

156. Zellweger generally discloses a method of generating menus that users of a system use to locate information, such as information stored in databases. Ex. 1007 (Zellweger), Abstract, 3:66-4:6. Specifically, Zellweger discloses a method that reduces the size of lengthy lists that are displayed in pulldown (also known as dropdown) menus (like the pick boxes in Cochran and querylists in HyperSQL). *Id.*, Abstract, 2:30-43.

157. The method disclosed by Zellweger involves shortening the voluminous lists into “Alphabet Lists” or “Content Lists” that are navigable by the menu developer (also referred to simply and interchangeably as “developers” in Zellweger) and end-users. Ex. 1007 (Zellweger), 4:21-53. In Zellweger, “menu developers” are users that use Zellweger and can also customize and regenerate the menus disclosed in Zellweger. *See e.g., id.*, Abstract, 2:30-43 (“With the present invention the menu developer has full control over how the original list is divided into intervals and how the tab menu list portray these divisions.”), 2:57-59 (“FIGS. 6a and 6b depict the interactive **user** interface of the configuration window in the present invention employed by a developer to generate a tab list menu.”) (emphasis

added), 4:7-52 (“So when an end-user selects the ‘Abdul-Hak ... Berg’ entry, the content menu generates a new list menu that starts with ‘Abdul-Hak’ and ends with ‘Berg’.”).

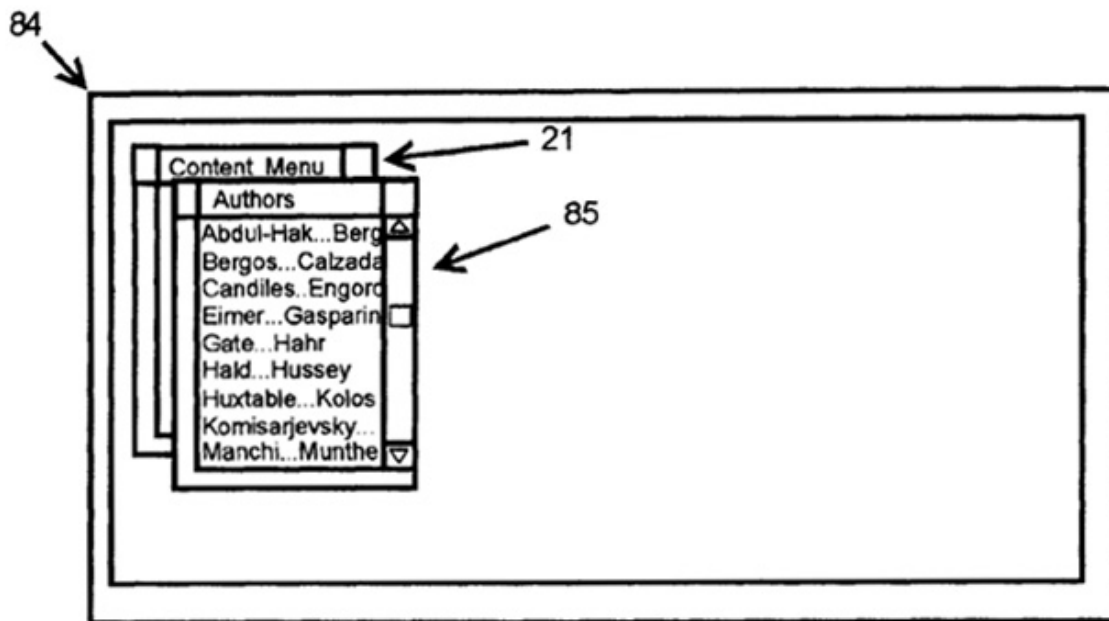
158. Alphabet Lists reduce the volume of the original list by truncating the items on the list down to a defined number of leading letters and organizing them into common groupings, shown in Figure 6b below. Ex. 1007 (Zellweger), 4:21-53.



Id., Figure 6b.

159. For example, Figure 6b shows the case where there are 1,231 entries for “Authors,” those entries are each truncated to one letter, and then the entries are categorized together in tab lists along with Authors which were truncated to the same characters (*i.e.*, A, B, C, etc.). Ex. 1007 (Zellweger), Figure 6b, 4:17-26, 4:36-44. Figure 6b shows that there are ten authors starting with “A” within the “A” tab list, 36 authors starting with “B” within the “B” tab list, and so forth. *Id.*

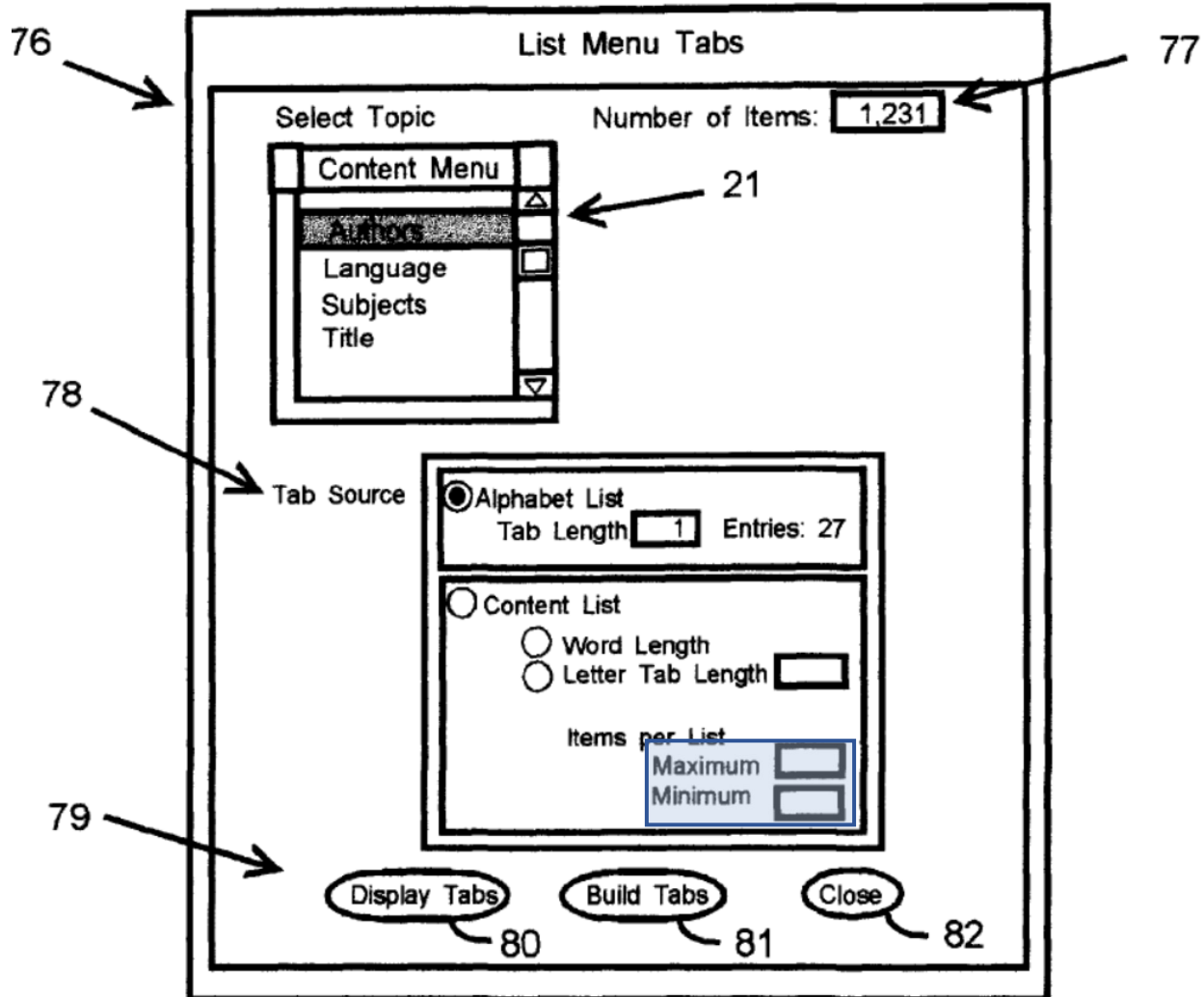
160. Content Lists organize the entries into intervals by truncating the entries down to either a full word or a specified length of letters. Ex. 1007 (Zellweger), Figure 7, 4:26-31, 4:45-53.



Id., Figure 7.

161. For example, Figure 7 above shows the case where the Authors seen (in Figure 6b above) related to the Alphabet List are truncated to one word length in a Content List, such that the tab entries of the Content List show ranges or intervals using the author's last name as the first and last elements in each range/interval in the list 85. Ex. 1007 (Zellweger), 4:45-53.

162. For Content Lists, users “can specify the maximum and minimum number of entries in each tab list menu.” Ex. 1007 (Zellweger), 4:27-31. The input boxes for the maximum and minimum setting can be seen in Figure 6a:



Id., Figure 6a (annotated).

4. Using Microsoft Access 97, Special Edition, by Roger Jennings (1997) ("Microsoft Access")

163. *Using Microsoft Access 97, Special Edition*, by Roger Jennings is a textbook published and sold by Que Corporation in 1997 that contains detailed information regarding the use of software published by Microsoft called Access 97 (Ex. 1009, "Microsoft Access"). A certificate from the Library of Congress shows

that a copy of Microsoft Access (Library of Congress Catalog No. 96-71443) was placed into, made available, and cataloged within their collections with a call number of QA 76.9.D3 J474 1997, Copy 1, marked with a stamp dated to January 10, 1997. Ex. 1015. Microsoft Access was cataloged under “Microsoft Access” and “Database management” subjects and assigned Dewey class no. 005.75/65. Ex. 1010. I understand that this means Microsoft Access is prior art under at least §102(b).

164. Microsoft Access was a popular, multifaceted software by Microsoft. Several versions of Microsoft Access were released in the 1990s alone (*e.g.*, 1.x, 2.x, 95, and 97). During that time, Microsoft Access gained in popularity for making it much easier to create, manage, and use databases. *See* discussion of its history and evolution at https://en.wikipedia.org/wiki/Microsoft_Access (last accessed November 25, 2023).

165. The first version of Microsoft Access was so popular that Microsoft “received orders for more than 750,000 copies of Access 1.0 between its release data [sic] in mid-November 1992, and January 31, 1993.” Ex. 1009 (Microsoft Access), 1. That momentum continued through the release of subsequent versions of Access and “[i]n the fall of 1996, Microsoft announced that the firm had sold a cumulative total of 10 million copies of Access.” *Id.*, 2. It is commonly acknowledged that Microsoft Access 97 itself “was a very popular version of the Access product.”

Microsoft Access 97, WinWorldPC.com,

[https://winworldpc.com/product/microsoft-access/97#:~:text=Microsoft%20Access%2097%20\(8.0\)%20was,number%20of%20desktop%20database%20applications](https://winworldpc.com/product/microsoft-access/97#:~:text=Microsoft%20Access%2097%20(8.0)%20was,number%20of%20desktop%20database%20applications)” (last accessed November 25, 2023).

166. In one aspect, Microsoft Access 97 functioned as a relational database management tool that users could leverage to quickly make desktop and client/server database applications. Ex. 1009 (Microsoft Access), 1. Naturally, this manual contains a detailed discussion of many aspects of how to use Microsoft Access 97. It also discloses many well-known operations related to the usage of databases. *Id.*, 291. For example, it discloses the use of some of the most basic Boolean operations it supported, such as the logical AND and OR operation, “to combine the results of two or more comparison expressions into a single result.” *Id.*

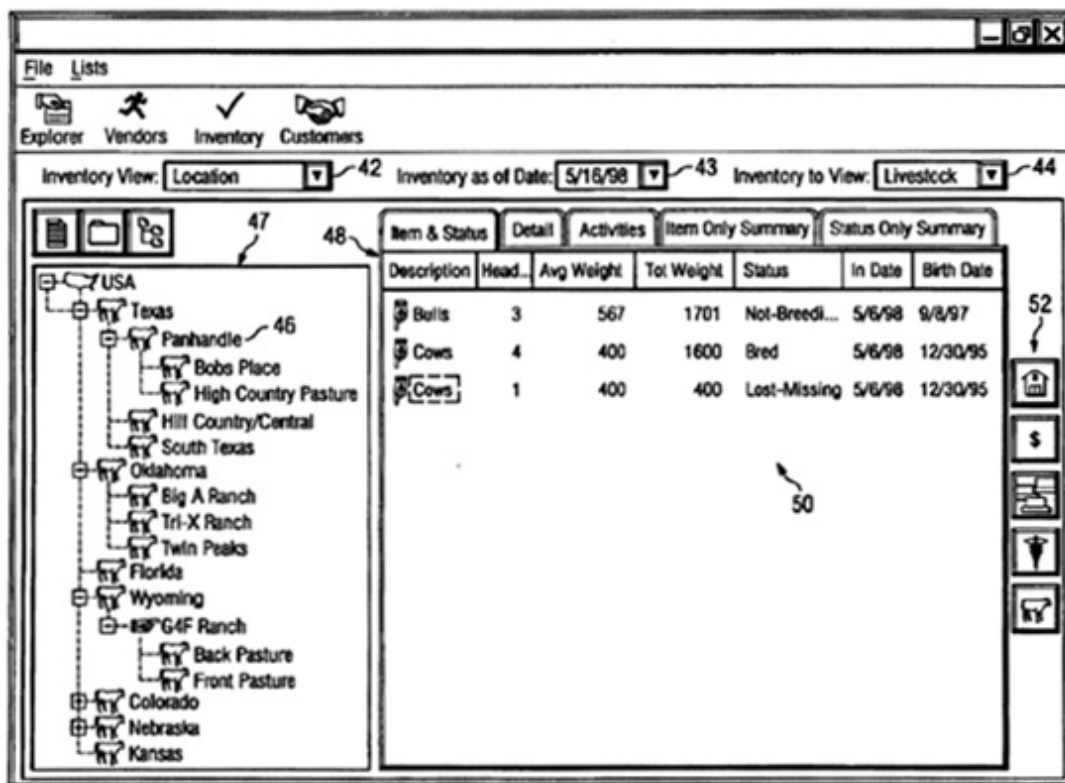
5. United States Patent No. 6,642,946 to Janes, Filed Aug. 12, 1999 (“Janes”)

167. Janes is a patent that issued from an application filed on August 12, 1999. I understand this means that Janes is prior art under §102(e).

168. Janes discloses using a graphical user interface (“GUI”) to manage data in a database—specifically in Janes’s exemplary case, it is used to manage a database containing livestock and material inventory for ranches and farms. Ex. 1012 (Janes), Title, Abstract, 2:10-62. Janes teaches a GUI that uses a hierarchical menu system with “nodes” in one part of the GUI, where the data is grouped together by node.

Id., 2:10-62, 4:30-5:60. A user may click on a node, upon which the GUI retrieves and displays the data associated with the node as icons and text in another part of the GUI. *Id.*

169. Janes also teaches that entities (*e.g.*, icons and text) in the GUI may be dragged and dropped onto other entities (*e.g.*, icons and text) to complete actions. Ex. 1012 (Janes), 2:48-62, 5:51-60, 11:22-12:44. While the drag-and-drop functionality may be used to initiate any actions, Janes specifically discloses using it to conveniently perform business transactions such sales of livestock. *Id.*, Figure 2A, 2:10-19, 12:34-41.



Id., Figure 2A.

170. In Figure 2A above, the “nodes” in the hierarchical menu represent certain locations or geographical regions that logically group together livestock and inventory data. For example, in Figure 2A, the G4F Ranch in Wyoming has been selected, so the livestock and inventory shown consists of any livestock and inventory within the G4F Ranch, which is shown to be certain numbers of bulls and cows.

6. United States Patent No. 6,629,079 to Spiegel, Filed June 25, 1998 (“Spiegel”)

171. Spiegel is a patent that issued from an application filed on June 25, 1998. I understand this means Spiegel is prior art under §102(e).

172. Spiegel is an Amazon patent that discloses an e-commerce system utilizing a GUI. Ex. 1013 (Spiegel), Abstract, 2:61-3:40, Figure 1. Specifically, the GUI relies on shopping carts that are visually represented on the GUI, each shopping cart may have associated billing and shipping information. *Id.*, Abstract, 2:61-3:40, 4:19-28. The GUI includes shopping cart icons that a user may use to graphically add items for purchase via a dragging-and-dropping operation—a user would, graphically, click on and drag items for purchase over shopping cart icons displayed on Spiegel’s GUI and ‘drop’ them into the shopping cart by releasing the button used to ‘click’ on the item for purchase. *Id.*, 5:39-6:16, 6:48-58, 10:31-54, Figures 1, 3; *see also* §VII.E above.

XII. GROUND 1: COCHRAN IN VIEW OF ZELLWEGER, AND THE KNOWLEDGE OF A POSITA RENDERS OBVIOUS CLAIMS 1-7, 15-20, AND 27-30

173. It is my opinion that a POSITA would have considered Cochran and Zellweger to be within the scope of the prior art and highly relevant to the patentability of the 720 Patent's claims.

174. It is my opinion that Cochran in view of Zellweger, especially when considering the knowledge of a POSITA at the time, renders obvious Claims 1-7, 15-20, and 27-30. The basis for my opinion is set forth in more detail below.

175. A POSITA would have been motivated to combine the teachings of Cochran and Zellweger, which are analogous art, and would have had a reasonable expectation of success when doing so as detailed in §XVII.A below.

A. Claim 1

1. [1.PRE] “A method implemented on a computer for searching databases, comprising:”

176. Cochran discloses “[a] method for managing the presentation of search terms on a computer terminal ... so that large computer databases can be searched efficiently.” Ex. 1005 (Cochran), Abstract; *see also id.*, Figures 7b, 3a, 4a, 4b, 4d, 1:30-37, 4:10-12. Cochran discloses in Figure 1 “the general configuration of a computer communications network used” according to the invention in Cochran. *Id.*, 4:65-67. Cochran’s Figure 1 shows a “server computer 710 ... for implementing the software program of” Cochran’s invention. *Id.*, 5:65-6:10, Figure 1.

177. In sum, Cochran clearly discloses “a method implemented on a computer” (*i.e.*, “on a computer terminal”) “for searching databases” (“so that large computer databases can be searched efficiently”).

2. [1.A] “determining a database schema for a database, wherein the database includes database fields;”

178. Cochran generally discloses “a method for accessing records from a computer database.” Ex. 1005 (Cochran), 1:15-16. The “[i]nformation is stored in the database as a collection of records, each record having multiple fields (also referred to as indexes or list identifiers) containing the data or information.” *Id.*, 1:23-29. These disclosures show that “the database” of Cochran “includes database fields.”

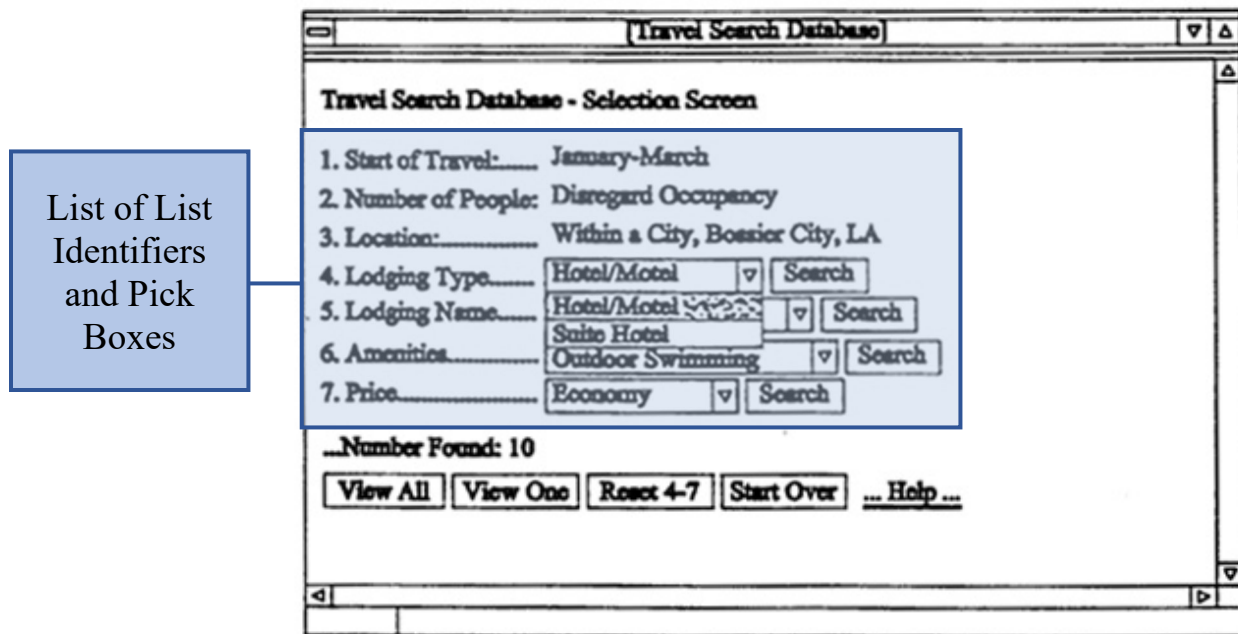
179. Cochran discloses that it “display[s] a list of all available choices” to “inform the user of available options before any search is initiated.” Doing this alleviates the issue where it is “difficult for the user to know how [the] database was organized without the visual presentation” of the data available in the database that Cochran presents to the user. Ex. 1005 (Cochran), 11:38-48. For Cochran to display such a list of *all* available choices, Cochran must know how to query the database, which involves knowing how the choices are organized (*i.e.*, what the database field names are called and how they are organized relative to each other).

180. A POSITA would have understood that SQL databases are organized collections of information and use a schema to describe the database's structure, without which it would be highly inefficient to manage the database. *See also* §VII.C above.

181. In sum, a POSITA would have thus understood from these disclosures that Cochran must “*determine a database schema for a database*”; otherwise, Cochran would not know how to tell the database to inform it of “all available choices” for the relevant database fields against which a user may need to apply search terms.

3. [1.B] “providing a list of the database fields, wherein the list includes a descriptor indicating a data category;”

182. Cochran displays for the user a list of “list identifiers” and “pick boxes” on a graphical user interface (“GUI”). Cochran states that the “list identifiers” “identify[] the various categories of information.” Ex. 1005 (Cochran), 7:19-28. This is shown in the annotated version of Figure 7b below.



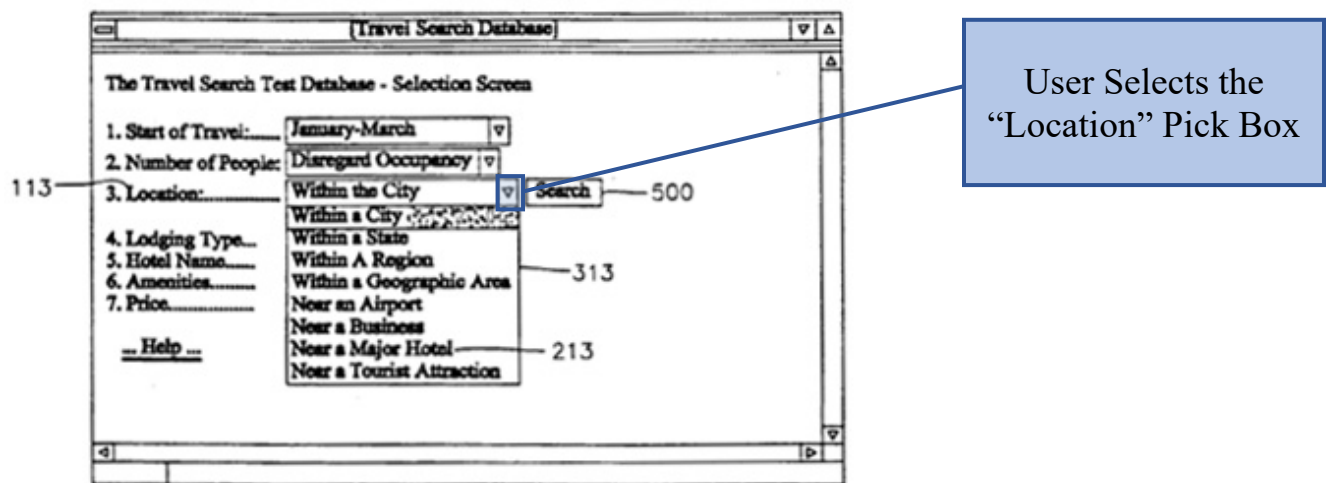
Id., Figure 7b (annotated).

183. For example, Figure 7b above lists fields/descriptors such as those for “Location,” “Lodging Type,” “Lodging Name,” etc. Ex. 1005 (Cochran), Figure 7b; *see also id.*, 3a-4b, 4d-9d, Abstract, 7:13-47, 11:22-28.

184. In sum, these disclosures from Cochran show that Cochran is “providing a list of the database fields” when it displays the list of “list identifiers” and “pick boxes” on the GUI and that “the list includes a descriptor indicating a data category” because each database field has an associated label that indicates that category to which the data belongs, such as indicators like “Location,” “Lodging Type,” “Lodging Name,” “Price,” and so on.

4. [1.C] “receiving a search selection for a database field on the provided list of the database fields;”

185. Cochran discloses that when a user clicks on one of the listed pick boxes, the selected pick box expands to display all the relevant “search terms” related to the associated database field. Ex. 1005 (Cochran), 9:60-10:4. Cochran receives a search selection for the selected pick box because otherwise Cochran would not have been able to acquire the relevant search terms from the database and populate the pick box with all of the search terms for displaying in the pick box. *See id.*, 1:30-47, 7:9-19, 6:31-39, 8:45-47.



Id., Figure 3d (annotated); *see also id.*, Figures 3a-3c, 4a-4b, and 4d-9d.

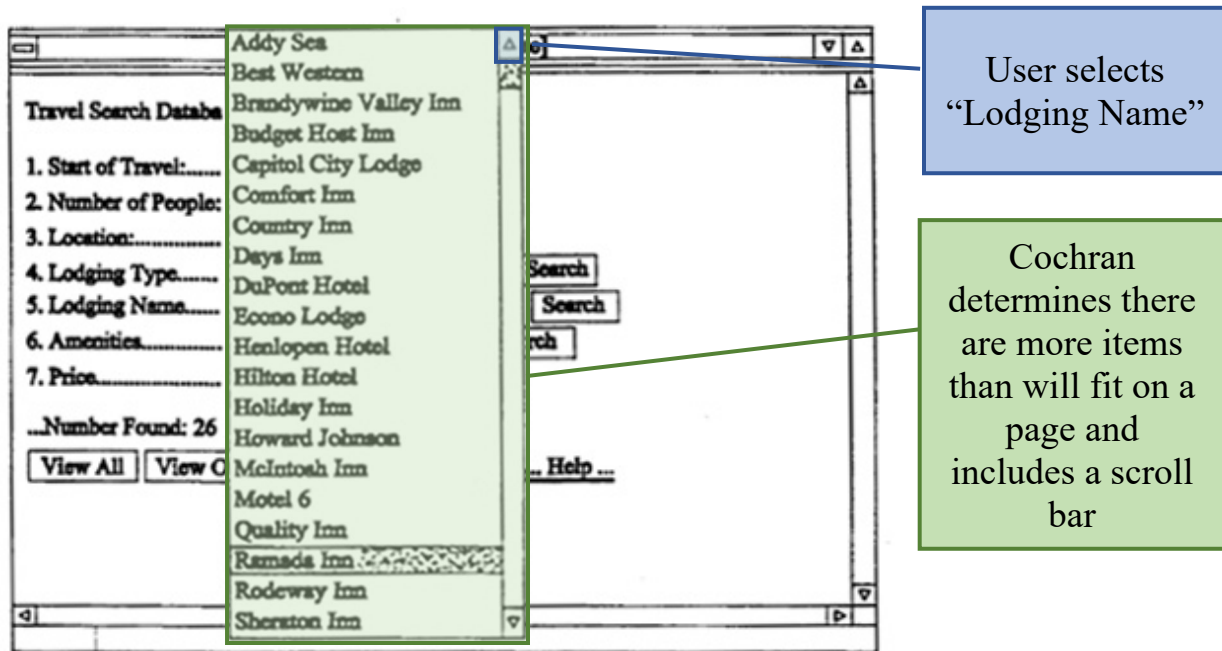
186. As seen in Figure 3d above the user clicked on the pick box for “Location” and expanded it to reveal all the search terms that are available under Location. The user similarly may expand the pick boxes for the other fields such as

Lodging Type, Hotel Name, Amenities, and Price, etc., that are shown in the embodiment of Figure 3d.

187. In sum, Cochran discloses “*receiving a search selection*” to obtain the search values displayed in the pick boxes and those selections are related to the “*database field on the provided list of the database fields*” because each pick box relates to an underlying database field and is on the list of similar pick boxes related to underlying database fields.

5. [1.D] “determining a quantity of entries in the selected database field;”

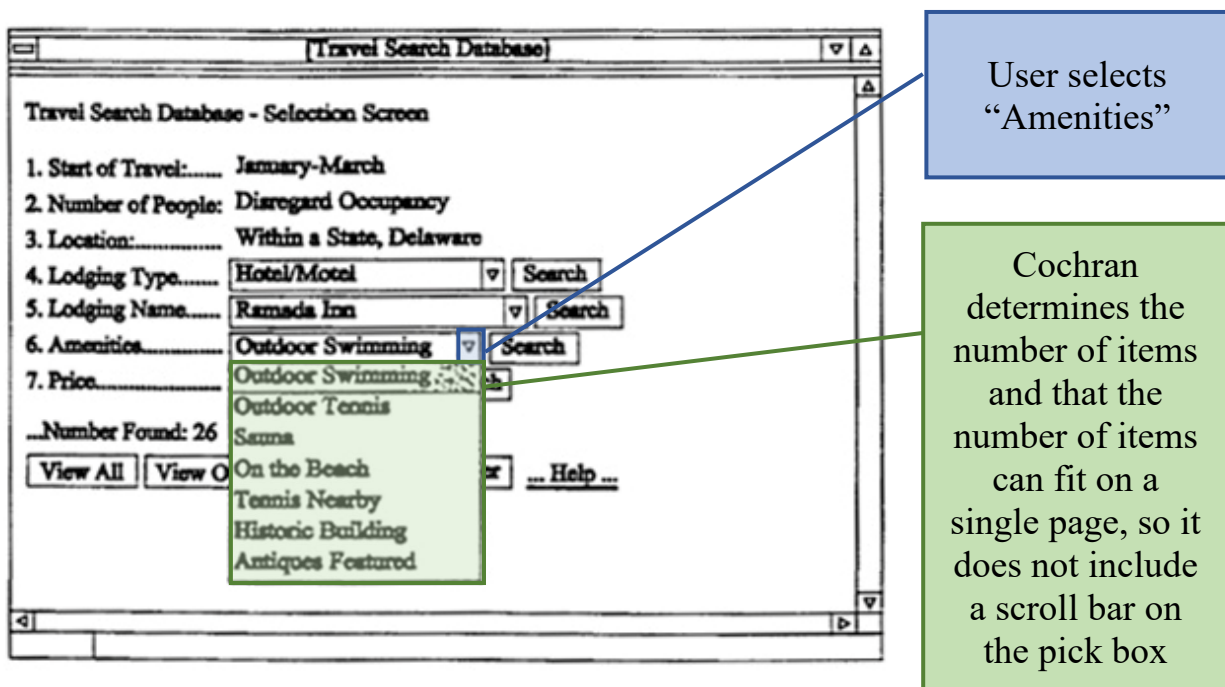
188. Cochran discloses that “[o]nly choices or search terms that are actually available are presented to the user.” Ex. 1005 (Cochran), 4:29-35. Cochran’s Figure 5d shows that if Cochran determines that the number of entries it needs to display in a pick box exceeds the number that can be displayed in the space provided on the screen then Cochran adds a scroll bar. *Id.*, 10:57-65 (“If the expanded pick-box area is too small to display all of the available search terms, then scrolling is permitted within the confines of the expanded pick box.”), Figure 5d.



Id., Figure 5d (annotated).

189. As seen in Figure 5d above, when the user selects the Lodging Name's pick box and clicks on the button to expand that pick box (in blue), Cochran determines that there are more search terms than can be displayed on one page; then, it includes a scroll bar in the pick box to allow the user to scroll to any hidden search terms (in green).

190. However, if all of the entries that need to be displayed in the pick box can be displayed on a single list in the space provided on the screen, Cochran decides to not display a scroll bar. *See* Ex. 1005 (Cochran), Figures 3a-9d.



See *id.*, Figure 5e (annotated).

191. As seen in Figure 5e above, the user selects the Amenities pick box and clicks on the button to expand that pick box (in blue); but Cochran now determines that the number of search terms to display in the expanded pick box *can* fit onto the screen on one page and so Cochran uses this determination to decide to not include a scroll bar on the pick box (in green).

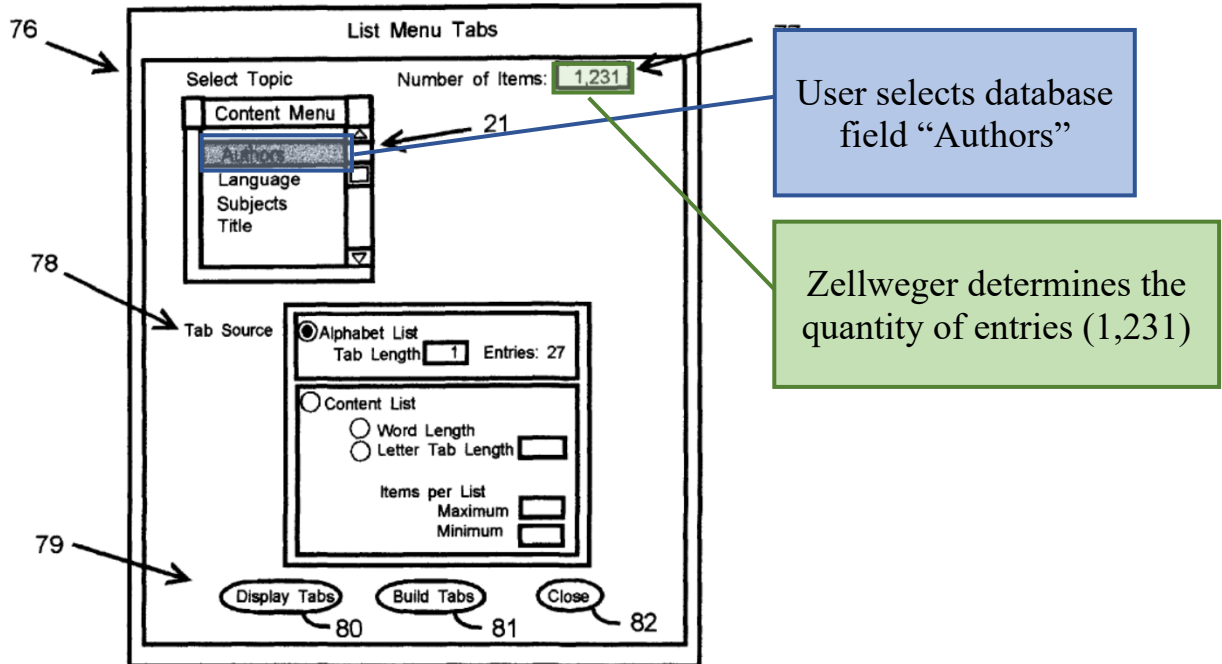
192. These disclosures make it clear (or at least obvious) that Cochran determines the quantity of entries in the database field. If Cochran did not determine the quantity of entries in the database field that are being displayed in the pick box Cochran would be unable to determine whether or not to include a scroll bar. Thus, Cochran determines the number of “search terms” (“*determining a quantity of*

entries”) to display in an expanded pick box (“*in the selected database field*”) when Cochran generates the display of the expanded pick box. *See also* §VII.E above.

193. Additionally, and in general, a POSITA would have understood that database queries may return multiple items, and computer software must count how many items are returned, for example to ensure that ample memory buffers and display space are allocated to hold the items. *See also* §§VII.B and VII.E above.

194. Zellweger similarly discloses determining a number of items (“*quantity of entries*”) associated with a “*selected database field.*” Zellweger discloses this in two ways.

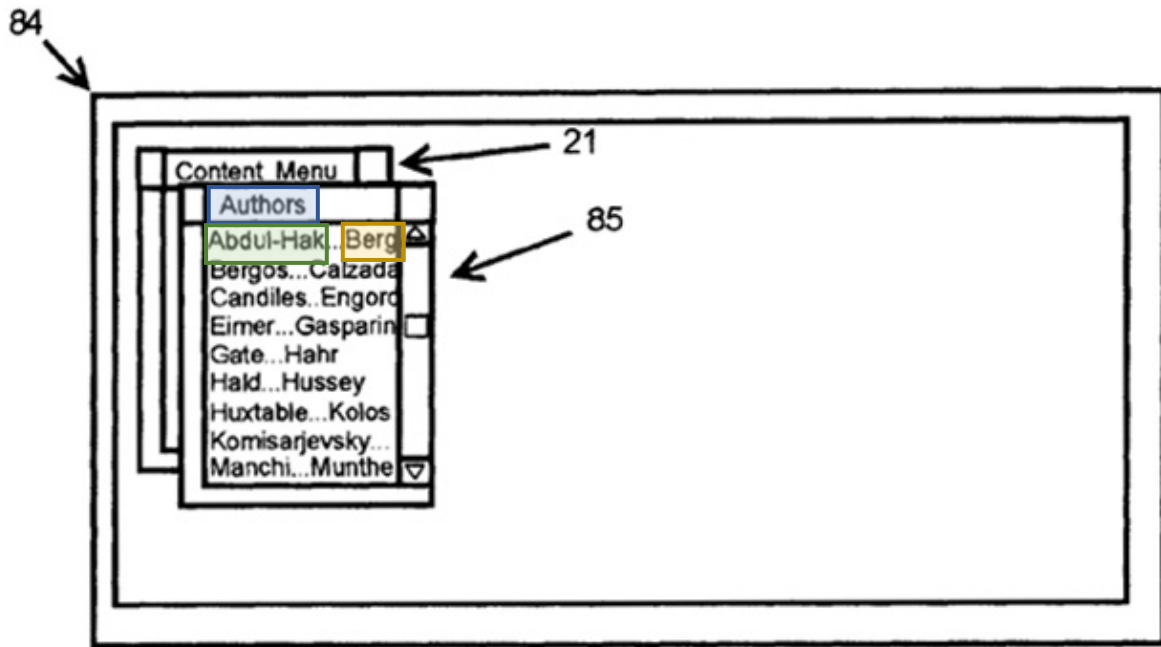
195. First, a user in Zellweger can select a database field in the Content Menu, and upon this selection, Zellweger will determine and display for the user how many entries are in that field. Ex. 1007 (Zellweger), 2:30-35, 4:7-16, Figure 6a.



Id., Figure 6a (annotated).

196. Figure 6a above shows a user selecting “Authors” (in blue) and Zellweger determining that there are 1,231 entries in Authors (in green).

197. Second, Zellweger discloses this limitation when generating a “Content List.” Zellweger discloses there are two kinds of Content Lists that can be created: an “Alphabet List” and a “Content List.” Ex. 1007 (Zellweger), 4:17-35. If the user opts to make a Content List, all the entries are divided and grouped into intervals as shown below:



Id., Figure 7 (annotated).

198. Figure 7 above shows an example using the Authors database field (in blue) and the last name of the authors (in green and yellow) such that all the intervals are listed “using the author’s last name as the first and last elements.” Ex. 1007 (Zellweger), 4:45-53. For example, the first interval in the list includes authors with last name “Abdul-Hak” (in green) and goes to “Berg” (in yellow, although it’s cut off so whether the entire last name for the final last name in the interval is “Berg” or simply starts with “Berg” is unknown; this is irrelevant to the point being made here; see four lines below for the name range “Gate...Hahr” which is not cut off). *Id.*, Figure 7. If a user clicks on an entry, Zellweger discloses that the user will open a

newly generated “new list menu that starts with” the first entry and ends with the last entry in the selected interval. *Id.*

199. A Zellweger user has some settings available to them (in green in annotated Figure 6a below). Specifically, the user can set a maximum and a minimum number of items per list for the content list. Ex. 1007 (Zellweger), 4:26-31. For Zellweger to be able to determine how to segment the entries into the Content List intervals, and to determine whether the maximum or minimum is satisfied for a list, Zellweger must necessarily determine the quantity of entries for the associated database field (*e.g.*, Authors).

☒ Alphabet List
Tab Length Entries: 27

☐ Content List

☐ Word Length

☐ Letter Tab Length

Items per List
Maximum
Minimum

Id., Figure 6a (annotated).

200. A POSITA, in applying the methods of Zellweger to the pick boxes of Cochran, would have understood that such a combined system also “*determines a*

quantity of entries in the selected database field.” Zellweger’s improved menu system uses functionality from Zellweger that performs this limitation. Thus, an integration of Zellweger’s improved menu teachings into Cochran may alternatively use Zellweger’s functionality to determine how many items are in the database field. See §XII.A.6 below.

201. In sum, Cochran discloses “*determining a quantity of entries in the selected database field*” to determine how to format the display of pick boxes. Zellweger discloses “*determining a quantity of entries in the selected database field*” because it explicitly shows determining the number of entries, and because it must determine how many items are in a list to determine whether the maximum or minimum as set by the user is satisfied.

6. **[1.E] “if the quantity exceed a specified amount, truncating data, and displaying the truncated data wherein the truncating reduces characters in one or more entries in the selected database field and the truncated data represents each of the entries in the selected database field; and”**

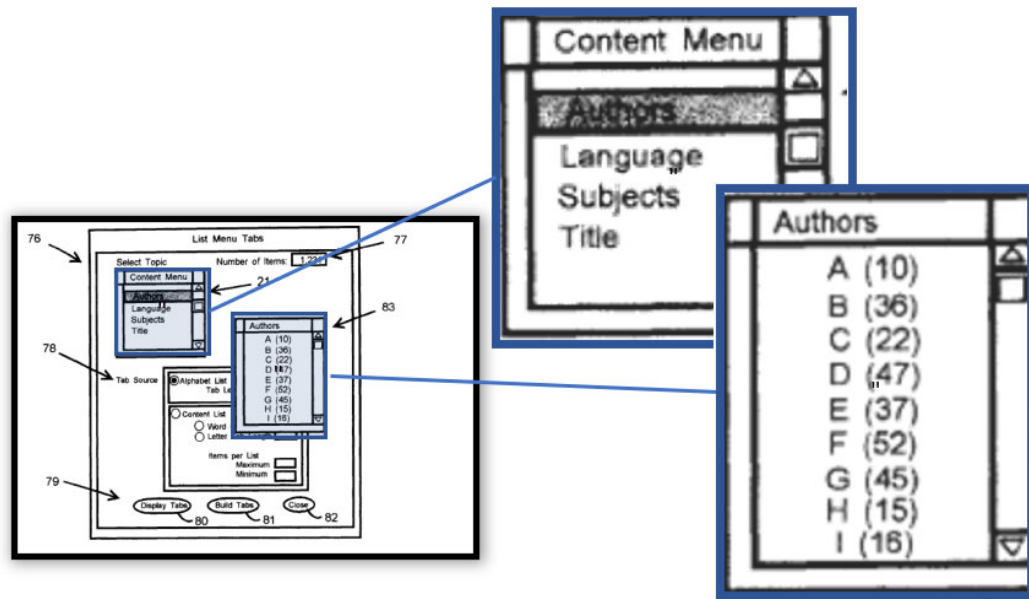
202. As an initial matter, a POSITA would have known that truncating data entries by reducing their size, truncating strings or numbers, rounding numbers, and enabling an assortment of forms of summarization of data types (*e.g.*, dates) was already very well known by 1999. See §VII.D above.

203. Cochran discloses that, in order for it to display “more [search terms] ... than can fit on the display screen simultaneously,” it must make use of a scroll bar

with “scroll buttons” to allow the user to “scroll[] to display additional search terms.” Ex. 1005 (Cochran), 13:58-14:7; *see also id.*, 10:48-65, 14:45-49, 15:4-11, 16:15-21. This allows a user to view and select from all the search terms for a specific database field even if all the terms cannot fit into the pick box on screen at the same time.

204. Zellweger discloses various methods that are used to display voluminous lists such as the lists in Cochran: the latter are so voluminous that all the entries in the list may not fit on the screen at the same time. Specifically, Zellweger discloses two separate types of lists known as “Alphabet Lists” and “Content Lists,” each of which “*truncat[es]*” the text of entries in a list by “*reduc[ing] characters in one or more entries of the selected database field*” and displaying them grouped into intervals (“*displaying the truncated data*”). *See* §XII.A.5 above (Claim [1.D]).

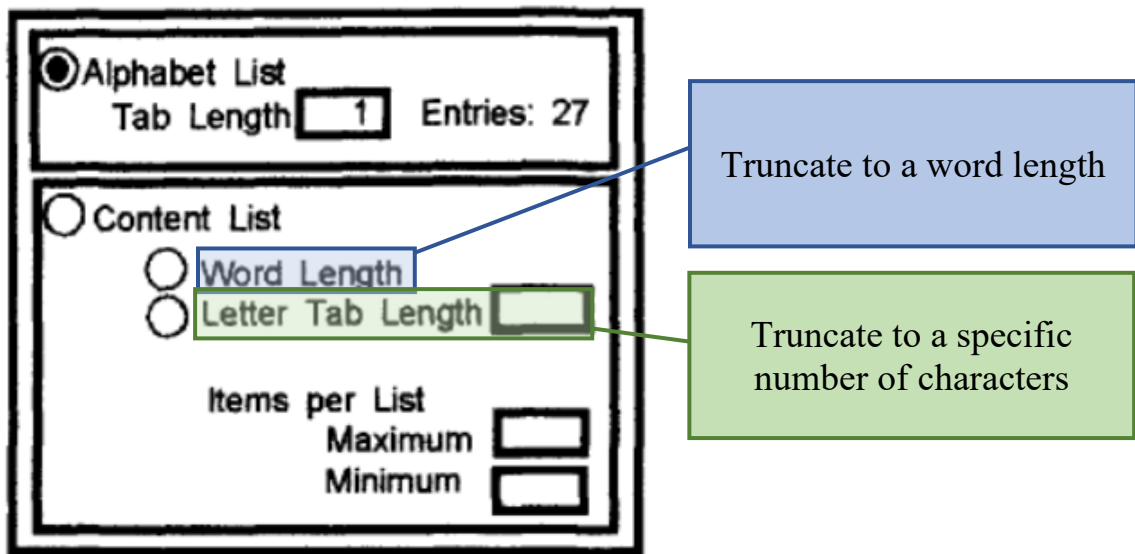
205. Zellweger’s Alphabet List first reduces the number of entries in the list to a user-defined number of letters and organizes the entries into groups based on their first letters (up to the user-defined number of letters). Figure 6b is an example of this, showing the truncation of the entries for “Authors” to one letter (and listing the count of elements that start with each author’s letter in parenthesis):



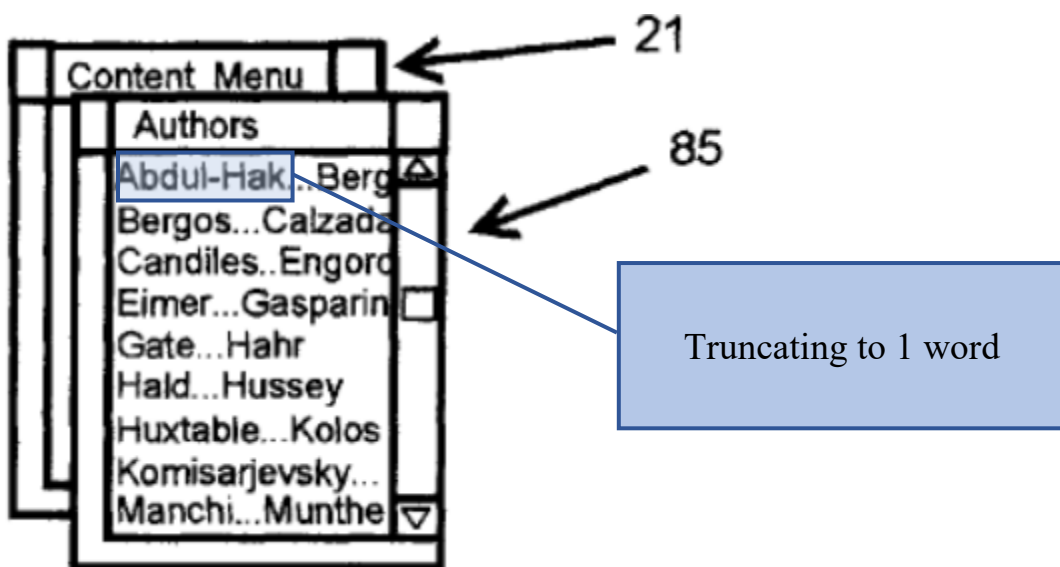
Ex. 1007 (Zellweger), Figure 6b (annotated, excerpts zoomed in), 4:36-44.

206. Here, all entries in the field “Authors” were truncated to a single letter (the first, or leading, letter) and then organized into a group. For example, Figure 6b shows that there are 10 entries in the Authors field that start with the letter A, 36 that start with the letter B, and so forth.

207. Zellweger’s Content List works a bit differently. Specifically, when a Content List is used, the entries are truncated down to a word (blue) or a number of leading characters (green) and then organized into a set of intervals, as seen in the figures below:



Ex. 1007 (Zellweger), Figure 6a (annotated).



Id., Figure 7 (annotated).

208. Figure 7 above shows how the entries for the “Authors” database field are truncated to one word length and organized into ranges or intervals. Ex. 1007 (Zellweger), 4:45-53.

209. The truncated data in either the Alphabet List or Content List therefore “represents” each of the authors (“entries”) in the “Authors” field (“selected database field”). Specifically, each of the entries are *represented* in the Alphabet List grouping lists or the Content List interval lists. That is, in either of these lists, each entry is encapsulated by one of the set of groups or intervals.

210. As I just explained for Claim [1.D] (§XII.A.5 above), Cochran discloses modifying the pick box list (*e.g.*, by deciding to add a scroll bar) when the list of search values/entries for the pick box exceeds the number of entries that can be fit onto the display screen all at once. A POSITA would have found it obvious to “truncate” the search values/entries of Cochran using Zellweger’s Alphabet List or Content List when the number of entries exceeds that specified amount (*i.e.*, when the number of entries are so great that they cannot all be displayed in the expanded pick box simultaneously), as it would have made viewing, selecting, or processing large amounts of data more efficient for users. *See also* §VII.D above.

211. Alternatively, Zellweger teaches using a user-determined setting called “minimum” which specifies the smallest number of entries for the Content List. A POSITA would have understood or found obvious to truncate based on any predetermined number (*e.g.*, 10, 50, 100) that’s defined for this minimum setting (“if the quantity exceed a specified amount”). This permits the user additional flexibility to customize the level of truncation to fit a specific case (*e.g.*, small databases may

need little to no truncation, whereas larger ones may require more aggressive truncation). Ex. 1007 (Zellweger), 4:26-35, Figure 6a, 7; *see* §XII.A.5 above (Claim [1.D]). Thus, Cochran in view of Zellweger teaches truncating when a specified amount (number) of entries will be in a list (*“if the quantity exceed a specified amount”*).

212. In sum, Cochran in view of Zellweger discloses *“if the quantity exceed a specified amount”* because Cochran discloses determining if the quantity of search terms are greater than the number that can be displayed simultaneously on screen and Zellweger discloses determining the number of entries (to display the number or to calculate whether a minimum value has been satisfied) and Cochran in view of Zellweger discloses *“truncating data, and displaying the truncated data wherein the truncating reduces characters in one or more entries in the selected database field and the truncated data represents each of the entries in the selected database field”* because Zellweger discloses truncating the entries in a list and displaying them in a truncated format in either an Alphabet List or Content List.

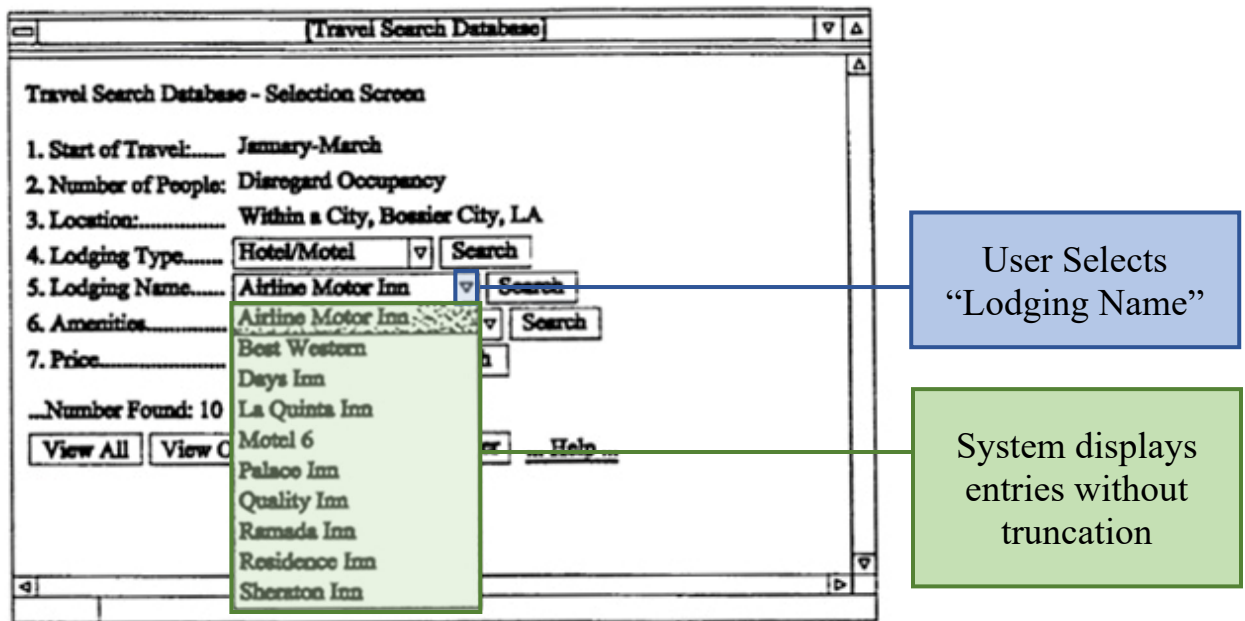
7. [1.F] “if the quantity does not exceed the specified amount, displaying contents of the database field.”

213. As I explained above regarding Claim [1.E], Cochran in view of Zellweger discloses or renders obvious truncating the list of entries if the list exceeds a specified number of entries. *See* §XII.A.6 above (Claim [1.E]). Specifically, I

explained that the combined system of Cochran in view of Zellweger would truncate when the number of entries exceeds the number of lines available on a display screen. *See* §XII.A.6 above (Claim [1.E]).

214. Conversely, there is the situation where the number of search terms to display in a pick box does not exceed that number (*e.g.*, the number of entries that can be displayed in the pick box on screen simultaneously). A POSITA would have understood or found obvious that, in this event, the combined system/method displays the list without truncation (“*displaying contents of the database field*”) in this case, when the number of entries does not exceed the specified amount (“*if the quantity does not exceed the specified amount*”). *See* §XII.A.6 above (Claim [1.E]). *See also* §VII.E above.

215. Indeed, Cochran already discloses that when the number of entries in the pick box does not exceed the number (which is the maximum number of entries that can be displayed on screen simultaneously), Cochran simply displays all the entries without truncation, seen in the figure below (in green).



Ex. 1005 (Cochran), Figure 7c (annotated).

216. For example, Figure 7c above shows displaying entries for “Lodging Name” after a user selects it and clicks on the button to expand its pick box (in blue) without truncation/modification because the list does not exceed the number of lines displayable on the screen (in green); *see also* Ex. 1005 (Cochran), 11:22-48, 15:64-66, 16:15-21.

217. In sum, Cochran in view of Zellweger discloses this because Cochran discloses displaying the search terms in an expanded pick box without truncation; the system of the Cochran/Zellweger combination would use this (displaying entries without truncation) if the number of entries for display is smaller than the “*specified amount.*”

B. Claim 2

1. “The method of claim 1, further comprising providing a key word search.”

218. Cochran discloses that its database “may be searched by specifying the desired information sought as a ‘search term.’” Ex. 1005 (Cochran), 1:30-37, 2:25-28, 9:18-22. Specifically, when a user selects an entry from a pick box, that entry is a “search term” used to retrieve data from the database. *Id.*, 10:13-18. Therefore, a POSITA would have understood that the selection of Cochran’s search terms in the pick boxes constitutes providing “key words” for searching Cochran’s database and thus selecting a search term in a pick box constitutes “*providing a key word search*” to search the Cochran database. *See also* §VII.F above.

C. Claim 3

1. [3.PRE] “A method implemented on a computer for searching a database, comprising:”

219. Cochran in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claim [1.PRE]. *See* §XII.A.1 above (Claim [1.PRE]).

2. [3.A] “generating a list of data fields;”

220. Cochran in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claim [1.B]. *See* §XII.A.3 above (Claim [1.B]).

3. [3.B] “receiving a first data field selection from the list of data fields;”

221. Cochran in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claim [1.C]. *See* §XII.A.4 above (Claim [1.C]). Any difference in terminology between Claim 1 and Claim 3 is immaterial.

A POSITA would have understood that these terms are equivalent:

- “search selection” (Claim 1) and “first data field selection” (Claim 3); and
- “for a database field on the provided list of the database fields” (Claim 1) and “from the list of data fields” (Claim 3).

222. Cochran discloses these limitations for the same reasons as stated for Claim 1. *See* §XII.A.4 above (Claim [1.C]).

4. [3.C] “determining a first quantity indicative of a number of entries of the selected data field;”

223. Cochran in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claim [1.D]. *See* §XII.A.5 above (Claim [1.D]). Any difference in terminology between Claim 1 and Claim 3 is immaterial.

A POSITA would have understood that these terms are equivalent:

- “quantity of entries” (Claim 1) and “first quantity indicative of a number of entries” (Claim 3); and
- “in the selected database field” (Claim 1) and “of the selected data field” (Claim 3).

224. Cochran and Zellweger disclose these limitations for the same reasons as stated for Claim 1. *See* §XII.A.5 above (Claim [1.D]).

5. **[3.D] “if the first quantity exceeds a specified limit, reducing a size of data to be displayed from the selected data field wherein the reducing reduces characters in one or more entries in the selected data field and the size-reduced data represents each of the entries in the selected field; and”**

225. Cochran in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claims [1.E] and [3.E]. *See* §XII.A.6 above (Claim [1.E]) and §XII.C.6 below (Claim [3.E]). Any difference in terminology between Claim 1 and Claim 3 is immaterial. A POSITA would have understood that these terms are equivalent:

- “quantity” (Claim 1) and “first quantity” (Claim 3);
- “specified amount” (Claim 1) and “specified limit” (Claim 3);
- “truncating data” (Claim 1) and “reducing a size of data” (Claim 3);
- “the truncating” (Claim 1) and “the reducing” (Claim 3);
- “database field” (Claim 1) and “data field” (Claim 3);
- “truncated data” (Claim 1) and “size-reduced data” (Claim 3); and
- “selected database field” (Claim 1) and “selected field.” (Claim 3).
- Cochran and Zellweger disclose these limitations for the same reasons.

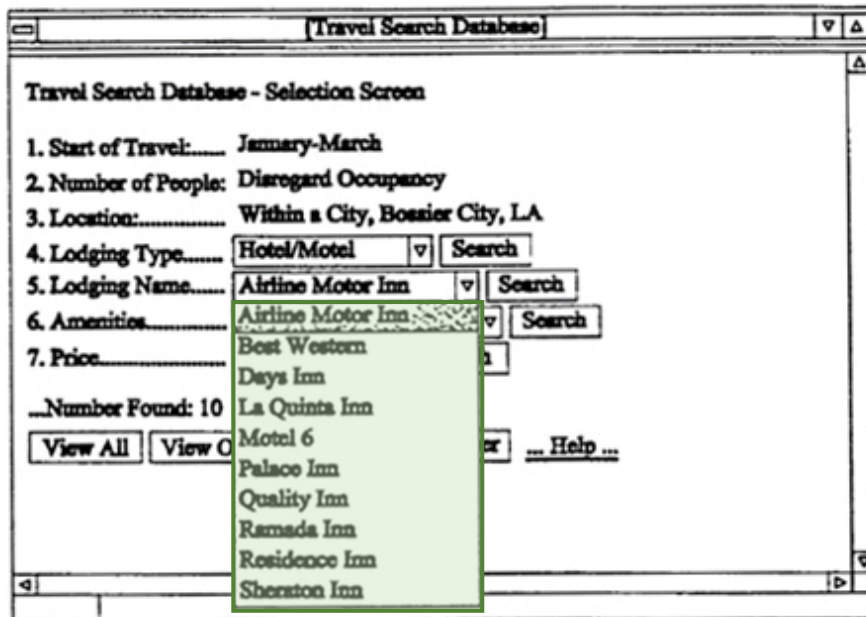
226. Cochran and Zellweger disclose these limitations for the same reasons as stated for Claim 1 and below at Claim [3.E]. See §XII.A.6 above (Claim [1.E]) and §XII.C.6 below (Claim [3.E]).

227. Claim 3.D contains two terms that do not have an antecedent basis: (1) “selected field” and (2) “date.” I make the following assumptions: (1) “the selected field” refers to “the selected data field” and (2) “date” is a typo and should be read as “data.”

6. [3.E] “displaying data from the selected data field.”

228. Cochran in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claims [1.D] and [1.E]. Specifically, in §§XII.A.5 above and XII.A.6 above (Claims [1.D] and [1.E], respectively), I discuss the various ways in which the combined system “*display[s] data from the selected data field.*”

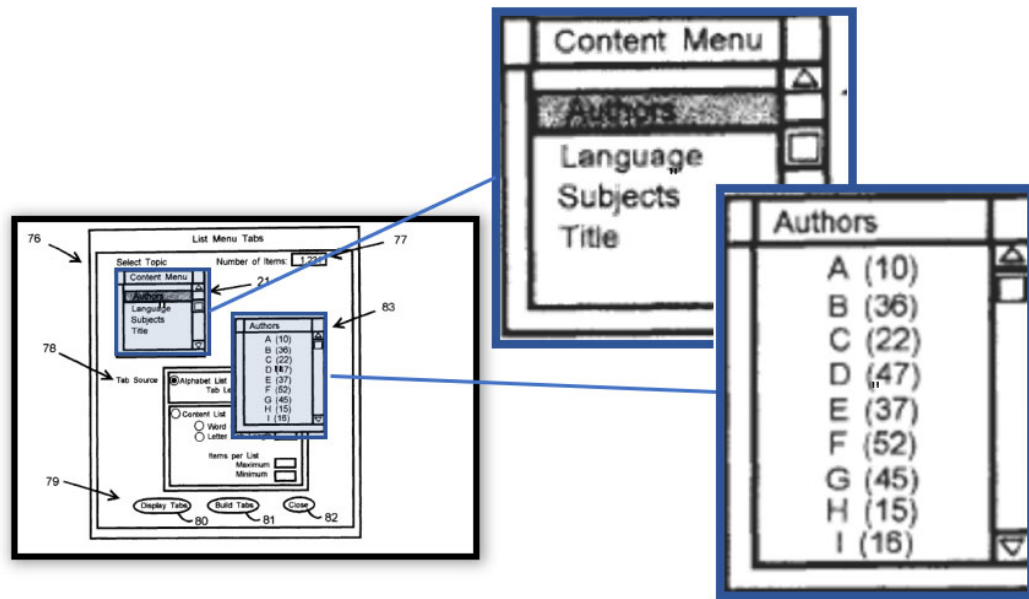
229. Specifically, Cochran discloses displaying search terms (“*displaying data*”) in a user-selected pick box (“*from the selected data field*”):



Ex. 1005 (Cochran), Figure 7c (annotated).

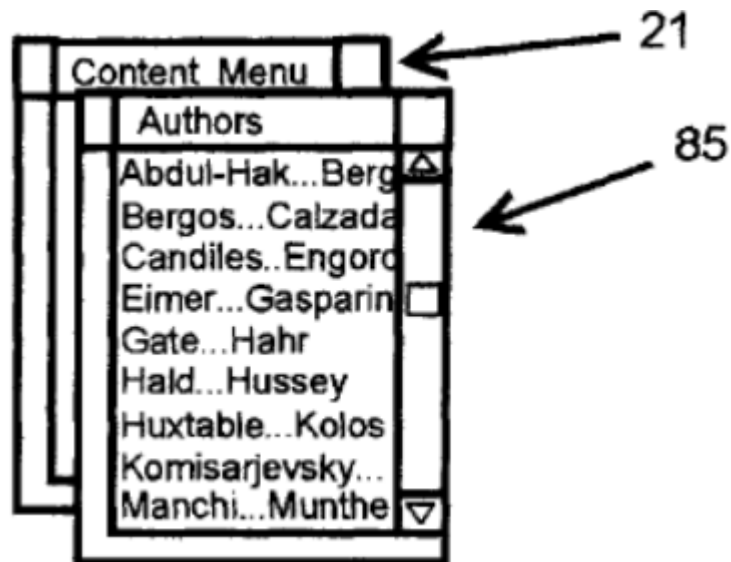
230. Figure 7c shows that the data (search terms) for the data field associated with “Amenities” is displayed in the expanded pick box (in green).

231. Zellweger discloses displaying the entries (“*displaying data*”) organized in either an Alphabet List or Content List for the selected topic (“*from the selected data field*”):



Ex. 1007 (Zellweger), Figure 6b (annotated, excerpts zoomed in).

232. Figure 6b shows that “Authors” is the “selected data field.” See Ex. 1007 (Zellweger), 3:66-4:26.



Id., Figure 7.

233. Figure 7 again shows that “Authors” is the “*selected data field*” with a scrollable list of author ranges/intervals. *See* Ex. 1007 (Zellweger), 4:26-31.

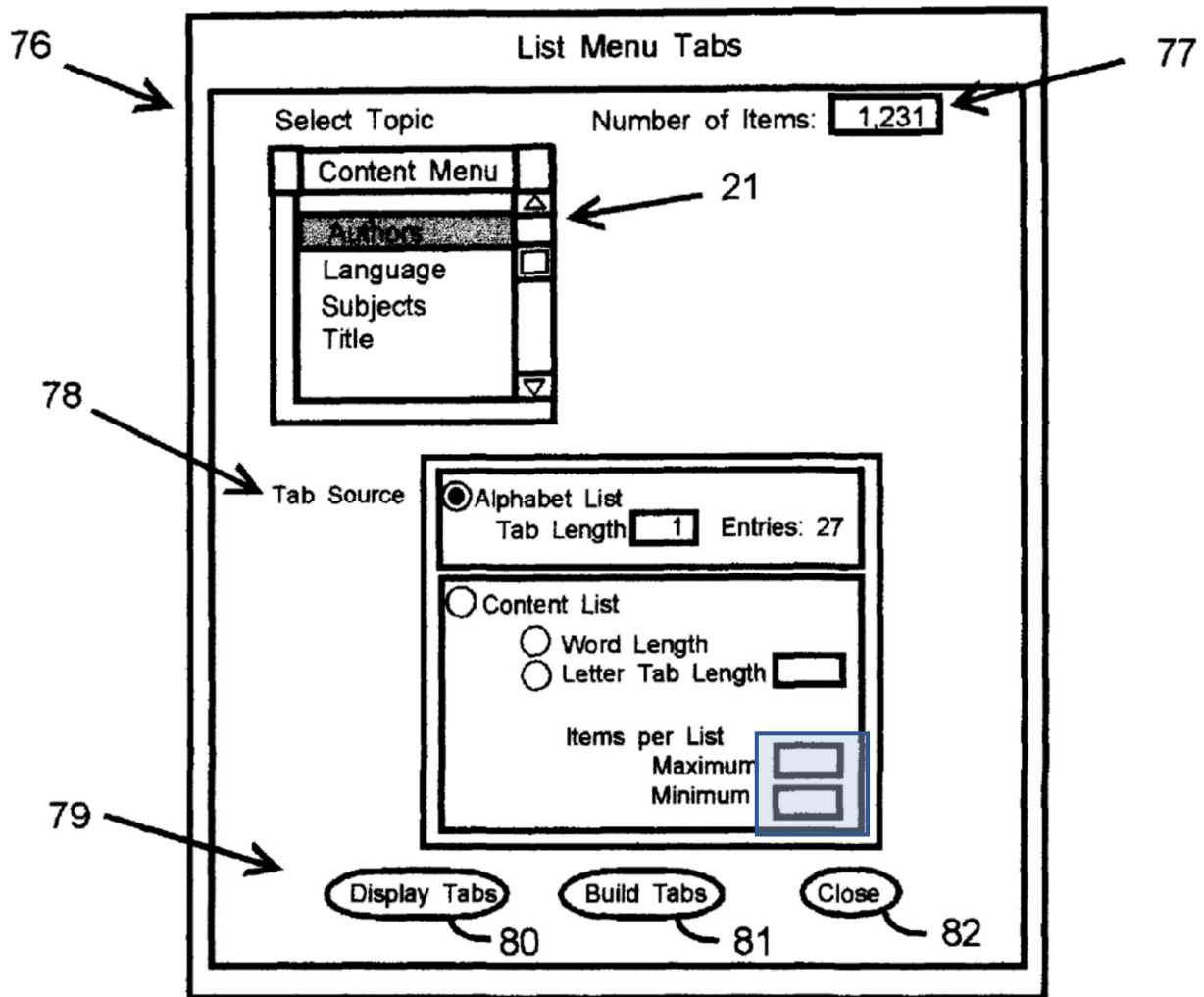
234. In sum, Cochran discloses “*displaying data from the selected data field*” because it discloses displaying the search terms for the selected data field in its expanded pick boxes and Zellweger discloses “*displaying data from the selected data field*” because it discloses displaying the entries in Alphabet Lists and Content Lists for the selected data field.

D. Claim 4

1. “The method of claim 3, wherein the specified limit is fixed.”

235. As I discussed above in §§XII.A.5 above and XII.A.6 above (Claims [1.D] and [1.E], respectively), Zellweger discloses a method of splitting a list of entries into intervals using “Content Lists” and a user may specify the minimum and maximum number of “Items per List.” *See* Ex. 1007 (Zellweger), Abstract (“The present invention discloses a menu authoring system component that enables a developer to generate and build a tab list menu. The tab menu divides a target list menu into tab intervals that can include a specified number of letters or full words. The present invention discloses an interactive configuration window and the program logic that enables a developer to specify settings for these tab menus.”), 2:30-43, 4:26-52, Figure 6a.

236. Zellweger shows the “Minimum” and “Maximum” settings are plain text input fields (highlighted in blue below).



Ex. 1007 (Zellweger), Figure 6a (annotated).

237. Figure 6a shows that the Maximum and Minimum values are set by a user typing text into plain text entry boxes (in blue). A POSITA would have understood that such plain text input fields allow a user to input whatever they want

into the field. The specified limit is “fixed” when the user sets the minimum items per list to one value and does not change it.

E. Claim 5

1. “The method of claim 3, wherein the specified limit is variable.”

238. As I discussed above in §XII.D above regarding Claim 4, a user in Zellweger can set a minimum and maximum number of items per list in the “Maximum” and “Minimum” text input boxes.

239. Zellweger’s minimum and maximum values are variable because the user may change either one at will and the values are thus “variable” when the values are changed. *See* §XII.D above (Claim 4). Additionally, as I discussed above, Cochran teaches that the specified limit may be based on the size of the screen (*i.e.*, the number of entries that can fit on the screen simultaneously), which is variable. *See* §XII.A.6 above (Claim [1.E]).

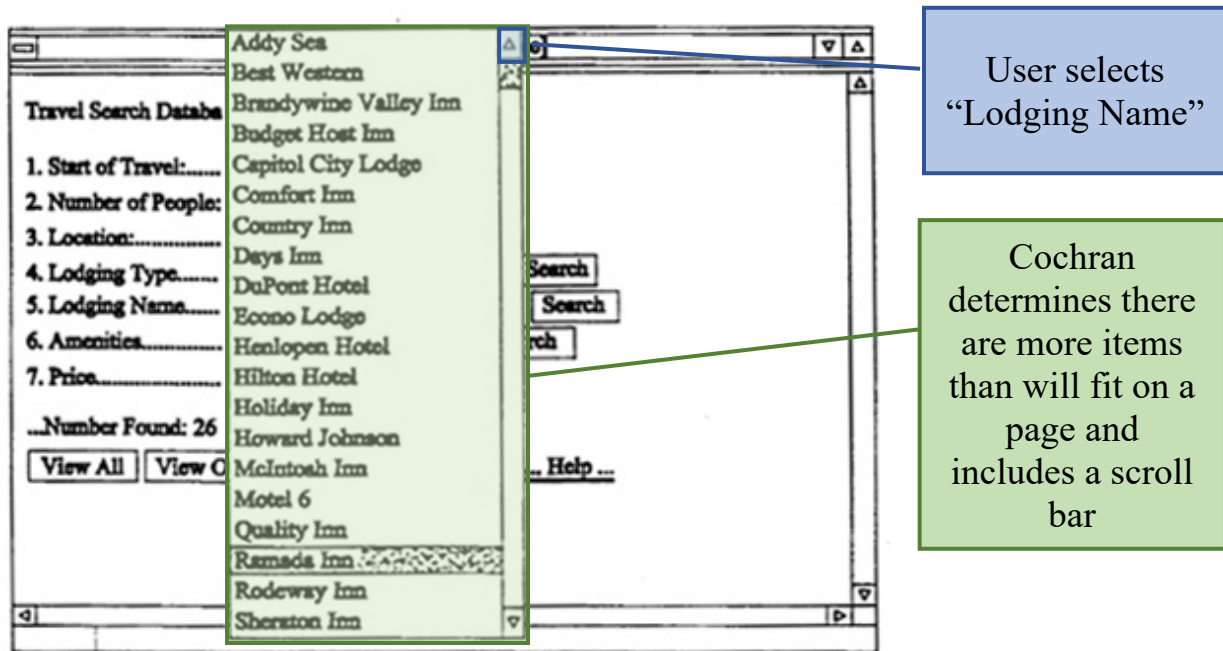
F. Claim 6

1. “The method of claim 3, wherein the data are displayed on a terminal, and wherein the specified limit is determined dynamically, based on a characteristic of the terminal.”

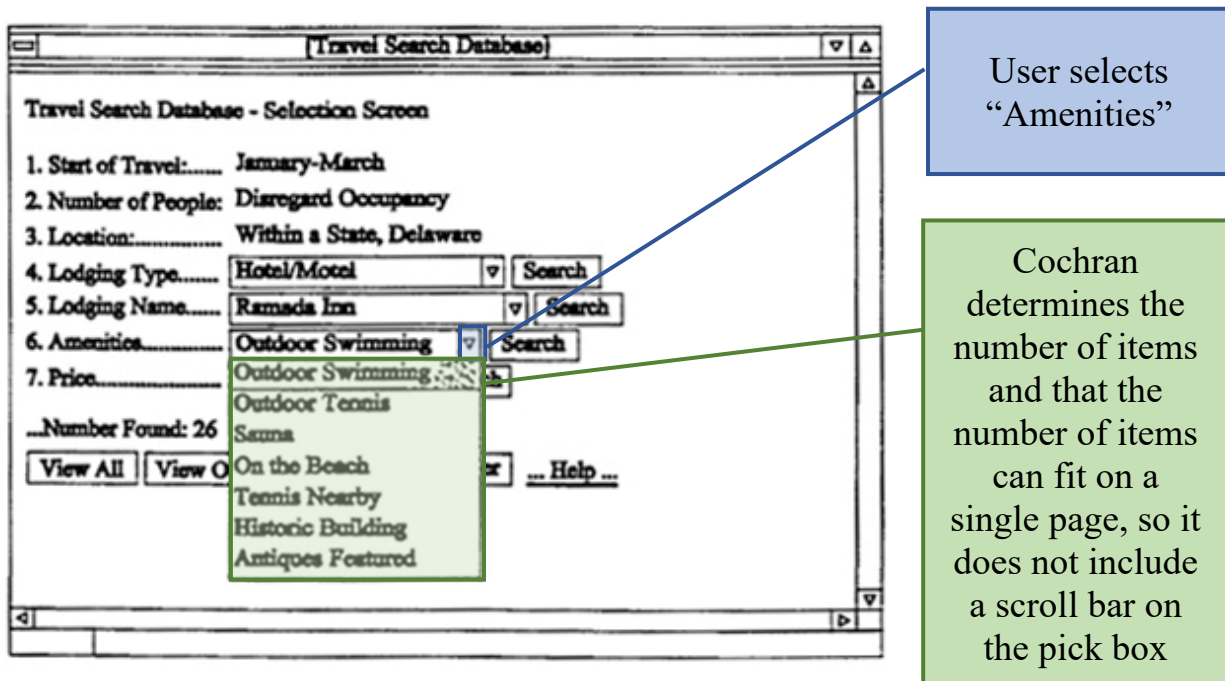
240. As an initial matter, a POSITA would have known that every display or terminal device has specified dimensions (*e.g.*, number of vertical and horizontal characters or pixels), and that operating systems include several mechanisms for

users and applications to find those dimensions and act accordingly to display information that fits therein. *See* §VII.B above.

241. Cochran discloses determining whether all entries to be displayed in a pick box are displayable simultaneously on the screen and modifying the pick box list (*e.g.*, adding a scroll bar if there are too many entries) if the number of entries exceeds that which will fit on the display screen. *See* §§XII.A.5 and XII.A.6 above (Claims [1.D] and [1.E], respectively). The number of entries which fit on the screen is shown in the pick box's list (*i.e.*, the “*data are displayed on a terminal*”) and the remaining entries can be accessed using the scroll bar added to the pick box list. As a recap I have included below annotated versions of Figures 5d (listing items using a scroll bar) and 5e (listing items without a scroll bar) from Cochran.



Ex. 1005 (Cochran), Figure 5d (annotated).



See *id.*, Figure 5e (annotated).

242. As seen in Figures 5d and 5e above, when a user selects one of the data fields and clicks to expand the pick box (in blue), Cochran will determine whether or not the search terms for display in the expanded pick box will fit on the screen on one page simultaneously. If they will, Cochran decides to not include a scroll bar; if they will not, Cochran decides to include a scroll bar (in green).

243. From this behavior, a POSITA would have understood that Cochran determines how many items can fit on the screen at once to decide whether or not to include a scroll bar on the pick box list. The number of entries that can fit on the screen will depend on the display capabilities of the terminal (“*wherein the specified limit is determined dynamically, based on a characteristic of the terminal*”) because some terminals may be smaller or larger than expected or may be running at a lower or higher display resolution than expected. For example, if a terminal is running at a lower resolution than expected, fewer entries may be displayable simultaneously. A POSITA would have understood that, to allow Cochran to properly determine the “specified limit” to trigger inclusion of a scroll bar, Cochran must dynamically determine the number based on the terminal’s display characteristics.

244. A POSITA would have further understood that software which displays any information on a display, whether text-based or graphical, would have to identify the display’s size and capabilities: it would make no sense to attempt to display more

items than can fit on the display screen, as those items would not be visible to the user. *See also* §§VII.B and VII.F above.

G. Claim 7

1. **“The method of claim 3, wherein the specified limit is a user-determined limit.”**

245. As I discussed for Claims 4 and 5 in §§XII.D and XII.E above, respectively, Zellweger discloses that a user “can specify the maximum and minimum” value for the Content List “items per list.” Ex. 1007 (Zellweger), 4:26-31, Figure 6a. Because the user specifies these values by typing them into the associated text box for the “maximum” or “minimum” they are necessarily “user-determined.”

H. Claim 15

1. **[15.A] “The method of claim 3, further comprising: receiving a first constraint, wherein the first constrain[t] is related to a data element in a data field; and”**

246. As an initial matter, a POSITA would have known that databases and search engines are designed to allow one or more search constraints, with a versatile set of logical operators to refine and narrow down the search criteria. *See* §§VII.C and VII.F above.

247. I previously discussed how Cochran discloses that a user selects a search term. *See* §XII.A.4 (Claim [1.C]) and §XII.C.3 above (Claim [3.B]). A POSITA would have understood that each of the selected search terms constitutes a

“constraint” because each term limits (or constrains) the scope of the result set returned by the user’s search query. Ex. 1005 (Cochran), Abstract (“A search of the database is conducted based upon selected search terms from at least one of the multiple lists. The subset of records identified by the search is used to form new lists of search terms that are then displayed on the display device.”), 8:45-65. Cochran discloses that a user may click on a “View All” button which applies the search terms and “locate[s]” and “presents the entire subset of records to the user.” *Id.*, 12:66-13:14. Thus, when a user selects a *first* search term Cochran is “*receiving a first constraint.*”

248. In Cochran, the search terms are the “terms that are actually available in the database.” Ex. 1005 (Cochran), Abstract. That is, each search term is data that is stored in the database for the associated database field. For example, the price search terms are related to the Price database field, the hotel name search terms are related to the Lodging Name database field, and so forth for each list identifier displayed in Cochran’s Selection Screen—showing the various list identifiers and pick boxes. *See, e.g., id.*, 4:2-9, 4:29-35, 12:44-65, 14:13-15:16, Figures 4b, 4d-4g, and 5b-5f. A POSITA would have understood from these disclosures that each of the search terms in a pick box is “*related to*” a corresponding “*data element in a data field.*” Thus, whatever search term is selected as the “first constraint” is also “*related to*” its corresponding “*data element in a data field.*”

249. In sum, Cochran discloses “*receiving a first constraint, wherein the first constrain[t] is related to a data element in a data field*” because it receives a search term selected by the user and that search term comes from the database as a potential value for the corresponding field in the database.

2. [15.B] “receiving one or more subsequent constraints, wherein search results are generated based on a combination of the first and the one or more subsequent constraints.”

250. As a natural extension to the discussion above, a POSITA would have understood that Cochran receives “*one or more subsequent constraints*” when a user selects multiple additional search terms (“*constraints*”). See §XII.H.1 above (Claim [15.A]). Take as one example when a user first selects a search term “Within a City, Boston, MA” for the list identifier “Location” and then *subsequently* selects “Economy” as the search term for the list identifier “Price” (or any other additional constraints as applicable and desired (*e.g.*, “Amenities,” etc.)). Ex. 1005 (Cochran), 16:9-17:13, Figures 3d, 8a-8e. The subsequent selection of the “Economy” is one of potentially many “*subsequent constraints*” that the user may select and apply to their search.

251. Cochran shows that the search results are “*generated based on a combination of the first*” and “*subsequent constraints*” in several ways. First, the user’s selected search terms may be applied in combination to arrive at a narrowed list of options in the other pick boxes (*e.g.*, the “Lodging Type”). Ex. 1005

(Cochran), Figures 8a-9d, 16:38-17:13. For example, Cochran discloses that a user may have “only one choice at Lodging Type: Hotel/Model, instead of [] 5 choices available in Boston, Mass. before Economy was chosen.” *Id.* This shows that as a user makes selections in the pick boxes, the search terms are iteratively applied as search term constraints to narrow down the search term options displayed in the other pick boxes.

252. Second, Cochran discloses the user may click on the “View All button 121” which “presents the entire subset of records to the user.” Ex. 1006 (Cochran), 12:66-13:3. This means that when the user selects multiple constraints (*e.g.*, Start of Travel, Number of People, Location and Price), and subsequently clicks on the View All button, Cochran will generate a set of search results that is found by the application of the search terms as a combined set of constraints on the data set in the database. *Id.*

253. Lastly, a POSITA would have recognized that narrowing down a search to multiple terms is common human experience (*e.g.*, searching for a specific address, first by state, then city, then street, then house number). Moreover, computers and databases have long permitted refining searches, using conjunctive (logical AND) operators. *See* §VII.C above.

254. In sum, Cochran discloses “*receiving one or more subsequent constraints, wherein search results are generated based on a combination of the first*

and the one or more subsequent constraints” because Cochran receives several search terms when a user selects them in different pick boxes and Cochran generates a set of search results after applying those search terms against the database to constrain the set of results based on the search terms.

I. Claim 16

- 1. “The method [of] claim 3, wherein each display of data comprises a search cycle, and further comprising receiving a refresh command, the refresh command resetting all search cycles.”**

255. As explained below in this section, Cochran discloses that a user selects search terms from the pick boxes, where a selection from one pick box initiates a search query to retrieve other search terms to be further used in additional pick boxes, to incrementally refine a search query. Cochran updates the terms in the other pick boxes to account for a narrowing of the relevant terms by the selection of additional search terms.

256. The discussion below, associated with the annotated reproductions of Figures 8a, 8e, 9a, and 9c, shows an example of selections from individual pick boxes constituting a “*search cycle*.” This example illustrates how the selection of search terms in pick boxes individually may initiate searches to ensure that “lists that are displayed never include search terms that are not available after taking into account choices that have already been made.” Ex. 1005 (Cochran), 16:28-37.

The screenshot shows a window titled "Travel Search Database" with a subtitle "Travel Search Database - Selection Screen". It contains a list of seven search criteria, each with a text input field and a "Search" button. A blue rectangular box highlights the "Search" buttons for criteria 4 through 7. Below the criteria, it says "...Number Found: 73". At the bottom, there are five buttons: "View All", "View One", "Reset 4-7", "Start Over", and "... Help ...".

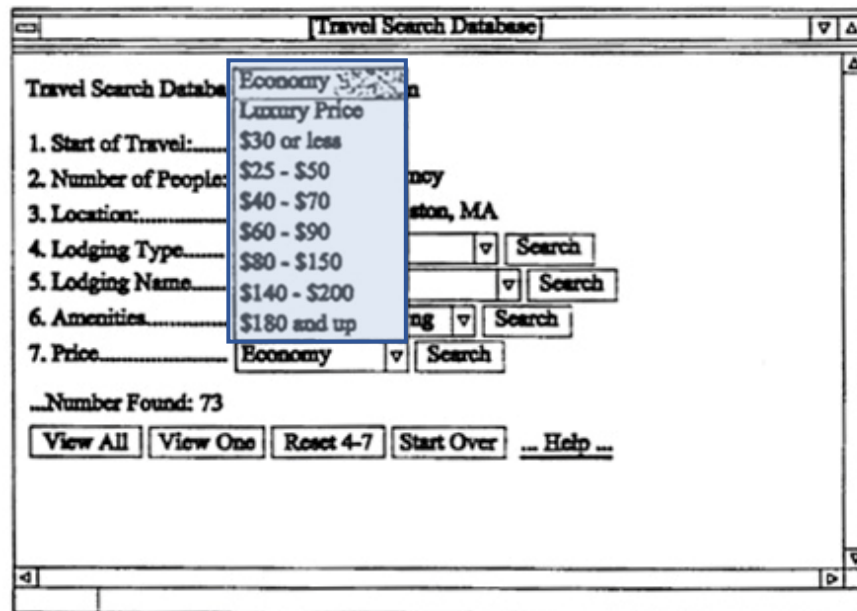
Criteria	Value	Action
1. Start of Travel:.....	January-March	
2. Number of People:	Disregard Occupancy	
3. Location:.....	Within a City, Boston, MA	
4. Lodging Type.....	Hotel/Motel	Search
5. Lodging Name.....	Best Western	Search
6. Amenities.....	Outdoor Swimming	Search
7. Price.....	Economy	Search

...Number Found: 73

View All View One Reset 4-7 Start Over ... Help ...

Id., Figure 8a (annotated).

257. Figure 8a above shows that the user received search terms to display in the pick boxes for the data fields “Lodging Type,” “Lodging Name,” “Amenities,” and “Price” (in blue) as a part of a first cycle. *See* Ex. 1005 (Cochran), 16:9-37.



Id., Figure 8e (annotated).

258. Figure 8e above shows that the user expanded the pick box to display the values associated with the data field “Price” (in blue). *See* Ex. 1005 (Cochran), 16:24-27.

The screenshot shows a window titled "Travel Search Database" with a subtitle "Travel Search Database - Selection Screen". It contains seven numbered search criteria, each with a text input field and a dropdown menu. The values entered are: 1. Start of Travel: January-March; 2. Number of People: Disregard Occupancy; 3. Location: Within a City, Boston, MA; 4. Lodging Type: Hotel/Motel; 5. Lodging Name: Comfort Inn; 6. Amenities: Outdoor Swimming; 7. Price: Economy. Each dropdown menu has a "Search" button next to it. Below the criteria, the text "...Number Found: 13" is displayed, with "13" highlighted in red. At the bottom, there are five buttons: "View All", "View One", "Reset 4-7", "Start Over", and "... Help ...".

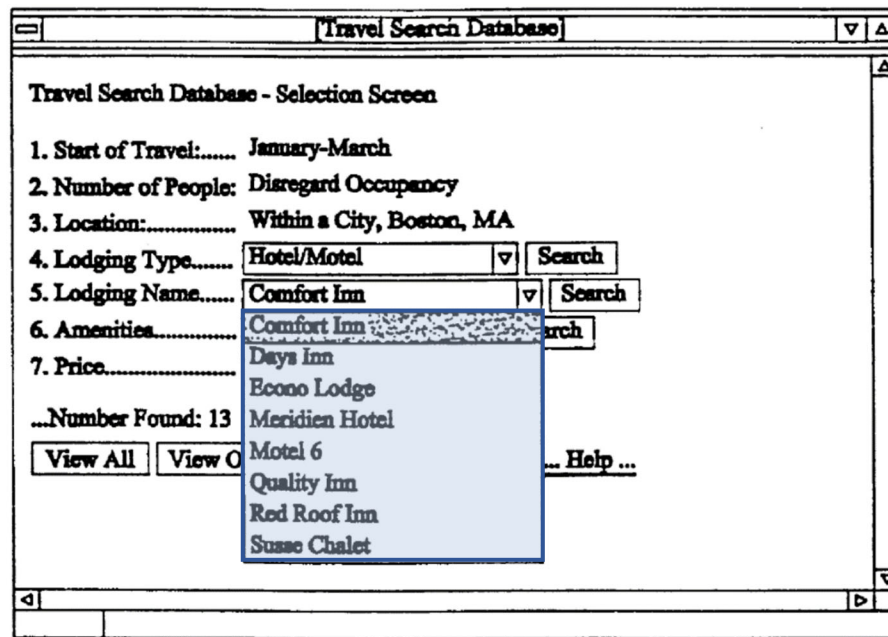
Criteria	Value
1. Start of Travel	January-March
2. Number of People	Disregard Occupancy
3. Location	Within a City, Boston, MA
4. Lodging Type	Hotel/Motel
5. Lodging Name	Comfort Inn
6. Amenities	Outdoor Swimming
7. Price	Economy

...Number Found: 13

View All View One Reset 4-7 Start Over ... Help ...

Id., Figure 9a (annotated).

259. Figure 9a above shows that the user selected “Economy” as the search term from the pick box associated with the “Price” data field (in blue). This kicks off a second (narrowing) search for the second search cycle to update the list of search terms for the other pick boxes, as confirmed by the fact the “Number Found” value was reduced from 73 to 13 (in red). *See* Ex. 1005 (Cochran), 16:39-52.

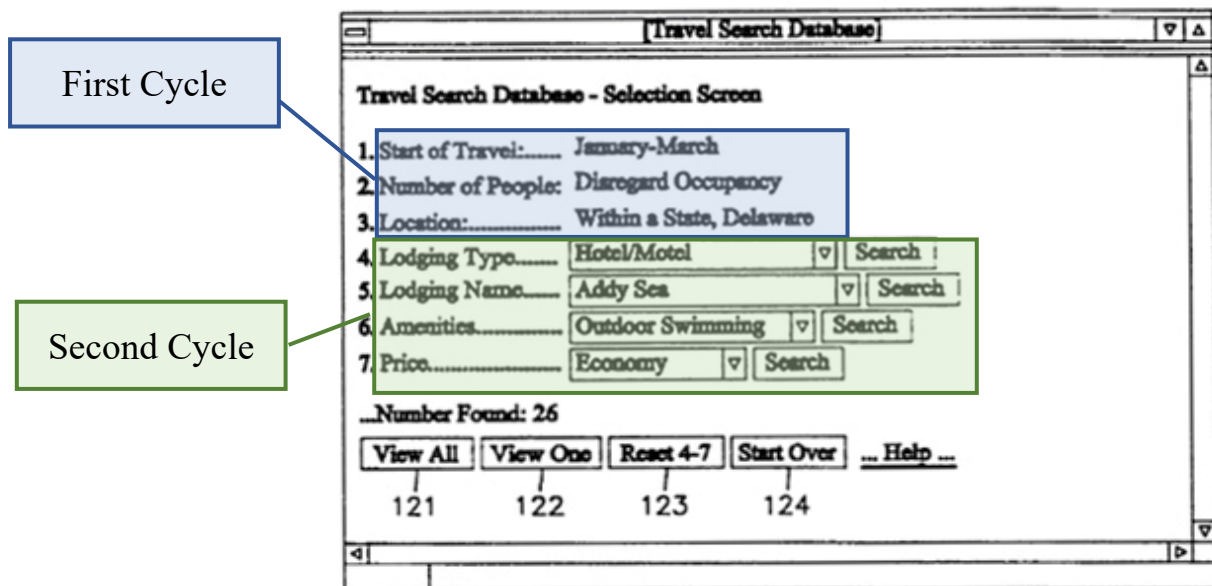


Id., Figure 9c (annotated).

260. Figure 9c above shows that as a part of this second search cycle, the display of the search terms in the pick box related to “Lodging Name” (in blue). *See* Ex. 1005 (Cochran), 16:66-17:13. In the same way described above (for Figures 8a, 8e, 9a, and 9c), a user may follow the same process using the pick boxes associated with, *e.g.*, “Amenities” and “Lodging Type,” instead of the pick box for “Price,” to act as the second search cycle; the user may also choose to perform additional search cycles, which includes displaying data in any pick boxes for which a user has yet to make a selection. *See id.*, 16:39-46 (“After looking at the search terms available in the four list identifiers for Boston, Mass. and deciding to narrow the selection so that the subset of selected records more closely matches the user’s needs. The user may use any list.”).

261. A user may also select several search terms, where that set of selections causes other pick box search terms to be updated (“*each display of data comprises a search cycle*”). Ex. 1005 (Cochran), 8:45-65.

262. The discussion immediately below, in conjunction with the annotated reproduction of Figure 5b, illustrates an example of the selection of several search terms constituting a “*search cycle*”:

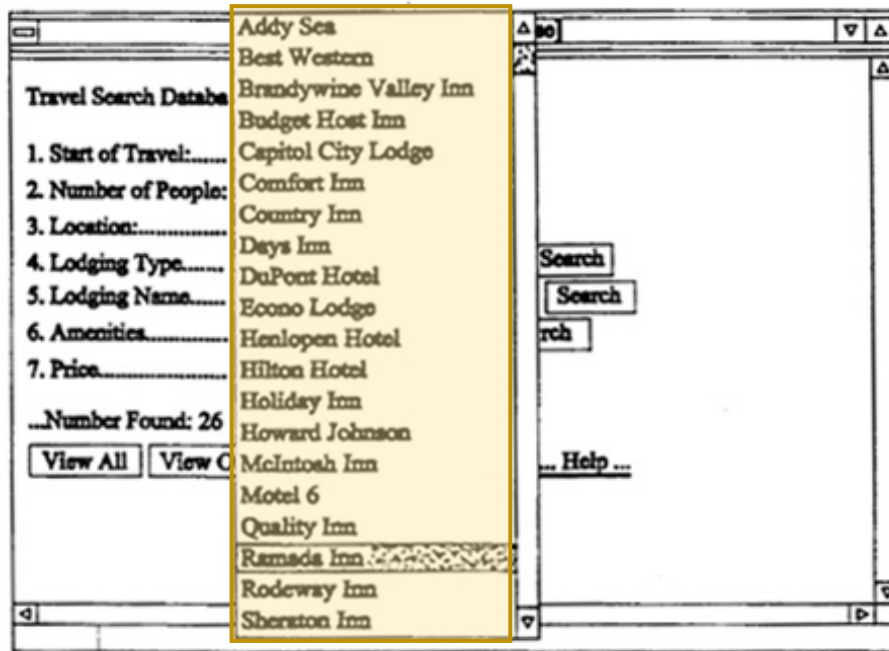


Ex. 1005 (Cochran), Figure 5b (annotated), 14:45-58.

263. In the annotated version of Figure 5b above, the user performs two cycles where the second cycle (in green) includes the display of the data for the search terms related to the Lodging Type, Lodging Name, Amenities, and Price list item database fields. Ex. 1005 (Cochran), Abstract (“The search terms contained in the lists are only those terms that are actually available in the database. A search of

the database is conducted based upon selected search terms from at least one of the multiple lists. The subset of records identified by the search is used to form new lists of search terms that are then displayed on the display device.”); 9:18-28 (“the user may refine the search and reduce the subset of records to a smaller set by returning to step 14. At step 14, the user either accepts the new default search terms and conducts a new search 24 or views the additional lists 16 and selects additional search terms 18.”).

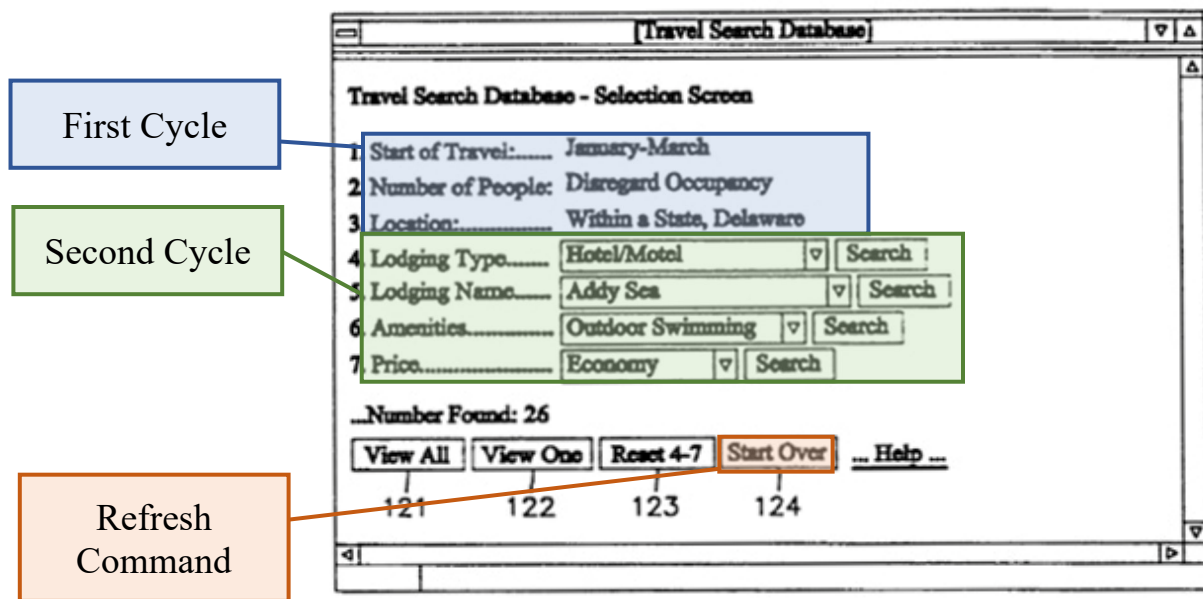
264. The user selected “January-March,” for “Start of Travel,” “Disregard Occupancy” for “Number of People,” and “Within a Region, Atlantic States: DE, DC, MD, PA, NJ, VA, WV” for “Location” in the first cycle (in blue), and then received the terms for the pick boxes associated with “Lodging Type,” “Lodging Name,” “Amenities,” and “Price” (in green) for the second cycle. Ex. 1005 (Cochran), 8:54-65 (“For instance, in the current example, after a search has been conducted based on search terms associated with list identifiers ‘Start of Travel’, ‘Number of People’, and ‘Location’, updated lists of search terms would then be generated for each of the remaining list identifiers, ‘Lodging Type’, ‘Lodging Name’, ‘Amenities’ and ‘Price’.”).



Id., Figure 5d (annotated), 14:50-15:11.

265. In the annotated version of Figure 5d above, the user displays the data for “Lodging Name” in the corresponding pick box (in yellow) as a part of the second cycle.

266. Cochran also discloses that its Selection Screen has a “Start Over” button. When the user clicks on the “Start Over” button, a command will be issued which causes Cochran to reset all pick box search terms selections, which constitutes “receiving a refresh command, the refresh command resetting all search cycles.” Ex. 1005 (Cochran), 13:19-26 (“The Start Over button 124 will reset the system back to the beginning display of FIG. 3a by ... returning to the initial step 12.”), Figures 5b, 3a, and 2.



Id., Figure 5b (annotated).

267. Figure 5b above shows a button “Start Over” 124 (orange) that resets the first (blue) and second (green) cycles. A POSITA would have recognized the convenience of being able to reset a multi-step search to be able to easily start all over. An inconvenient alternative would be to go to each search term or cycle, click on the drop-down menu, and select a new term.

268. Of course, this applies equally to the discussion above regarding Figures 8a, 8e, 9a, and 9c and the situation where selection of terms from and display of data in individual pick boxes each constitute a search cycle because the “Start Over” button “will reset the system back to the beginning display.” Ex. 1005 (Cochran), 13:19-26; *see also id.*, 8:9-14.

269. In sum, Cochran discloses “*wherein each display of data comprises a search cycle, and further comprising receiving a refresh command, the refresh command resetting all search cycles*” because each display and selection of a search term (or set of search terms) constitutes a search cycle, a user may select options from several sets of pick boxes, and the user may activate the “Start Over” button to reset all pick boxes to their original state.

J. Claim 17

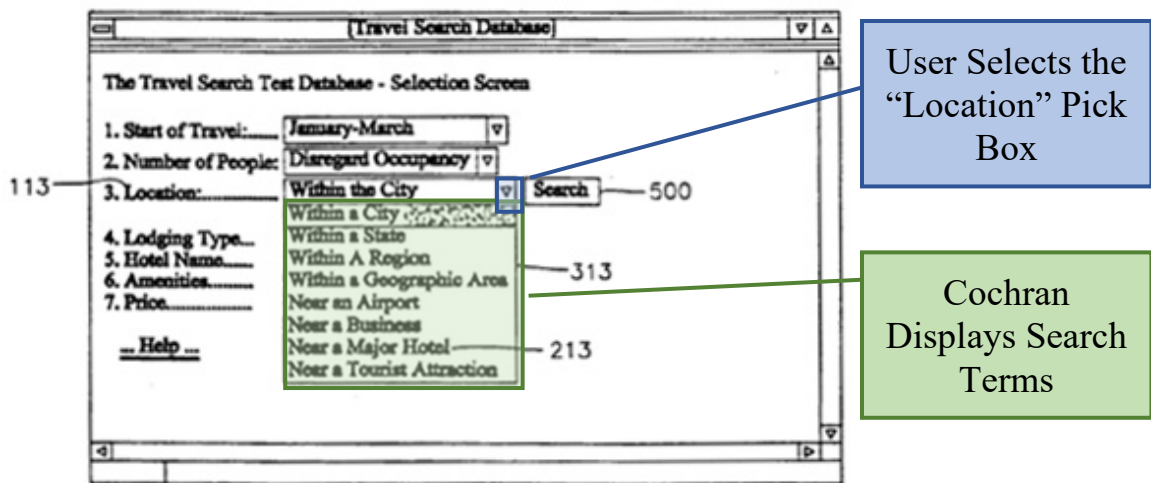
- 1. “The method [of] claim 3, further comprising displaying the list of data fields and a representation of the selected data field, wherein the representation of the selected data field comprises one or more elements from the selected data field.”**

270. As previously explained, Cochran displays a set of list identifiers and pick boxes (which constitutes “*displaying the list of data fields and a representation of the selected data field*”) where the pick boxes contain search terms for the data field (which constitutes “*wherein the representation of the selected data field comprises one or more elements from the selected data field*”). See §§XII.A.4 (Claim [1.C]), XII.A.5 (Claim [1.D]), and XII.C.2 above (Claim [3.A]).

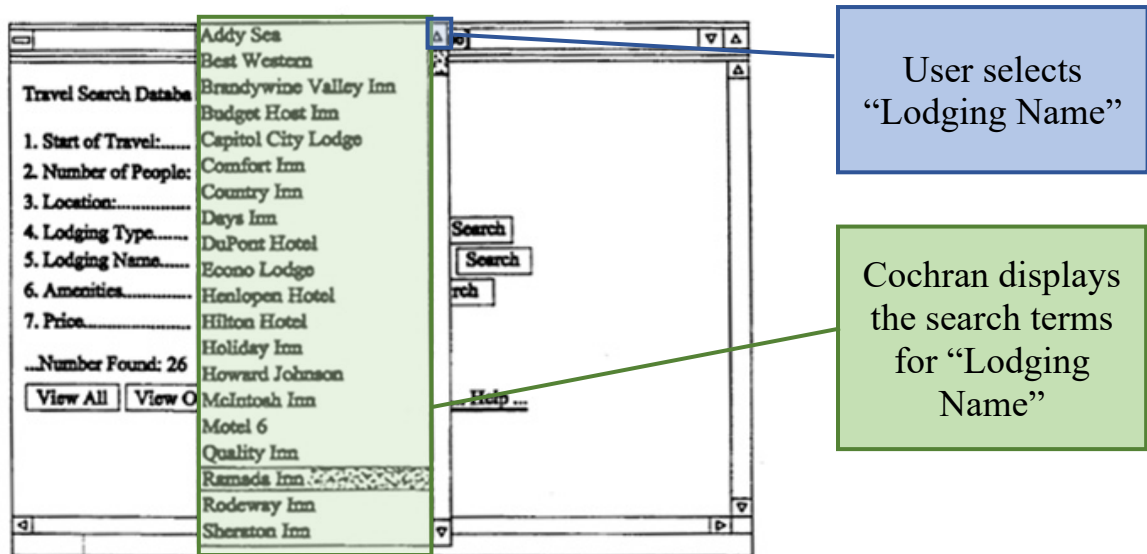
271. A user may select one of the listed items by clicking on the pick box’s drop-down button, which results in displaying a list of search terms associated with the listed item (from the data field) in a drop-down menu. Ex. 1005 (Cochran), 7:36-47 (“If the user selects to display a list at step 14, multiple search terms associated

with the list identifier are read from the computer's memory 717 and all are displayed at step 16."), 10:13-19 ("Pick boxes are standard elements of modern windowing technology. Pick boxes are spaces on the display that can be enlarged to display additional information and then 'closed' to their original size and/or position."), 10:43-11:65 ("The pick boxes 311-313 may be selectively expanded by clicking on an expand field icon 411-413, respectively, with a mouse-controlled cursor or other selection device (not shown). ... Activation of the expand button 411-413 will open the associated pick box 311-313 and reveal a list of previously concealed search terms (step 16).").

272. A POSITA would have understood that the list of search terms is a "*representation of the selected data field*" comprising "*one or more elements from the selected data field*" because it lists one or more elements from the given data field. This is shown in the figures below for "Location" and "Lodging Name":



Ex. 1005 (Cochran), Figure 3d (annotated).



Id., Figure 5d (annotated).

273. Figures 3d and 5d above show an example of expanding the pick boxes to display search terms/elements (in green) from data fields corresponding to "Location" and "Lodging Name" (in blue).

274. In sum, Cochran discloses “*displaying the list of data fields and a representation of the selected data field*” because Cochran displays a screen containing list identifiers, each with pick boxes, and discloses “*wherein the representation of the selected data field comprises one or more elements from the selected data field*” because Cochran displays a list of search terms in the pick boxes for the selected data field.

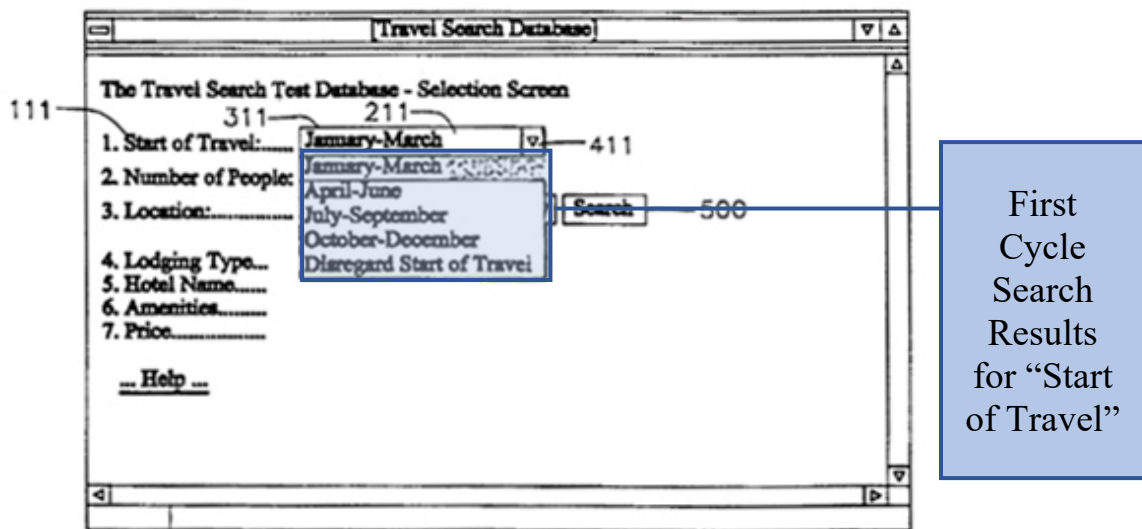
K. Claim 18

- 1. “The method of claim 3, wherein the method comprises executing a plurality of search cycles, each search cycle of the plurality of search cycles providing a search result list comprising selected data elements from the data base.”**

275. Cochran allows a user to select from multiple pick boxes (“*executing a plurality of search cycles*”), with each pick box showing a list of search terms. The search terms are a search result list of the data in the database associated with the data field for which the pick box applies (*e.g.*, the pick box for the Lodging Type data field is expanded in Figure 4d) (“*providing a search result list comprising selected data elements from the data base*”). Ex. 1005 (Cochran), Abstract (“Multiple lists of search terms, such as words, symbols or phrases, are displayed adjacent to list-identifiers (also referred to as index names or indexes). The search terms contained in the lists are only those terms that are actually available in the database. A search of the database is conducted based upon selected search terms

from at least one of the multiple lists. The subset of records identified by the search is used to form new lists of search terms that are then displayed on the display device.”), 4:29-35 (“Only choices or search terms that are actually available are presented to the user. ... Only search terms that will return a ‘hit’ are presented on the display device.”); *see also id.* 8:45-65, 13:50-57, 14:64-66, Figure 4d; §XII.I above (Claim 16).

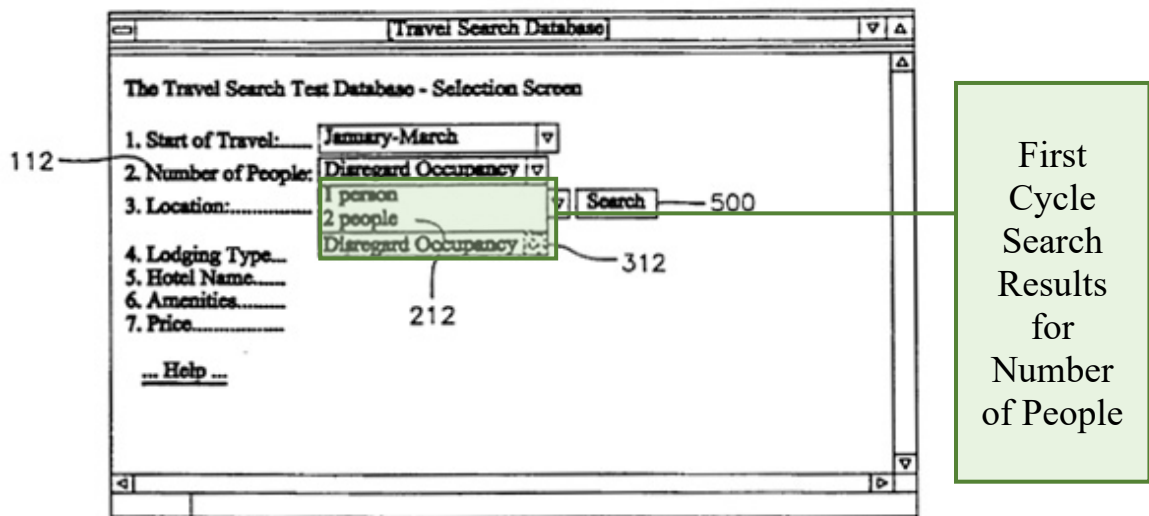
276. Cochran discloses that each selection (*i.e.*, expansion) of a pick box causes Cochran to display the search terms (*i.e.*, a cycle’s search results) in the pick box drop-down for each cycle. This is exemplified in the series of figures below, that show two (“*a plurality of*”) search cycles:



Ex. 1005 (Cochran), Figure 3b (annotated), 10:66-11:16.

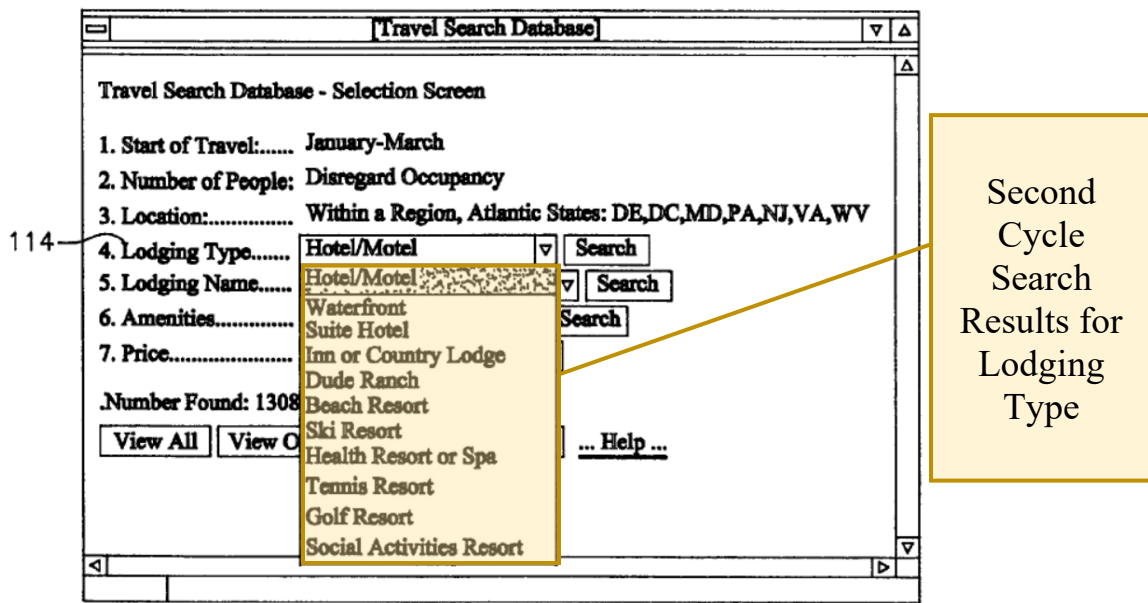
277. In the first search cycle, Cochran received the pick box search terms for “Start of Travel,” “Number of People” and “Location.” Figure 3b above shows that

the results of the first search pertaining to the “Start of Travel” data field are presented in the pick box with list identifier “Start of Travel” (in blue) which lists the search terms “January-March,” “April-June,” “July-September,” “October-December,” and “Disregard Start of Travel.”



Ex. 1005 (Cochran), Figure 3c (annotated), 11:17-21.

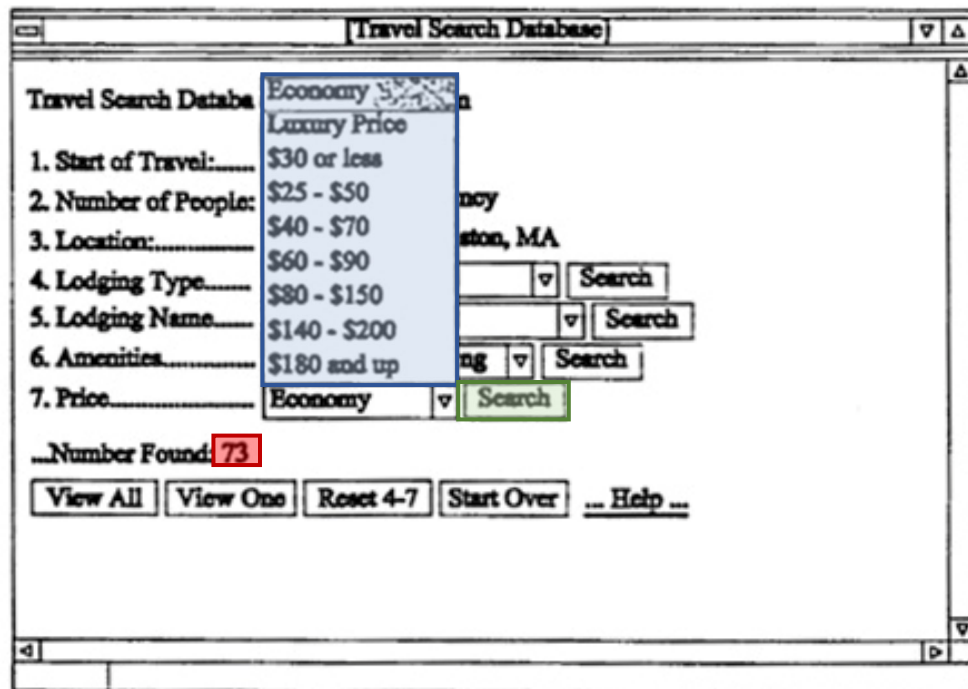
278. Figure 3c above shows the first cycle search results for the “Number of People” pick box (in green), which has search terms “1 person,” “2 people,” and “Disregard Occupancy.” Similarly, Cochran discloses receiving the entries for Location in the first cycle as well. See Ex. 1005 (Cochran), Figures 3d, 4a, 14:8-20.



Id., Figure 4d (annotated), 12:27-43, 13:50-57, 14:13-20.

279. Figure 4d above shows that, after the user has made selections for “Start of Travel,” “Number of People,” and “Location,” Cochran performs another search cycle to retrieve the search terms for the remaining pick boxes (“Lodging Type,” “Lodging Name,” “Amenities,” and “Price”). Specifically, Cochran discloses that “in the current example, after a search has been conducted based on search terms associated with list identifiers ‘Start of Travel’, ‘Number of People’, and ‘Location’, updated lists of search terms would then be generated for each of the remaining list identifiers, ‘Lodging Type’, ‘Lodging Name’, ‘Amenities’ and ‘Price’.” Ex. 1005 (Cochran), 8:54-9:43. This second search constitutes the “second cycle,” and Figure 4d above shows the second cycle search results pertaining to Lodging Type in yellow.

280. The above discussion in conjunction with Figures 3b, 3c, and 4d is presented only as an illustration of the general process. For example, the user may perform further search cycles by choosing other search terms to “refine the search and reduce the subset of records” and cause “newly generated lists [to be] transmitted back to the user’s computer terminal.” Ex. 1005 (Cochran), 9:10-23. Specifically, the user may “activate[] the associated search button” beside a pick box and then the “server 710 will search the subset of records.” *Id.*, 14:21-26. This is exemplified below using the search terms in the pick box associated with the data field “Price.” Furthermore, as I explained regarding Claim 16 in this Ground, a search cycle may constitute the selection and display of search terms in individual pick boxes. *See* §XII.I above (Claim 16).



Ex. 1005 (Cochran), Figure 8e (annotated), 16:22-27.

281. Figure 8e above shows in a search cycle after the first search cycle the user expanding the “Price” pick box and viewing the variety of applicable search terms (in blue). Specifically, Cochran states that the prices shown are the “list of search terms available for Price in Boston, Mass.” Ex. 1005 (Cochran), 16:22-27. That is, the search terms in Price are results from a second search cycle conducted after search terms from the pick boxes associated with “Start of Travel,” “Number of People,” and “Location” have been selected.

282. The figure shown below results after the user selects “Economy” from the list in blue and clicks on the “search” button highlighted in green in Figure 8e above.

The screenshot shows a web browser window titled "[Travel Search Database]". The main content area is titled "Travel Search Database - Selection Screen". It contains a list of search criteria with their current values:

- 1. Start of Travel:..... January-March
- 2. Number of People: Disregard Occupancy
- 3. Location:..... Within a City, Boston, MA
- 4. Lodging Type:..... Hotel/Motel [dropdown arrow] [Search]
- 5. Lodging Name:..... Comfort Inn [dropdown arrow] [Search]
- 6. Amenities:..... Outdoor Swimming [dropdown arrow] [Search]
- 7. Price:..... Economy [dropdown arrow]

Below the criteria, it says "...Number Found: 13", where "13" is highlighted in red. At the bottom, there are five buttons: "View All", "View One", "Reset 4-7", "Start Over", and "... Help ...".

Ex. 1005 (Cochran), Figure 9a (annotated), 16:39-50.

283. Figure 9a above shows after the user has selected “Economy” (in blue) for “Price” and clicked on the “Search” button (in green in annotated Figure 8e above), which performs a subsequent (*e.g.*, third) search cycle. A user may do similar steps to perform subsequent search cycles for “Lodging Type,” “Lodging Name,” and “Amenities.” Ex. 1005 (Cochran), 16:47-55. That the “Number Found” reduced from 73 in Figure 8e (in red) to 13 in Figure 9a (in red) confirms that this additional, third, (narrowing) search cycle occurred.

284. The search terms displayed in the pick boxes represent each cycle’s “*search result list*.” See, *e.g.*, Ex. 1005 (Cochran), Abstract, 4:20-50, 8:15-9:22, 12:66-13:13; 15:53-17:13. Specifically, Cochran states that “[t]he search terms

contained in the lists are only those terms that are actually available in the database.”

Id., Abstract. “Each search request will result in a subset of records” that are marked at the server and then subsequent “[s]earch terms derived from the subset of records marked at the server [] are then transmitted to ... the computer terminals,” *id.*, 6:40-52, which would then be presented as the subsequent cycle search terms in the pick boxes.

285. A POSITA would have understood this to mean that as the user selects a search term from the pick box during each search cycle, a search is conducted with the selected search terms to arrive at a result set of still-applicable search terms for the remaining pick boxes, to ensure that the user is still only selecting, in each cycle, records that are applicable given the (progressively narrowing) set of search terms already selected by the user. *See e.g.*, Ex. 1005 (Cochran), 14:50-58, 16:9-14, 16:56-65.

286. In sum, Cochran discloses “*executing a plurality of search cycles, each search cycle of the plurality of search cycles providing a search result list comprising selected data elements from the data base*” because when users of Cochran select search terms from pick boxes, they cause Cochran to search for search terms for other pick boxes, and return the search terms for display in other pick boxes.

L. Claim 19

1. “The method of claim 18, further comprising resetting one or more search cycles of the plurality of search cycles.”

287. As an initial matter, I note that “resetting one or more search cycles” includes resetting all search cycles.

288. As I just explained above for claim 18, Cochran discloses a multi-cycle search process where a user selects search terms for data elements. §XII.K above (Claim 18).

289. Cochran further discloses that when a user clicks “Reset 4-7,” Cochran “will clear any search terms previously selected for list identifiers 114-117 [(“Lodging Type,” “Lodging Name,” “Amenities,” and “Price”)] and replace them with default terms.” Ex. 1005 (Cochran), 13:14-18. A POSITA would have understood this to mean resetting some, but not all of the search terms.

290. Moreover, when a user presses “Start Over,” Cochran will “restart the selection process from the beginning” by “reset[ting] the system back to the beginning display ... and returning to the initial step.” Ex. 1005 (Cochran), 13:19-26. A POSITA would have understood that by pressing the “Start Over” button, the user can reset all search cycles which is also “one or more” of the search cycles.

291. In sum, Cochran discloses that the user can “reset all or part of the search” using the “Reset 4-7” or “Start Over” buttons, each of which constitutes “*resetting one or more search cycles of the plurality of search cycles.*”

M. Claim 20

1. [20.A] “The method of claim 3 further comprising: determining a schema of the database;”

292. Cochran determines a schema of the database for the same reasons I expressed for Claim [1.A]. *See* §XII.A.2 above (Claim [1.A]).

2. [20.B] “comparing the determined schema to a capability of a terminal for displaying the data; and”

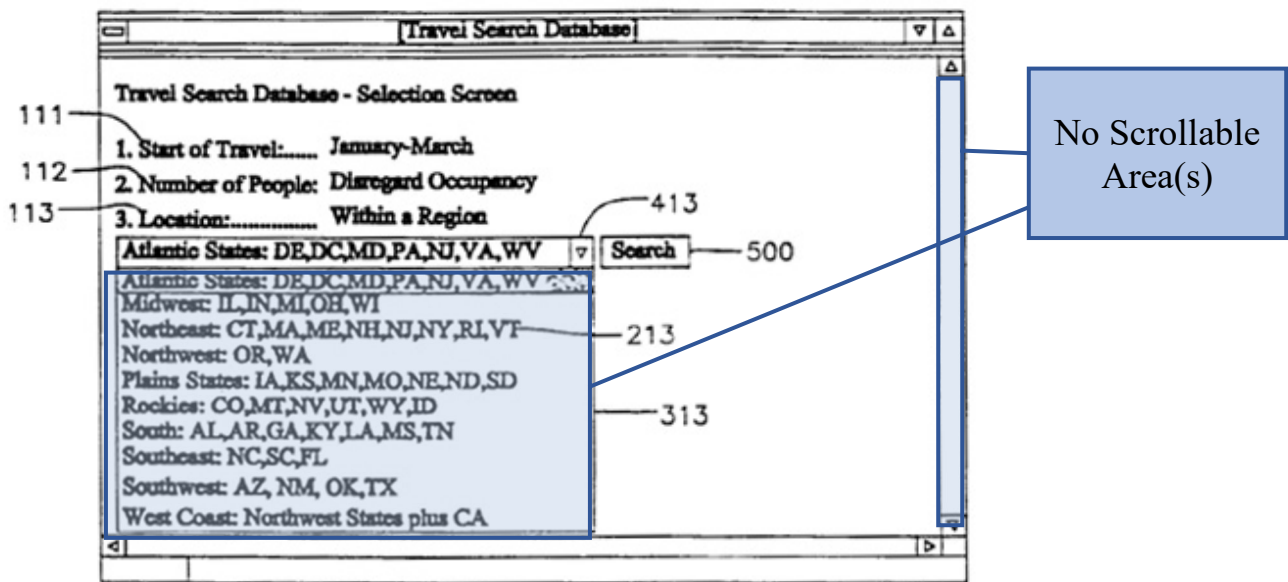
293. As an initial matter, a POSITA would have known that every display or terminal device has specified dimensions (*e.g.*, number of vertical and horizontal characters or pixels), and that operating systems include several mechanisms for users and applications to find those dimensions and act accordingly to display information that fits therein. *See* §VII.B above.

294. I also note that this limitation is nonsensical, effectively comparing “apples to oranges.” A terminal’s capability of displaying is denoted by attributes such as its vertical and horizontal dimensions (*i.e.*, numbers). But a schema is a high-level description of a database’s structure (*i.e.*, meta-data) and is not a number that can be compared. *See* §VII.C above. A database may have an arbitrary number of data fields (*e.g.*, columns in a table) across one or more tables. The number of

data fields *is* a number that can be compared. Therefore, at this time, I am assuming that the number of data fields is what gets compared to the terminal's capability for displaying the data.

295. I previously described how Cochran's Selection Screen displays a list of labels and their accompanying pick boxes, which constitutes "*the determined schema*." See §§XII.A.4 (Claim [1.C]), XII.A.5 (Claim [1.D]), and XII.C.6 above (Claim [3.E]). Cochran discloses that it is meant to be used on "a computer terminal or other display device." Ex. 1005 (Cochran), Abstract, 4:10-15, 6:11-12, Figure 1. Cochran also discloses that it may be used on various "computer terminal[s]," each with its own "display device" that needs to be able to display the labels and pick boxes. *Id.*, 6:11-12, Figure 1. Thus, a POSITA would have understood or at least found obvious that the system of Cochran would accommodate various display capabilities of an arbitrary "computer terminal or other display device."

296. To make this accommodation, Cochran would need a way to figure out what to do when there are too many search terms such that it cannot display them all on a single page. Cochran's figures show that the Selection Screen may allow the user to scroll but that when a terminal's display device can fit all labels (fields) and pick boxes on one page, Cochran displays a bar with no scrollable area:



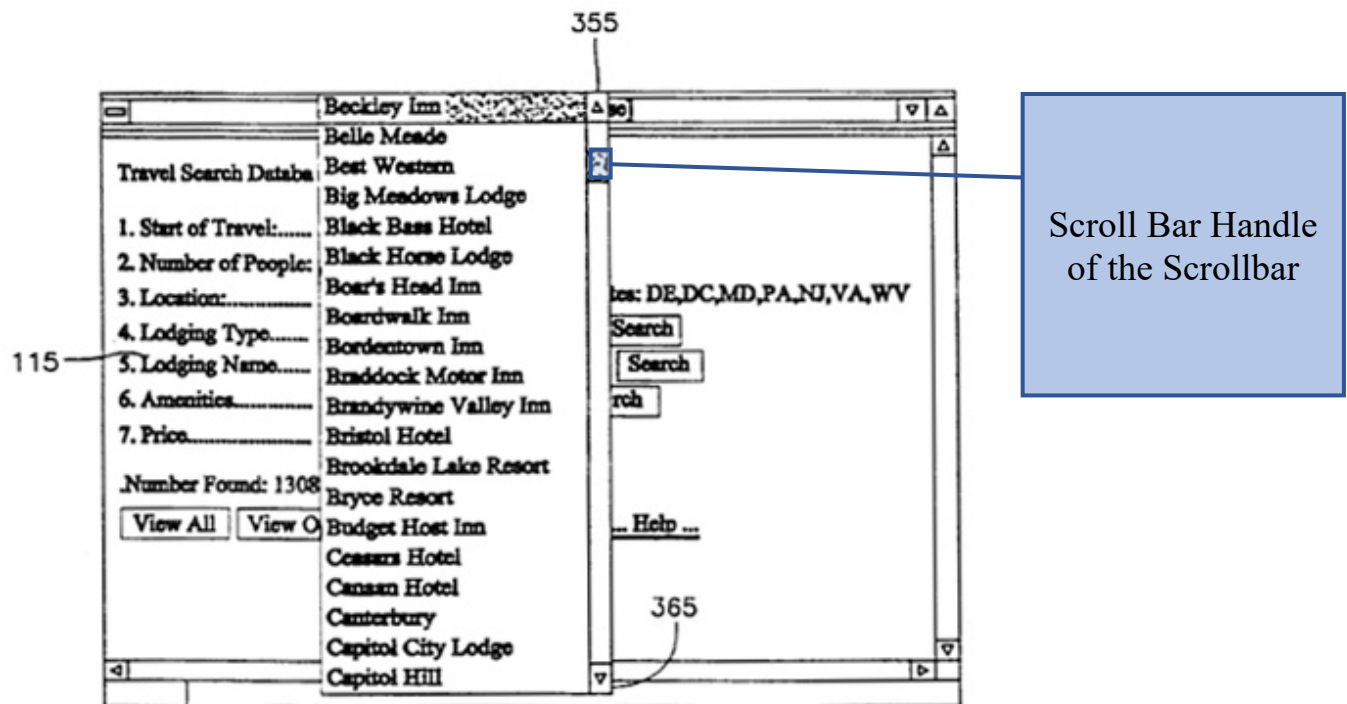
Ex. 1005 (Cochran), Figure 4a (annotated).

297. Figure 4a above shows a solid white bar without a handle (in blue on the far right) sandwiched between two scroll buttons, which indicates that Cochran determined that there is no need to allow the user to scroll (*i.e.*, all the list identifiers and pick boxes fit on a single screen). The figure also shows (in blue on the lower-left side) a short list of Location items, again without a scroll bar, because this list is short enough that a scroll bar is unnecessary.

298. It would have been obvious to a POSITA that when Cochran cannot display all list identifiers and pick boxes on one page simultaneously, Cochran would display a scroll bar including a scrollable area and scroll bar handle (also known as a “scroll thumb”). This is consistent with how Cochran handles pick boxes

with so many search terms they cannot all be presented simultaneously, *see* §XII.A.5 above (Claim [1.D]); *see also* §VII.E above.

299. A POSITA would have reasonably expected and found it obvious that Cochran would handle the analogous circumstance involving list identifiers and pick boxes in a similar fashion:



Ex. 1005 (Cochran), Figure 4e (annotated).

300. Figure 4e above shows that when Cochran determines that there are too many search terms to fit in the display for an expanded pick box, Cochran provides a scrollbar with a scrollbar handle (in blue). A POSITA would have found it obvious to implement a similar functionality in the analogous circumstance involving list

identifiers and pick boxes (*i.e.*, when the number of fields/identifiers/pickboxes exceeds the number that can be displayed on a single screen).

301. A POSITA would understand that this would be accomplished automatically and only if Cochran is able to “*compare*” how many list identifiers (fields) and pick boxes there are (*i.e.*, how many database fields there are) (“*the determined schema*”), and how many records are retrieved for each field, to how much content can fit on the screen of each “computer terminal or other display device” simultaneously (“*to a capability of a terminal for displaying the data*”). Without being able to ascertain these values and compare them, Cochran would not be able to determine whether to include a scroll bar for the user to scroll through more items than can fit on the given terminal or display. *See also* §VII.E above.

302. In sum, Cochran discloses “*comparing the determined schema to a capability of a terminal for displaying the data*” because Cochran provides a scrollbar on the screen displaying the list identifiers and pick boxes and that screen may or may not have a scrollable area depending on Cochran’s comparison of the number of list identifiers and pick boxes (“*the determined schema*”) and the capability of the terminal for displaying data on the screen.

3. [20.C] “formatting a search request from the terminal to correspond to the determined schema.”

303. I recognize that this claim limitation contains no reference to the comparison in Claim [20.B]. Thus, the formatting performed in this limitation need not rely on or make use of the results of the comparison done in Claim [20.B].

304. I also note that this limitation appears to merely claim what databases and SQL have been doing for many years before the alleged invention of the 720 Patent: formatting search requests such as SELECT queries, which of course must correspond to the database’s schema. A search request that does not correspond to a database’s scheme will likely fail or return an error message because one cannot search for categories of items that do not even exist. *See also* §VII.C above.

305. A POSITA would have understood that Cochran’s search requests must “correspond to the determined schema” because the names used for and organization of data fields is an aspect of the schema. Thus, if the search requests did not “correspond to the determined schema.” the search requests would not refer to the correct data fields in the database. *See* Ex. 1005 (Cochran), 1:21-29. Thus, Cochran must compose “search requests” (“*formatting a search request*”) in such a way that it “*correspond[s] to the determined schema.*” *Id.*, 6:40-52, 7:5-8:65; *see also* §§XII.A.2-XII.A.4 (Claims [1.A]-[1.C]) and XII.M.1-XII.M.2 above (Claims [20.A]-[20.B]).

306. In sum, Cochran discloses “*formatting a search request from the terminal to correspond to the determined schema*” because otherwise Cochran would not be able to query the database and retrieve the “search terms” to populate the pick boxes.

N. Claim 27

- 1. “The method of claim 3, further comprising displaying the data on a terminal, the terminal including one of a handheld device, a cellular phone, a geosynchronous positioning satellite (GPS) device, a wrist-worn device, an interactive phone device, a household appliance, a television, a television set top box, a handheld computer, a main frame computer and a personal computer.”**

307. Cochran displays graphical user interfaces, which includes the Selection Screen interface that includes the list identifiers and search terms (“*displaying the data*”), on a “computer terminal” that has a “display device” which constitutes at least a “*a handheld computer*” or “*a personal computer.*” *See, e.g.,* Ex. 1005 (Cochran), Title (“Apparatus and method for selecting records from a computer database by repeatedly displaying search terms from multiple list identifiers before either a list identifier or a search term is selected”), Abstract (“A method for managing the presentation of search terms on a computer terminal or other display device so that large computer databases can be searched efficiently. Multiple lists of search terms, such as words, symbols or phrases, are displayed adjacent to list-identifiers (also referred to as index names or indexes).”), 4:10-19

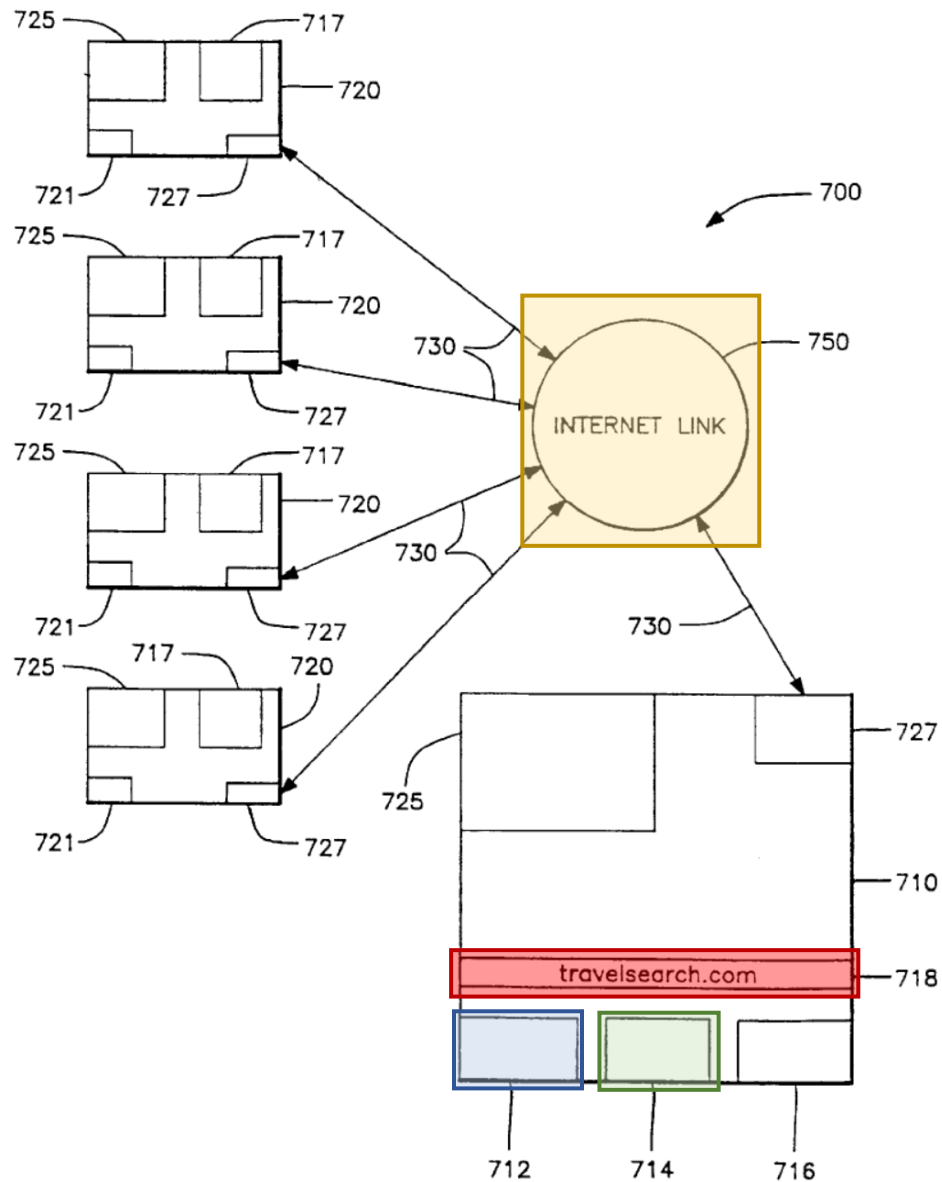
(“It is our object of the present invention to provide a search system that efficiently displays information to users that are searching for particular information in a database. The database source is ordinarily at a location different from the location of the display devices, which are computer screens associated with a computer terminal.”), 5:4-11 (“FIGS. 3a-3d illustrate a selection screen on a computer display device for selecting search terms used to search a database having travel information in accordance with a preferred embodiment of the present invention.”), 5:65-6:52 (“FIG. 1 shows a preferred overall implementation of the invention in a computer network 700 in accordance with the preferred embodiment. A plurality of computer terminals 720 are networked to a remotely located server computer 710 via a communications link 750, preferably the Internet. ... Each computer terminal 720 has memory space 717 and a monitor or display device 725.”); *see also id.*, 7:13-19, 9:60-67, 11:9-16, 12:44-53, 14:13-20, 14:45-49, 14:64-15:3, 15:17-27, 15:37-44, Claims 1, 2, 8.

308. A POSITA would have understood that “computer terminals” can at least include personal computers, handheld computers, desktops, servers, and mainframes. *See also* §VII.A above.

O. Claim 28

- 1. “The method of claim 3, wherein the database is coupled to an Internet website, the method further comprising accessing the Internet website.”**

309. Cochran shows that its system can be coupled to a server “accessed through a communications medium such as the Internet.” Ex. 1005 (Cochran), 5:39-49. Cochran’s explicitly discloses applying its teachings to a travel search database that is accessed through an Internet “Web Page addressed at ‘http://travelsearch.com.’” *Id.*, 6:31-64. Cochran includes a figure showing the “general configuration of a computer communications network used in accordance with the [] invention. *Id.*, Figure 1, 4:65-67; *see also id.*, 6:6-39, 8:45-47. The server hosts the website and database. *Id.*



Id., Figure 1.

310. Figure 1 above shows the general configuration of a computer communication network used in accordance with Cochran. Specifically, it shows server computer 710, database 712 (in blue), software code 714 (in green), and Web

Site address 718 (in red), and accessing these components via the communications link (INTERNET LINK) 750 (in yellow).

311. A POSITA would have understood that because the Cochran invention may be accessed via Internet websites, and because it stores its data in “database 712,” meaning that the Internet website and database are at least in communication with one another, the database and Internet website are therefore “*coupled*” together. See §§VII.A and VII.F above.

P. Claim 29

1. **“The method of claim 3, wherein the database is coupled to a node in a telecommunications network, the method further comprising accessing the node in the telecommunications network.”**

312. Cochran may be implemented in software on a server that also hosts a database (“*the database is coupled to a node in a telecommunications network*”) and is accessed via the Internet (“*accessing the node in the telecommunications network*”). See §XII.O above (Claim 28). Cochran discloses “server 710” (which hosts “database 712”) “includes a processor 729 for implementing ... the present invention” and “[c]omputer terminal 720 includes a processor 721 for implementing software ... and instructions received from server 710.” Ex. 1005 (Cochran), 6:6-10, 6:31-39, 8:45-47. Further, the “[c]omputers 720 are coupled to [a] communications link 750 through a bidirectional communication line 730, such as a

telephone line or an Integrated Services Digital Network (ISDN) connection.” *Id.*, 6:12-15; *see also id.*, 6:16-19, 6:20-39.

313. In sum, a POSITA would have understood that using a telephone communication line, ISDN, or the Internet are examples of a “*telecommunications network*,” and that Cochran’s webserver discloses a “*node in a telecommunications network*.” Furthermore, Cochran shows that the webserver is “*accessed*” over a telephone communication line, ISDN, or the Internet. §XII.O above (Claim 28). *See also* §VII.A above.

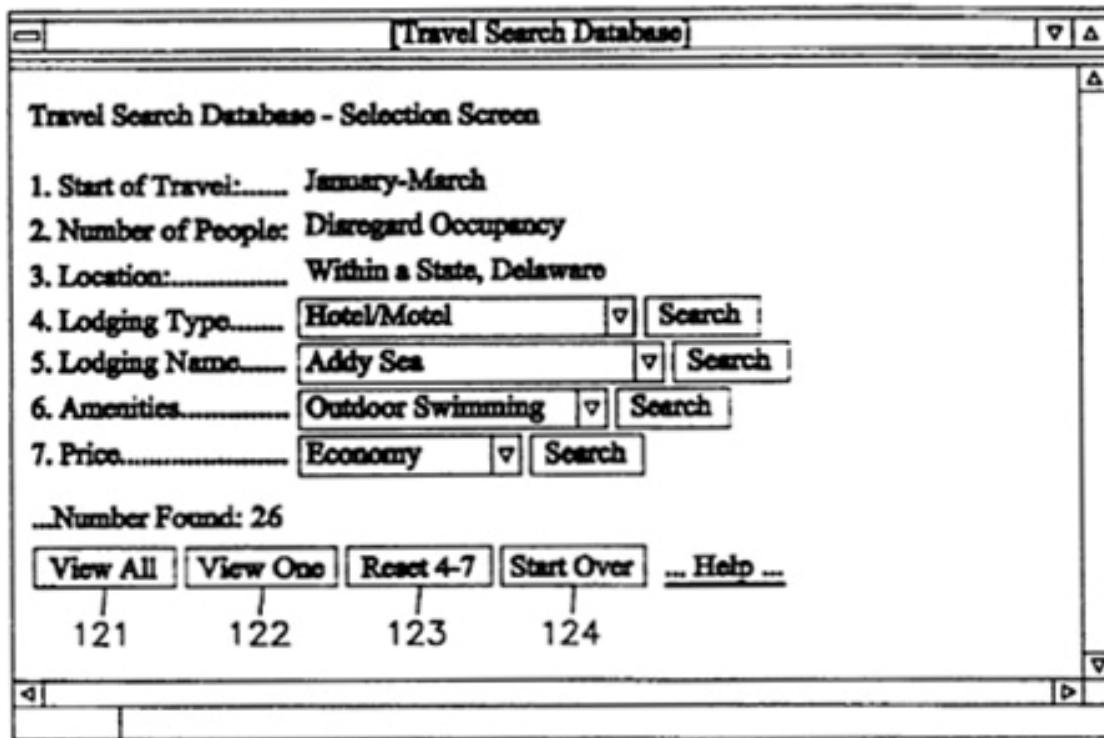
Q. Claim 30

1. “The method of claim 3, wherein the database is coupled to a terminal for displaying the data.”

314. Cochran discloses a method of accessing a database information through a graphical user interface on a web page displayed in a Web browser. Ex. 1005 (Cochran), Abstract (“A method for managing the presentation of search terms on a computer terminal or other display device so that large computer databases can be searched efficiently.”), 6:31-39 (“In the preferred embodiment of the invention, the database at server 710 is accessed via a World Wide Web page by a Web Browser.”). Cochran’s database is connected (“*coupled*”) to a computer terminal for displaying the data from the database (“*terminal for displaying the data*”). *See id.*, 6:49-52 (“Search terms derived from the subset of records marked

at server 710 are then transmitted to the memory 717 of the computer terminals 720 where the user initiated the search.”), 7:9-18 (“In response to a request by a user at remote computer terminal 720, the server 710 transmits information, at step 10, to the user’s computer 720. The request to start the program is made by dialing the address of the remote server 710. The information transmitted from server 710 to the user’s computer 720 includes a plurality of list identifiers and lists of search terms associated with those list identifiers.”); *see also id.*, 9:10-12, 12:66-13:13, 18:14-16, 19:12-20:4.

315. Cochran shows that the database data (*e.g.*, search results and the search terms in the pick boxes) is displayed in a graphical user interface on the computer terminal. Ex. 1005 (Cochran), Figures 3a-10b.



Id., Figure 5b (annotated).

316. The entirety of the screen shown in Figure 5b comprises the GUI shown on the terminal.

317. A POSITA would have understood that the “*database is coupled*” (connected) to the computer terminal because “the server 710 transmits information ... to the user’s computer 720,” Ex. 1005 (Cochran), 7:9-18, where the server hosts the database, and Cochran’s computer terminal is the “*terminal*.” *Id.*, 6:31-39. As discussed for Claim 28 (§XII.O above), databases are often coupled to web servers. A POSITA would have recognized that terminals such as graphical web browsers running on a device such as a computer are coupled to web servers,

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

and therefore such terminals are also coupled to databases (at least indirectly via web servers).

XIII. GROUND 2: HYPERSQL IN VIEW OF ZELLWEGER AND THE KNOWLEDGE OF A POSITA RENDERS OBVIOUS CLAIMS 1-5, 7, 15-19, AND 27-30

318. It is my opinion that a POSITA would have considered HyperSQL and Zellweger to be within the scope of the prior art and highly relevant to the patentability of the 720 Patent's claims.

319. It is my opinion that HyperSQL in view of Zellweger, especially when considering the knowledge of a POSITA at the time, renders obvious Claims 1-5, 7, 15-19, and 27-30. The basis for my opinion is set forth in more detail below.

320. A POSITA would have been motivated to combine the teachings of HyperSQL and Zellweger, which are analogous art, and would have had a reasonable expectation of success when doing so as detailed in §XVII.B below.

A. Claim 1

1. [1.PRE] “A method implemented on a computer for searching databases, comprising:”

321. HyperSQL discloses a system for creating “forms- and hypertext-based database interfaces for non computer experts” that “permit the user to query databases by filling out query forms selected from menus.” Ex. 1006 (HyperSQL), 329-30, 335-36, Figures 8, 9. HyperSQL discloses that its system operates on a Web server. *Id.*, Figures 8, 9.

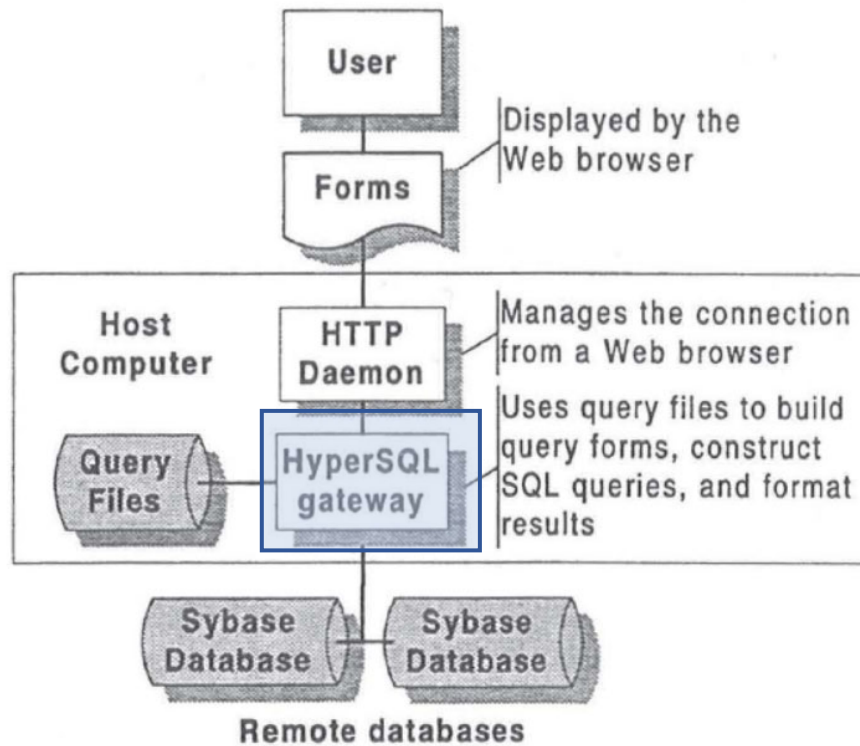


Figure 8. HyperSQL architecture

Id., Figure 8 (annotated).

322. Figure 8 above shows that HyperSQL (in blue) is software implemented and running on the “Host Computer.” Moreover, a POSITA would have known that a “daemon” and “gateway” refer to software. See §VII.A above.

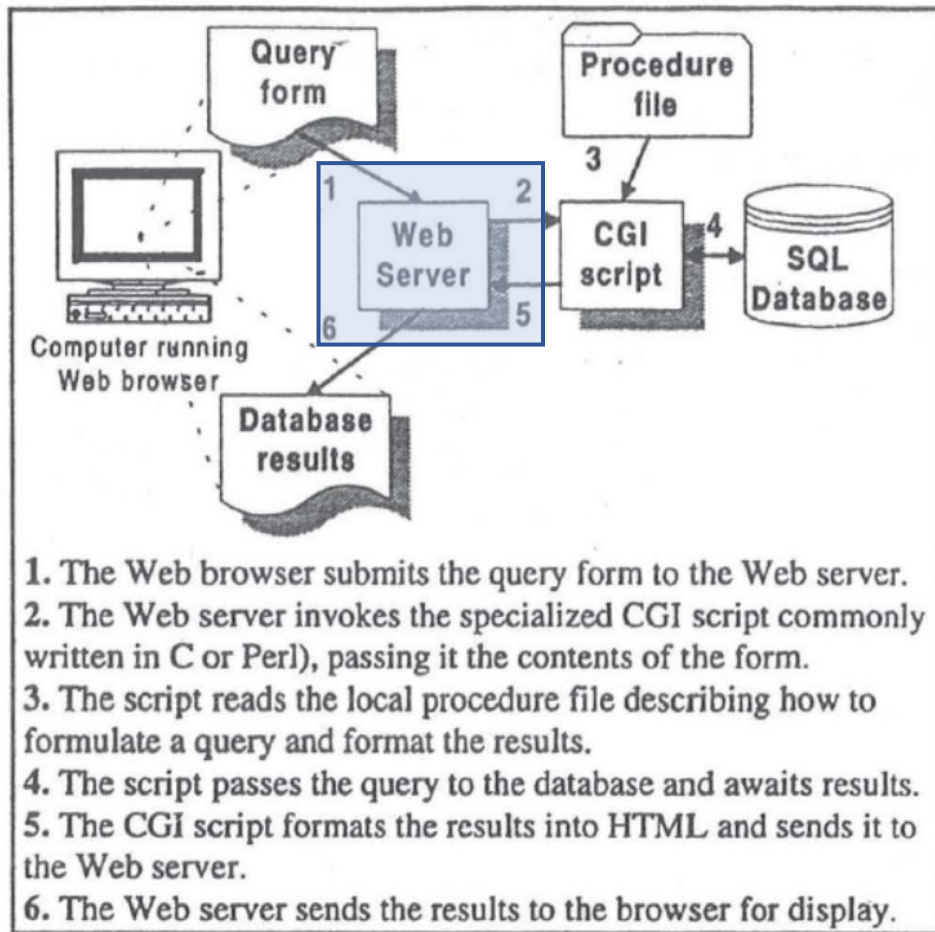


Figure 9. How CGI-based query interfaces work

Ex. 1006 (HyperSQL), Figure 9 (annotated).

323. Figure 9 above shows the Web Server—a computer—upon which HyperSQL is implemented and running.

2. [1.A] “determining a database schema for a database, wherein the database includes database fields;”

324. HyperSQL discloses that part of what makes it easy to use is that it eliminates “direct exposure to SQL and low-level database tools,” meaning that HyperSQL takes care of that work for the user. Ex. 1006 (HyperSQL), 330.

HyperSQL discloses that such tools may be used “for extracting the database schema from the database and automatically creating a working template form definition file” (“*determining a database schema for a database*”) such that “[n]o knowledge of SQL is required because the interface automatically generates SQL queries from the user’s input.” *Id.*, 337, 330.

325. A POSITA would have understood that the underlying databases include “database fields.” Indeed, HyperSQL contains several references to “fields” of a database: “the Keyword Search button brings up a form that accepts a text string to be located by searching all data fields,” Ex. 1006 (HyperSQL), 331; “HyperSQL also supports keyword search, making it possible to locate records that contain keyword(s) within any data field,” *id.*; “BROWSE: ... specifies formatting of a header, data fields, and footer,” *id.*, 335; “values of data fields returned from the database,” *id.*, 337; “An entity is a collection of fields from one or more tables, views, or other entities, grouped together in a logical set. Fields are permitted to contain links to other databases,” *id.*

326. A complete HyperSQL interface includes a section of the interface (a web page) called a “query form” which contains a “form body.” The “form body” “specifies what prompts and data entry fields” (*i.e.*, “*database fields*” in the “*database*”) “should be presented to the user.” Ex. 1006 (HyperSQL), 333. The form body may include a variety of types of data entry fields, including “scrollable

lists that are filled with information acquired automatically from the database” called “querylists,” as seen in the figure below:

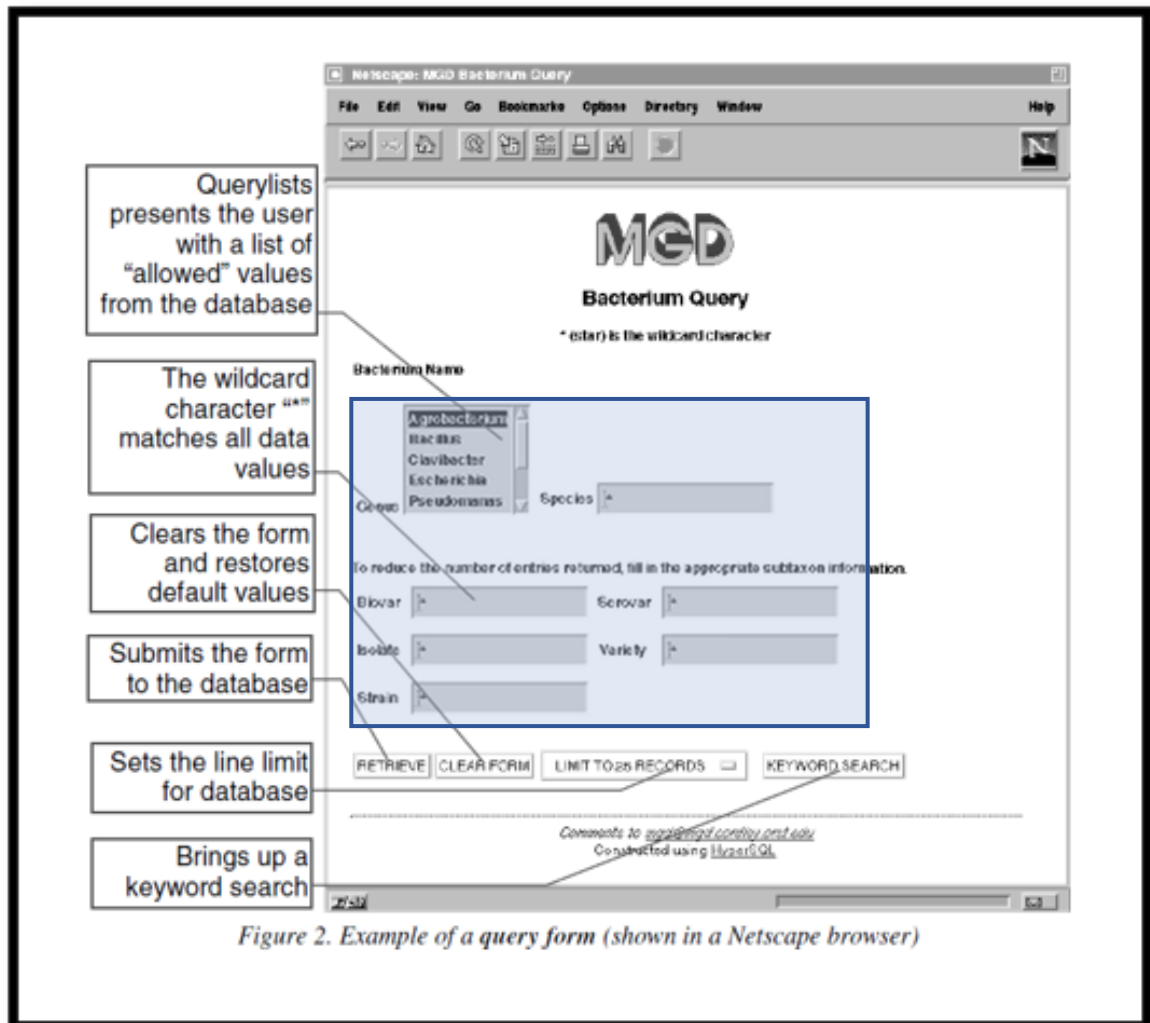


Figure 2. Example of a query form (shown in a Netscape browser)

Id., 331, Figure 2 (annotated).

327. Figure 2 is annotated to emphasize the set of prompts and data entry fields, highlighted in blue. Figure 2 shows seven separate sets and using only one querylist, for “Genus.”

328. A user may use the querylists to “fill in fields” (*i.e.*, database fields) “incrementally by selecting from [the] querylists,” where each querylist contains a “list of ‘allowed’ values from the database.” Ex. 1006 (HyperSQL), 331.

329. In sum, HyperSQL discloses “*determining a database schema for a database, wherein the database includes database fields*” because HyperSQL must determine how the data is organized and what database fields are in the database, to be able to present the user with a “query form.” Furthermore, HyperSQL discloses that it leverages tools for determining schemas. *See also* §VII.C above.

3. [1.B] “providing a list of the database fields, wherein the list includes a descriptor indicating a data category;”

330. HyperSQL discloses that the “form body section specifies what prompts [*“descriptors”*] and data entry fields [*“database fields”*] should be presented to the user.” Ex. 1006 (HyperSQL), 333. For example, Figure 2 shows an “[e]xample of a query form” that depicts this.

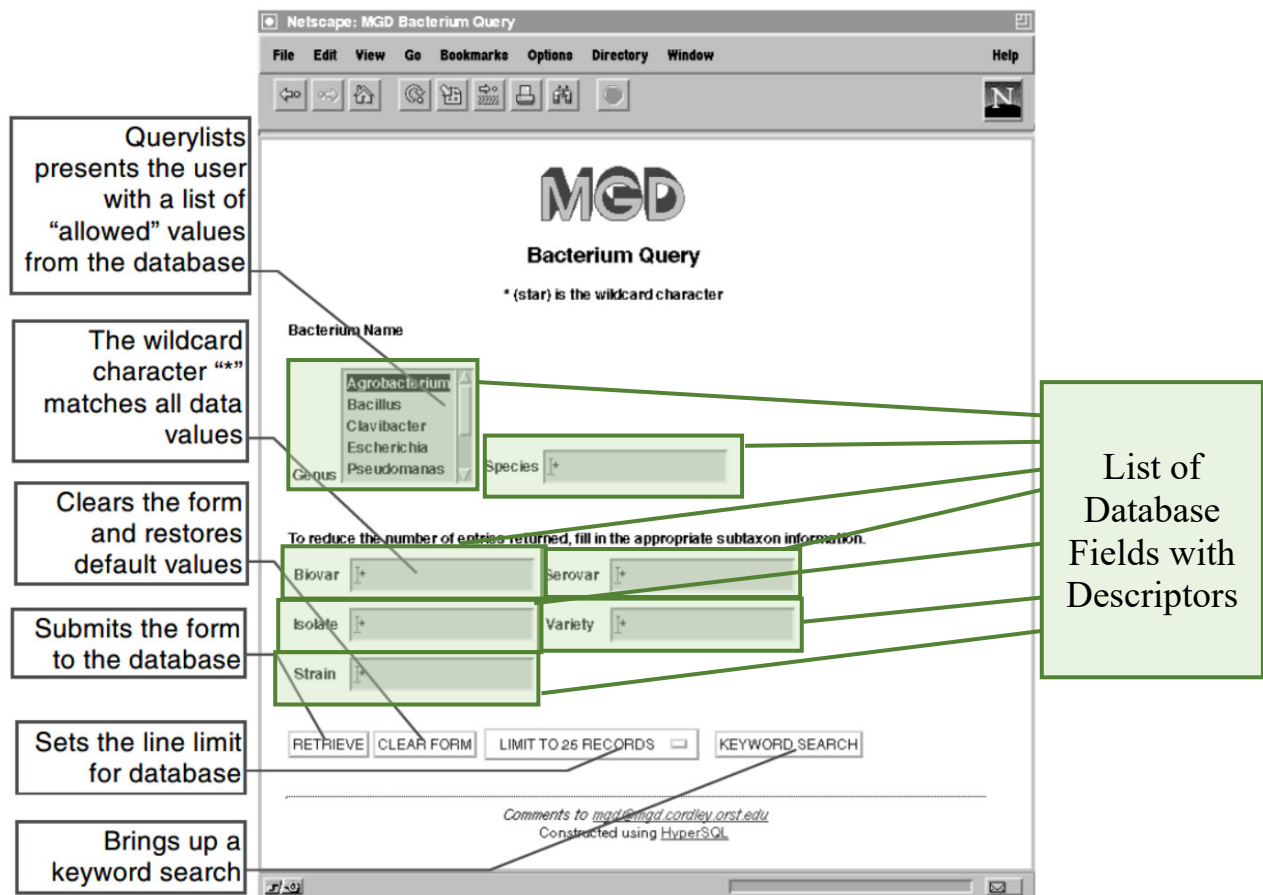


Figure 2. Example of a **query form** (shown in a Netscape browser)

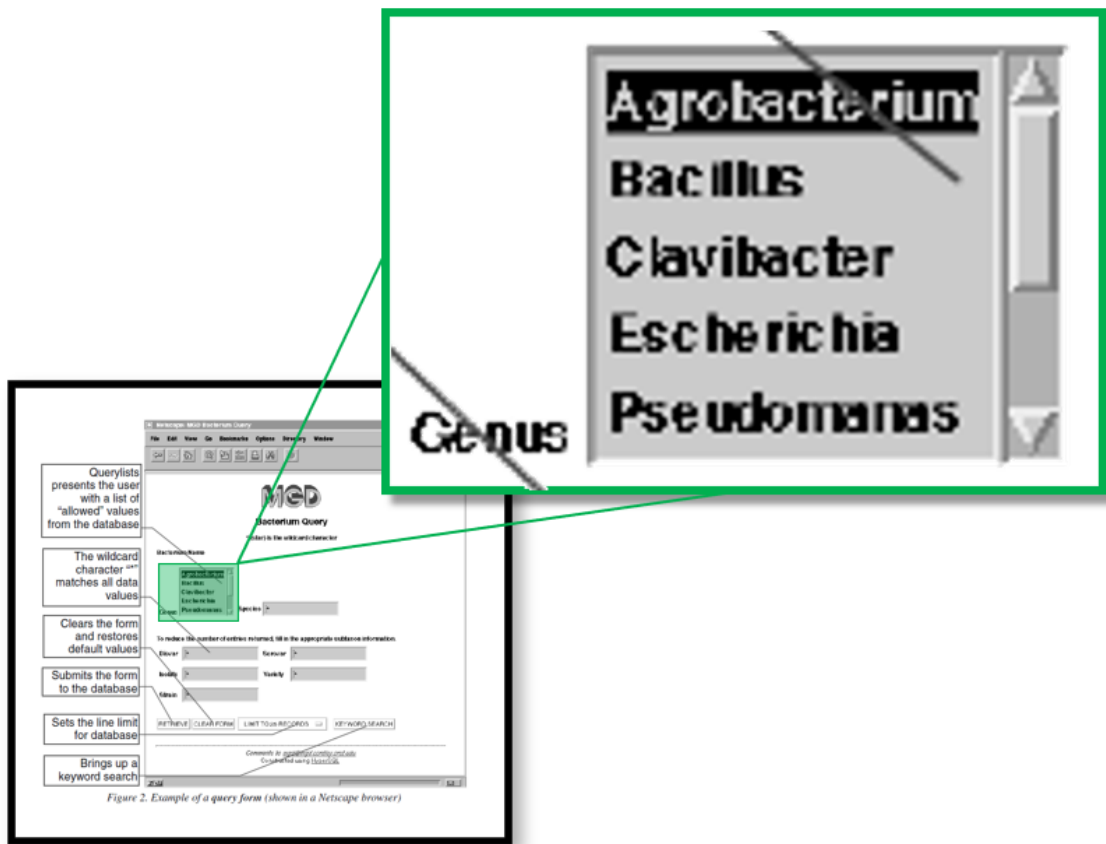
Id., Figure 2 (annotated); *see also id.*, Figure 5, 229, 331-32; §XIII.A.2 above (Claim [1.A]).

331. Specifically, the query form of Figure 2 above shows a list of seven separate sets of prompts and inputs (in green), each one relating to a database field. For each entry in the list, the prompt provides a “*descriptor indicating a data category*” because it *describes* to what data field the input is related. For example, we know that the first querylist is related to the Genus data category because the prompt beside the querylist reads “Genus.”

332. In sum, HyperSQL discloses “*providing a list of the database fields, wherein the list includes a descriptor indicating a data category*” because it presents the user with a list of prompts and inputs including querylists where the prompt is a descriptor indicating the data category.

4. [1.C] “receiving a search selection for a database field on the provided list of the database fields;”

333. HyperSQL users can select a database field by interacting with the field on the query form. This constitutes “*receiving a search selection for a database field on the provided list of database fields.*” Specifically, HyperSQL describes querylists as “scrollable lists that are filled with information acquired automatically from the database.” Ex. 1006 (HyperSQL), 333. This means that the entries in the querylists are the data from the database for the corresponding database field.



Id., Figure 2 (annotated and zoomed in); *see also id.*, Figure 5.

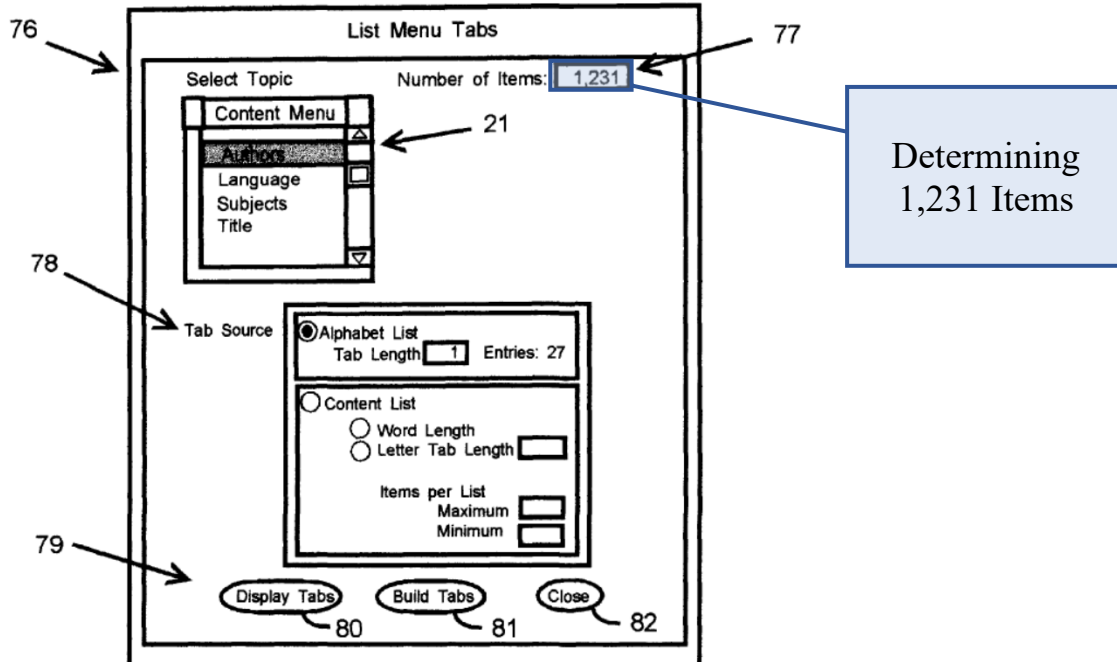
334. Figure 2, annotated above, shows that the querylist for “Genus” contains several values that a user can choose from including “Agrobacterium,” “Bacillus,” “Clavibacter,” “Escherichia,” “Pseudomonas,” and other values that are obscured and accessible using the scroll bar or buttons on the right side of the querylist. HyperSQL had received a search selection for the database field “Genus” where the “allowed” values from the database start with “Agrobacterium.” If HyperSQL had *not* received such a search selection, HyperSQL would be unable to retrieve the values from the database to populate the list of values in Genus.

335. In sum, HyperSQL discloses “*receiving a search selection for a database field on the provided list of the database fields*” because HyperSQL receives from the user a selection for the database field corresponding to the prompt and input (*e.g.*, prompt and querylist) when the user selects the input on the list of prompts and inputs.

5. [1.D] “determining a quantity of entries in the selected database field;”

336. In my discussion of Ground 1, I already explained how Zellweger discloses determining a number of items (“*quantity of entries*”) in a “*selected database field*” in two ways. For the same three reasons expressed there, Zellweger discloses the same here. *See* §XII.A.5 above (Claim [1.D]).

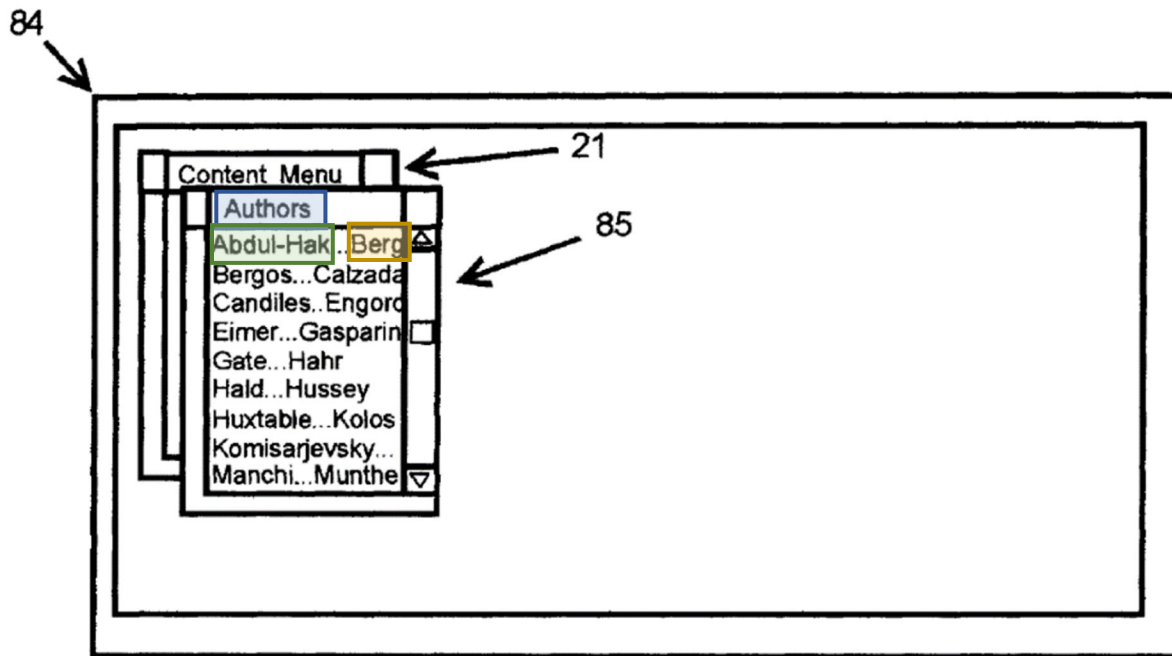
337. First, Zellweger explicitly discloses determining the number of entries items (entries) when a user selects one of the database fields.



Ex. 1007 (Zellweger), Figure 6a (annotated).

338. Zellweger's Figure 6a shows that Zellweger determined that there are 1,231 entries in the database field corresponding to Authors (in blue).

339. Second, Zellweger discloses this limitation when generating a "Content List." Zellweger discloses there are two kinds of Content Lists that can be created: an "Alphabet List" and a "Content List." Ex. 1007 (Zellweger), 4:17-35. If the user opts to make a Content List, all the entries are divided and grouped into intervals as shown below:



Id., Figure 7 (annotated).

340. Figure 7 above shows an example using the “Authors” database field (in blue) and the last name of the authors (in green and yellow) such that all the intervals are listed “using the author’s last name as the first and last elements.” Ex. 1007 (Zellweger), 4:45-53. For example, the first interval in the list includes authors with last name “Abdul-Hak” (in green) and goes to “Berg” (in yellow, although it’s cut off so whether the entire last name for the final last name in the interval is “Berg” or simply starts with “Berg” is unknown; this is irrelevant to the point being made here; see four lines below for the name range “Gate...Hahr” which is not cut off). *Id.*, Figure 7. If a user clicks on an entry, Zellweger discloses that

the user will open a newly generated “new list menu that starts with” the first entry and ends with the last entry in the selected interval. *Id.*

341. A Zellweger user has some settings available to them (in green in annotated Figure 6a below). Specifically, the user can set a maximum and a minimum number of items per list for the content list. Ex. 1007 (Zellweger), 4:26-31. For Zellweger to be able to determine how to segment the entries into the Content List intervals, and to determine whether the maximum or minimum is satisfied for a list, Zellweger must necessarily determine the quantity of entries for the associated database field (*e.g.*, Authors).

The image shows a user interface with two main sections. The top section is titled 'Alphabet List' and has a radio button that is selected. Below it, there is a 'Tab Length' field with the value '1' and 'Entries: 27'. The bottom section is titled 'Content List' and has an unselected radio button. Below this, there are two more radio buttons: 'Word Length' and 'Letter Tab Length'. The 'Letter Tab Length' field is empty. A green rectangular box highlights the 'Items per List' section, which contains 'Maximum' and 'Minimum' fields, both of which are empty.

Id., Figure 6a (excerpted and annotated).

342. A POSITA, in applying the methods of Zellweger to the querylists of HyperSQL, would have understood that such a combined system also “*determines a*

quantity of entries in the selected database field.” Zellweger’s improved menu system relies on functionality that performs this limitation. Thus, an integration of Zellweger’s improved menu teachings into HyperSQL requires Zellweger’s functionality to determine how many items are in the database field. *See* §XIII.A.6 below.

343. In sum, Zellweger discloses “*determining a quantity of entries in the selected database field*” in two ways. The first is Zellweger’s explicit disclosure of determining a number of items when the user selects the database field (*e.g.*, “Authors”). The second is because Zellweger must determine the number of items in a Content List to be able to compute whether the number of items meets the “maximum” or “minimum” value.

6. **[1.E] “if the quantity exceed a specified amount, truncating data, and displaying the truncated data wherein the truncating reduces characters in one or more entries in the selected database field and the truncated data represents each of the entries in the selected database field; and”**

344. As an initial matter, a POSITA would have known that truncating data entries by reducing their size, truncating strings or numbers, rounding numbers, and enabling an assortment of forms of summarization of data types (*e.g.*, dates) was already very well known by 1999. *See* §VII.D above.

345. Although HyperSQL does not provide an explicit disclosure of “*truncating data*” when the quantity of entries for a selected field exceeds a specified amount as required by Claim [1.E], Zellweger’s menu system discloses this.

346. Indeed, HyperSQL discloses that the querylists may need to display many values, requiring the user to scroll through the list to select the desired value(s), which would be increasingly cumbersome to a user as the number of values in the querylist increases. Specifically, HyperSQL notes that it is targeted towards being used with “large-scale biological collections [] stored in Sybase databases” and that the workstations running databases are “capable of managing large volumes of data.” Ex. 1006 (HyperSQL), 329. Further, HyperSQL Figure 5 shows an example query form where the querylist’s small scroll bar handle size indicates that there are many items to scroll through, which would force a user to waste time scrolling through the list:

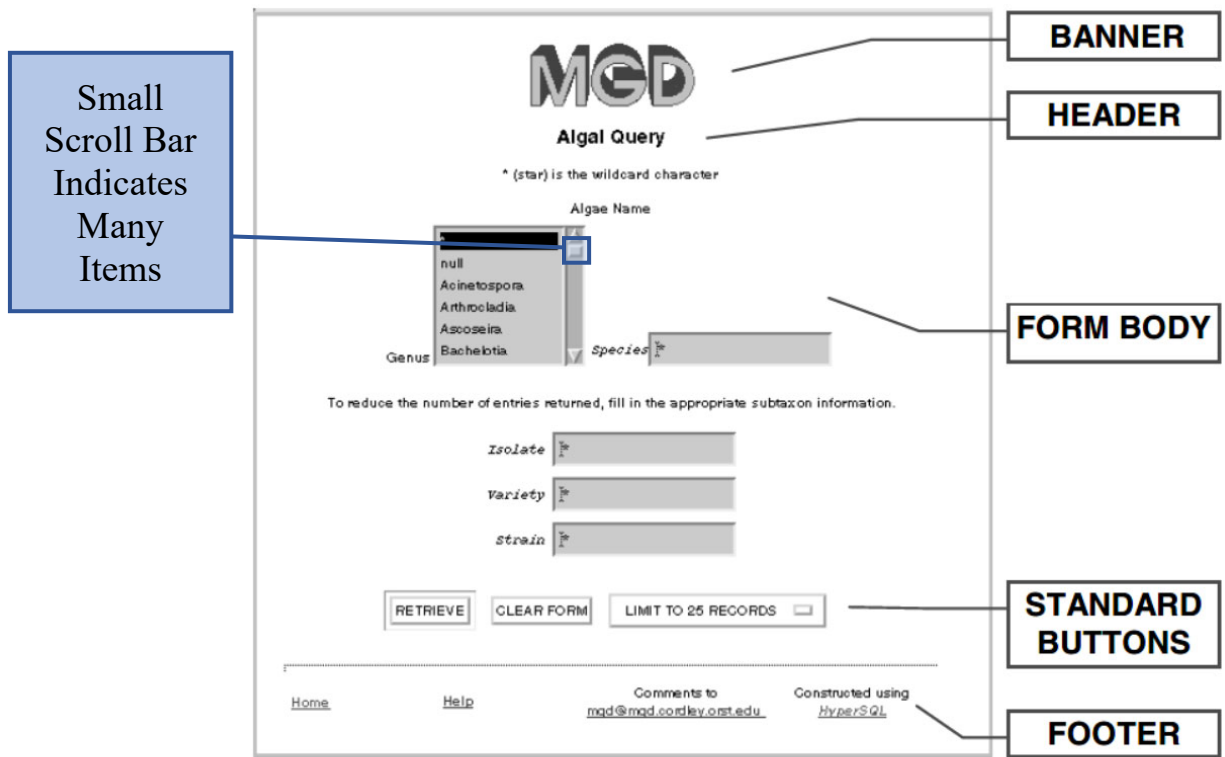


Figure 5. Layout of a sample query form

Id., Figure 5 (annotated).

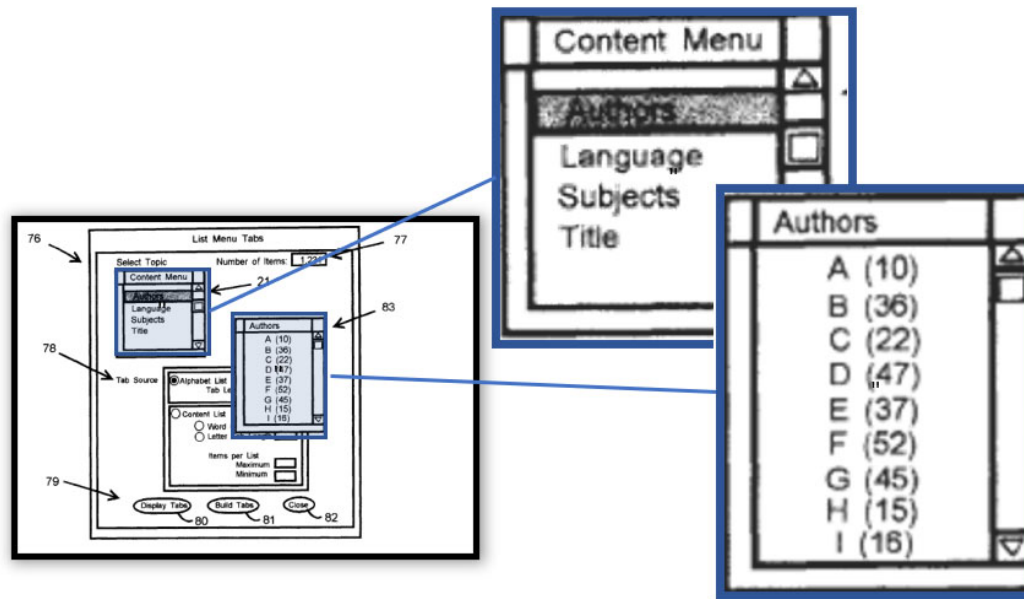
347. HyperSQL's Figure 5 shows a querylist for "Genus" that has many items, as indicated by the small size of the scroll bar handle (in blue).

348. Zellweger discloses using "Alphabet Lists" or "Content Lists" to provide an easy-to-use menu structure for when there are many items that must be provided on the list. I discussed this in Ground 1 at Claim [1.E]. *See* §XII.A.6 above (Claim [1.E]). For the same reasons expressed there, Zellweger discloses reducing ("truncating") lists that are voluminous (*i.e.*, where "*the quantity [of entries] exceed[s] a specified amount*") into Alphabet Lists or Content Lists. As I also explained in Ground 1 at Claim [1.E], Zellweger discloses that its Alphabet Lists

and Content Lists represent all available entries and shows them using letters or tabs; this corresponds to the claimed behavior in which “*the truncating reduces characters in one or more entries in the selected database field and the truncated data represents each of the entries in the selected database field.*” See §XII.A.6 above (Claim [1.E]).

349. Specifically, Zellweger discloses various methods that are used to display voluminous lists such as the querylists in HyperSQL, which are so voluminous that all the entries in the list may not fit on one page at the same time. Specifically, Zellweger discloses two separate types of lists known as “Alphabet Lists” and “Content Lists,” each of which “*truncat[es]*” the text of entries in a list by “*reduc[ing] characters in one or more entries of the selected database field*” and displaying them grouped into intervals (“*displaying the truncated data*”).

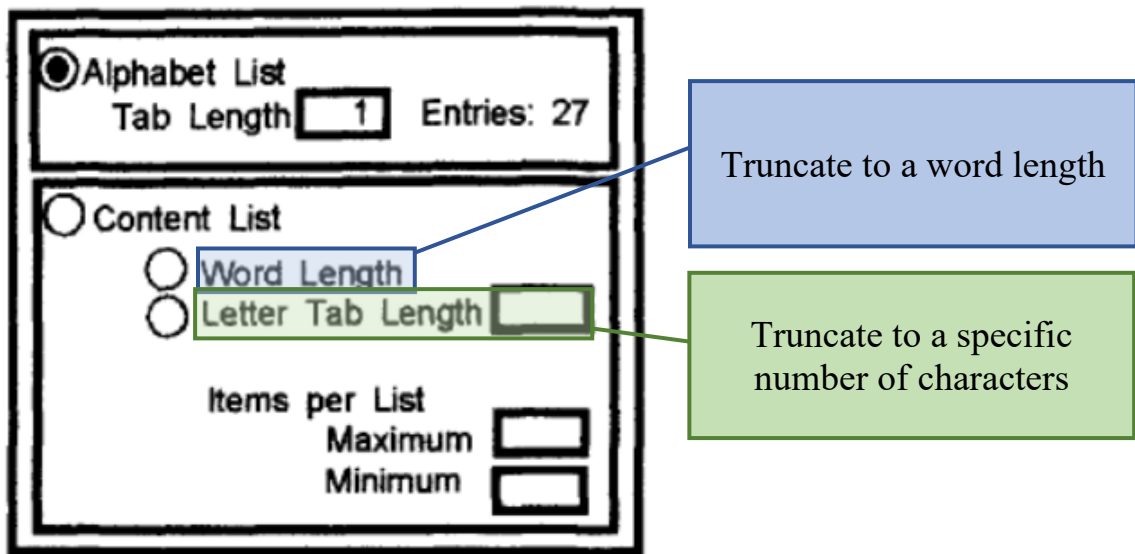
350. Zellweger’s Alphabet List first reduces the number of entries in the list to a user-defined number of letters and organizes the entries into groups based on their first letters (up to the user-defined number of letters). Figure 6b is an example of this, showing the truncation of the entries for “Authors” to one letter (and listing the count of elements that start with each author’s letter in parenthesis):



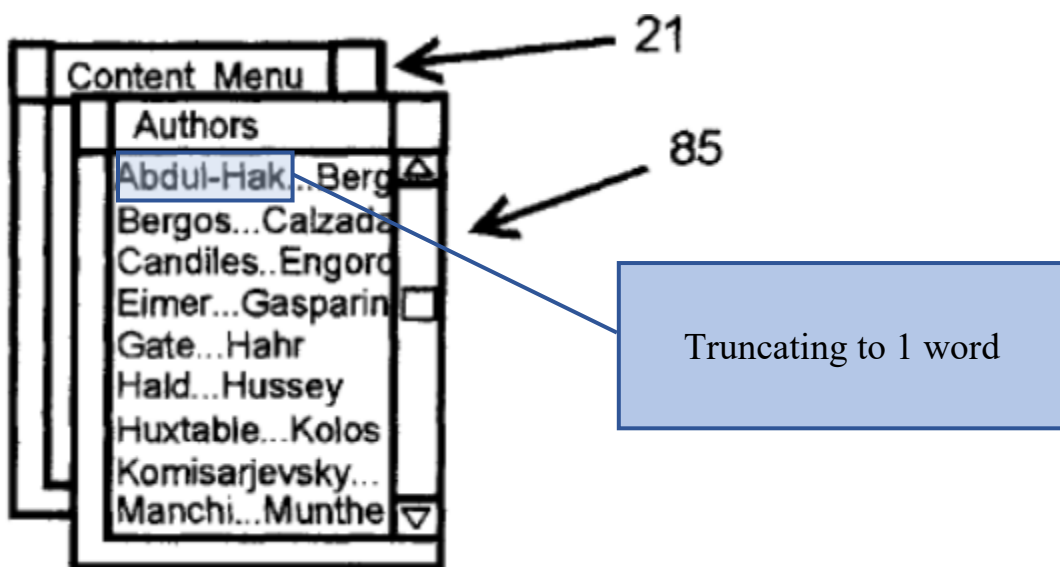
Ex. 1007 (Zellweger), Figure 6b (annotated, excerpts zoomed in), 4:36-44.

351. Here, all entries in the field “Authors” were truncated to a single letter (the first, or leading, letter) and then organized into a group. For example, Figure 6b shows that there are 10 entries in the Authors field that start with the letter A, 36 that start with the letter B, and so forth.

352. Zellweger’s Content List works a bit differently. Specifically, when a Content List is used, the entries are truncated down to a word (blue) or a number of leading characters (green) and then organized into a set of intervals, as seen in the figures below:



Ex. 1007 (Zellweger), Figure 6a (excerpted and annotated).



Id., Figure 7 (annotated).

353. Figure 7 above shows how the entries for the “Authors” database field are truncated to one word length and organized into ranges or intervals. Ex. 1007 (Zellweger), 4:45-53.

354. The truncated data in either the Alphabet List or Content List therefore “represents” each of the authors (“entries”) in the “Authors” field (“selected database field”). Specifically, each of the entries is *represented* in the Alphabet List grouping lists or the Content List interval lists. That is, in either of these lists, each entry is encapsulated by one of the set of groups or intervals.

355. A POSITA would have understood that when integrating Zellweger’s teachings of an improved menu into HyperSQL, the combination discloses that the user may input a “specified amount” into Zellweger’s “Minimum” setting so that when the number of entries to be displayed in a querylist is greater than that “specified amount,” the entries are truncated and displayed using a Zellweger-style list in which the entries are displayed using the truncated representation. See §XII.A.6 above (Claim [1.E]).

356. A POSITA also would have understood that because these inputs in Zellweger are simple text input boxes, a user could select any threshold limit when converting the lists to Alphabet or Content Lists. Ex. 1007 (Zellweger), 4:17-52, Figures 6a-7. For example, when representing the querylist values in a Zellweger Alphabet List, the user may progressively reduce the Tab Length value to arrive at a desired threshold limit. When representing the querylist values in a Zellweger Content List, the user may simply insert a desired value for the minimum (and maximum) Items per List. This would have made sense because computer displays

for different devices have limited space to display items: any software that does not recognize a display's characteristics and size limitations would risk displaying information beyond the display's ability to show it (*e.g.*, "off screen"). *See also* §VII.B above.

357. In sum, the combined HyperSQL/Zellweger system discloses "*if the quantity exceed a specified amount, truncating data, and displaying the truncated data wherein the truncating reduces characters in one or more entries in the selected database field and the truncated data represents each of the entries in the selected database field*" because if the number of values to display in a querylist exceeds the "minimum" value set for a Zellweger Content List, the combined system would truncate the querylist values by reducing characters in the one or more values and displaying them in Zellweger Content List, which represents each of the values in the corresponding database field (*e.g.*, the database field for "Genus").

7. [1.F] "if the quantity does not exceed the specified amount, displaying contents of the database field."

358. I just described in this Ground for Claim 1E how HyperSQL in view of Zellweger discloses or renders obvious truncating the list of entries if the list exceeds a specified number of entries. *See* §XIII.A.6 above (Claim [1.E]). Therefore, HyperSQL in view of Zellweger does not truncate the list of entries if it can fit in the display. *See also* §VII.B above.

359. In sum, the combined HyperSQL/Zellweger system displays the list without truncation (“*displaying contents of the database field*”) when the list it does not exceed the specified amount (“*if the quantity does not exceed the specified amount*”). Specifically, when the quantity of values to display in a querylist does not exceed the specified amount, the values would just be displayed in an untruncated HyperSQL querylist.

B. Claim 2

1. “The method of claim 1, further comprising providing a key word search.”

360. HyperSQL discloses that its query form includes a “KEYWORD SEARCH” button which “brings up a form that accepts a text string to be located by searching all data fields” when a user clicks on it (“*providing a key word search*”). Ex. 1006 (HyperSQL), 331, Figure 2.

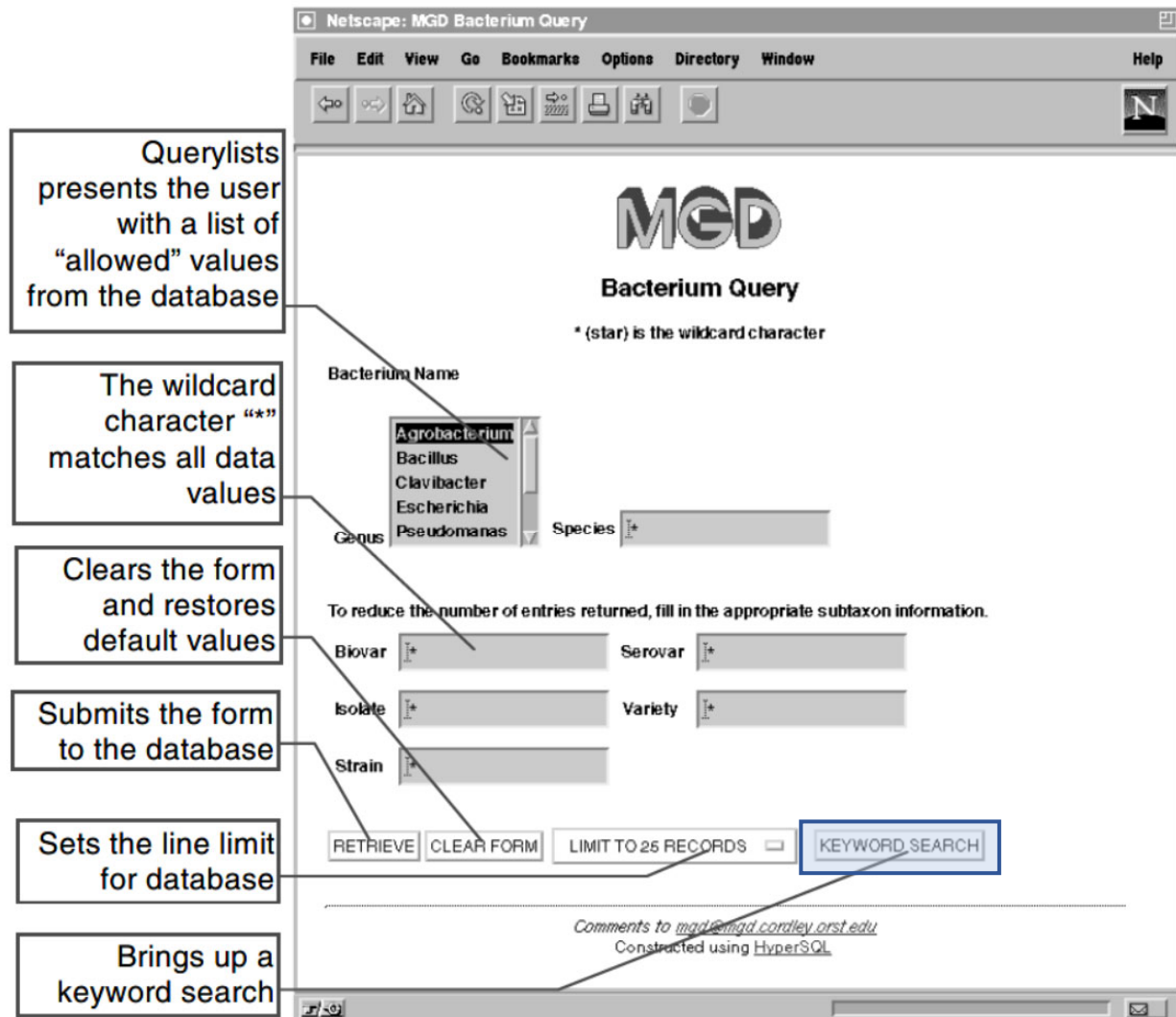


Figure 2. Example of a *query form* (shown in a Netscape browser)

Id., Figure 2 (annotated).

361. The KEYWORD SEARCH button is shown in the annotated version of Figure 2 in blue.

362. In sum, HyperSQL discloses "*comprising providing a key word search*" because it discloses this "KEYWORD SEARCH" button.

C. Claim 3

1. [3.PRE] “A method implemented on a computer for searching a database, comprising:”

363. HyperSQL in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claim [1.PRE]. *See* §XIII.A.1 above (Claim [1.PRE]).

2. [3.A] “generating a list of data fields;”

364. HyperSQL in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claim [1.B]. *See* §XIII.A.3 above (Claim [1.B]).

3. [3.B] “receiving a first data field selection from the list of data fields;”

365. My discussion at §XII.C.3 above (regarding Claim [3.B] in Ground 1) about the equivalence of some of the terms used in Claim 1 and Claim 3 applies here as well. As stated there, the difference in word choice is immaterial and a POSITA would have understood that these terms are equivalent.

366. HyperSQL in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claim [1.C]. *See* §XIII.A.4 above (Claim [1.C]).

4. [3.C] “determining a first quantity indicative of a number of entries of the selected data field;”

367. My discussion at §XII.C.4 above (regarding Claim [3.C] in Ground 1) about the equivalence of some of the terms used in Claim 1 and Claim 3 applies here as well. As stated there, the difference in word choice is immaterial and a POSITA would have understood that these terms are equivalent.

368. HyperSQL in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claim [1.D]. *See* §XIII.A.5 above (Claim [1.D]).

5. [3.D] “if the first quantity exceeds a specified limit, reducing a size of data to be displayed from the selected data field wherein the reducing reduces characters in one or more entries in the selected data field and the size-reduced data represents each of the entries in the selected field; and”

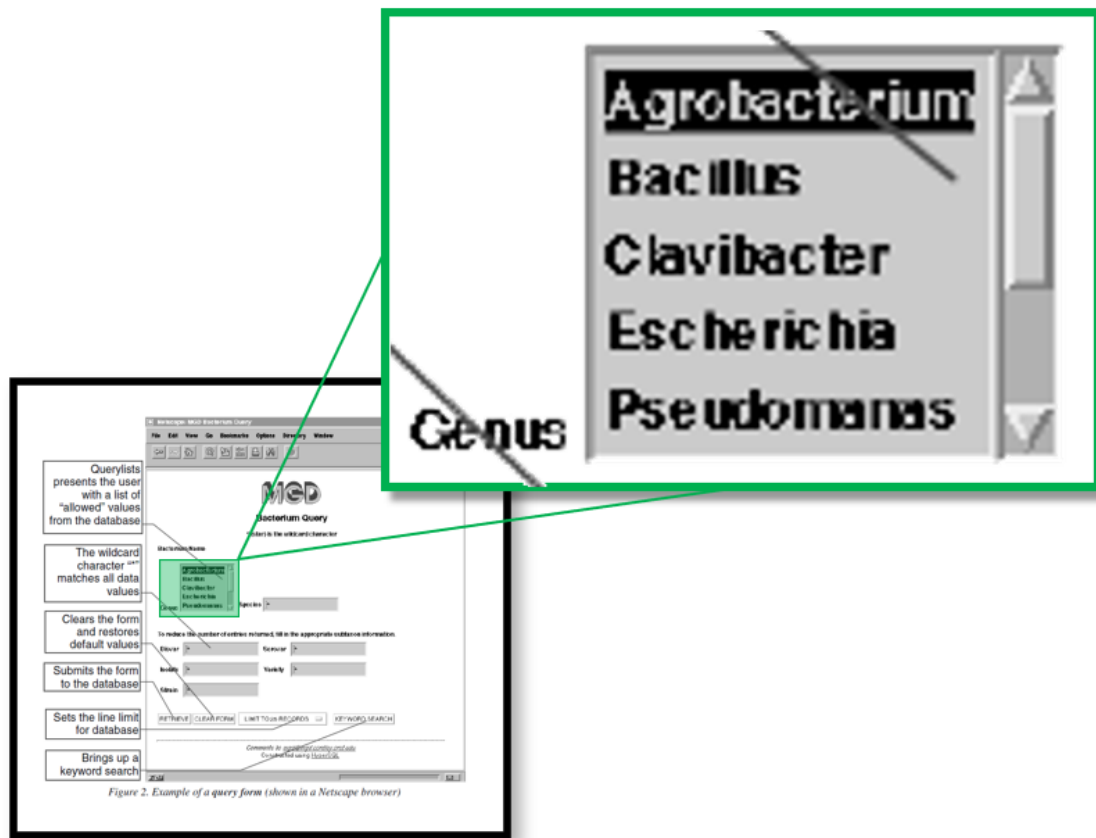
369. The discussion at §XII.C.5 above (regarding Claim [3.D] in Ground 1) about the equivalence of some of the terms used in Claim 1 and Claim 3 as well as some apparent errors in Claim 3 applies here as well. As stated there, the difference in word choice is immaterial and a POSITA would have understood that these terms are equivalent. I have had to make the same assumptions as to the apparent errors to make the claim limitations intelligible.

370. HyperSQL in view of Zellweger discloses this limitation for the same reasons expressed in this Ground regarding Claims [1.E] and [3.E]. *See* §XIII.A.6 above (Claim [1.E]) and §XIII.C.6 below (Claim [3.E]).

6. [3.E] “displaying data from the selected data field.”

371. In this Ground for Claims [1.E] and [1.D] I already discussed how HyperSQL and Zellweger each disclose displaying data from the data field selected by the user. *See* §§XIII.A.5-XIII.A.6 above (Claims [1.D]-[1.E]). I provide a brief recap below for clarity.

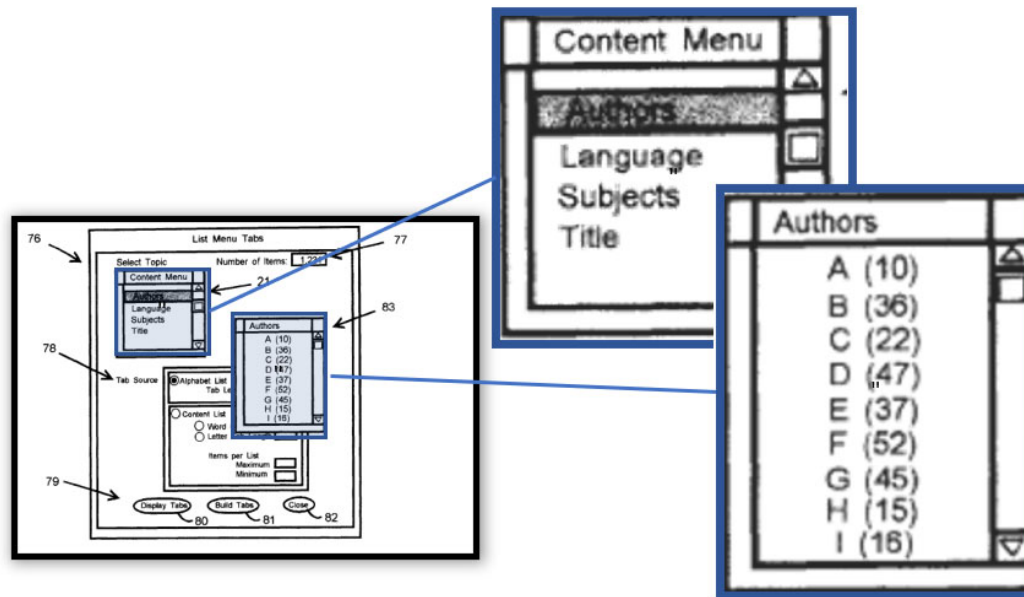
372. HyperSQL displays the inputs including querylists as shown in HyperSQL Figure 2.



Ex. 1006 (HyperSQL), Figure 2 (annotated, excerpts zoomed in).

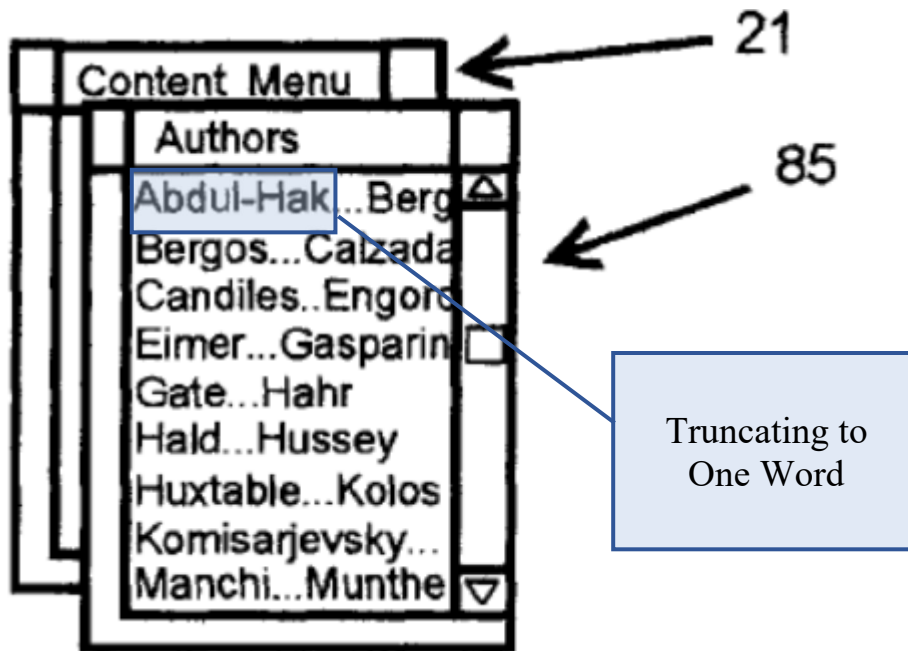
373. This figure shows that the data for the database field “Genus” is displayed in a querylist, which is blown up in the annotated figure. Specifically, the Genus field has data entries such as “Agrobacterium,” “Bacillus,” “Clavibacter,” “Escherichia,” and “Pseudomonas.” In the zoomed in version of the Genus querylist, we see some data entries displayed on screen as well as a scrollbar. The scrollbar indicates there are more entries in the querylist that are not displayed, but to which the user may scroll using the scroll bar handle or scroll bar arrow buttons.

374. Zellweger also discloses this in a couple of ways. Specifically, Zellweger discloses displaying entries in Alphabet Lists and Content Lists (“displaying data from the selected data field”), as shown in the figures below:



Ex. 1007 (Zellweger), Figure 6b (annotated, excerpts zoomed in).

375. Figure 6b shows that Authors is the “selected data field” and the data from Authors is displayed as a truncated representation within the Alphabet List, each value being categorized based on the first letter (“displaying data from the selected data field”). Ex. 1007 (Zellweger), 4:21-25.



Id., Figure 7 (annotated).

376. Here, in Figure 7, Authors is still the “*selected data field*” but the Authors are displayed in a different truncated representation within a Content List menu structure, where the entire list of values are split up and grouped into intervals (“*displaying data from the selected data field*”). Ex. 1007 (Zellweger), 4:26-31.

377. In sum, HyperSQL discloses “*displaying data from the selected data field*” because HyperSQL displays values from the database for the selected database field in querylists. Furthermore, Zellweger discloses “*displaying data from the selected data field*” because Zellweger discloses entries for the selected database field in Alphabet Lists and Content Lists.

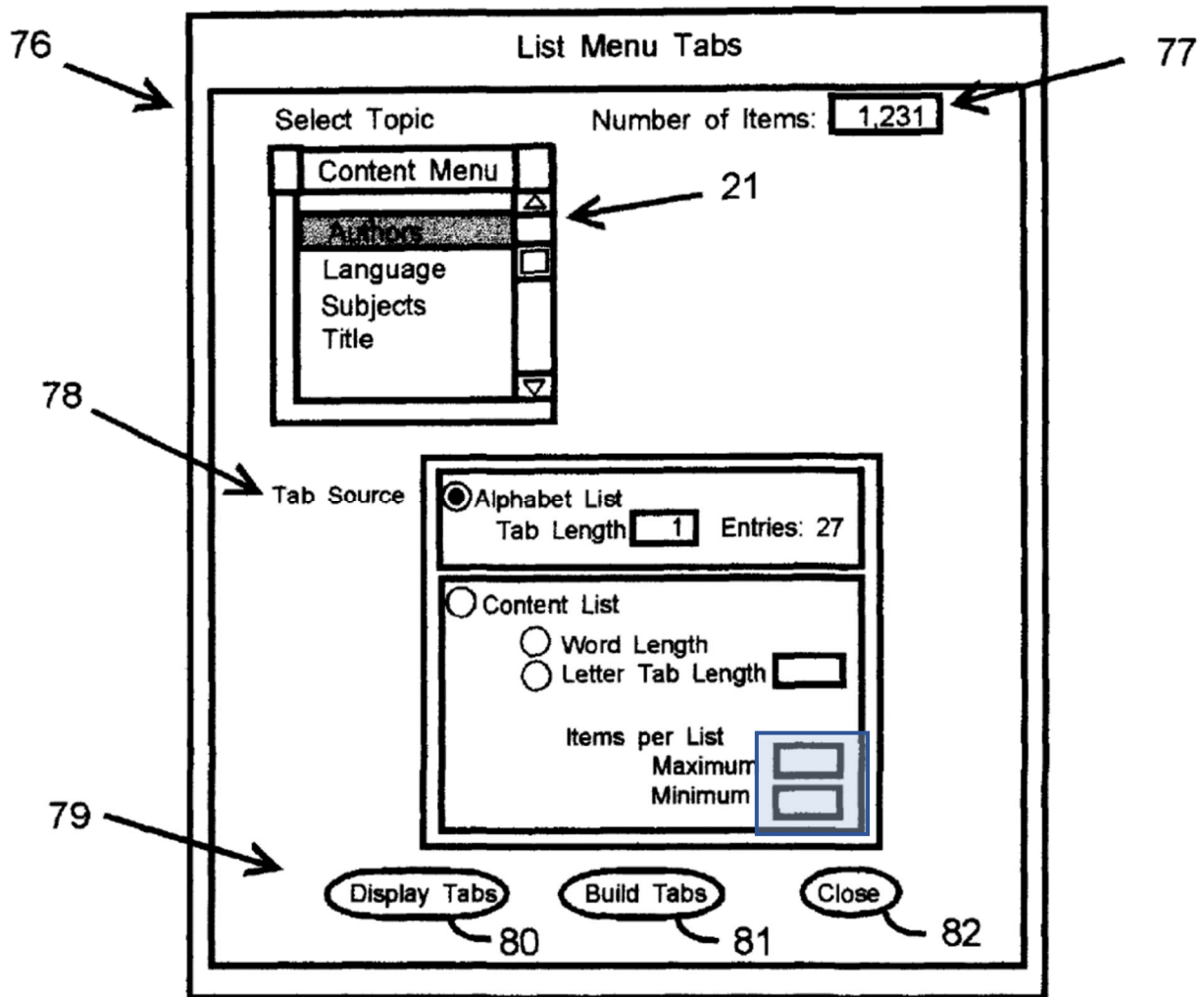
D. Claim 4

1. “The method of claim 3, wherein the specified limit is fixed.”

378. As explained in this Ground regarding Claim [1.E], HyperSQL may need to display many items in a querylist, which would require cumbersome scrolling through a potentially lengthy list. *See* §XIII.A.6 above (Claim [1.E]). Further, as I explained in Ground 1 for claim 4, Zellweger discloses this limitation when handling such voluminous lists and the reasons expressed there are equally applicable here. *See* §XII.D above (Claim 4).

379. Specifically, Zellweger discloses a method of splitting a list of entries into intervals using “Content Lists”; a user may specify the minimum and maximum number of “Items per List.” *See* Ex. 1007 (Zellweger), Abstract (“The present invention discloses a menu authoring system component that enables a developer to generate and build a tab list menu. The tab menu divides a target list menu into tab intervals that can include a specified number of letters or full words. The present invention discloses an interactive configuration window and the program logic that enables a developer to specify settings for these tab menus.”), 2:30-43, 4:26-52, Figure 6a.

380. Zellweger shows that the “Minimum” and “Maximum” settings are plain text input fields.



Ex. 1007 (Zellweger), Figure 6a (annotated).

381. Figure 6a shows that the Maximum and Minimum values are set by a user typing text into plain text entry boxes (in blue). A POSITA would have understood that such plain text input fields allow a user to input whatever they want into the field. The specified limit is “fixed” when the user sets the minimum items per list to one value and does not change it.

E. Claim 5

1. “The method of claim 3, wherein the specified limit is variable.”

382. HyperSQL in view of Zellweger discloses this limitation. In Ground 1 for Claim 5, I explained how Zellweger discloses this limitation and the reasons expressed there are equally applicable here. *See* §XII.E above (Claim 5).

383. Specifically, Zellweger’s minimum and maximum values are variable because the user may change either one at will and the values are thus “variable” when these values are changed. *See also* §XIII.D above (Claim 4).

F. Claim 7

1. “The method of claim 3, wherein the specified limit is a user-determined limit.”

384. Zellweger discloses that a user “can specify the maximum and minimum” value to set the maximum and minimum “items per list” for the Content List (“*the specified limit is a user-determined limit*”). Ex. 1007 (Zellweger), 4:26-31; *See also* §§XIII.D (Claim 4) and XIII.E above (Claim 5). Both the “maximum” and “minimum” values are set and input by the user so they are each necessarily “user-determined.”

G. Claim 15

1. [15.A] **“The method of claim 3, further comprising: receiving a first constraint, wherein the first constrain[t] is related to a data element in a data field; and”**

385. HyperSQL discloses that a user may select a value from querylists provided on the query form, where the values in the querylist are “‘allowed’ values from the database.” Ex. 1006 (HyperSQL), 331, 333, Figure 2. These “allowed values” are the actual entries for the data field associated with the querylist, so a POSITA would have understood that any such value (“*constraint*”) is “*related to a data element in a data field*” for any given querylist. *Id.*, 329-33, 333-34, Figure 2. Furthermore, a user’s selection of a value from a first querylist constitutes the “*receiving a first constraint*.” An example is shown below with a query form that currently shows one querylist:

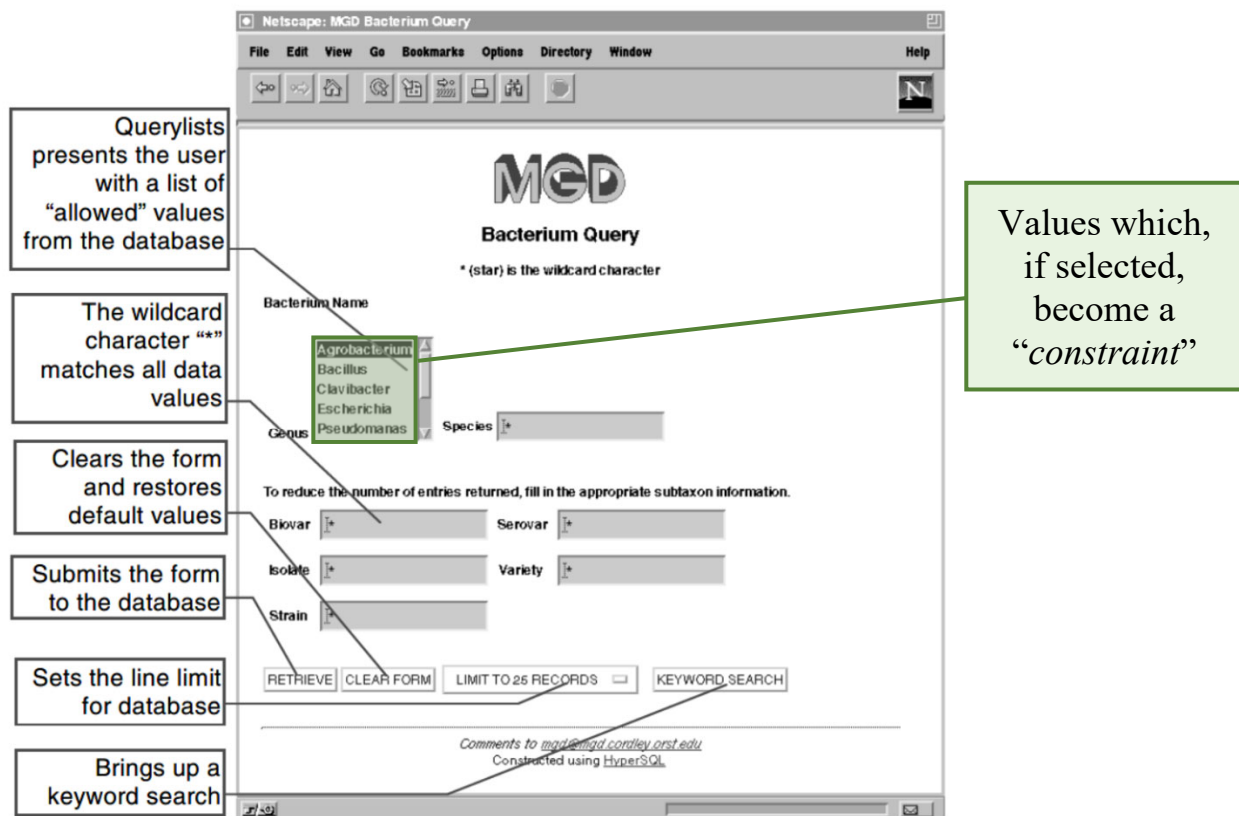


Figure 2. Example of a *query form* (shown in a Netscape browser)

Ex. 1006 (HyperSQL), Figure 2 (annotated).

386. For example, and using Figure 2 above as a reference, a user may search for information in a biological bacterium database using the Bacterium Query “query form” of Figure 2 where the database has data fields such as “Genus” and “Species.” Ex. 1006 (HyperSQL), Figure 2. The user picks a “*a first constraint*,” for example, when the user clicks on “Agrobacterium” for the “Genus” data field. *Id.* That “*constrain[t] is related to a data element*” in the “Genus” “*data field*.” *Id.*, 332-33, Figure 2. HyperSQL itself describes that the “query forms” are “where users enter query constraints.” *Id.*, 337.

387. In sum, HyperSQL discloses “*receiving a first constraint, wherein the first constrain[t] is related to a data element in a data field*” because HyperSQL receives a selection from a user of value(s) that are related to the data field indicated by the input’s associated prompt, and those value(s) become a constraint on the user’s search.

2. [15.B] “receiving one or more subsequent constraints, wherein search results are generated based on a combination of the first and the one or more subsequent constraints.”

388. HyperSQL discloses that a user can pick or select values for multiple inputs such as picking “Agrobacterium” as the constraint for database field “Genus.” *See* §XIII.G.1 above (Claim [15.A]). The user can then pick subsequent constraints for other database fields like “Species,” “Biovar,” “Serovar,” *etc.*, that each progressively narrow the search being conducted against the database (“*receiving one or more subsequent constraints*”).” Ex. 1006 (HyperSQL), 332-33, Figure 2. Again, HyperSQL itself directly states that the “query forms” are “where users enter query constraints.” *Id.*, 337.

389. HyperSQL also explains that it has a “RETRIEVE” button on the query form. When that button is clicked, HyperSQL “[s]ubmits the form to the database” and causes it to “return[] results.” Ex. 1006 (HyperSQL), 331, Figure 2. The constraints are combined and applied in a search of the database to produce a set of results from the database(s) that has been narrowed by the combined constraints

(“search results are generated based on a combination of the first and the one or more subsequent constraints”). *Id.*, 331-333, Figures 2-4 and 6.

390. HyperSQL further discloses that a query form may have multiple querylists used in a process called “query refinement” in which the user “fill[s] in fields incrementally by selecting from ‘querylists’” (“search results are generated based on a combination of the first and the one or more subsequent constraints”). Ex. 1006 (HyperSQL), 331. HyperSQL explains that “querylists are useful when the user does not know what the database contains, or simply wants to select from a list of values rather than typing.” *Id.*

391. These disclosures mean that on a form with multiple querylists, a user can choose a value from a first querylist as a first constraint; this causes a refresh of the other querylists incrementally as values are selected. The user can then choose a value from a subsequent querylist as the next constraint. Once the user has selected all the values/constraints the user desires, the user can click on the “RETRIEVE” button as explained above to run the search and retrieve the final results, applying all values as search constraints. Ex. 1006 (HyperSQL), 331, Figure 2.

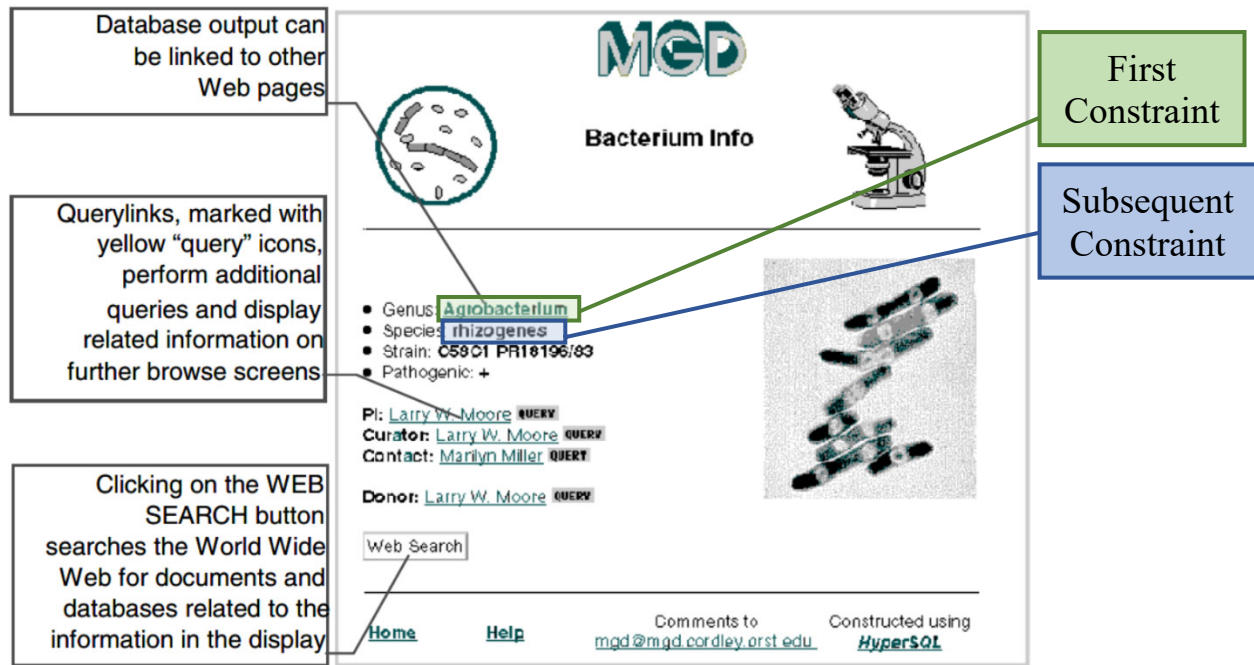


Figure 4. Example of a browse screen

Id., Figure 4 (annotated); *see also id.*, Figure 3.

392. Figure 4 reproduced and annotated above shows a screen that depicts a result on the browse screen, found after performing a search, where Genus was set to *Agrobacterium* (the “*first constraint*”) and Species was set to *rhizogenes* (the “*subsequent constraint*”).

393. In sum, HyperSQL discloses “*receiving one or more subsequent constraints, wherein search results are generated based on a combination of the first and the one or more subsequent constraints*” because HyperSQL teaches that a user may select a second value (“*receiving one or more subsequent constraints*”) and run a search that applies both the first and second values as constraints on the returned

result set (“*wherein search results are generated based on a combination of the first and the one or more subsequent constraints.*”).

H. Claim 16

- 1. “The method [of] claim 3, wherein each display of data comprises a search cycle, and further comprising receiving a refresh command, the refresh command resetting all search cycles.”**

394. As I discussed above in this Ground regarding Claim 15, a HyperSQL user can perform a search in database(s) using values (entries/constraints) from multiple querylists (database fields). *See* §XIII.G.2 above (Claim [15.B]).

The screenshot shows a web form titled "MGD Algal Query". The form includes a "Genus" dropdown menu with options: null, Acinetospora, Arthrocladia, Ascosaira, and Bachelotia. Next to it is a "Species" text input field. Below these are three more text input fields labeled "Isolate", "Variety", and "Strain". At the bottom of the form are three buttons: "RETRIEVE", "CLEAR FORM", and "LIMIT TO 25 RECORDS" with a checkbox. The footer contains links for "Home", "Help", "Comments to mgd@mgd.cordley.orst.edu", and "Constructed using HyperSQL".

Annotations on the right side of the form:

- BANNER**: Points to the "MGD" logo.
- HEADER**: Points to the "Algal Query" title.
- FORM BODY**: Points to the "Genus" dropdown and "Species" input field.
- STANDARD BUTTONS**: Points to the "RETRIEVE", "CLEAR FORM", and "LIMIT TO 25 RECORDS" buttons.
- FOOTER**: Points to the footer links and text.

Ex. 1006 (HyperSQL), Figure 5.

395. HyperSQL Figure 5 shows multiple querylists and fields for which a user may select values/entries/constraints. Although Figure 5 indicates only one querylist on the query form, HyperSQL discloses multiple querylists on a query form. See §XIII.G.2 above (Claim [15.B]).

396. As I discussed above in this Ground for Claim 15, when a user selects one of the querylists/fields (or at least value(s) therein), HyperSQL searches the available values/entries and displays them in the querylist (“wherein each display of

data comprises a search cycle”) to ensure that the querylists are displaying the “‘allowed’ values from the database.” See §XIII.G.2 above (Claim [15.B], discussing an example search using Genus and Species); Ex. 1006 (HyperSQL), 331 (“query forms supports [sic] ‘query refinement’ allowing the user to fill in fields incrementally by selecting from ‘querylists’ (lists of allowed valued).”), 333 (querylists are “filled with information acquired automatically from the database”), Figure 2 (“Querylists presents the user with a list of ‘allowed’ values from the database”).

397. HyperSQL also discloses that the query form contains a “CLEAR FORM” button. Ex. 1006 (HyperSQL), Figures 2, 5.

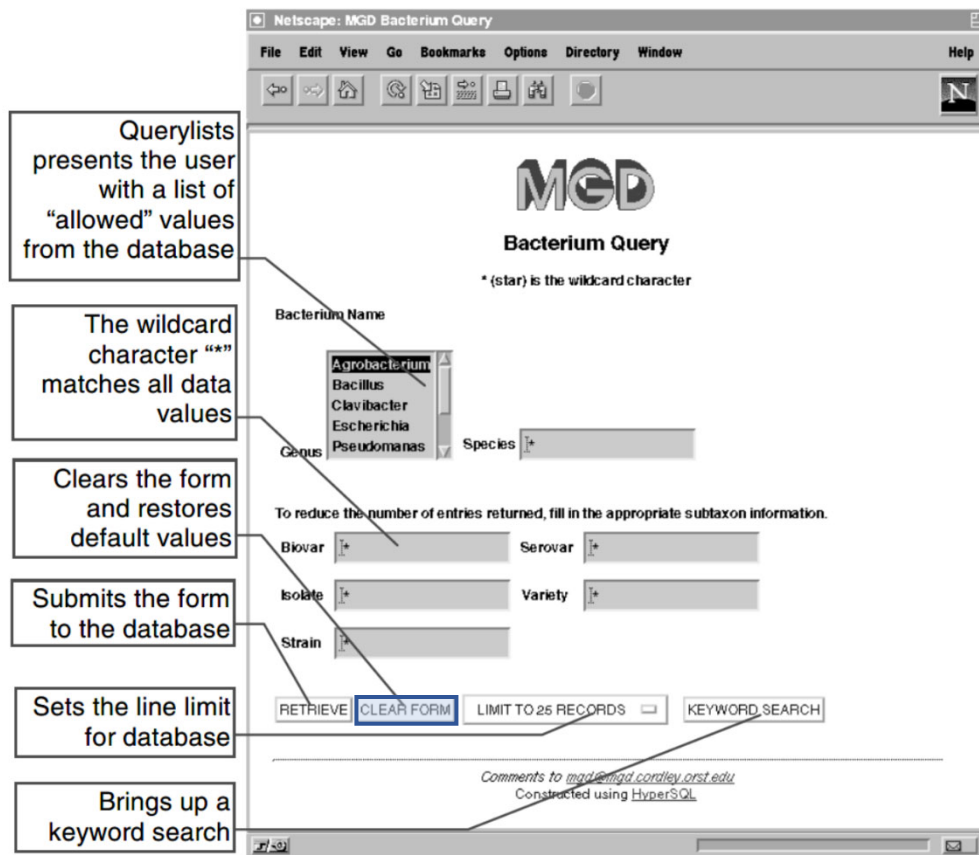


Figure 2. Example of a *query form* (shown in a Netscape browser)

Id., Figure 2 (annotated).

398. HyperSQL explains that when the user activates the CLEAR FORM button (seen in blue within the annotated copy of Figure 2 above), HyperSQL “erases any information entered into the form, replacing them with default values,” which constitutes the claimed “*receiving a refresh command, the refresh command resetting all search cycles.*” Ex. 1006 (HyperSQL), 330-31, Figure 2. This action resets the search cycles because once the information “entered into the form” is “erase[d]” and “replac[ed] [] with default values,” each of the prior operations comprising a “*search cycle*” (user selecting a querylist, user selecting a value from

the querylist, and HyperSQL refreshing the values in the other querylists) will have been undone (*i.e.*, “reset”).

399. In sum, HyperSQL discloses “*wherein each display of data comprises a search cycle*” when a user uses a query form with multiple querylists. Specifically, in this scenario, the user will select values from a first querylist, then a second querylist, and so on. At each selection, HyperSQL performs an additional search to ensure only allowed values are in the other querylists and then proceeds to display only those allowed values in the querylists. Furthermore, HyperSQL discloses “*further comprising receiving a refresh command, the refresh command resetting all search cycles*” because HyperSQL discloses a CLEAR FORM button to reset the selections that the user made in the querylists such that the query form returns to its original state with the default values.

I. Claim 17

- 1. “The method [of] claim 3, further comprising displaying the list of data fields and a representation of the selected data field, wherein the representation of the selected data field comprises one or more elements from the selected data field.”**

400. HyperSQL teaches displaying a query form that has a “form body section” that presents the user with several inputs, where the inputs “includ[e] radio lights ..., buttons, pulldown menus, and querylists ..., as well as simple text entry

fields.” Ex. 1006 (HyperSQL), 333. Thus, a query form may have multiple inputs including multiple querylists, which corresponds to “*displaying the list of data fields.*”

401. HyperSQL further discloses that a user can select a particular querylist to display the allowed values for that querylist. That is, the selected querylist is “filled with information [(values)] acquired automatically from the database.” Ex. 1006 (HyperSQL), 333. Moreover, the querylist values are those “‘allowed’ values from the database” for that database field. *Id.*, Figure 2. A user selecting a particular querylist to display the allowed valued for that querylist corresponds to “*displaying ... a representation of the selected data field [querylist], wherein the representation of the selected data field comprises one or more elements from the selected data field.*”

402. This is shown in the figure below:

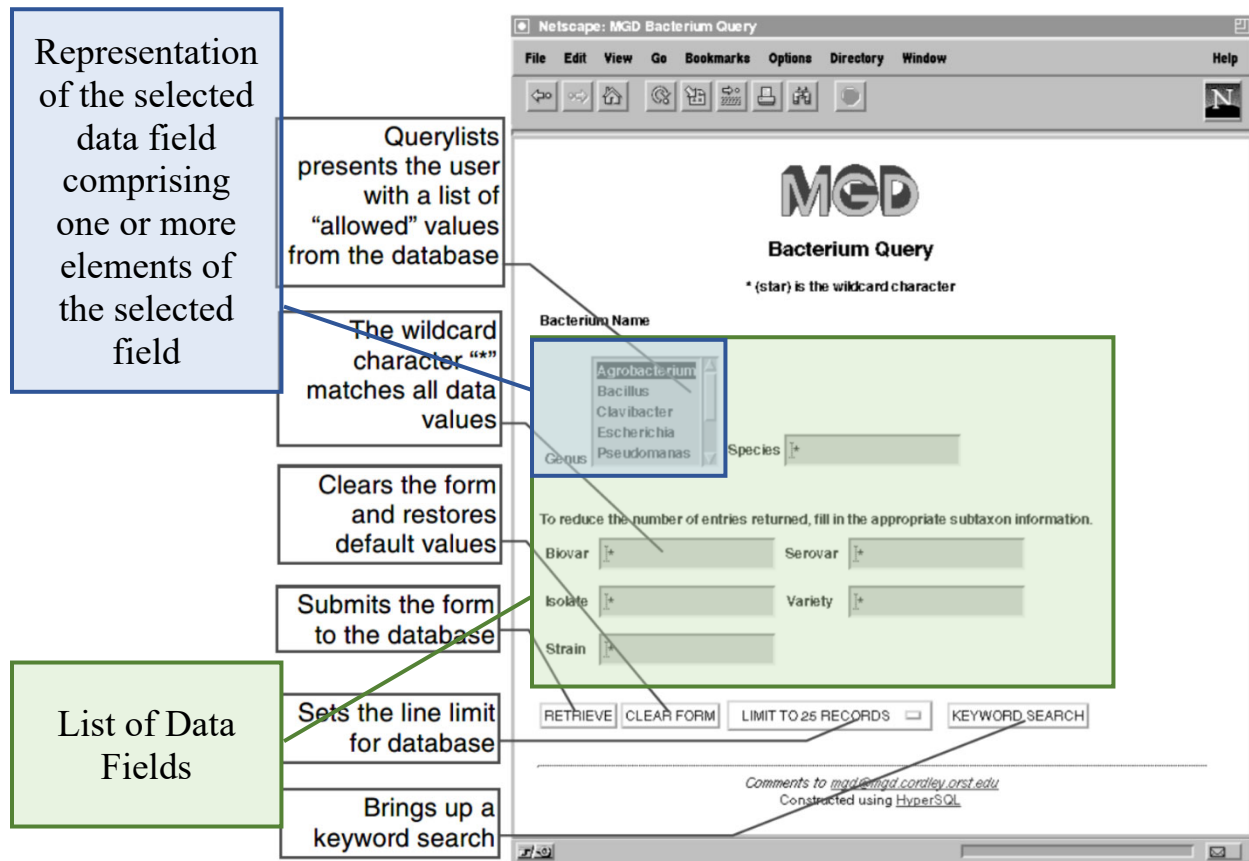


Figure 2. Example of a **query form** (shown in a Netscape browser)

Ex. 1006 (HyperSQL), Figure 2 (annotated); *see also id.*, 331-33, 338, Figure 5.

403. HyperSQL Figure 2, above, shows that the query form contains a list of prompts and inputs (in green, which may be querylists) associated with various database fields such as Genus, Species, and more. Each of the querylists lists the allowed values from the database, which consists of one or more of the values (“*elements*”) from the selected database field (in blue).

404. In sum, HyperSQL’s list of allowed values in each querylist is a “*representation of the selected data field*” comprising “*one or more elements from*

the selected data field” because each list of values in a querylist lists the values for the corresponding database field and the values are “from the database” for that field.

J. Claim 18

- 1. “The method of claim 3, wherein the method comprises executing a plurality of search cycles, each search cycle of the plurality of search cycles providing a search result list comprising selected data elements from the data base.”**

405. As I discussed above in this Ground for Claims [15.B] and 16, HyperSQL teaches a “query refinement” feature through which a user selects values (entries/constraints) from a plurality of querylists (database fields), and each time a user selects a different querylist (database field), the system executes a “*search cycle*.” See §§XIII.G.2 and XIII.H above (Claims [15.B] and 16, respectively).

406. Specifically, and again as I discuss this above for Claims [15.B] and 16, HyperSQL performs a search (“*search cycle*”) to populate other querylists with the “allowed values” “automatically from the database” whenever the user selects a value from a querylist (“*each search cycle of the plurality of search cycles providing a search result list comprising selected data elements from the data base*”). See §§XIII.G.2 and XIII.H above; Ex. 1006 (HyperSQL), 331, 333.

407. A POSITA would have understood that a user selecting values from multiple querylists as taught by HyperSQL teaches “*executing a plurality of search*

cycles,” given HyperSQL’s “query refinement” behavior. Ex. 1006 (HyperSQL), 331, Figure 2.

K. Claim 19

1. “The method of claim 18, further comprising resetting one or more search cycles of the plurality of search cycles.”

408. As an initial matter, I note that “resetting one or more search cycles” includes resetting all search cycles.

409. As discussed immediately above in this Ground for Claim 18, HyperSQL discloses a search process with multiple cycles when it describes using multiple querylists on a query form and the query refinement feature. *See* §XIII.J above (Claim 18).

410. Furthermore, I previously discussed how HyperSQL discloses the “CLEAR FORM” button on HyperSQL’s “query form,” which a user can activate to reset the search cycles. *See* §XIII.H above (Claim 16). The activation of the “CLEAR FORM” button corresponds to “*resetting one or more search cycles of the plurality of search cycles*” for the same reasons expressed in this Ground regarding Claim 16.

L. Claim 27

- 1. “The method of claim 3, further comprising displaying the data on a terminal, the terminal including one of a handheld device, a cellular phone, a geosynchronous positioning satellite (GPS) device, a wrist-worn device, an interactive phone device, a household appliance, a television, a television set top box, a handheld computer, a main frame computer and a personal computer.”**

411. The HyperSQL interface consists of “a set of screens” which are “displayed by the browser” of a computer. Among these screens is the “query form” which is used to display inputs such as querylists. Querylists display the “‘allowed’ values from the database.” Thus, the display of the values in querylists on a screen rendered on a browser corresponds to “*displaying the data on a terminal.*” Ex. 1006 (HyperSQL), 330-31, 334-36; Figures 2-6 and 8-9. This is shown in Figure 9, reproduced below:

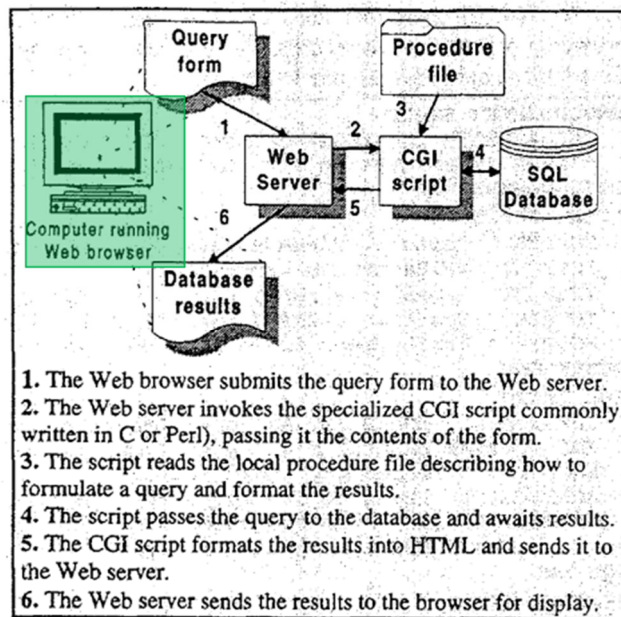


Figure 9. How CGI-based query interfaces work

Id., Figure 9 (annotated).

412. Figure 9 shows, for example, that the Web server sends database search results to “the browser for display” on the “Computer” highlighted in green in Figure 9 above. Ex. 1006 (HyperSQL), Figure 9. From the disclosures in HyperSQL, a POSITA would have understood that the “Computer running Web browser” would be at least one of a “*handheld computer ... [or] a personal computer.*” Figure 9’s choice of graphics support this because the “Computer” in that figure is depicted using a graphical representation often used to represent personal computers. *See also* §VII.A above.

M. Claim 28

1. **“The method of claim 3, wherein the database is coupled to an Internet website, the method further comprising accessing the Internet website.”**

413. HyperSQL discloses that it is a “Web-based Query Interface[]” and is thus implemented on the Web (Internet). Ex. 1006 (HyperSQL), 329-30.

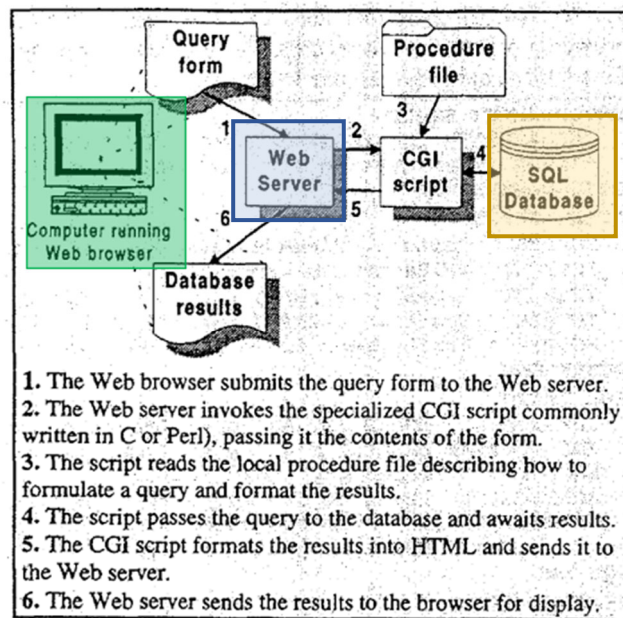


Figure 9. How CGI-based query interfaces work

Id., Figure 9 (annotated).

414. Figure 9 confirms that HyperSQL’s database(s) (in yellow) communicate with a web server (in blue), which is where the website is located, and the Computer (in green) accesses the database information using the Query form and Web server. *See also* Ex. 1006 (HyperSQL), Figure 8, 333. Thus, the HyperSQL “database is coupled to an Internet website.”

415. HyperSQL discloses that a user performs searches by accessing the website using a “Computer running Web browser” (“*accessing the Internet website*”). Ex. 1006 (HyperSQL), 329, Figure 9. Figure 9 explicitly states the “Web browser submits the query form to the Web server” in step 1, and then the following steps use that data to send the “query to the database,” after which the Web server sends “format[ted]” “results to the browser [on the Computer] for display.” *Id.* This is confirmed in Figure 8 which shows an analogous flow:

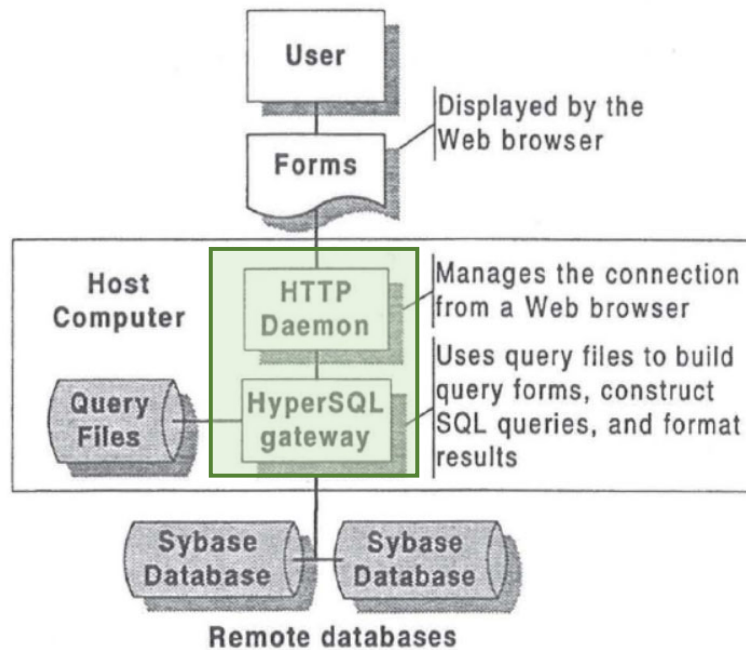


Figure 8. HyperSQL architecture

Id., Figure 8 (annotated).

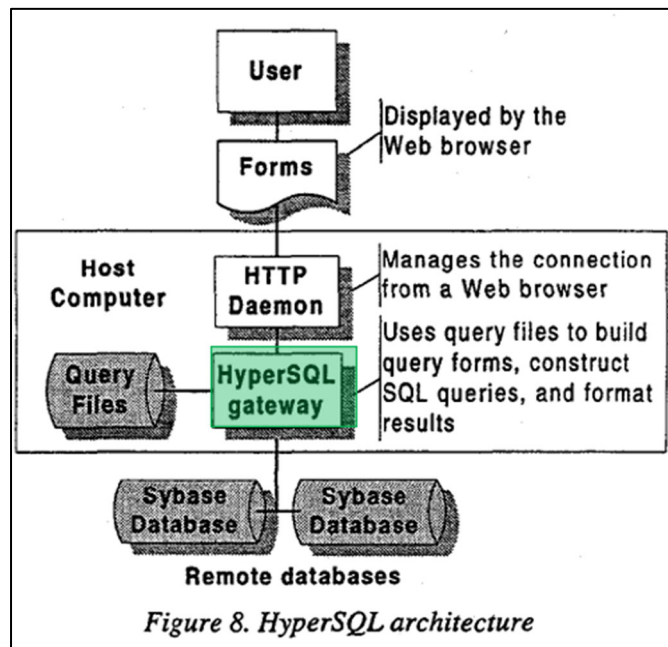
416. Specifically, Figure 8 shows that a user uses a “Web browser” on their device to access the website running on the Web Server (*e.g.*, HTTP Daemon and HyperSQL gateway in green at the “Host Computer”).

N. Claim 29

- 1. “The method of claim 3, wherein the database is coupled to a node in a telecommunications network, the method further comprising accessing the node in the telecommunications network.”**

417. HyperSQL is implemented on the Web (Internet). *See* §XIII.M above (Claim 28). A POSITA would have understood the Internet to be “[a] *telecommunications network*,” and that communications between computers and website servers would occur through “nodes,” like servers, computers, and networking hardware. *See also* §VII.A above.

418. The HyperSQL database(s) communicate with (“*is coupled to*”) other computers and devices, including the “HyperSQL gateway” (Figure 8) and “Web Server” (Figure 9) (“*the database is coupled to a node in a telecommunications network*”). Ex. 1006 (HyperSQL), 333. Specifically, the Web Server with the HyperSQL gateway is “*a node in a telecommunications network*.”



Id., Figure 8 (annotated).

419. As shown above in Figure 8 the “HyperSQL gateway” (in green) is something (*e.g.*, software) running on a Host Computer (“Web Server”). *See also* §VII.A above.

420. The user accesses the Web Server by, *e.g.*, inputting and submitting information on the query form (“*accessing the node in the telecommunications network*”). *Id.*

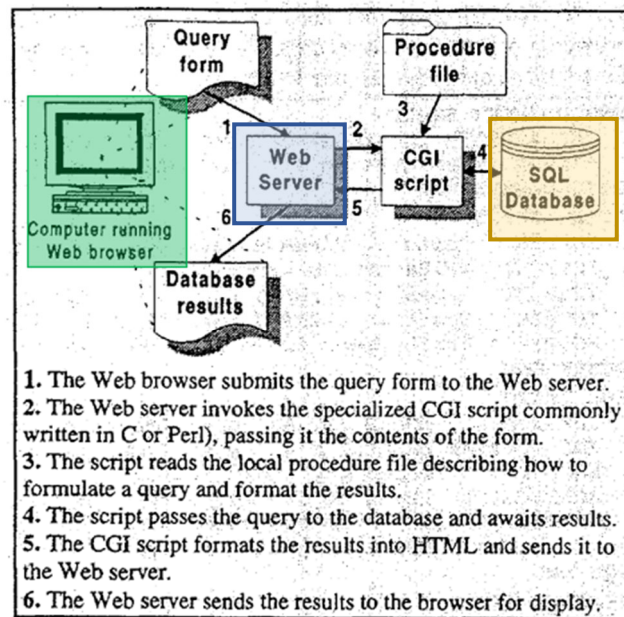


Figure 9. How CGI-based query interfaces work

Ex. 1006 (HyperSQL), Figure 9 (annotated).

421. As shown in annotated Figure 9 above, the user of the highlighted “Computer” (in green) accesses the “Web Server” (in blue) by inputting information on “Query form” and receiving “Database results” (from the database(s), annotated in yellow) back from the “Web server.”

422. In sum, HyperSQL discloses “*the database is coupled to a node in a telecommunications network, the method further comprising accessing the node in the telecommunications network*” because the SQL database(s)/Sybase Database(s) communicate with the Web Server over a telecommunications network such as the Internet to allow a user on a Computer with a Web browser to retrieve information from the databases by accessing the Web Server using the HyperSQL interface.

O. Claim 30

1. “The method of claim 3, wherein the database is coupled to a terminal for displaying the data.”

423. A user can leverage HyperSQL to create “forms- and hypertext-based database interfaces for non computer experts.” Ex. 1006 (HyperSQL), 329. HyperSQL seeks “to make it easier for scientists to retrieve information from [] databases using common Web browsers.” *Id.*, 330. Thus, the database communicates data to (“*the database is coupled to*”) a “Computer running Web browser” (“*a terminal for displaying the data*”). *See id.*, 329-30, 333, 335-36, Figures 2-6 and 9. This is plainly seen from any of the figures of HyperSQL showing the values “from the database” displayed in querylists, or the data shown in the results and browse screens, a representative subset are reproduced below with blue color highlighting some of the “data”:

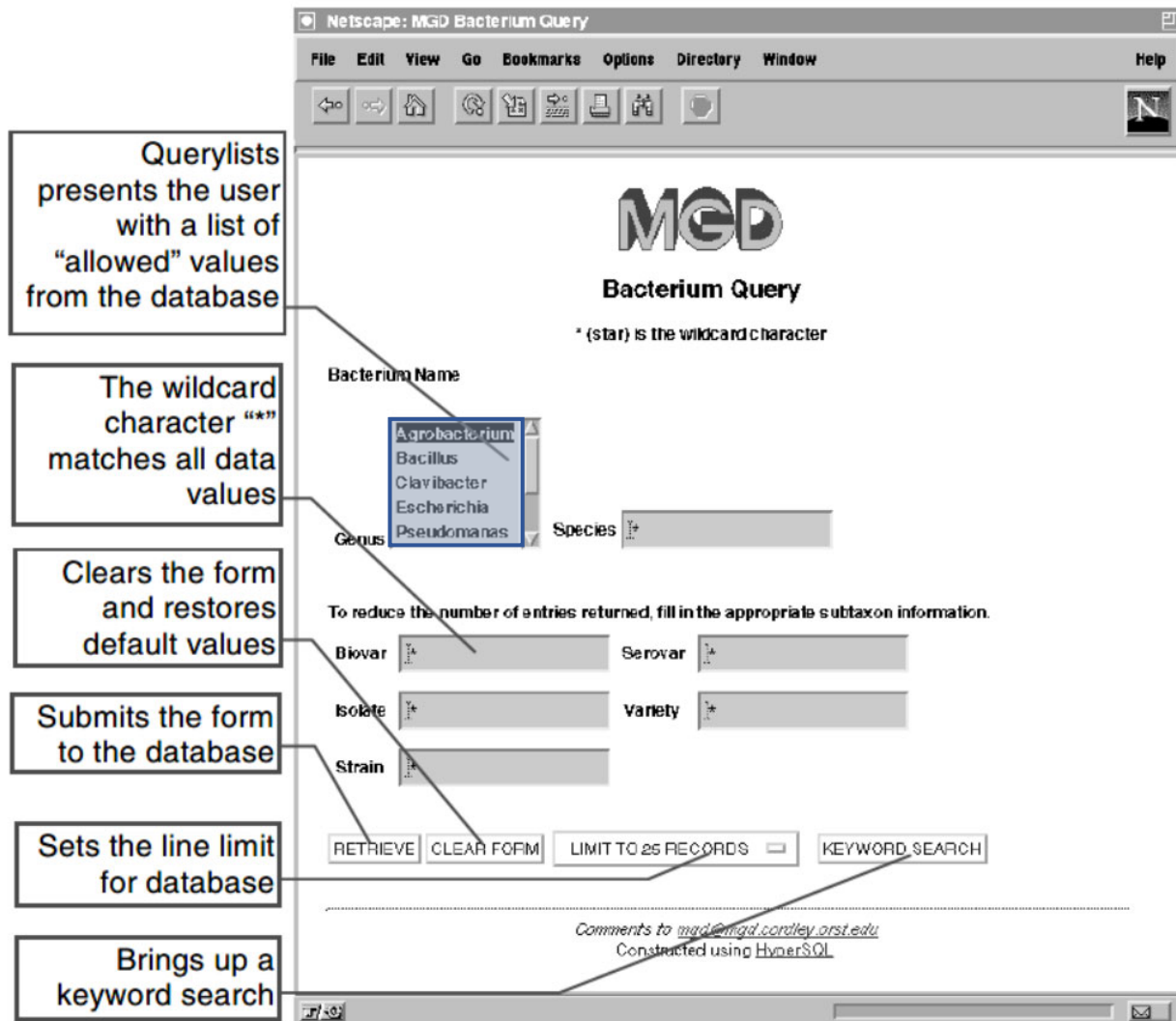


Figure 2. Example of a *query form* (shown in a Netscape browser)

Id., Figure 2 (annotated).

424. Figure 2 above shows data displayed in a querylist (in blue) on a query form.

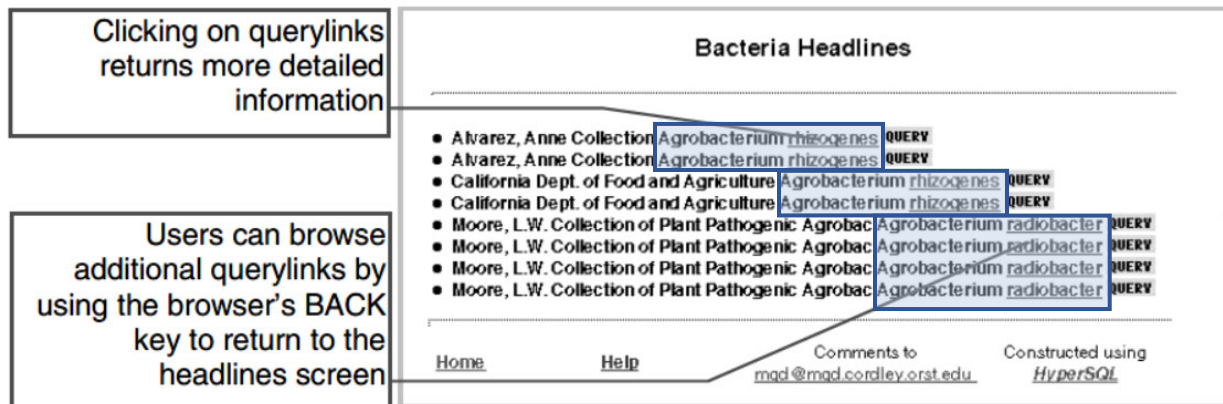


Figure 3. Example of a **results** screen

Ex. 1006 (HyperSQL), Figure 3 (annotated).

425. Figure 3 above shows data displayed in a results screen (in blue).

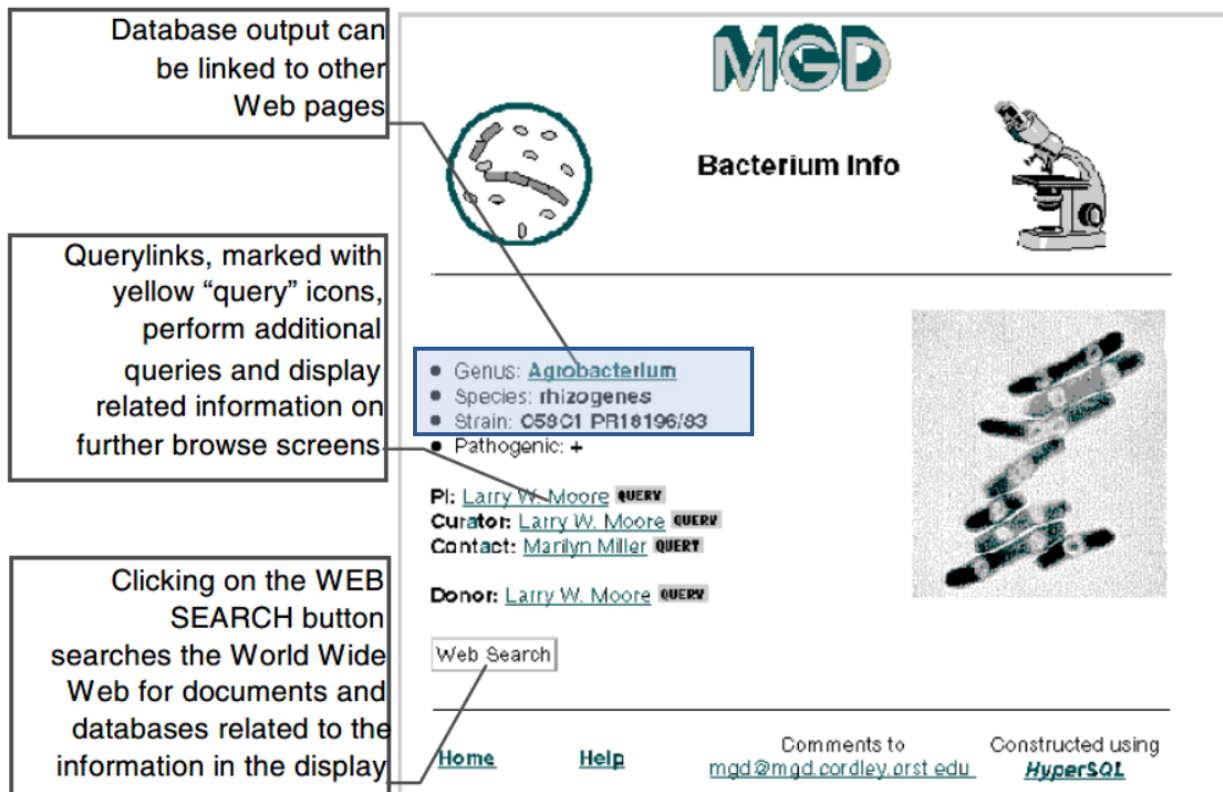


Figure 4. Example of a **browse** screen

Ex. 1006 (HyperSQL), Figure 4 (annotated).

426. Figure 4 above shows data displayed in a browse screen (in blue).

427. In sum, HyperSQL discloses “*the database is coupled to a terminal for displaying the data*” because the database communicates the data to the terminal for display in a browser at the terminal using various HyperSQL screens (including on a query form in a querylist, on a results screen, and a browser screen).

XIV. GROUND 3: HYPERSQL IN VIEW OF ZELLWEGER, MICROSOFT ACCESS, AND THE KNOWLEDGE OF A POSITA RENDERS OBVIOUS CLAIMS 21-23

428. It is my opinion that a POSITA would have considered HyperSQL, Zellweger, and Microsoft Access to be within the scope of the prior art and highly relevant to the patentability of the 720 Patent's claims.

429. It is my opinion that HyperSQL in view of Zellweger and Microsoft Access, especially when considering the knowledge of a POSITA at the time, renders obvious Claims 21-23. The basis for my opinion is set forth in more detail below.

430. A POSITA would have been motivated to combine the teachings of HyperSQL, Zellweger, and Microsoft Access, which are analogous art, and would have had a reasonable expectation of success when doing so as detailed in §XVII.C below.

A. Claim 21

1. **[21.A] “The method of claim 3, wherein the method for searching the database comprises searching a first database and a second database, and further comprising:”**

431. HyperSQL explicitly discloses searching two databases—a first “Sybase Database” and a second “Sybase Database”—which discloses “*the method for searching the database comprises searching a first database and a second*

database.” Ex. 1006 (HyperSQL), 335, Figure 8; *see also id.*, 329-30. Specifically, Figure 8 shows the following:

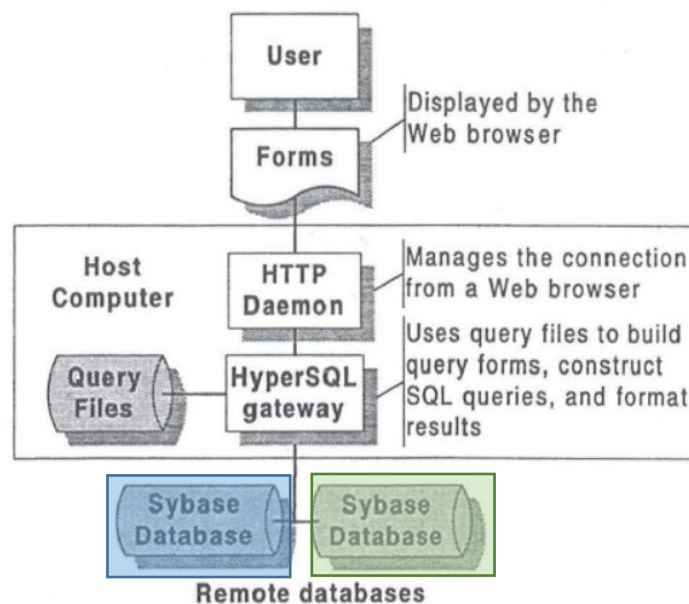


Figure 8. HyperSQL architecture

Id., Figure 8 (annotated).

432. As seen in Figure 8 above, there are two separate “Databases,” here highlighted in blue and green. Figure 8 also discloses that the “HyperSQL gateway” “uses query files to build query forms, construct SQL queries, and format results” from the databases.

433. HyperSQL discloses searching against non-centrally, remotely located “databases ... maintained under the control of individual scientists/collectors.” Ex. 1006 (HyperSQL), 329. HyperSQL gives examples of such databases: “Genome

Database,” “Microbial Germplasm Database,” “National Fungus Collection,” and “Nature Conservancy Database.” *Id.* HyperSQL’s abstract also affirms that HyperSQL will need to query multiple “databases” (emphasis to pluralization added) and says that HyperSQL has already been “successfully implemented for three major NSF-sponsored biological databases.” *Id.* A POSITA would have understood that HyperSQL searches multiple, separate, remotely located databases.

2. [21.B] “receiving a first search result from the search of the first database;”

434. Because HyperSQL is an interface used to query multiple databases, it would have been clear to a POSITA that HyperSQL queries constructed pursuant to a user filling in querylists and submitting HyperSQL query forms would have been sent to, and results returned from, at least one (a first) Sybase Database, which constitutes “*a first search result from the search of the first database.*” *See* §XIV.A.1 above (Claim [21.A]). Indeed, HyperSQL confirms that it is “for constructing Web-based query interfaces to SQL databases” to retrieve results from the databases. Ex. 1006 (HyperSQL), 330-31, 333.

3. [21.C] “receiving a second search result from the search of the second database; and”

435. During my discussion of Claims [21.A] and [21.B] above, I explained how HyperSQL discloses multiple databases and that HyperSQL queries at least a first of the databases for a result. *See* §§XIV.A.1-XIV.A.2 above (Claims [21.A]-

[21.B])). In addition, a POSITA would have known how to submit search queries to multiple databases. *See* §VII.F above.

436. HyperSQL further discloses that it supports “multi-database management capabl[e]” database systems. Ex. 1006 (HyperSQL), 335. HyperSQL discloses that it would be used in situations “requiring data from multiple sources.” *Id.*, 329.

437. It would have been well understood to a POSITA that in a multi-database system described by HyperSQL, where data is stored in multiple sources, the queries are sent to and fulfilled by various databases, depending on where the requested data is stored. For example, when retrieving “data from multiple sources” (such as the system of two Sybase Databases in HyperSQL Figure 9, or the “three major NSF-sponsored biological databases,” Ex. 1006 (HyperSQL), 329), a POSITA would have understood that HyperSQL queries must be sent to, and results returned from, at least the second Sybase Database (“*a second search result from the search of the first database*”) in addition to the first Sybase Database; otherwise the search would be unable to retrieve the “data from multiple sources” as disclosed by HyperSQL. *See also* §XIV.A.1 above (Claims [21.A]).

4. [21.D] “combining the first and the second search results.”

438. HyperSQL states that some uses may “requir[e] data from multiple sources.” Ex. 1006 (HyperSQL), 329. A POSITA would have understood from this

and other disclosures in HyperSQL that HyperSQL necessarily combines results from the first and second Sybase Databases, which is the claimed “*combining the first and the second search results.*” Because HyperSQL can use at least two Sybase databases, a POSITA would have understood that records responsive to a search may be stored in only one database (*e.g.*, when a particular collection of samples has unique samples in its database) or in both databases (*e.g.*, when multiple collections/donors have some of the same samples in their databases). See §§XIV.A.2-XIV.A.3 above (Claims [21.B]-[21.C]). This is further supported by HyperSQL Figure 3, reproduced below.

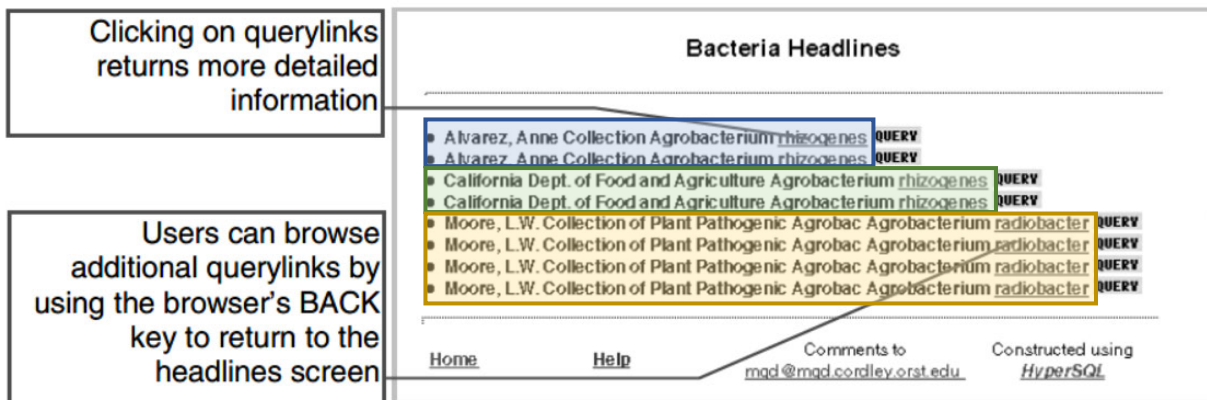


Figure 3. Example of a *results* screen

Ex. 1006 (HyperSQL), Figure 3 (annotated).

439. Figure 3 above shows a listing of samples from three distinct collections or donors: (1) Alvarez, Anna Collection Agrobacterium (in blue), (2) California Dept. of Food and Agriculture Agrobacterium (in green), and (3) Moore, L.W. Collection of Plant Pathogenic Agrobacterium (in yellow). As HyperSQL

discloses, the records for each of the collections/donors are likely stored in separate “databases [which] are not likely to be centrally located.” Ex. 1006 (HyperSQL), 329. This is further supported by the fact that the three collections listed in Figure 3 above appear to be owned by different entities: Alvarez, Anne; California Dept. of Food and Agriculture, and Moore, L.W.

440. Because the user would be searching and receiving results across several distinct databases, the results from the databases must be combined together when a user presses the “RETRIEVE” button on the query form to “submit[] the form and return[] results.” Ex. 1006 (HyperSQL), 331.

441. Figure 3 above shows that the results of querying the three different collections is shown in a single concise display. A POSITA would have understood that this unified display therefore “combines” the different results into a single output. I note that the claim does not appear to require an explicit SQL JOIN command to logically merge the results into a single data set. Nevertheless, a POSITA would have understood that SQL databases support several JOIN commands that could easily be used to logically merge results into a single data set. *See also* §§VII.C and VII.F above.

442. Next, Microsoft Access also discloses combining search results. Specifically, Microsoft Access discloses “[l]ogical operators (also called *Boolean operators*) are used most often to **combine the results** of two or more comparison

expressions into a single result,” Ex. 1009 (Microsoft Access), 291 (emphasis added), which comprises the “*combining the first and the second search results.*” Indeed, a POSITA would have known that such logical operators (*e.g.*, AND, OR, XOR, NOR, etc.) are frequently used in database queries. *See also* §VII.C above.

443. In sum, HyperSQL discloses “*combining the first and the second search results*” because a user may search multiple databases and therefore HyperSQL would have to combine results from the multiple databases to provide a results screen. Microsoft Access discloses “*combining the first and the second search results*” because it explicitly discusses using “[l]ogical operators” to “combine the results of two or more comparison expressions.”

B. Claim 22

1. “The method of claim 21, wherein the combining step comprises a Boolean AND operation.”

444. Microsoft Access discloses multiple “[b]oolean operators” including an “And” operation:

Table 9.3 Logical Operators			
Operator	Description	Example 1 Example 2	Result 1 Result 2
And	Logical and	True And True True And False	True False
Or	Inclusive or	True Or False False Or False	True False
Not	Logical not	Not True Not False	False True
Xor	Exclusive or	True Xor False True Xor True	True False

Ex. 1009 (Microsoft Access), 291.

445. A POSITA would have understood that the “And” operation disclosed in Microsoft Access would be implemented in the HyperSQL/Zellweger combination to generate a result set containing results that exist in both Sybase Databases, which comprises “*the combining step comprises a Boolean AND operation.*” This would capture any *redundant* results across the different databases. This may occur, for example, when a user wants to identify samples that have been collected by multiple donors (*e.g.*, where each donor/collection has its own database). See Ex. 1006 (HyperSQL), Figures 3, 4.

446. For example, Figure 3 shows several results from various donors/collections:

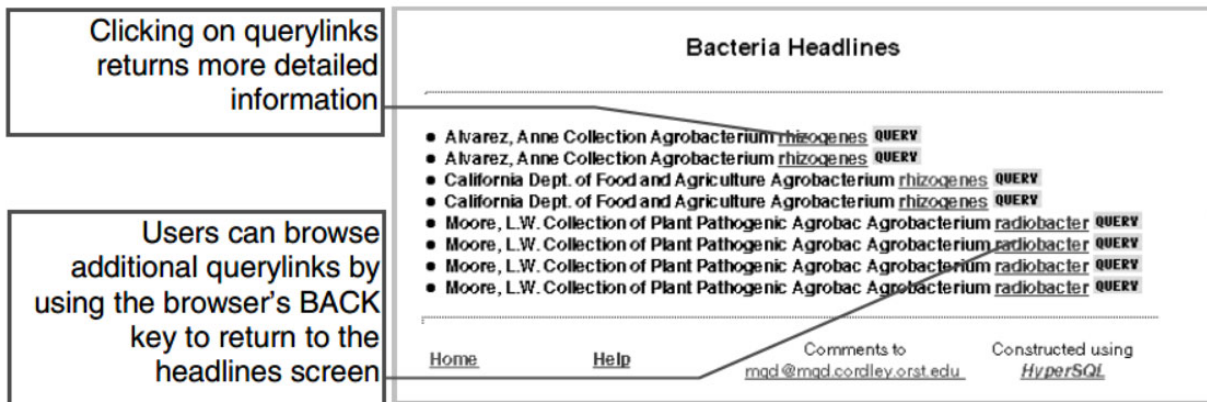


Figure 3. Example of a results screen

Ex. 1006 (HyperSQL), Figure 3.

447. Figure 3 shows that the “Alvarez, Anne Collection” and “California Dept. of Food and Agriculture” collections each store results for the criteria “Agrobacterium” and “rhizogenes,” and thus likely may have overlapping results. A POSITA would have used an AND combination operation to identify results with such overlap. *See also* §VII.C above.

C. Claim 23

1. “The method of claim 21, wherein the combining step comprises a Boolean OR operation.”

448. Microsoft Access discloses multiple “[b]oolean operators” including an “Or” operation:

Table 9.3 Logical Operators			
Operator	Description	Example 1 Example 2	Result 1 Result 2
And	Logical and	True And True True And False	True False
Or	Inclusive or	True Or False False Or False	True False
Not	Logical not	Not True Not False	False True
Xor	Exclusive or	True Xor False True Xor True	True False

Ex. 1009 (Microsoft Access), 291.

449. A POSITA would have understood that the “Or” operation would be implemented in the HyperSQL/Zellweger combination by returning results where each result is in either the first *or* the second Sybase Database, which comprises “*the combining step comprises a Boolean OR operation.*” This would capture *every* result in either database. This may occur, for example, when a user wants to identify samples that have been collected by *any* donors (*e.g.*, where each donor/collection has its own database). *See* Ex. 1006 (HyperSQL), Figures 3, 4.

450. For example, Figure 3 shows several results from various donors/collections:

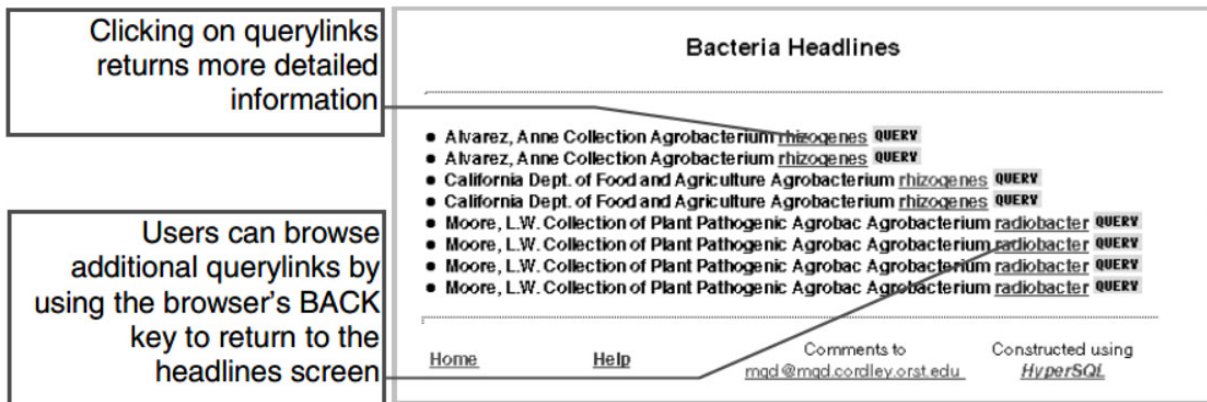


Figure 3. Example of a *results* screen

Ex. 1006 (HyperSQL), Figure 3.

451. Figure 3 shows that the “Alvarez, Anne Collection,” “California Dept. of Food and Agriculture,” and “Moore, L.W. collection of Plant Pathogenic Agrobac” collections each store results for the criteria “Agrobacterium” as “Genus” and “rhizogenes” or “radiobacter” as “Species.” If a user runs a search to capture all results for the Genus of “Agrobacterium” that have been collected by *any* donor, the POSITA would need to use an OR combination operation to ensure that HyperSQL returns all results in any of these collections databases. *See also* §VII.C above.

XV. GROUND 4: HYPERSQL IN VIEW OF ZELLWEGER, MICROSOFT ACCESS, JANES, AND THE KNOWLEDGE OF A POSITA RENDERS OBVIOUS CLAIM 25

452. It is my opinion that a POSITA would have considered HyperSQL, Zellweger, Microsoft Access, and Janes to be within the scope of the prior art and highly relevant to the patentability of the 720 Patent's claims.

453. It is my opinion that HyperSQL in view of Zellweger, Microsoft Access, and Janes, especially when considering the knowledge of a POSITA at the time, renders obvious Claim 25. The basis for my opinion is set forth in more detail below.

454. A POSITA would have been motivated to combine the teachings of HyperSQL, Zellweger, Microsoft Access, and Janes, which are analogous art, and would have had a reasonable expectation of success when doing so as detailed in §XVII.D below.

A. Claim 25

1. **“The method of claim 21, wherein the first search result is represented by a first icon and the second search result is represented by a second icon, and wherein the combining step comprises overlaying the first and the second icons.”**

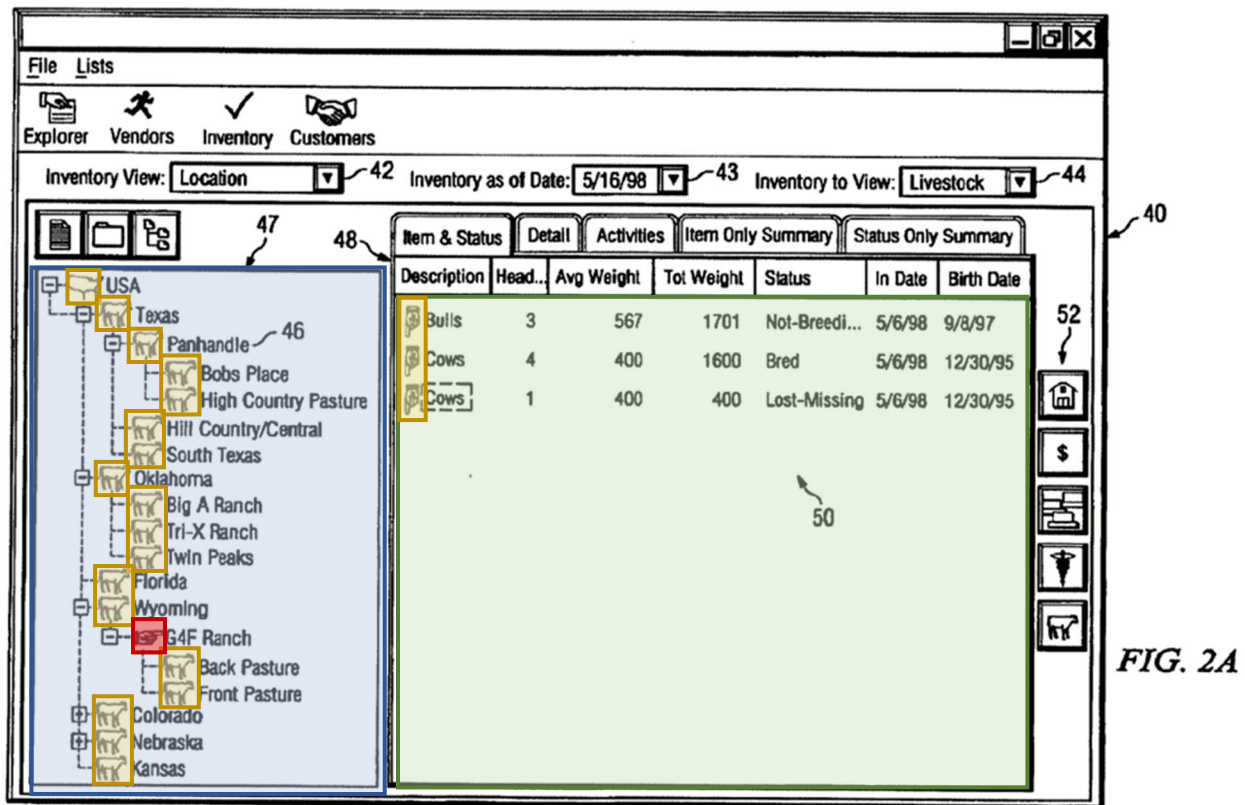
455. HyperSQL discloses that the results it displays “are automatically formatted as graphics and hypertext.” Ex. 1006 (HyperSQL), 329. A POSITA would have understood this to mean that HyperSQL may display the first result and second result as described related to Claims [21.A]-[21.C] (at §§XIV.A.1-XIV.A.3 above), each “formatted as graphics and hypertext,” which corresponds to the

limitation “*the first search result is represented by a first icon and the second search result is represented by a second icon.*”

456. Janes discloses an “inventory system having a database storing inventory data” and “a graphical user interface operable to display data.” Ex. 1012 (Janes), Abstract. Janes’s interface includes “displayed elements” such as “clickable icons and buttons.” *Id.*, 4:30-38, 8:61-9:11, 9:41-45, 11:44-55, 12:29-33.

457. Janes also discloses a drag-and-drop system that users can use to click on and “drag” an element over another element and “drop” it to associate the two elements. Ex. 1012 (Janes), 2:48-62. Janes explains that the “drag and drop method ... may be used to select any displayed element as the data source ... to any other displayed element as the data target” *Id.*, 12:34-44. A POSITA would have understood from this disclosure that Janes’s teachings of completing operations via drag-and-drop may be accomplished with any elements displayed on screen. *See also* §§VII.E and VII.F above.

458. For example, Janes discloses applying these teachings in the context of a livestock management system. *See e.g.*, Ex. 1012 (Janes), Title, Abstract, 1:13-16, 2:10-19, Figures 2A-2J, 4A-11. This is suggested by Janes’s title—“Livestock Inventory and Materials System With Interactive Graphical User Interface.” *Id.*, Title. Janes is complete with many figures showing this graphical system; an exemplary figure, Figure 2A, is pasted below for purposes of discussion:



Id., Figure 2A.

459. As seen in Figure 2A above, farms and groups of farms are displayed as nodes in a hierarchical menu on the left (in blue) and the inventory of the selected node is shown on the right (in green). Each node and inventory item is displayed with an icon (in yellow). Ex. 1012 (Janes), 4:48-5:60. Figure 2A shows the state of the screen when the “G4F Ranch” in “Wyoming” has been selected (indicated with a left-pointed finger icon in red). On the right side of the GUI is listed one row of Bulls, and two rows of Cows, which is the listing of the inventory associated with the G4F Ranch. *Id.*, 5:15-38.

460. In this system, Janes discloses that the drag-and-drop feature may be used to allow users to easily (and graphically) complete operations such as “moving livestock and other inventory from one location to another.” Ex. 1012 (Janes), 5:51-60. Using drag-and-drop a user can simply “pick up inventory from one location and drop it in another location ... using a drag and drop technique.” *Id.*

461. It would have been obvious to a POSITA to use the drag-and-drop techniques taught by Janes to perform operations graphically, and that these techniques could be integrated into HyperSQL. With such a combination, a user would search multiple databases/collections, receive the results from the various databases/collections, and then combine those results using an easy-to-use graphical interface.

462. Specifically, a user would easily drag and drop an icon or text representation of one search result onto the other search result to visually combine them, which comprises the claimed “*combining step comprises overlaying the first and the second icons.*” For example, a Janes user could first drag-and-drop the Bulls icon in Figure 2A above, from its current location (a source ranch) to another (destination) ranch, thus indicating the desired movement of said Bulls over to the new ranch location. Thereafter, the user could select, say, the second row of Cows in the source ranch, and drag that icon over to the new destination ranch, the same as the Bulls. Doing so will effectively combine the results of two searches, each

denoted with its own icon, onto a single destination ranch, thus overlaying the icons onto a single destination location.

463. In sum, HyperSQL discloses “*the first search result is represented by a first icon and the second search result is represented by a second icon*” because HyperSQL’s results from searches (*i.e.*, first search results from a first database and second search results from a second database) are formatted as graphics and hypertext. Alternatively, Janes discloses “*the first search result is represented by a first icon and the second search result is represented by a second icon*” because the results from the search for the nodes in the hierarchy (first search results) and the results for the inventory at a specific node in the hierarchy (second search results) are displayed with icons. HyperSQL in view of Janes discloses “*wherein the combining step comprises overlaying the first and the second icons*” because Janes discloses interacting with database data by dragging and dropping the icons representing search results.

XVI. GROUND 5: COCHRAN IN VIEW OF ZELLWEGER, JANES, SPIEGEL, AND THE KNOWLEDGE OF A POSITA RENDERS OBVIOUS CLAIM 26

464. It is my opinion that a POSITA would have considered Cochran, Zellweger, Janes, and Spiegel to be within the scope of the prior art and highly relevant to the patentability of the 720 Patent's claims.

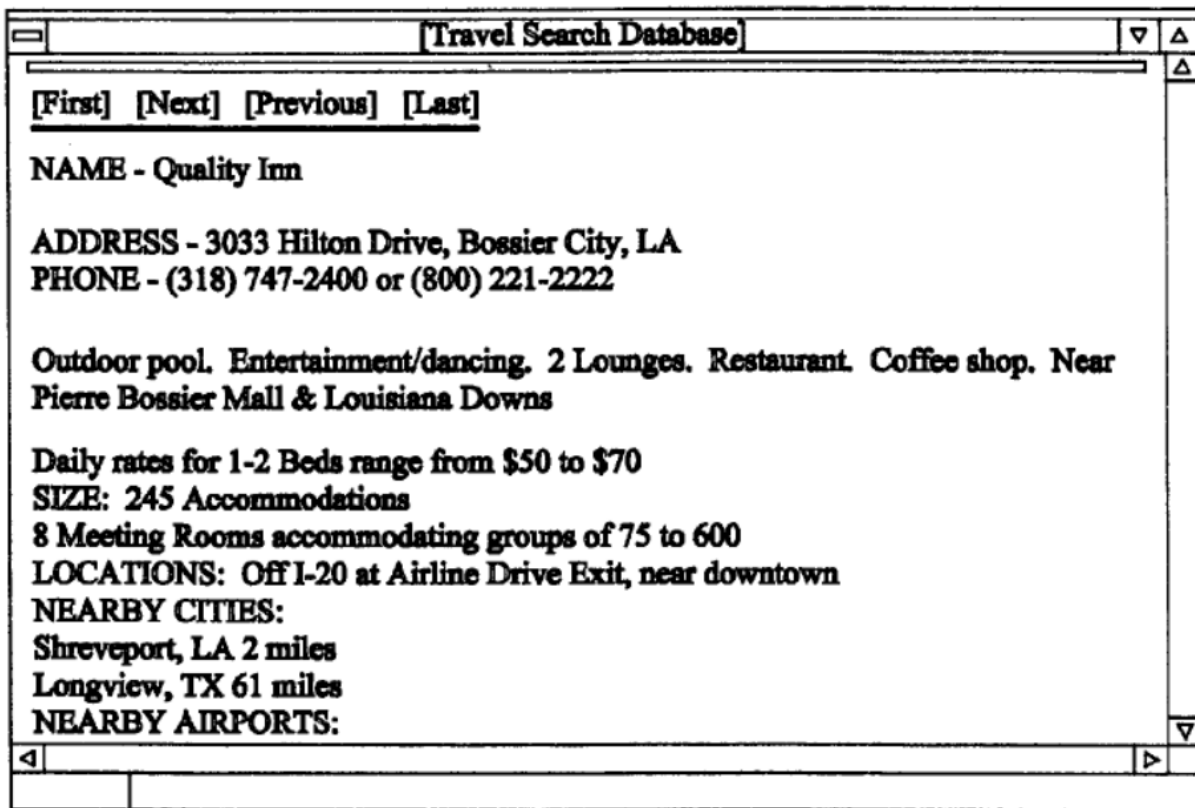
465. It is my opinion that Cochran in view of Zellweger, Janes, and Spiegel, especially when considering the knowledge of a POSITA at the time, renders obvious Claim 26. The basis for my opinion is set forth in more detail below.

466. A POSITA would have been motivated to combine the teachings of Cochran, Zellweger, Janes, and Spiegel, which are analogous art, and would have has a reasonable expectation of success when doing so as detailed in §XVII.E below.

A. Claim 26

1. [26.A] “The method of claim 3, wherein the displayed data includes one or more data records,”

467. Cochran discloses a screen to “view the [search result] records” when the user clicks the “View All” button. Ex. 1005 (Cochran), 13:2-14. Thus, Cochran displays “*data where the data includes data records.*” An example of this is shown in Figure 4c, although that figure just shows “[p]art of a record.” *Id.*



Id., Figure 4c.

468. Figure 4c above depicts a part of search records. Where there are several records on the search result records screen, the user can “scroll the screen” to view the other records returned pursuant to the user’s search. Ex. 1005 (Cochran) 13:2-7.

469. In sum, Cochran discloses “*wherein the displayed data includes one or more data records*” because it discloses a screen which may display multiple records on a screen after a user presses the “View All button.”

2. **[26.B] “each of the one or more data records represented by one or more of text title and an icon, and further comprising overlaying a text title or an icon on a symbol representing an online purchase,”**

470. As I discussed related to Claim 25, and discussed in my summary of the prior art, Janes discloses an inventory system where data records are “*represented by one or more of text title[s] and an icon.*” See §§XI.A.5 (Janes prior art) and XV.A above (Claim 25).

471. Spiegel, like Janes, discloses dragging and dropping. However, Spiegel discloses using dragging and dropping as a part of purchasing items online. Specifically, Spiegel teaches a “computer method and system of electronic commerce.” Ex. 1013 (Spiegel), Title, Abstract, 1:6-9. Spiegel recognizes that “[t]he World Wide Web portion of the Internet is especially conducive to conducting electronic commerce” and presents an improved “system for conducting electronic commerce” on the Internet. *Id.*, 2:1-20, 2:61-3:16. Spiegel suggests “dragging-and-dropping items into ... shopping carts” to provide a more user-friendly way to purchase items online. *Id.*, 10:31-54. See also §§VII.E and VII.F above.

472. A POSITA would have modified the Cochran/Zellweger combination with Janes’s and Spiegel’s teachings such that users of Cochran’s Travel Search system could purchase stays at hotels through the Cochran system. Ex. 1005 (Cochran), 6:58-64. First, after applying Janes’s teachings, Cochran’s records

screen would display the data records for the “hotels, resorts and inns” as a combination of icons and text. *Id.* Janes already discloses that such icons and texts can be hooked into a drag-and-drop system. Ex. 1012 (Janes), 2:48-62. Specifically, Janes states that the “drag and drop method ... may be used to select any displayed element as the data source ... to any other displayed element as the data target” *Id.*, 12:34-44.

473. Second, after applying Spiegel’s teachings, Cochran would include a shopping cart icon on screen such that a user may drag the icons and/or text for the “hotels, resorts and inns” over the shopping cart and then drop it onto the shopping cart (*e.g.*, the shopping cart in Spiegel) to purchase stays at one of the hotels, resorts, or inns listed on Cochran’s search results page, which constitutes the claimed “*overlaying a text title or an icon on a symbol representing an online purchase.*”

474. In sum, Cochran, Janes, and Spiegel disclose “*each of the one or more data records represented by one or more of text title and an icon, and further comprising overlaying a text title or an icon on a symbol representing an online purchase*” because in the combined system a user may drag and drop a record (using an icon and text representation as taught by Spiegel) for a hotel, resort, or inn using the drag and drop techniques taught by Spiegel—onto a shopping cart as taught by Spiegel—to complete a purchase.

3. [26.C] “wherein the step of overlaying initiates an online purchase of an item represented by the text title or icon.”

475. Spiegel discloses “dragging-and-dropping items into [] various shopping carts.” Ex. 1013 (Spiegel), 10:43-54. The “purchaser” (*i.e.*, the user) places items into the shopping cart to hold items to buy. *Id.*, 2:21-34. A “shopping cart [may] already ha[ve] the appropriate billing and shipment information” associated with it. *Id.*, 4:19-24. That is, a user may provide billing and shipping information and associate it with a shopping cart, such that the cart has all information necessary to complete a purchase.

476. A POSITA would have understood that given the shopping cart has all the information needed to complete a purchase, the user could make a trivial adjustment to the system such that that a purchase is initiated when a user drags and drops a product’s text or icon onto the shopping cart, which would make the shopping process quicker and more efficient. This corresponds to the claimed “*the step of overlaying initiated an online purchase of an item represented by the text title or icon.*” See also §§VII.E and VII.F above.

477. In sum, the combination of Cochran, Janes, and Spiegel teach “*wherein the step of overlaying initiates an online purchase of an item represented by the text title or icon*” because Spiegel teaches dragging and dropping onto a shopping cart to

Declaration of Erez Zadok, Ph.D.,
In Support of *Inter Partes* Review of U.S. Patent No. 6,760,720

initiate a purchase because shopping carts contain all information required to make a purchase (*i.e.*, billing and shipping information).

XVII. MOTIVATIONS TO COMBINE

478. Below I discuss why it is my opinion that a POSITA would have been motivated to combine the prior art references in each of the combinations for the Grounds presented in §§XII, XIII, XIV, XV, and XVI above.

A. Motivations to Combine Cochran and Zellweger

479. A POSITA would have been motivated to combine Cochran and Zellweger for at least six reasons as described next.

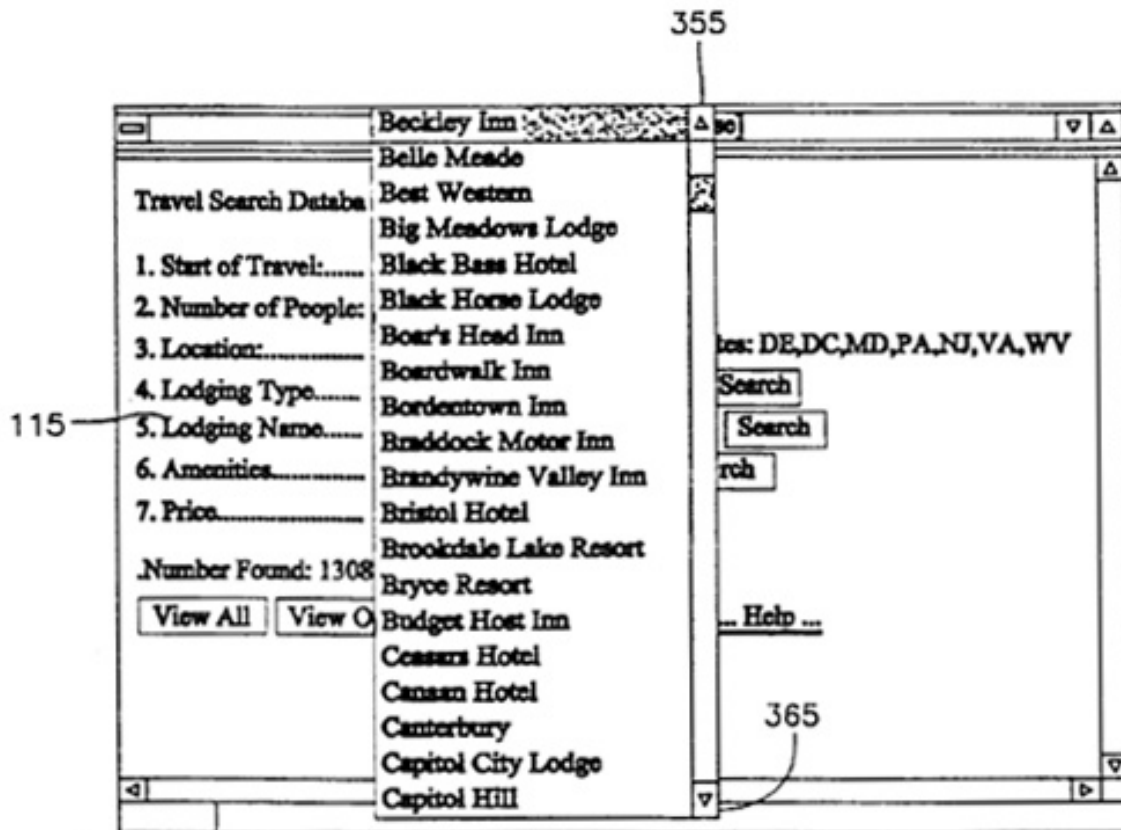
480. First, Cochran and Zellweger are both directed to the same field (accessing information in databases) and address similar concepts (making information more easily accessible using graphical interfaces). Cochran and Zellweger also both teach several of the same concepts. For example, Cochran and Zellweger both teach graphical user interfaces to access or display database information. *Compare Ex. 1005 (Cochran)*, Abstract (“A method for managing the presentation of search terms on a computer terminal or other display device so that large computer databases can be searched efficiently.”), 1:14-20 (“The present invention relates to a method for accessing records from a computer database. More specifically, the present invention relates to a method for accessing records from a computer database by first displaying a plurality of lists of search terms available in the database.”), 4:2-57 (“It is our object of the present invention to provide a search system that efficiently displays information to users that are searching for particular

information in a database.”); *with* **Ex. 1007 (Zellweger)**, Abstract (“The present invention discloses a menu authoring system component that enables a developer to generate and build a tab list menu. The tab menu divides a target list menu into tab intervals that can include a specified number of letters or full words.”), 2:1-6 (“Zellweger’s disclosures show how to automatically generate networks of lists and paths from external sources of data such as database structures...”), 2:30-43 (“The present invention discloses a menu authoring system that generates a tab list menu that links to a set of sublists which, previously, represented one long continuous list of menu items.”).

481. Second, both Cochran and Zellweger are implemented as computer software. *Compare* **Ex. 1005 (Cochran)**, 5:40-49 (“The system and method of the present invention is implemented by computer software which permits the accessing of data from a computer database.”) *with* **Ex. 1007 (Zellweger)**, 1:35-40 (“The present invention relates to computer software used to build and maintain a menu system ...”). A POSITA would have appreciated that Cochran’s and Zellweger’s software teachings were well-known and complementary to one another.

482. Third, Cochran states that it endeavors to “provide an efficient and user-friendly method of searching computer databases” using “a search system that efficiently displays information to users.” **Ex. 1005 (Cochran)**, 4:4-19. Cochran acknowledges that there are times that it has to display more items than a pick box

can display at one time, which forces a user to, scroll through all the options, which could be cumbersome if there are many options. *Id.*, 10:58-65 (“If the expanded pick-box area is too small to display all of the available search terms, then scrolling is permitted within the confines of the expanded pick box.”). A POSITA would have understood that the lists in Cochran may be fairly long and that they would benefit from some means to improve the ability to navigate through the options in the menu.



Id., Figure 4e.

483. The expanded pick box list in Figure 4e above show a list of Lodging Names which could very well have hundreds or thousands of hotel names depending on the other criteria already selected by the user.

484. This is a problem that Zellweger sought to alleviate. Specifically, Zellweger discloses “menus” that can be used for lists with “hundreds, if not thousands of entries, [where] scrolling for a particular entry can be extremely tedious.” Ex. 1007 (Zellweger), Abstract.

485. Accordingly, a POSITA implementing the system of Cochran would have looked to Zellweger to improve the usability of the pick boxes. Specifically, a POSITA would have been motivated to create Zellweger’s Alphabet or Content Lists instead of scroll bars when, for example, the number of available entries exceeds the number of entries that can be presented on a given screen.

486. Fourth, this combination would be nothing more than using one known method (Zellweger’s menus) to improve another known method (Cochran’s database search system) using the express teachings of Zellweger. Indeed, this is the problem that Zellweger was designed to alleviate. Ex. 1007 (Zellweger), Abstract (“Hierarchical menus provide end-users with an easy way to locate information. The underlying hierarchical menu structure is often scalable as there are no limits on the number of entries in a list menu. ... As a list menu can contain hundreds, if not thousands of entries, scrolling for a particular entry can be extremely

tedious. The present invention discloses a menu authoring system component that enables a developer to generate and build a tab list menu.”).

487. Fifth, the combination would have yielded predictable results because of the straightforward application of Zellweger’s teachings in the manner envisioned by Zellweger (*i.e.*, using Zellweger’s menu to list out entries from a list that would be voluminous if listed in full). Moreover, Cochran expressly teaches that other methods for implementing pick boxes known in the art could be used to display “wide ranges” of terms. *See* Ex. 1005 (Cochran), 10:61-65 (“The expansion of pick boxes to enable display of a wide range of terms may be done in accordance with routine programming steps that are apparent to persons of ordinary skill in the computer programming arts.”). A POSITA developing GUI software would have known, for example, to replace a call to display a scroll bar with a call to display menu list; this would have represented a small change to the code.

488. Sixth, a POSITA would have had a high expectation of success in adapting Cochran in view of Zellweger’s teachings of the Alphabet and Content List menus. Zellweger was created with the intention to improve systems such as Cochran that have graphical interfaces with long lists of entries. Furthermore, as I explained at the top of this section both Cochran and Zellweger involve displaying database information in a GUI. Because of this a POSITA would have understood that altering Cochran to truncate entries like in Zellweger’s list menus would require

nothing more than routine programming changes to update the type of menus being used in Cochran depending on how many entries need to be listed in a Cochran pick box.

489. Finally, a POSITA would have known that databases, web technologies, and GUIs—as well as technologies to search, summarize, and display that combine them—were all well known in the art, as I detail in the technology background §§VII.B-VII.F above.

B. Motivations to Combine HyperSQL and Zellweger

490. A POSITA would have been motivated to combine HyperSQL and Zellweger for at least six reasons as described next.

491. First, HyperSQL and Zellweger are both directed to the same field (accessing information in databases) and address similar concepts (making information more easily accessible using graphical interfaces). HyperSQL and Zellweger both teach several of the same concepts such as graphical user interfaces to access or display database information. *Compare Ex. 1006 (HyperSQL)*, 329 (“HyperSQL is an interoperability layer that enables database administrators to rapidly construct browser-based query interfaces to remote Sybase databases. ... HyperSQL can be used to create forms- and hypertext-based database interfaces for non computer experts (e.g., scientists, business users). Such interfaces permit the user to query databases by filling out query forms selected from menus. No

knowledge of SQL is required because the interface automatically composes SQL from user input.”), Figures 2-6; *with Ex. 1007 (Zellweger)*, Abstract (“The present invention discloses a menu authoring system component that enables a developer to generate and build a tab list menu. The tab menu divides a target list menu into tab intervals that can include a specified number of letters or full words.”), 2:1-6 (“Zellweger’s disclosures show how to automatically generate networks of lists and paths from external sources of data such as database structures...”), 2:30-43 (“The present invention discloses a menu authoring system that generates a tab list menu that links to a set of sublists which, previously, represented one long continuous list of menu items.”).

492. Second, HyperSQL and Zellweger are both implemented as computer software. *Compare Ex. 1006 (HyperSQL)*, 330 (“Our software makes it possible to layer browser-based query interfaces on top of normal SQL. ... With our software, forms and hypertext-based interfaces eliminate direct exposure to SQL and low-level database tools.”), 335 (“Figure 8 shows the architecture of HyperSQL. The software components of a HyperSQL query interface include the target Sybase database, the HyperSQL gateway, an HTTP ... daemon, a set of screens (displayed by the browser), and an end-user, assumed to be a scientist or other person who may be unfamiliar with SQL and the target database.”), Figures 1, 8-9; *with Ex. 1007 (Zellweger)*, 1:35-40 (“The present invention relates to computer software used to

build and maintain a menu system ...”). Thus, a POSITA would have appreciated that HyperSQL’s and Zellweger’s teachings were well-known and complementary to one another.

493. Third, HyperSQL states that its goal is “to make it easier for scientists to retrieve information from remote scientific databases” through the use of a graphical “[q]uery interface.” Ex. 1006 (HyperSQL), 329-30. HyperSQL discloses using “querylists” which list the “‘allowed’ values” from a database that a user may select as constraints when performing database searches. *Id.*, 330-31. HyperSQL’s exemplary use case is with biological databases (although it is not so limited, in HyperSQL, “[w]e are also considering adapting HyperSQL to support Oracle and possibly other databases”) where the querylists can be fairly lengthy and difficult to scroll through. *Id.*, 333, Figure 2. This is burdensome and makes selecting values in such querylists time consuming. HyperSQL Figure 5 shows this:

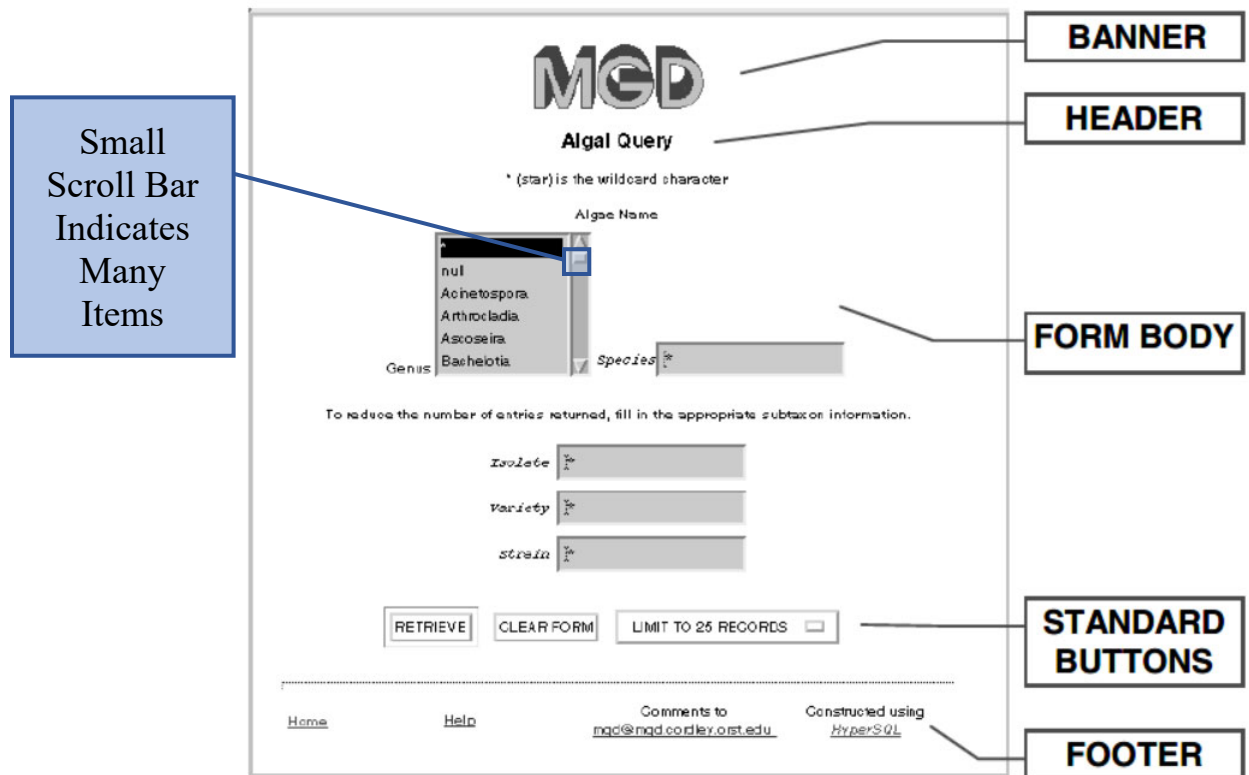


Figure 5. Layout of a sample query form

Id., Figure 5 (annotated).

494. Specifically, a POSITA would have understood that the “Genus” querylist contains a large number of search terms because the scrollbar handle (highlighted in blue in Figure 5 above) is shown as just a small sliver of the entire scrollbar space.

495. This is the same problem that Zellweger set out to alleviate. Zellweger discloses “menus” well suited for lists with “hundreds, if not thousands of entries, [where] scrolling for a particular entry can be extremely tedious.” Ex. 1007 (Zellweger), Abstract. Accordingly, a POSITA would have looked to Zellweger to

improve the usability of the HyperSQL querylists. Specifically, a POSITA would have been motivated to create Zellweger Alphabet or Content Lists instead of scroll bars when, for example, the number of “‘allowed’ values” for a querylist exceeded a specified amount.

496. Fourth, because this is the exact problem addressed by Zellweger, this combination would be nothing more than using one known method (Zellweger) to improve a known system (HyperSQL) according to the express teachings of Zellweger to yield predictable results. A POSITA developing GUI software would have known, for example, to replace a call to display a scroll bar with a call to display Alphabet/Content lists; this would have represented a small change to the code.

497. Fifth, A POSITA would have had a high expectation of success in adapting HyperSQL in view of Zellweger’s teachings of the Alphabet and Content List menus. Zellweger was created with the intention to improve systems such as HyperSQL that have graphical interfaces with long lists of entries. Furthermore, as I explained at the top of this section both HyperSQL and Zellweger involve displaying database information in a GUI. Because of this a POSITA would have understood that altering HyperSQL to truncate entries like in Zellweger’s list menus would require nothing more than routine programming changes to update the type of menus being used in HyperSQL depending on how many entries need to be listed in a HyperSQL querylist.

498. Sixth, a POSITA would have known that databases, web technologies, and GUIs—as well as technologies to search, summarize, and display that combine them—were all well known in the art, as I detail in the technology background §§VII.B-VII.F above.

C. Motivations to Combine HyperSQL, Zellweger, and Microsoft Access

499. A POSITA would have been motivated to combine HyperSQL and Zellweger, as discussed in §XVII.B above, and to add Access Manual’s teachings to the combination, for at least six reasons as described next.

500. First, Microsoft Access is from the same field and teaches similar concepts as HyperSQL and Zellweger. *See, e.g.*, §§XI.A.2, XI.A.3, and XI.A.4 above. For example, HyperSQL, Zellweger, and Microsoft Access relate to graphical means of accessing or displaying information from databases. *Compare Ex. 1006 (HyperSQL)*, 329 (“HyperSQL is an interoperability layer that enables database administrators to rapidly construct browser-based query interfaces to remote Sybase databases. ... HyperSQL can be used to create forms- and hypertext-based database interfaces for non computer experts (e.g., scientists, business users). Such interfaces permit the user to query databases by filling out query forms selected from menus. No knowledge of SQL is required because the interface automatically composes SQL from user input.”), Figures 2-6; *with Ex. 1007 (Zellweger)*, Abstract

(“The present invention discloses a menu authoring system component that enables a developer to generate and build a tab list menu. The tab menu divides a target list menu into tab intervals that can include a specified number of letters or full words.”), 2:1-18 (“Zellweger’s disclosures show how to automatically generate networks of lists and paths from external sources of data such as database structures... .”), 2:30-43 (“The present invention discloses a menu authoring system that generates a tab list menu that links to a set of sublists which, previously, represented one long continuous list of menu items.”); *and with Ex. 1009 (Microsoft Access)*, 1 (“Microsoft Access 97, Version 8.0 (called Access 97 in this book) is a powerful and robust 32-bit relational database management system (RDBMS) for creating desktop and client/server database applications”). Thus, a POSITA would have appreciated that HyperSQL’s, Zellweger’s, and Microsoft Access’s teachings were well-known and complementary, making them good candidates for combining.

501. Second, HyperSQL teaches that searches will be run across multiple databases. *See* §XIV.A above (Claim 21). For example, HyperSQL teaches that biological samples may come from one or more collections or donors, and a POSITA would have understood that each collection may have its own database. *See* Ex. 1006 (HyperSQL), 329 (“HyperSQL is an interoperability layer that enables database administrators to rapidly construct browser-based query interfaces to remote Sybase databases.”), 329 (“databases are not likely to be centrally located,

because they are typically maintained under the control of individual scientists/collectors”), 335 (“Sybase provides multi-database management capabilities”), Figures 3, 4, and 8.

502. Because of this multi-database organization, a POSITA would have understood or at least thought it obvious that the results from these searches can be combined in various ways. In some cases, for example, a user may want to identify samples that have been submitted by multiple donors/collectors for comparison. A POSITA would have understood that this requires an “AND” Boolean operation between two collections/databases. That is, the user would want to discover which samples are in a first database AND a second database.

503. Alternatively, the user may want to identify *any* samples that meet the proposed constraints. A POSITA would have understood that this requires an “OR” Boolean operation between two or more databases. That is, the user may want to discover every result regardless of whether it is in a first database OR a second database.

504. As I discussed regarding Claims 22 and 23, Microsoft Access discloses various ways in which database results could be combined using a well-known and commonly used database management system (DBMS), Microsoft Access 97. *See* §§XIV.B-XIV.C above (Claims 22-23). Specifically, Microsoft Access discloses some of the most common and obvious ways to combine search results—using

Boolean operators like “AND” and “OR.” Ex. 1009 (Microsoft Access), 291 (“*Logical operators* (also called *Boolean operators*) are used most often to combine the results of two or more comparison expressions into a single result”), as seen in Table 9.3 below:

Table 9.3 Logical Operators			
Operator	Description	Example 1 Example 2	Result 1 Result 2
And	Logical and	True And True True And False	True False
Or	Inclusive or	True Or False False Or False	True False
Not	Logical not	Not True Not False	False True
Xor	Exclusive or	True Xor False True Xor True	True False

Id.

505. Using the “Logical Operators” taught by Microsoft Access to implement the “multi-database management” described by HyperSQL would be nothing more than implementing well-known and obvious techniques that are suggested by HyperSQL, known to a POSITA, and expressly taught by Microsoft Access.

506. Third, Microsoft Access (97) and its earlier versions were very popular. Several versions of Microsoft Access were released in the 1990s alone (*e.g.*, 1.x, 2.x, 95, and 97). During that time, it gained its popularity for making it much easier to

create, manage, and use databases. *See* discussion of its history and evolution at https://en.wikipedia.org/wiki/Microsoft_Access (last accessed November 25, 2023).

507. Microsoft Access remained popular from its first version up to an including at least Microsoft Access 97, and thus a POSITA would have been encouraged to take inspiration from Microsoft Access for teachings to improve HyperSQL. Ex. 1009 (Microsoft Access), 1 (“Microsoft Corporation received orders for more than 750,000 copies of Access 1.0 between its release data [sic] in mid-November 1992, and January 31, 1993”), 2 (“In the fall of 1996, Microsoft announced that the firm had sold a cumulative total of 10 million copies of Access.”); *Microsoft Access 97*, WinWorldPC.com, [https://winworldpc.com/product/microsoft-access/97#:~:text=Microsoft%20Access%2097%20\(8.0\)%20was,number%20of%20desktop%20database%20applications](https://winworldpc.com/product/microsoft-access/97#:~:text=Microsoft%20Access%2097%20(8.0)%20was,number%20of%20desktop%20database%20applications) (last accessed Nov. 22, 2023) (“Microsoft Access 97 (8.0) was a very popular version of the Access product.”). Moreover, the processes taught for combining the results of searches from two or more databases were well-known to a POSITA regardless of the database management system being used.

508. Fourth, there were only so many ways to combine search results from two databases and using an “AND” or “OR” operation would be the most obvious. Accordingly, at a minimum it would have been “obvious to try” combining results

using “AND” and “OR,” two of the most basic Boolean operations available in computing with which any POSITA would be intimately familiar.

509. Fifth, a POSITA would have had a high expectation of success in adapting the HyperSQL/Zellweger combination to include Microsoft Access’s teachings. A POSITA would have understood that the teachings being adopted from Microsoft Access comprise two of the most basic and fundamental Boolean logic operations: “AND” and “OR.” A POSITA would have understood that implementing these teachings in the combination of HyperSQL/Zellweger would have required simple, routine, and trivial programming.

510. Sixth, a POSITA would have known that databases, web technologies, and GUIs—as well as technologies to search, summarize, and display that combine them—were all well known in the art, as I detail in the technology background §§VII.B-VII.F above.

D. Motivations to Combine HyperSQL, Zellweger, Microsoft Access, and Janes

511. A POSITA would have been motivated to combine HyperSQL, Zellweger, and Microsoft Access, as discussed in §§XVII.B and XVII.C above, and to add Janes’s teachings to the combination, for at least four reasons as described next.

512. First, Janes is from the same field, addresses the same general concept, and teaches similar concepts as HyperSQL, Zellweger, and Microsoft Access. *See, e.g., §§XI.A.2, XI.A.3, XI.A.4, and XI.A.5 above.* For example, HyperSQL, Zellweger, Microsoft Access, and Janes all relate to graphical means of accessing or displaying information from databases. *Compare Ex. 1006 (HyperSQL)*, 329 (“HyperSQL is an interoperability layer that enables database administrators to rapidly construct browser-based query interfaces to remote Sybase databases. ... HyperSQL can be used to create forms- and hypertext-based database interfaces for non computer experts (e.g., scientists, business users). Such interfaces permit the user to query databases by filling out query forms selected from menus. No knowledge of SQL is required because the interface automatically composes SQL from user input.”), Figures 2-6; *with Ex. 1007 (Zellweger)*, Abstract (“The present invention discloses a menu authoring system component that enables a developer to generate and build a tab list menu. The tab menu divides a target list menu into tab intervals that can include a specified number of letters or full words.”), 2:1-18 (“Zellweger’s disclosures show how to automatically generate networks of lists and paths from external sources of data such as database structures... .”), 2:30-43 (“The present invention discloses a menu authoring system that generates a tab list menu that links to a set of sublists which, previously, represented one long continuous list of menu items.”); *with Ex. 1009 (Microsoft Access)*, 1 (“Microsoft Access 97,

Version 8.0 (called Access 97 in this book) is a powerful and robust 32-bit relational database management system (RDBMS) for creating desktop and client/server database applications”); *and with Ex. 1012 (Janes)*, Abstract (“A livestock and material inventory system having a database storing inventory data, at least one data explorer operable to access predetermined portions of data in the database, and a graphical user interface operable to display data accessed by the at least one data explorer to a user is provided.”), 2:27-48 (“In another aspect of the invention, a livestock and material inventory system comprises a database storing inventory data, at least one data explorer operable to access predetermined portions of data in the database, and a graphical user interface operable to display data accessed by the at least one data explorer to a user.”). Thus, a POSITA would have appreciated that HyperSQL’s, Zellweger’s, Microsoft Access’s, and Janes’s teachings were well-known and complementary, making them good candidates for combination.

513. Second, As I explained regarding Claims [21.A]-[21.C] (§§XIV.A.1-XIV.A.3 above), the resulting combined HyperSQL/Zellweger/Microsoft Access system teaches searching multiple databases and combining the results (*e.g.*, using AND/OR Boolean operations as taught by Microsoft Access).

514. However, what is lacking in this combination is a teaching of a GUI mechanism that a user can operate to request the system to perform these operations. HyperSQL uses such a graphical database interface intended to “make it easier ...

to retrieve information from ... databases” and give more “meaningful feedback during the query process.” Ex. 1006 (HyperSQL), 330. A POSITA implementing the system of HyperSQL would have turned to references like Janes (which teach graphical user interfaces used to search databases) and been motivated to integrate Janes’s generalized drag-and-drop teachings into the HyperSQL/Zellweger/Microsoft Access combination to allow a user to perform the combinations in HyperSQL’s graphical user interface, giving the user more “meaningful feedback” and “mak[ing] it easier” (*i.e.*, more user-friendly) to conduct queries and combine the results.

515. Third, a POSITA would have had a high expectation of success in adapting the HyperSQL/Zellweger/Microsoft Access combination to include Janes’s teachings of representing search results as icons and text and allowing a user to drag and drop these representations together to visually combine the results. A POSITA would have understood that all of these changes to implement Janes’s teachings in the HyperSQL/Zellweger/Microsoft Access combination would have required routine programming adjustments to the graphical user interface to display icons and implement drag and drop behaviors.

516. Fourth, a POSITA would have known that databases, web technologies, and GUIs—as well as technologies to search, summarize, and display that combine

them—were all well known in the art, as I detail in the technology background §§VII.B-VII.F above.

E. Motivations to Combine Cochran, Zellweger, Janes, and Spiegel

517. A POSITA would have been motivated to combine Cochran and Zellweger, as described in §XVII.A above, and to add Janes’s and Spiegel’s teachings to the combination, for at least five reasons as described next.

518. First, Janes and Spiegel are from the same field and teach similar concepts as Cochran and Zellweger. *See, e.g.*, §§XI.A.1, XI.A.3, XI.A.5, and XI.A.6 above. For example, Cochran, Zellweger, Janes, and Spiegel relate to graphical means of accessing or manipulating information in databases. *Compare Ex. 1005 (Cochran)*, Abstract (“A method for managing the presentation of search terms on a computer terminal or other display device so that large computer databases can be searched efficiently.”), 1:14-20 (“The present invention relates to a method for accessing records from a computer database. More specifically, the present invention relates to a method for accessing records from a computer database by first displaying a plurality of lists of search terms available in the database.”), 4:2-57 (“It is our object of the present invention to provide a search system that efficiently displays information to users that are searching for particular information in a database.”); *with Ex. 1007 (Zellweger)*, Abstract (“The present invention discloses a menu authoring system component that enables a developer to generate and build

a tab list menu. The tab menu divides a target list menu into tab intervals that can include a specified number of letters or full words.”), 2:1-18 (“Zellweger’s disclosures show how to automatically generate networks of lists and paths from external sources of data such as database structures...”), 2:30-43 (“The present invention discloses a menu authoring system that generates a tab list menu that links to a set of sublists which, previously, represented one long continuous list of menu items.”); *with* **Ex. 1012 (Janes)**, Abstract (“A livestock and material inventory system having a database storing inventory data, at least one data explorer operable to access predetermined portions of data in the database, and a graphical user interface operable to display data accessed by the at least one data explorer to a user is provided.”), 2:27-48 (“In another aspect of the invention, a livestock and material inventory system comprises a database storing inventory data, at least one data explorer operable to access predetermined portions of data in the database, and a graphical user interface operable to display data accessed by the at least one data explorer to a user.”); *and with* **Ex. 1013 (Spiegel)**, 6:59-7:13 (“FIG. 4 is a block diagram illustrating an embodiment of the present invention. This embodiment supports electronic commerce with multiple contexts over the Internet using the World Wide Web. The server system 410 includes a server engine 411, various Web pages 412, a user database 413, and the multiple electronic commerce context (‘MECC’) system (or multiple shopping cart system in one embodiment).”). A

POSITA would have appreciated that Cochran's, Zellweger's, Janes's, and Spiegel's teachings were well-known and complementary, making them good targets for combination.

519. Second, a goal of Cochran was to “provide an efficient and user-friendly” means of searching databases. Ex. 1005 (Cochran), 4:2-19. Because of this goal, a POSITA would have sought out further ways to make the system of Cochran more “efficient and user-friendly.” It would have been immediately obvious to a POSITA that integrating Janes's teachings of using icons and text for data entries (search results) would aid the user when they have to “scroll[ing] the screen to view the records,” *Id.*, 13:2-7, which Cochran discloses may be extremely voluminous given that the exemplary travel database of Cochran is “using thousands of search terms” and may consist of “more than 13,000 hotels, resorts and inns.” *Id.*, 6:58-67. A POSITA would have understood that, for example, use of icons would provide a way for a user to quickly discern different records—similar to adding bullet points to a list assists the reader in parsing the list. *See id.*, Figure 4c.

520. Third, A POSITA would have been further motivated to integrate the functionality from Spiegel to allow a user to make purchases in the Cochran system given that Cochran is applied to commerce databases, such as Cochran's exemplary Travel Search database. Ex. 1005 (Cochran), 6:58-64. It would have been immediately apparent to a POSITA that integrating Spiegel's teachings about

dragging-and-dropping items onto shopping carts to allow a user to make purchases would further increase Cochran's utility in an easy-to-use way. Specifically, a POSITA would have understood that by allowing users to drag-and-drop the icons or text of a search result (*e.g.*, for one of the "hotels, resorts and inns" stored in the exemplary travel search database in Cochran) onto another icon (*e.g.*, Spiegel's shopping cart) to make a purchase (*e.g.*, allowing the user to book a stay at a hotel, resort or inn of Cochran's Travel Search embodiment) would further Cochran's goal of providing a useful yet easy-to-use means of accessing such databases.

521. Fourth, A POSITA would have had a high expectation of success in modifying the combination of Cochran/Zellweger to include the teachings from Janes and Spiegel related to icons and text representations, dragging-and-dropping, and usage of shopping carts to facilitate purchases. Indeed, it would have been immediately apparent to a POSITA that implementing Janes's and Spiegel's teachings in the Cochran/Zellweger combination would require routine programming changes to (1) include an icon for each of Cochran's record on the view records screen and (2) add an additional purchasing icon that initiates a purchase after a user drags and drops a hotel's text and/or icon onto the purchasing icon.

522. Fifth, a POSITA would have known that databases, web technologies, and GUIs—as well as technologies to search, summarize, and display that combine

them—were all well known in the art, as I detail in the technology background §§VII.B-VII.F above.

XVIII. SECONDARY CONSIDERATIONS OF NON-OBVIOUSNESS

523. During my review of the materials discussed herein, I did not become aware of any evidence suggesting the existence of any secondary indicia or considerations of non-obviousness. In the event that Patent Owner proffers any such evidence, I shall provide my opinions on such evidence in a reply declaration.

XIX. DECLARATION

In signing this declaration, I understand that this declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the USPTO.

I am aware that willfully false statements and the like made by me in connection with this declaration are punishable by fine or imprisonment, or both (18 U.S.C. §1001) and may jeopardize the validity of the Petition. All statements herein made by me of my own knowledge are true and all statements made on information and belief are believed to be true. I declare under penalty of perjury that the foregoing is true and correct.

Executed this 30th day of November 2023 at Stony Brook, New York.

A handwritten signature in black ink, appearing to read 'Erez Zadok', is positioned above a horizontal line.

Erez Zadok, Ph.D.