WILLOW: Technical Overview

Matt Freedman
mattf@cac.washington.edu

University of Washington,
COMPUTING & COMMUNICATIONS
Information Systems

University of Washington, Seattle, Washington
willow
WASHINGTON INFORMATION LOOKER-UPPER LAYERED OVER WINDOWS

A summary of the implementation of Willow, the Washington Information Looker-upper Layered Over Windows. Willow is a cooperative project qof University of Washington Computing & Communications and the UW Libraries, with contributions by the Massachusetts Institute of Technology.

Willow is a general purpose information retrieval tool. It provides a single, easy-to-use graphical user interface to any number of text-based bibliographic databases. With it, you can use a point-and-click style application with even the most arcane command-oriented text-retrieval database system. Since early 1990, the Willow interface has evolved in a spiral of test versions and refinements, based on user-analysis and feedback. It can be compiled and run on most any standard Unix/X machine, as long as the Motif libraries are available[1].

Willow is the primary interface to the University of Washington online database collection (including the Library Catalog), and has been installed on a large scale since 1992. As of fall of 1994, there are approximately 250 public and staff NCD X-Terminals running Willow in the library system. All campus X users have Willow available to them, either installed on a Unix box in their department, or via dedicated campus willow servers. Many people with higher-end networked PC's and Macs also run Willow via X-emulators. Since Willow was released outside of the UW in April of 1994, over 700 sites spread all over the internetted world have downloaded it.

This document provides a description of the Willow interface from the user's perspective, and basic instructions for using and configuring the program; the technical details of its implementation; some related software, such as the character-cell and MS-Windows versions of Willow; Willow's compatibility with standard Internet information retrieval tools such as WWW/Mosaic, WAIS, and Z39.50; how to demo and download Willow; and our plans for future developments.

1. Actually, pre-built binaries are available for a number of popular platforms, so you do not really even need the Motif libraries on your system See *Availability* on page 21.

1.0 Using Willow

The Willow interface is an X Window application, built with the Motif toolkit (see *Related Software* on page 18 for information on Willow for non-X platforms). The paradigm used for Willow's architecture is that of "database-drivers", modeled after the concept of Unix device drivers. With this model, Willow itself does not need to be changed at all for it to speak to a new database, no matter how unique the database is. Instead, a database-driver is written to act as an intermediary between Willow and the host database. One important database driver allows Willow to speak the standardized database access protocol known as Z39.50. See *Architecture* on page 13 for more detailed information.

The Willow interface is designed to simplify the process of connecting to and searching a heterogenous collection of databases. Willow also adds new functionality by performing result-retrieval in the background while you examine the initial matches. Willow makes it easy to develop your search strategy incrementally. Successful strategies can easily be saved for future re-execution. Capturing results to local or remote systems is also greatly simplified by the Willow interface. Willow fully supports standard X copy and paste, so text can be easily interchanged with other standard X application.

There are five windows in Willow -- the main **Search** window where you compose your search strategy in either advanced or basic mode, and have overall control (see FIGURE 1 and FIGURE 7); the **List Browser** window where you can browse lists of possible values for certain fields in a database (see FIGURE 4); the **Summary List** window where you get a concise overview of the results of your search (see FIGURE 8); the **Record Retrieval** window where you can view your retrieved records in full detail (see FIGURE 9); and the **Help** window, where, well you guessed it (see FIGURE 11).

1.1 Connecting to a Database

Once Willow is started, to connect to a particular database you simply selects its name from the **Database** menu (a standard Motif hierarchical pulldown). As far as you are concerned, that is all there is to it. No messing around with telnet, typing in account/password information, setting terminal types, and remembering obscure operating system commands and/or database system commands to get the appropriate data-set loaded and ready. Willow handles all of that transparently.

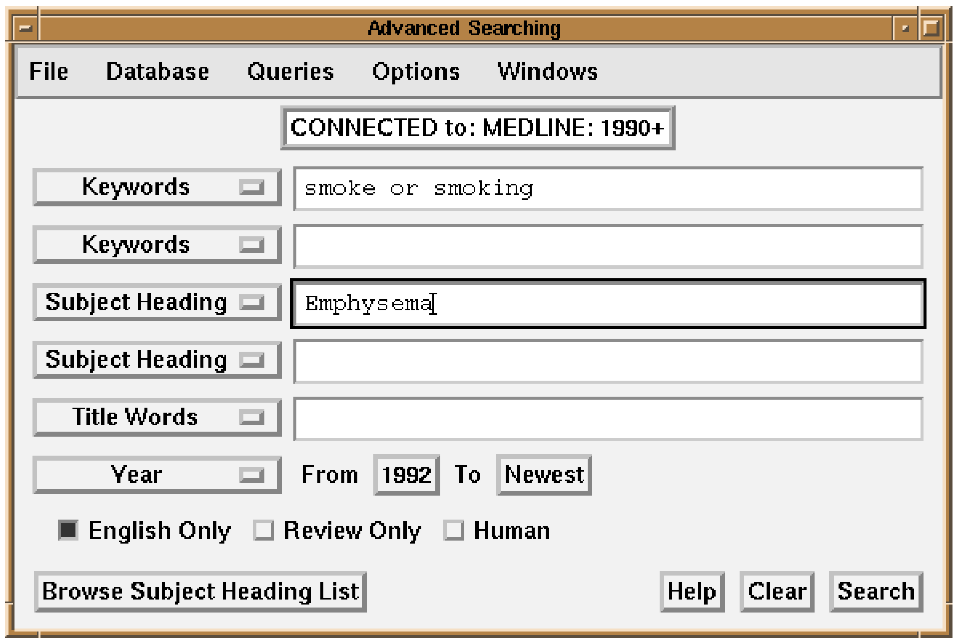
At any point, you can select a different database from the **Database** menu and connect to that one instead. For example, you could do a search on *Books in Print*, then select your local library catalog, and in short order, run the same query again.

Willow can also be configured to be launched by Mosaic, or any other WWW browser. Thus you could browse a hypertext document that included embedded Willow links, and start a Willow database session simply by clicking on the database name as it appears on the Mosaic page. See *Willow and WWW/Mosaic* on page 20.

1.2 Composing a Search

The search composition area of the **Search** window (see FIGURE 1) consists of six text fields, and up to four limit fields. A text field consists of a name of a particular field that is found in the target database, and a blank area for you to type in words that you are interested in finding in that field. For example, if the current database is MEDLINE[1], the default set-up might be two fields labeled "Keywords" (meaning ignore field boundaries in the database entirely), two labeled "Subject Heading", one labeled "Title Words", and one labeled "Author." You can instantly re-configure the layout of the text fields if you do not like the defaults (the field labels are actually pop-up menus, with the less frequently used fields initially hidden, see FIGURE 2). The field labels automatically configure themselves appropriately when you switch databases.

FIGURE 1 Willow Search Window.

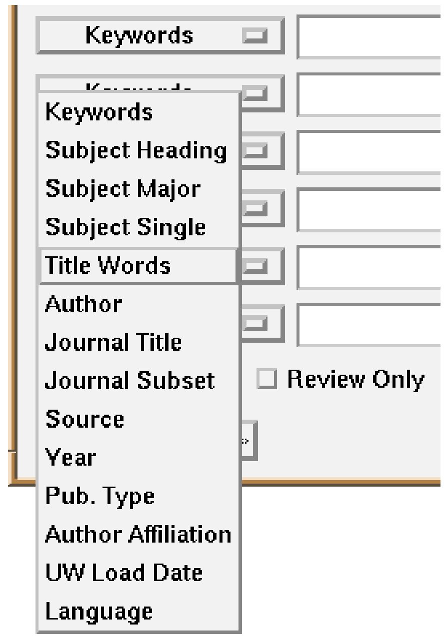


When the search is performed, the individual lines are joined with an "AND" operator. For example, if you type "smoking" in one of the Keywords fields, "emphysema" in a Subject line, and "Smith" in the Author line, Willow retrieves all citations that have the word "smoking" anywhere, and "Emphysema" in the Subject field, and "Smith" in the author field. Whatever other search operators the host database supports can be manually entered if needed (e.g. OR, NOT, ADJ, WITH, etc.) Note that Willow does not currently support operators other than AND between different fields. You can search for smoke OR smoking in the Subject field, but you can not search smoke in the Subject field OR Smith in the Author field.

Limit fields are simple labeled check boxes. For example, one might be "English Only" to limit retrievals to citations of English language articles. There is also a graphical interface for limiting search results to a range of publication dates.

1. The National Library of Medicine's collection of medical journal citations.

FIGURE 2 Changing a field-label.



Once the search is ready, you press the **Search** button to send it off to the host database. A typical pattern for using Willow is to put one or two very broad terms into the **Search** window and launch the search. You then get many more hits than you want, but you can quickly scan through the list of results using the **Summary List** window (as discussed below). When you see something that you are interested in, you can simply use X paste to copy the relevant terms from the **Summary List** back to the **Search** window, and then start a new, more specific search, which should retrieve only relevant citations.

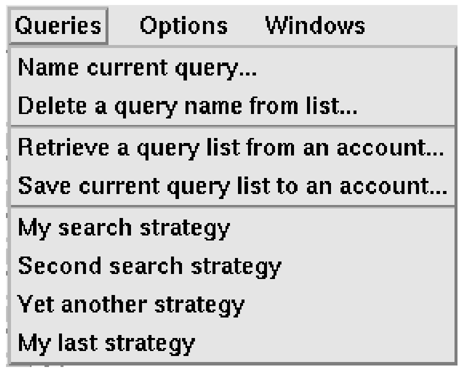
1.2.1 Saving a Search Strategy

Any search strategy can be named, and added to the **Queries** menu (see FIGURE 3). Selecting a query name from that list causes the search composition area to return to the exact state it was in when the strategy was saved. Additionally, all the strategies saved in the menu at any given time can be written to a file. And of course, an old file of search strategies can be opened at any time. Thus for a constantly updated database such as MEDLINE, it is simple for you to keep up to date on newly added citations which match your favorite search criteria.

1.3 Browsing Lists of Search Terms

Willow provides an ability to browse lists of possible values for the various search fields. For example, many bibliographic databases use only terms from an official list in the Subjects fields of their records. MEDLINE uses the MeSH list (Medical Subject Headings). If they did not use this controlled list of subject terms, and instead, individual authors just picked their own, half the articles about heart attacks might have “Heart Attack” in the Subject field, and the other half might have “Myocardial-Infarction.”

FIGURE 3 Willow Queries menu.



With the controlled list of subjects, if you search "Myocardial-Infarction" in the Subjects field, you should retrieve all the records about heart attacks.

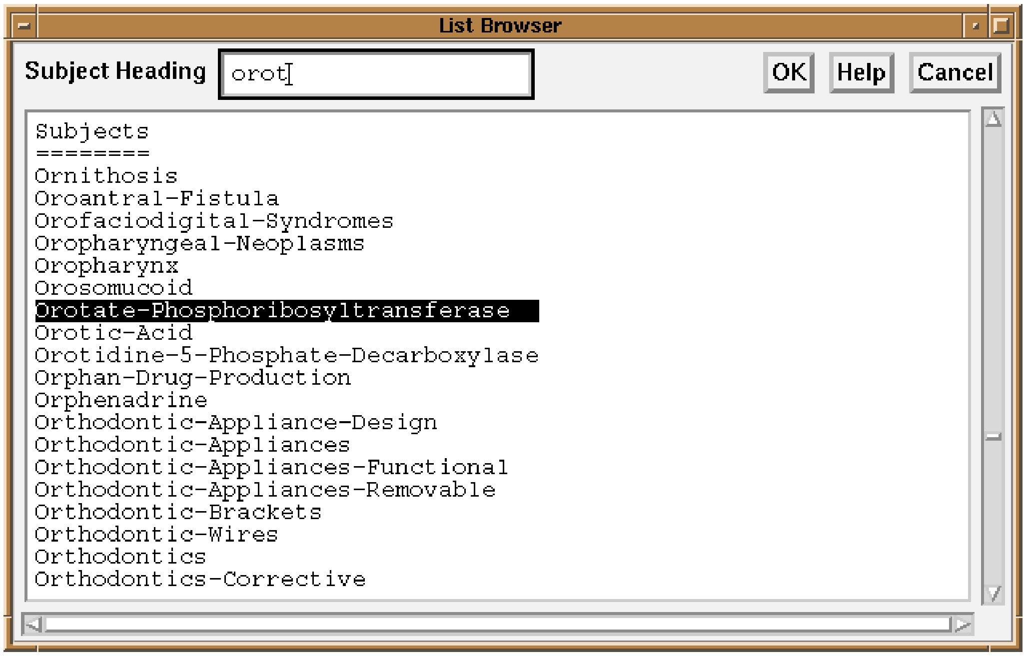
In addition to subject terms, you may also want to browse alphabetical lists of the authors, titles, or other fields that comprise a given database. Picking search terms from a list instead of typing them yourself also protects you from your own typographical errors[1]. Willow has a **List Browser** window that allows you to quickly find the items you are interested in. If a particular field has a corresponding browse-list available, you can use the **Browse** button on the lower left-hand corner of the **Search** window to view it. The label on the **Browse** button corresponds to the field you most recently clicked the mouse in.

Once you have popped-up the **List Browser**, you just type the first few letters of the word you want. The list automatically scrolls to match what you have typed. For example, in the MEDLINE Subject Headings list, just typing 'orot' will get you to 'Orotate-Phosphoribosyltransferase' (as shown in FIGURE 4). You can also use the scrollbar to move around in the list. Once you have selected the item(s) you are interested in, you click "OK", and the chosen entries are automatically transferred back to the **Search** window. Though this all appears quite simple to the user, there is *alot* going on behind the scenes. See *List Browser Implementation* on page 17 for details.

One browse list of particular interest is the UW Libraries Catalog titles list, sorted by Call Number. Unlike most lists which are sorted and searched in alphabetical order, this list is in true call number order. A typical use is to do a normal Willow search of the Catalog, find an item of interest in the **Summaries** window, then use X to select and paste the call number from the displayed result into the call number browser. The list will then jump to display the title you already found, and surrounding it in the list will be other items with closely related call numbers. In effect, it is like going to the stacks and finding your book, then looking around to see what is shelved near it. But unlike the real shelf, even books that are currently checked out will be there, and you will see the items available in more than 20 branches of the UW Libraries. See FIGURE 5.

1. On the other hand, displaying lists of all the existing values of a given field in your database will expose the embarrassing typos in your data!

FIGURE 4 Willow List Browser.



1.4 Basic Searching Mode

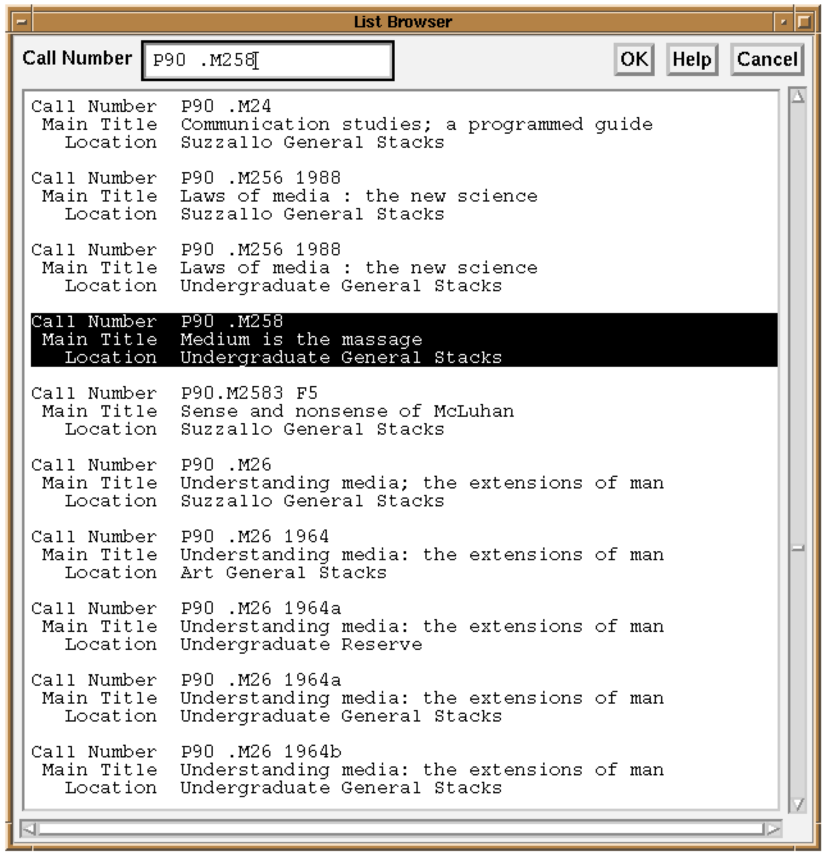
Willow has a **Basic Searching** mode, which is designed especially for use with library catalogs. With these databases, many users want to look up a known book or author. In this case the Willow **Search** screen can be confusing, and it provides more power than necessary. All you really want to do is pick an already-known item out of a list, and see where to find it in the library, and whether it is checked out. In **Basic Searching**, you first see a screen with a few of the most commonly used fields listed, as shown in FIGURE 6. You click the button corresponding to the kind of search you want to perform.

After you click a button, the window grows downwards, and a type-in area appears. As you type the first few letters of the item you want to look up, the window grows again, and a browse list (as discussed above) appears, as shown in FIGURE 7. When you press the **Search** button any items you have selected are looked up (with an OR between them, if more than one).

1.5 Viewing Summaries

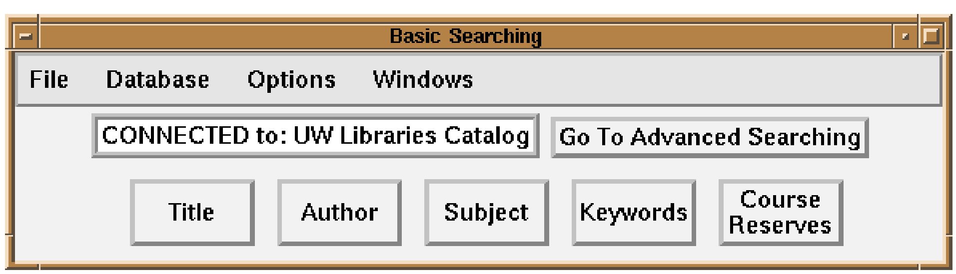
While the search is executing, you are treated to the endlessly amusing spectacle of a spinning globe. Once the search has finished, the **Summaries List** window is brought to the top. Willow starts the process of retrieving a summary of each matching database entry. As each summary is retrieved, a title line is added to a scrollable list. This line is configurable, and can contain several fields -- at UW we use 20 characters of the Author or Journal Title field (depending on the database), 60 characters of the Title field, and four characters for the Year of Publication. You can scroll through this list as it grows, scanning for interesting titles. When you see one, and click on it, the summary information for that citation appears below the title list. The summary is also configurable, but typically consists of Author, Title, Source

FIGURE 5 Browsing the UW Catalog by Call Number.



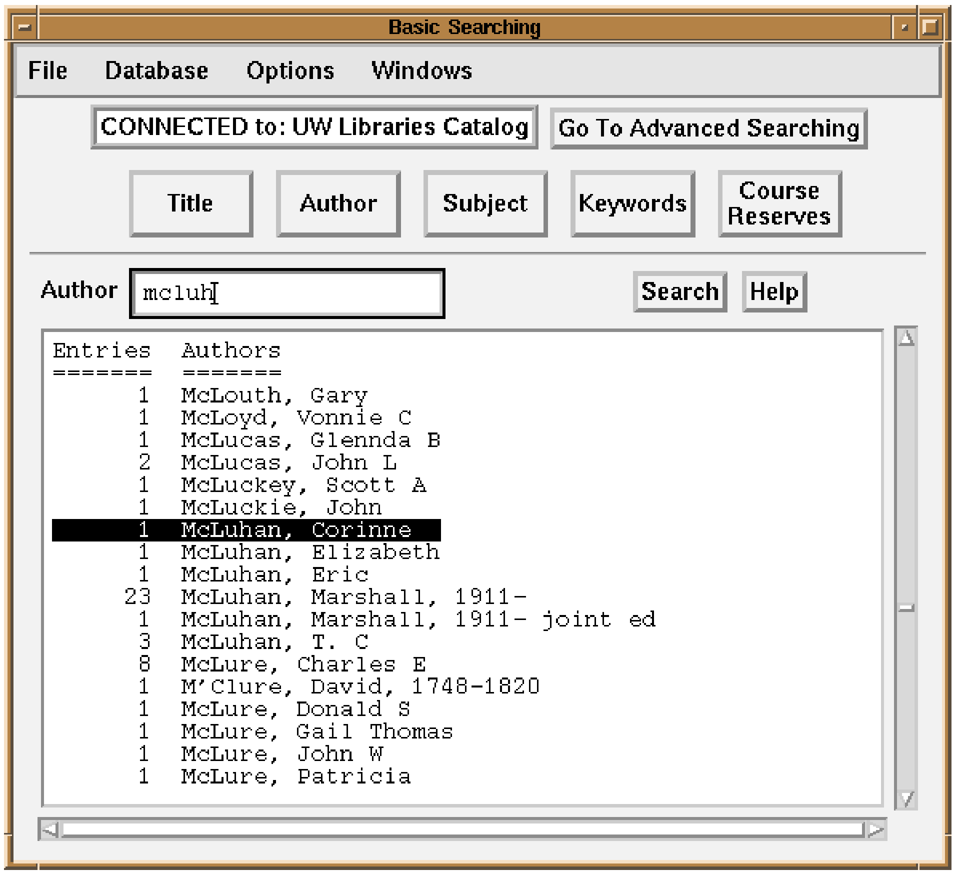
and Subject fields, as shown in FIGURE 8. To alleviate pressure on the underlying database system, no more than 30 summaries are retrieved at a time. You can press a button labeled **30 More** to view more titles.

FIGURE 6 Willow Basic Searching, initial view.



As you find summaries you are interested in, you can print them or save them, individually or in groups. Saving can be done to a local file, as an electronic mail message, or

FIGURE 7 Willow Basic Searching, view after typing the first part of an Author name.

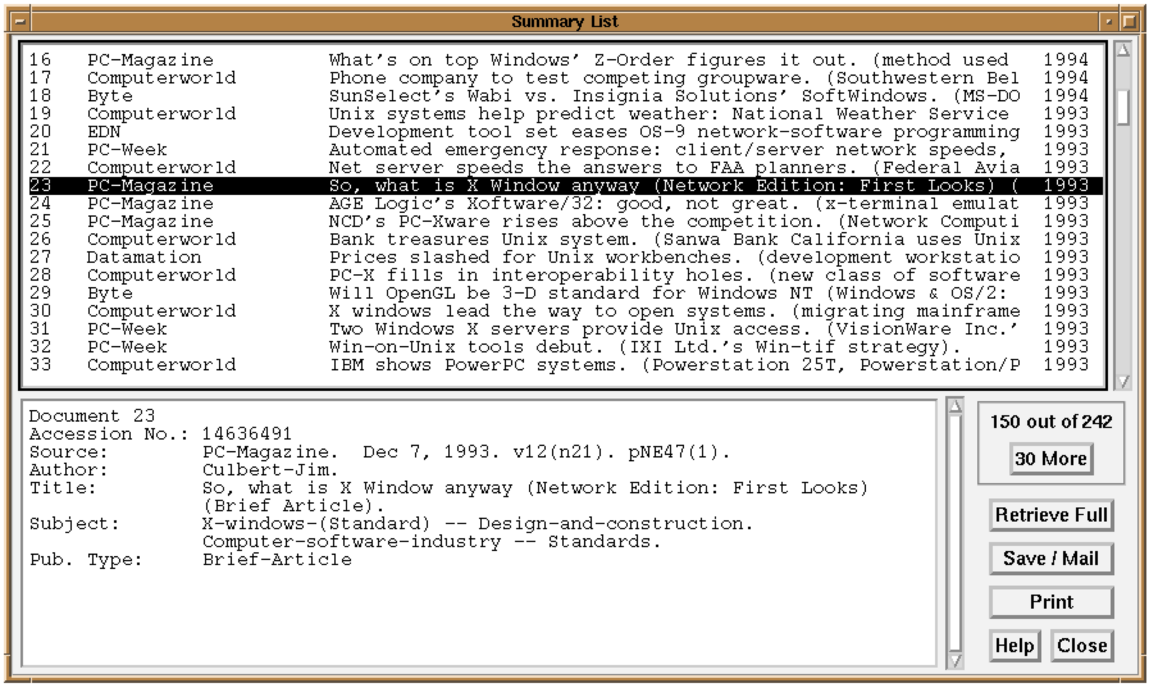


over the network using ftp (internet File Transfer Protocol). You can also press the **Retrieve Full** button to retrieve the full-records of all the selected titles. The records are displayed in the **Record Retrieval** window.

1.6 Viewing a Full Record

The contents of the full record vary greatly depending on the database. For a library catalog, it might just add a few additional esoteric fields that were not included in the summary display, but for MEDLINE and other databases of bibliographic citations, the full record usually includes the abstract of the cited article, and for some databases it could have the full text (in ASCII form) of the article. FIGURE 9 shows the Record Retrieval window with a MEDLINE record. With some databases, Willow allows you to choose which fields will appear in your full record. FIGURE 10 shows Willow's **Record Format Options** dialog box. This dialog box also lets you choose whether you prefer to view fully-expanded field names or two letter abbreviations (convenient for loading retrieved records into your own reference manager database), and whether you would like to include the search strategy that generated your results with your saved and printed records.

FIGURE 8 Willow Summary List. Figure shows results of a search for “X and Window” in the Information Access Company’s *Business Index* database.



The **Record Retrieval** window consists of a large scrolling text area, where the citation appears just as the host database sends it out. Words that match the query are highlighted[1]. In full-text databases such as Grolier’s Encyclopedia, the **Prev** and **Next** Match-Words buttons scroll the display to the next or previous occurrence of a highlighted match word. As with the summaries, the full-records can be printed, as well as saved via email, ftp, or to a local file.

1.7 Getting Help

Willow has on-line help available from each window. Pressing the button labeled **Help** will pop up a hierarchical help browser. The help system can display either plain text or bitmaps (as shown in FIGURE 11). It can also be configured so that you get different help-screens depending on what database you are connected to. This is very helpful for explaining searching techniques that only apply to a specific database.

2.0 Configuring Willow

Willow is an extremely customizable program, both at the level of the individual user, and for an entire site. This section contains an overview of the available customizations,

1. Highlighting of the matched words is not implemented in the Z39.50 driver, because indication of matched search terms is not part of the Z39.50 standard.

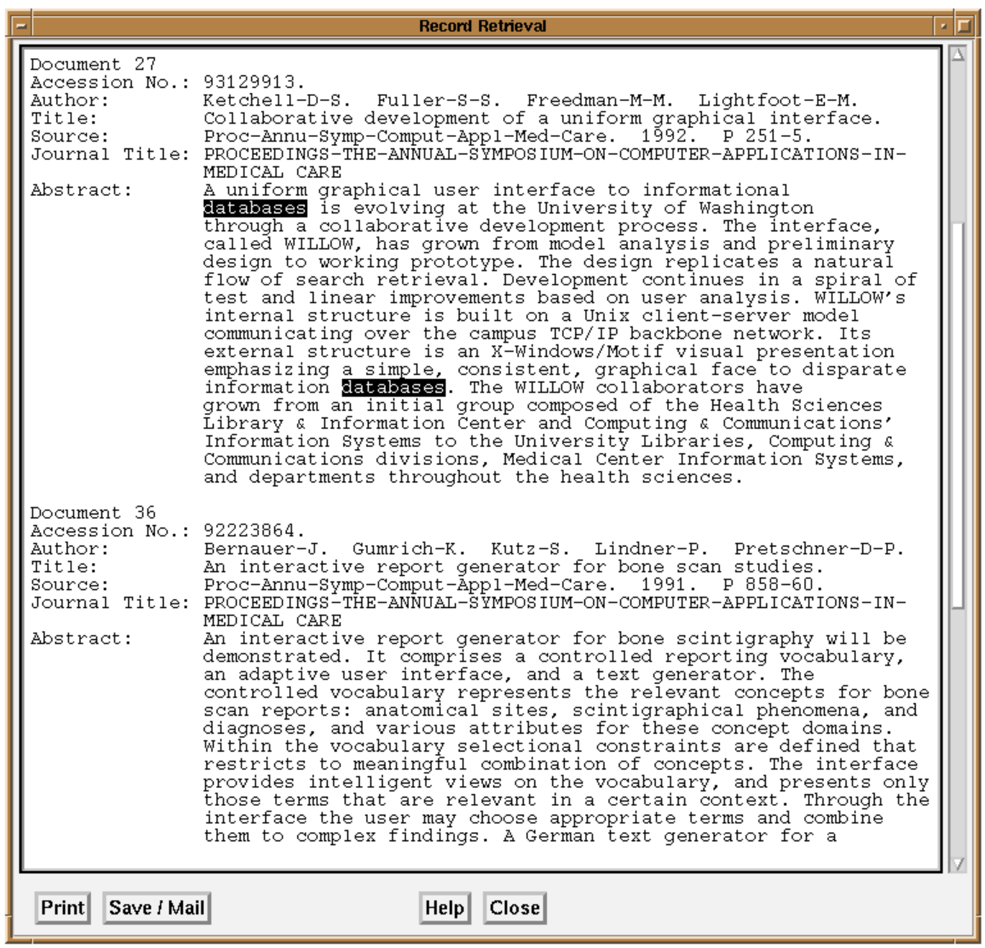
but for full documentation, refer to *doc/Customization* in the Willow distribution (see *Availability* on page 21 for information on getting the distribution).

2.1 User-level Customization

When you select **Save** from the **Options** menu, Willow writes a file called *.willowrc* in your home directory. In it, Willow records any customizations you have made, i.e. changes to the **Record Format Options** (FIGURE 10), your preferred printer, your preferred method of saving results (i.e. direct to file, email, or ftp) along with any necessary information for doing the saves (except for passwords for ftp, which must be re-entered each session), etc. When Willow starts up, it reads the *.willowrc* file, and configures itself accordingly.

In addition to the *.willowrc* file, Willow makes extensive use of X resources. All the options described above can be set via your *.Xdefaults* file as well. Plus, there are a number of customizations which are available only through X resources. You can set the *Willow*size* resource to *small, medium, large,* or *extra-large*. Willow then sets its geom-

FIGURE 9 Willow Record Retrieval Window.



etry and fonts accordingly. At *small*, all of Willow's windows fit fine in a screen as tiny as 640x480 pixels, and at *extra-large* Willow will take full advantage of the extra real estate of a 1280x1024 display. A value of *custom* is also available if you are not satisfied with the standard choices. Willow's color scheme and fonts are fully customizable as well, giving you free reign to make the display as nauseating as possible. Also available for customization are the various key-bindings and keyboard shortcuts used by Willow.

2.2 Site-level Customization

2.2.1 Database Configuration File

There are several other ways that Willow may be customized, but these will generally only apply if you are a system administrator, setting up Willow for an entire site. At start-up, Willow reads another text file, by default called *db.conf*, which contains the configuration information needed for each available database. Normally this is a global system file, though it is possible for you to have a personal one. The information in this file allows Willow to construct the **Database** menu. When you select a database, Willow knows which database driver to launch, and how to configure the field-labels and browse-lists, based on what it found in the *db.conf* file.

As a system administrator you will probably want to add new databases to Willow's repertoire, either your local data collections, or remote Internet-accessible ones. To do so, a new entry is made in *db.conf* for each database. See *doc/Customization* for complete documentation on the format of a *db.conf* file.

FIGURE 10 Willow Record Format Options dialog box.

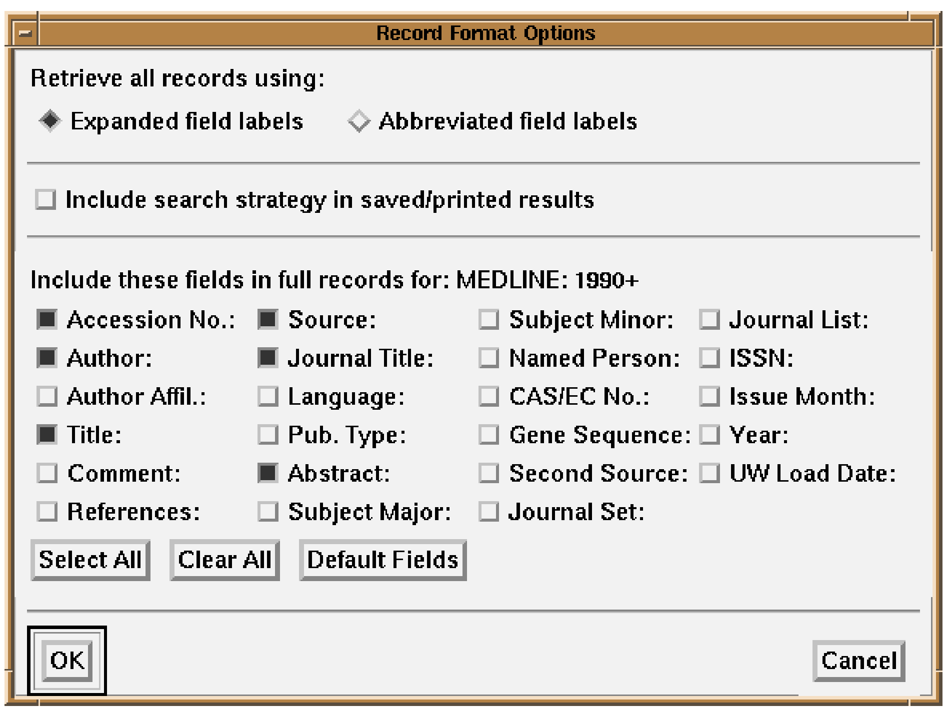
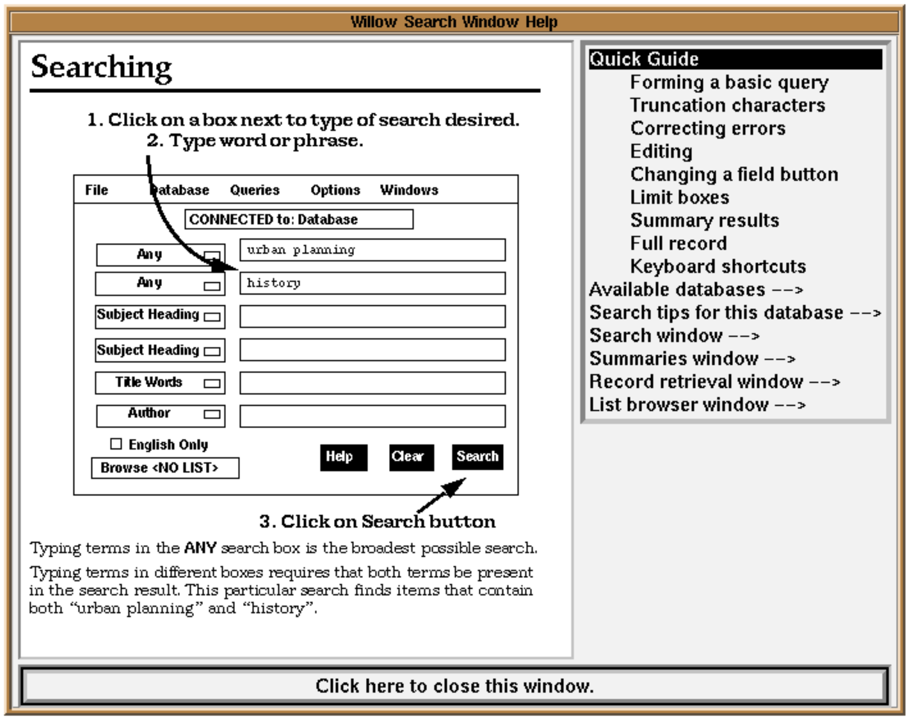


FIGURE 11 Willow Help System.



2.2.2 Interface Modes

Willow's interface can be run in three different modes, controlled by an X resource called *Willow*interfaceMode*. The mode described so far is desktop or full-featured mode. It is well-suited for use in individual departments, where the department is running Willow on their own Unix workstations, and individual users have personal accounts on these machines, and access to X-Windows displays.

However, as a huge institution with a very heterogenous collection of X-capable terminals and workstations, we have found that it makes the most sense to run Willow on a centrally maintained cluster of machines, which display the Willow session remotely on the user's terminal. Thus when you execute *willow* you are really just running a small script that uses *rsh* to automatically start up Willow on one of the cluster machines, using the power of X to display remotely on your screen. There are several advantages to this approach. New releases need only be compiled on a single architecture, and they do not have to be distributed to a large number of sites, more CPUs can be added to the cluster as load increases, or machines can be removed from the cluster if they develop hardware problems.

This approach only works well when the application can effectively be run anonymously, i.e. if the user does not have to have a personal account on the server machine. When *interfaceMode* is anonymous, Willow will assume you do not have an account on the machine it is running on. **Save** no longer appears on the **Options** menu, so you can not create a *.willowrc* file in the home directory of the anonymous account shared by many people. Naturally you can still set all the possible Willow options in your *.Xdefaults* file on the machine you really do have an account on. Also, in anonymous mode, retrieved records can only be saved via ftp or email -- not directly into a local file, and similarly, saved collections of search strategies can only be saved/loaded via ftp.

The third access method is walk-up use in the UW Libraries. In this case the system is controlled by a session manager that acts as "Big Brother". It gives the naive walk-up user lots and lots of help and encouragement, but at the same time severely limits what the user can actually do with the X environment. In session manager mode, Willow no longer has a **Quit** item on the **File** menu, and it accepts commands from the session manager telling it to connect, reset, quit, take itself off the screen, etc.

3.0 Architecture

Willow is designed to be a general-purpose bibliographic database front-end. It can be configured to act as an interface to virtually any network-accessible text-retrieval database. This flexibility is a direct consequence of Willow's architecture. The architecture is based loosely on the concept of Unix device-drivers. In Unix, any program can communicate with any physical input/output device (such as disks, printers, networks, terminals, keyboards etc.) through a few simple, standardized procedure calls. It does not matter that the physical devices may be wildly different from each other. This is because for each device somebody has written a program called a device-driver. The job of this program is to translate between the simple standardized procedure calls and the instructions that the hardware device expects to receive. Thus an application program can read data from a disk-drive with the exact same lines of code it uses to read from the keyboard, even though the two physical devices are entirely different from each other.

In Willow, instead of device-drivers, we have database-drivers. The Willow program itself does not know anything about the command syntax for any particular database. Instead, it knows how to communicate with a second program, the database driver. It is this program that actually knows how to make a connection to a particular database host, as well as the details of its commands, and how to parse the incoming stream of data.

Unlike Unix device-drivers, Willow and the database-drivers do not communicate via a procedure call interface. Instead, they use message-passing. More specifically, Willow opens a Unix pipe, and then splits off the driver (using standard Unix fork and exec functions). The two programs then send information back and forth in packets over the pipe[1].

Though a Willow database-driver is hopefully easier to write than a Unix device-driver, it is no trivial undertaking. However, the Z39.50 driver promises to alleviate the need to customize a driver for every unique database. See *Z39.50 Driver* on page 17 for details.

1. It is also possible to run the driver program on a separate computer, and communicate via a socket. See *Willow for MS-Windows* on page 18.

3.1 Willow Interface Implementation

The Willow interface is a Unix application, built with the Motif tool-kit, which is itself layered on top of the X Windows system. Because it is built with a standard user-interface tool-kit, the interface acts in a standard way[1]. Thus anybody who is familiar with other Motif applications will already be familiar with the basics of manipulating the Willow user interface.

The interface is set-up via the Motif User Interface Language (UIL), rather than in the C language code. Thus it is relatively easy to change the look of the interface without having to touch (or understand) real code. Willow has been successfully ported to a variety of standard Unix workstations including DEC Ultrix, Sun, IBM RS/6000, and Hewlett Packard HP/UX.

When Willow is running there are as many as three separate computers involved. Willow itself is running on a Unix box somewhere, but the user is often not actually sitting in front of that machine. Instead she is using the X Window System to display Willow on her own screen (X-Terminal, other workstation, or PC/Mac running an X server). And the database that she is searching is usually on a third machine. At the UW a typical configuration is: user sitting at an NCD 17c X-Terminal, remotely running Willow on a DECstation 5000/240, connected to a BRS[2] database running on an IBM RS/6000.

Like all X programs, Willow is event-driven. This means that Willow initially sets up lots of routines to handle all of the X Windows events it is interested in -- button clicks and key-strokes in various areas of the user-interface. After initial set-up, Willow spends most if its time waiting for the X system to pass incoming events to the specified handlers. In Willow's case though, not only are events coming in from the user-interface, but once it is connected to a database, packets are also coming in from the connection to the database-driver. The code to handle the packets is set up much like the code to handle X-events, i.e. a handler is declared for each type of packet that may come in.

Willow knows very little about interacting with databases -- it leaves that to the database drivers. For example, when the user selects a new database, Willow first sets up the pipe, forks the appropriate driver program, and sends a "connect" packet to it. Then Willow sits back and waits for events to come in -- either X-events such as button presses, or packet-events coming in from the driver. Eventually the driver will send a packet saying that the database has been started up, and is ready to search (or there was some specific problem starting it).

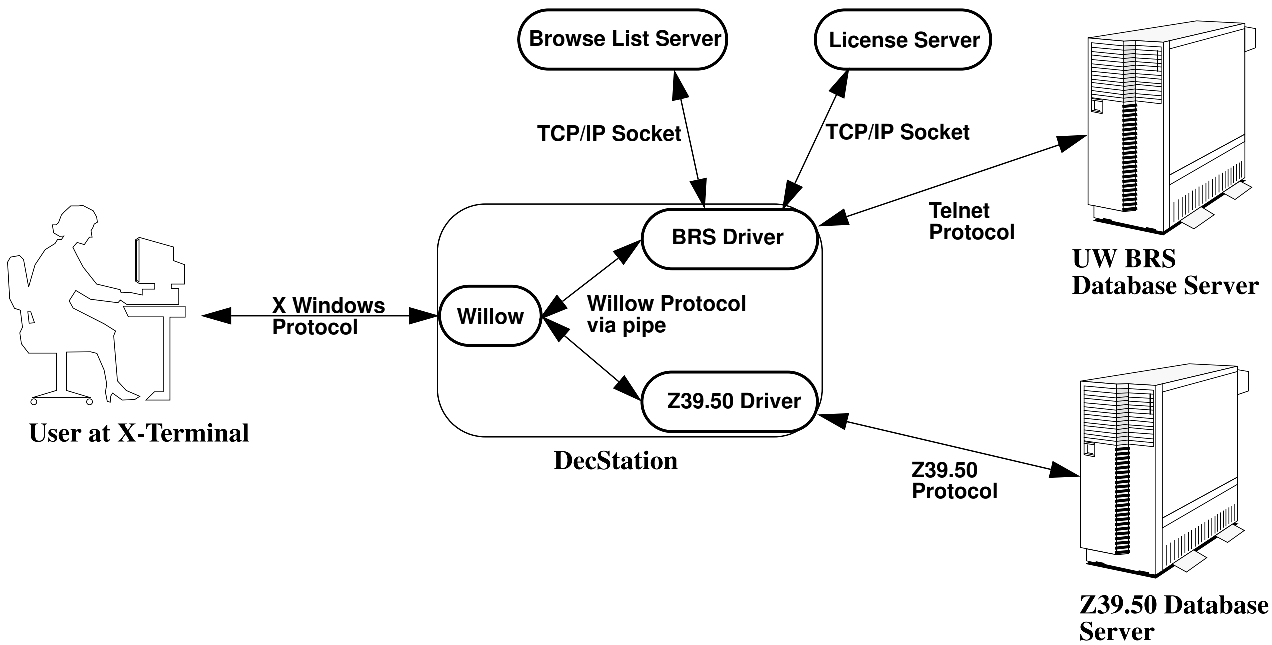
Searching is very similar. When the user presses the search button, Willow gathers up all the information the user has typed in, the names of the fields, and the settings of the limit buttons, and ships it all out to the driver. Willow lets the driver format all the information into the appropriate syntax for the current database system. Eventually the driver will send back a packet with some search result-statistics, followed by a stream of packets containing result-titles.

1. As the saying goes, the best thing about standards is that there are so many to choose from. There are several "standard" user interface paradigms, and Motif is only one of them. But at least it is one of the more popular.

2. BRS Search is the commercial database engine we employ at the UW.

FIGURE 12 Data Flow in a Willow Session.

This diagram illustrates the flow of data, and the different systems and protocols involved during a typical Willow session.



3.2 Database-Driver Implementation

The database-drivers are simple Unix applications. They know nothing at all about X Windows (and consequently are much smaller than Willow itself). The only requirement for a driver is that it understands the protocol that Willow uses. The implementor decides how a driver translates between the Willow protocol and the commands that a given database is expecting. However, Willow comes with a model driver, which with some modification, can probably be used for most bibliographic databases. In an attempt to make the model driver portable, its interactions with the host database system are table-driven. This section describes the implementation of the model driver, which communicates with BRS Search on an IBM RS/6000 running AIX.

The driver communicates with the database host by opening a pipe, and forking a telnet connection to the appropriate machine. Thus any database that is on the Internet can have a driver customized for it. It is certainly possible to write drivers which use other methods to reach non-internet databases (for example by dialing a modem, or releasing carrier pigeons) but there are no current plans to do so.

Like Willow itself, the driver has two sources of incoming events. One is the pipe to Willow, over which packets will periodically appear. The other is the pipe to the telnet process, over which a stream of characters will appear. The driver sends back command strings to the database, mimicking the actions of a human user. In fact, as far as the database is concerned, there is a human user at the other end of the telnet connection (a rather fast typist though).

Protocol packets coming in over the pipe from Willow are taken care of by event-handlers, one for each packet type, just as in Willow itself. The stream of characters from the database on the other hand, is handled by a finite state automaton (FSA). An FSA consists of a set of states, and for each state a set of input strings which might be encountered. Encountering a given input string causes the automata to move to another state (or return to the current state), and send out an output string.

For programming purposes the FSA is converted to a table. Thus at the core of the database driver is an FSA, in table form. The program has a variable called *state*, which indicates the current row of the table. For each state, there is a list of possible input strings that are currently "interesting". For each input string there is a corresponding output string which should be sent as a response, and a next state (row) to move to.

The driver continually scans the telnet connection, looking to see if the any of the listed input strings for the current state show up. If one does, the driver sends back the appropriate output string, then updates the state variable to the specified next state, and starts scanning again. Thus the command syntax of the host-database is not built into the driver directly, instead it is stored in the table.

3.2.1 Driver Complications

As you might suspect, the driver is actually rather more complicated than it sounds. Incoming packets from Willow can also trigger output strings. For example, search requests obviously cannot be built into the table in advance, since they are dynamically created by the user. In addition, there is another optional item in each row of the table -- a procedure to be called when the input string is discovered. For example, when the driver sees the string which tells it that the search is complete, it will automatically call a routine named in the table which analyzes the text that came back from the database, extracts the search's vital statistics, and sends them back to Willow for display to the user. In fact, there are a number of similar routines for analyzing the data stream, and putting it into a form that Willow is expecting. Some of these data-parsing routines can get quite ugly. These routines will typically all have to be re-written in order to port the driver to a new database.

The biggest problem with the current driver-architecture is that remote databases can change at any time. For example, if the driver is scanning for the prompt "PRESS ANY KEY:", and one day that prompt changes to "PRESS A KEY:" 99% of the human users would never notice, but it would stop the database-driver dead in its tracks. Many databases come with an API (Application Program Interface). If one is available, it would make sense to use that to communicate with the database rather than trying to mimic a human user by sending command-strings. An API could also make the data-parsing much simpler.

3.3 Z39.50 Driver

A much better solution to the complexities of writing a custom driver for every database would be if a single driver could speak to a wide variety of different databases using a standard protocol. Z39.50 is an ANSI standard network protocol for bibliographic database access over the Internet. The Massachusetts Institute of Technology has successfully interfaced the UW model Willow driver to the Stanford Z39.50 API library. This driver translates between the Willow/driver protocol and the Z39.50 protocol. Thus Willow is now a full-featured Z39.50 search client, and can talk to a wide variety of Z39.50-speaking databases across the Internet.

3.4 List Browser Implementation

As described in *Browsing Lists of Search Terms* on page 4, Willow provides a simple-to-use interface for browsing the possible search terms in some database fields. However the implementation of this system is far from simple. Because BRS, the database engine used at UW, does not handle this type of browsing at the speeds that are required, we pre-build the lists. I.e. for each field of each database that we want to provide browsing capability for, we have to extract every existing value for that field, and sort them. There are a large number of lists (as many as five or six per database), they can each be quite long (the list of titles held by the University of Washington Libraries is almost 100 Megabytes), and some of them need to be updated quite frequently. We have also created a centralized network browse-list server which our BRS database driver can talk to. This is a custom text-search engine, optimized for doing extremely fast searches of sorted lists[1]. Willow makes a browse query on every user keystroke in the *List Browser* window, and the server returns one screen's worth of the list, centered around the first item in the list which matches the prefix the user has typed. See FIGURE 4. (Actually, if the user is typing rapidly we wait until a pause to send the request). The user can also move through the list via the scrollbar. Motif is duped into thinking it is scrolling a huge file, when actually only a tiny fraction of the file is there at any time.

Note that the Willow program itself does not know anything about the BRS list-server. A list-browse is just a second type of search that gets sent down the pipe to the database-driver. It is the BRS driver which interacts with our list-server.

In the simplest case, a browse list consists of a simple list of homogenous items, i.e. just the subject terms. However Willow and our list-server can also handle multi-field lists, with a variety of display formats (do not confuse browse-list fields with the fields of the actual database). For example some of our lists also include a second field which indicates the number of records in which this term appears, i.e. how many hits you will get if you actually search this item (See FIGURE 7). Sometimes the text we want to display to the user is not directly searchable in our database. For example, in our list of titles in the Libraries Catalog, we want the user to be able to browse through entries such as "Music (vocal & instrumental) in the Works of Kalidasa", but as you can imagine, inputting a string with all those special characters would not go over well with the database engine. Instead, we want to search the string "Music-vocal-instrumental-in-the-works-of-Kalidasa" which has no special characters, and is pre-bound at database load-time so that the retrieval is very fast. In order to support this, certain browse-list fields are invisible. So what the user sees in the *List Browser* is not necessarily what gets transferred back to the *Search* window.

1. Using a simple binary search turns out to be plenty fast.

Other parameters of the list-server indicate whether secondary fields should be displayed on the same line as the main field, or whether every field should get its own line, where and how the list field-names should be displayed, and whether the text should be displayed right or left justified.

3.4.1 List Browsing and the Z39.50 Driver

Our list-server is definitely a home-brew system, and is not likely to be supported by non-UW database servers. However, the Z39.50 driver provides a more generic solution to the list-browse problem, via that protocol's SCAN functionality. When Willow is connected to a Z39.50 database, it sends the exact same list-browse requests down the pipe to the driver, but instead of forwarding the requests to the BRS list-server, the Z39.50 driver will send a SCAN directive to the host-database. The driver then packages the results of that search into the form that Willow expects, and sends it back. Not all of the options described above are available with the SCAN-based browse lists.

4.0 Related Software

There are several other interesting pieces of software developed at the University of Washington that are closely tied to Willow.

4.1 Wilco

WILCO -- the Washington Information Looker-upper - Character Oriented, is, as the name implies, a character-cell version of Willow. The idea is to provide as much as possible of Willow's functionality using only standard text-terminal emulation, so it can be run at home via a modem, or on other non-X-capable devices. Wilco uses the exact same protocol (and much of the code) as Willow, so it can run with identical driver programs. Thus Wilco is also a full-featured Z39.50 search client. Wilco has a very similar interface to Pine, the UW-developed electronic mail interface for character-cell environments. Of course much of Willow's power comes from the ability to have several windows on the screen at the same time, and from the convenience of using a mouse. Naturally both of these features are missing in Wilco. However user-feedback indicates that Wilco is still considered an improvement over standard text-based database interfaces. FIGURE 14 shows the Wilco search screen.

4.2 Willow for MS-Windows

A project to create a version of Willow for Microsoft Windows is under way. It is still in its infancy, but we hope to have a beta version for release some time in the first half of 1995. FIGURE 15 shows the MS-Windows Willow **Search** window. Versions of Willow for other non-Unix platforms (such as Macintosh) may also be developed in the future.

To help with porting to non-Unix platforms, the connection between Willow and the driver can be made a network socket, rather than a simple pipe. That way, the drivers can always run on a Unix box, and only the Willow interface itself needs to be ported.

FIGURE 14 Wilco Search screen.

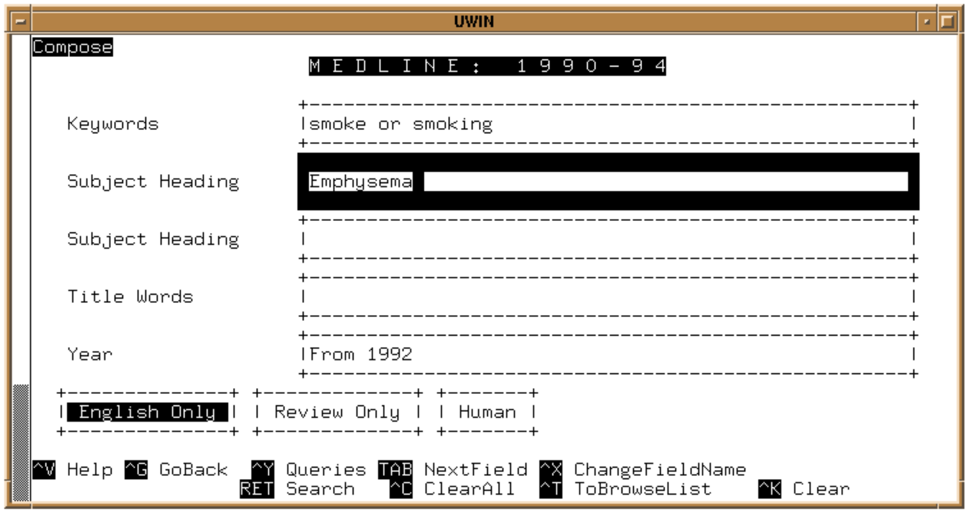
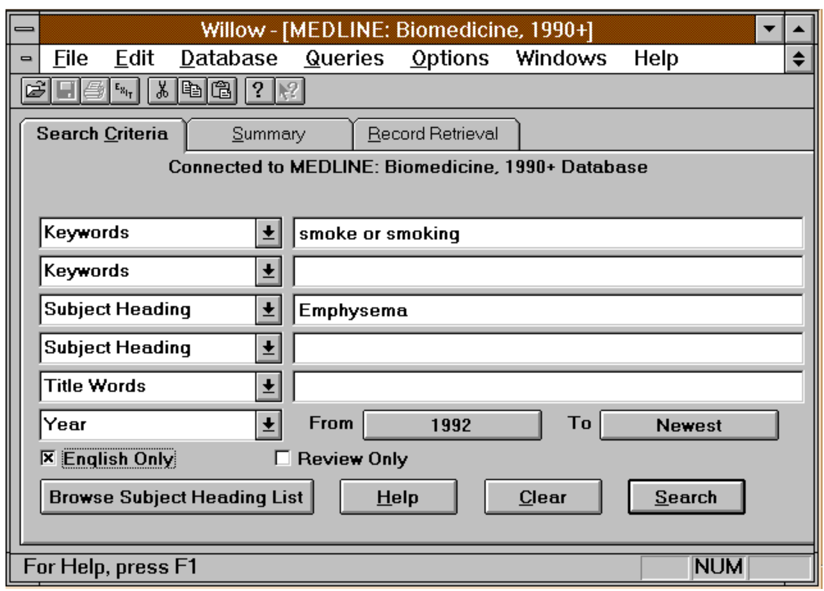


FIGURE 15 Willow for MS-Windows prototype Search screen.



4.3 Willow and WWW/Mosaic

Willow is designed to be compatible with the World Wide Web (WWW) and the Mosaic browser in particular. The situation is analogous to the way image files work with Mosaic. With those, you click on a hypertext link which causes a file with a *.gif* extension to be downloaded. Mosaic knows that when it has a GIF file, it should start the *xv* program to view it. Similarly, you can pick a WWW hypertext link that causes a file containing a single *db.conf* entry to be downloaded (with the extension *.dbcf*), and if you have configured Mosaic properly, it will know to start a Willow session to “view” that file.

4.4 Willow and WAIS

Willow can be used to search any database that speaks the standard Z39.50 information retrieval protocol (though these are not the only databases Willow can talk to). One notable example of a Z39.50 compatible system is ZDist, formerly called free-WAIS. ZDist provides a Z39.50 protocol engine that can be put in front of a standard search engine, such as WAIS. ZDist is produced by the Clearinghouse for Networked Information Discovery and Retrieval (CNIDR). ZDist is available via ftp, from *ftp.cnidr.org*, in *pub/NIDR.tools/zdist.*

4.5 Multimedia Extensions

In order to achieve 100% buzzword compatibility, a multimedia extension mechanism has been developed for Willow. Willow itself only deals with data in the form of plain ASCII text. However, more and more databases contain data in a more complex form -- images, sounds, animations, SGML, HTML, and Unicode, just to name a few. Rather than try to add a mode to Willow that can handle every possible data type, there are hooks that allow you to configure Willow so that it can start external programs to handle the non-standard portions of any retrieved information.

For example, many of our Libraries Catalog holdings are in non-Roman languages (i.e. Chinese, Japanese, Korean). We are developing a program that allows you to view library holdings in any of the languages in which they appear. When Willow retrieves a record that contains non-Roman text, it can start this viewer program to display it properly.

Similarly, the UW is participating in the TULIP project with Elsevier Publishing. They are supplying bitmap images of entire articles from selected journals. When a Willow-retrieved article citation matches a stored journal article, the user will be able to press a button and view the actual article using an external image viewer.

4.6 License Server

One complication the UW faced in the implementation of the database driver for our locally mounted BRS databases was the user-account problem. We wanted to support walk-in library users without forcing them to get personal accounts on the machine that runs BRS. Yet BRS requires each search session to run from a different account, so we could not just have a single public account. We implemented a general purpose network license server, and created pools of Willow accounts on the BRS host. When the user selects a BRS database, the BRS driver transparently contacts the license server and requests a BRS account and password. The account and password are then used to log into the host machine. The license server makes sure only one person is using an account at any time. We can also use the license server to make sure only a certain number of users are on a particular database at any time, which some contracts with database vendors require.

It is important to note the distinction between accounts on the database host system, and accounts on the Willow system. As discussed earlier, Willow itself is quite capable of running without accounts for individual users. The problem discussed here is with accounts on the machine where the database is running. Willow can also be configured to allow the user to use her own personal account and password on a database host, but this has become obsolete for us.

For increased security, the passwords are temporary -- they are generated on the fly by the license server, and can only be used for 30 seconds after they are issued. Also, the license server looks at the IP address of all requests, so we can deny non-UW requests for login-tickets to our licensed databases, but still allow a limited number of non-UW users into our Libraries Catalog.

5.0 Availability

The Willow release includes code produced and copyrighted by the University of Washington, the Massachusetts Institute of Technology, Stanford University, Online Computer Library Center Inc., and Carnegie Mellon University, but it is freely available for non-commercial purposes (i.e. even businesses are welcome to use it, but you can not re-sell it without permission). The *COPYRIGHT* file included in the Willow distribution contains the details. The source code (and pre-compiled binaries for a number of platforms) are available via ftp from *ftp.cac.washington.edu*, in the *willow* directory. The most up to date version of this document can also be found there. A few other items, such as source for the license server are also available. On the same ftp server, you will find a *wilco* directory, which contains a preliminary release of Wilco (the character-oriented version of Willow described above).

5.1 Willow Demo

If you have a good Internet connection, and an X display on a Unix machine, you can take Willow for a test drive. First, give the UW Willow server machines permission to put a program on your display, by issuing the command:

```
xhost +willow.u.washington.edu
```

The *xhost* command is usually found in */usr/bin/X11*. Next, use the Unix remote shell facility to tell the server to start a Willow session on your display. You will log in to the *demo* account, with the “-l” parameter (lower case L, not numeral 1). Your display is usually referenced by the name or IP address of the machine you are actually sitting at, with a “:0” appended at the end. So, if you are at an X Terminal called adonis.cs.wisc.edu, the command you would issue is:

```
rsh willow.u.washington.edu -l demo adonis.cs.wisc.edu:0
```

Our license server only lets a very limited number of outside users in at a time, so you may not be able to log into any UW databases once you get Willow up. Also, be aware that running Motif applications remotely over the Internet is quite bandwidth intensive, so Willow response may be sluggish. You will probably have better luck on both counts if you try this outside of business hours. If you do not have standard X11R4 colors and

75dpi fonts available, you may get some warnings about those, and Willow will come up looking pretty strange.

5.2 Wilco Demo

Wilco is available via UWIN (University of Washington Information Navigator), our text-based campus-wide information system. To access UWIN, use the command:

```
telnet uwin.u.washington.edu
```

UWIN is quite easy to use, and has extensive on-line help available. Go to the **Libraries** menu, and then to the **UW Libraries databases** section, and the UW Libraries Catalog will be available.

5.3 Mosaic Demo

An on-line hypertext version of this document, which includes live links to both the demos described above, examples of how to make Mosaic execute Willow directly, and more, is accessible with any WWW browser at:

```
http://www.cac.washington.edu/willow
```

5.4 Mailing Lists and Questions

Send any questions or comments about Willow that you may have, to:

```
willow@cac.washington.edu
```

To join the "willow-info" public mailing list for Willow release announcements and general Willow discussion, send a request to:

```
willow-info-request@cac.washington.edu
```

To send a message to everybody on that list, send it to

```
willow-info@cac.washington.edu
```

If you want to receive release announcements only, you can join the "willow-announce" list by sending a request to:

```
willow-announce-request@cac.washington.edu
```

6.0 Future Developments

Though Willow is now considered a mature system, there are still many enhancements we would like to add. For example, search history reference and sorting of result sets are not yet supported.

Further down the road, we need to figure out an intuitive and simple interface for allowing the naive user to construct complex boolean searches, i.e. a painless way of specifying ANDs, ORs, and NOTs, or even natural language processing. Parallel searches of multiple databases would also be a nice feature.

7.0 Willow Credits

- Primary design team: Matt Freedman, Scott Heyano, Ellen Jensen, Bill Jordan, Debra Ketchell, Ed Lightfoot and Jill McKinstry. Design ideas and feedback from many many others.
- Primary Willow development by Matt Freedman, UW Computing & Communications.
- Primary Wilco development by Scott Heyano, UW Computing & Communications.
- The Z39.50 driver is by Bill Cattey of MIT, based on the Stanford University Z39.50 API library by Harold Finkbeiner and incorporating code from the Online Computer Library Center, Inc.
- Willow for MS-Windows under development by Pete Libbey, UW Computing & Communications.
- The Willow help system programmed by Jim Fox of UW Computing & Communications. The help bitmaps were designed by Debra Ketchell of UW Health Sciences Library and Bill Jordan of the UW Libraries.
- Browse-list server programming by Leman Chung and Tom Unger of UW Computing & Communications. Call number browse algorithm by Bill Jordan, UW Libraries.
- The license server was designed and implemented by Steve Jones of UW Computing & Communications.
- User-analysis project coordinated by Debra Ketchell of the UW Health Sciences Libraries.
- BRS database programming for uwbrs driver by Dan Groves and Leman Chung, UW Computing & Communications.
- Willow Information Center banners and Willow logo by George Potratz.
- External support and mailing list coordination by Paul Reed Smith, UW Computing & Communication.