Chapter 6

MEASURING THE USE OF ELECTRONIC RESOURCES

This section discusses issues related to describing content—the scope of the library's electronic collection and how these materials are used.

Characterizing investments in electronic content

Libraries invest in many types of electronic content. One of the key measures of how a library stands in the continuum between the traditional and digital worlds involves the relative investments it makes in each type of content. Some types of electronic content typically procured by libraries include:

- Index and abstracting databases
- Individual electronic journals
- Aggregations of electronic journals
- Electronic theses and dissertations
- Electronic reference databases
- Business information services
- Electronic books
- Image databases
- News services

Almost all libraries already have reports that detail their spending for electronic versus traditional materials. As noted above, libraries that use the acquisitions module of their ILS for the procurement of electronic materials and code them in a way that makes them identifiable should be able to generate reports that characterize expenditures on electronic versus traditional materials. Those that do not make use of an ILS acquisitions module likely have spreadsheets or databases that track and summarize these expenditures.

Measuring access to electronic content

This section focuses on measuring how a library's patron makes use of electronic resources. Libraries have mastered the art of counting the use of books, journals, and other physical materials. ILS systems mostly do those functions automatically. With a book, for example, libraries can measure its use by the number of circulation checkouts and renewals. Libraries can know if a book was consulted but not checked out by scanning the book into the circulation system before reshelving it. The circulation module of the ILS then produces reports of the numbers of books used in any given period.

The process of measuring use is not so easy with electronic materials. But the library must develop a strategy so it has at least the same level of knowledge about usage of electronic resources as it has for books and other traditional materials.

Theoretically, the ability to measure access to electronic resources should be greater than traditional resources. As noted above, Web servers can be programmed to record each and every time an item of content is requested. The obstacles to comprehensive measurement of use tend be in the realm of coordination and organization, not in technical ability.

Vendor-supplied use statistics

A majority of the electronic content offered by libraries is provided through contracts or licenses with external vendors or content suppliers. To a large extent, the responsibility for reporting use can be placed on these vendors. Vendors are the ones, after all, who manage the servers involved and have control over the system logs, authentication systems, and other access management systems that record how their systems are used.

Many organizations have established guidelines on the types of use statistics that content suppliers must report. These organizations include:

The JSTOR Web Statistics Task force issued a report titled "Task Force To Develop Web-based Statistical Measures" in December 1997. Although this group worked directly with Journal Storage, The Scholarly Journal Archive (JSTOR), they intended their results to be applicable to other Web-based information resources. In this report the task force recommended that the following measurements be provided by vendors (paraphrased):

www.calstate.edu/SEIR/usge.stat.req.shtml

- The number queries or searches, both by database and IP address. Should this information be impossible to provide, the number of sessions or logins can be provided instead.
- The number of sessions turned away due to some limit set in the license or contract, such as exceeding the number of simultaneous users allowed.
- The number of items from many categories that were examined by the user. For abstract and index databases, report the number of citations examined. For full-text journals databases, record the number of tables of contents, abstracts, articles (mandatory), and other items viewed such as advertisements, images, and reviews.
- The use levels that correspond to the library's overall use of the resource for a given time period: queries, sessions, and turnaways, provided by day, month, year, and broken down by time of day. The vendor must also report peak simultaneous use if that is one of the measures stipulated in the contract. Report any downtime of the server, measured in hours, for each month.
- The report of the committee also addressed confidentiality and privacy issues, comparative statistics among institutions, and the access and delivery mechanisms.

The **International Coalition of Library Consortia** (ICOLC) produced a document entitled "Guidelines for Statistical Measures of Usage of Web-Based Indexed, Abstracted, and Full Text Resources" initially issued in November 1998. The guidelines of this report borrowed heavily from the JSTOR committee, introducing elements that specifically meet the needs of consortia as they collectively license information resources on behalf of their member libraries. Their requirements for statistical reporting by information providers are as follows (verbatim):

www.library.yale.edu/consortia/webstats.html

1. By each specific database of the provider
2. By each institutionally-defined set of IP addresses/locators to subnet level
3. By total consortium
4. By special data element passed by subscriber (for example, account or ID number)

5. By time period. Vendor's system should minimally report by month. For each month, each type of use should be reported by hour of the day, and vendor should maintain 24 months of historical data.

Use elements that must be provided are:

Number of queries (searches) categorized as appropriate for the vendor's information. A search is intended to represent a unique intellectual inquiry. Typically a search is recorded each time a search form is sent or submitted to the server. Subsequent activities to review or browse among the records retrieved or the process of isolating the correct single item desired do not represent additional searches, unless the parameter(s) defining the retrieval set is modified through resubmission of the search form, a combination of previous search sets, or some similar technique.

Number of menu selections categorized as appropriate to the vendor's system. If display of data is accomplished by browsing (use of menus), provide this measure. For example, an electronic journal site provides alphabetic and subject-based menu options in addition to a search form. The number of searches and the number of alphabetic and subject menu selections should be tracked.

Number of sessions (logins), if relevant, must be provided as a measure of simultaneous use. This number is not a substitute for either query or menu selection counts.

Number of turnaways, if relevant, as a contract limit (for example, requests exceed simultaneous user limit).

Number of items examined (that is, viewed, marked or selected, downloaded, emailed, or printed) to the extent these items can be recorded and controlled by the server rather than the browser:

1. Citations displayed (for abstracting and indexing databases)
2. Full text displayed broken down by title, ISSN with title listed, or other title identifier as appropriate
3. Tables of contents displayed
4. Abstracts displayed
5. Articles or essays, poems, chapters, and other works viewed (such as ASCII or HTML documents) or downloaded (such as PDF, email files)
6. Other (for example, image/AV files, ads, reviews, and others, as appropriate)

www.hpl.lib.tx.us/conferences/ala/outline.html

The **Public Library Association Research and Statistics Committee** developed a report that addresses statistics. The report included a section on the use statistics that should be reported by vendors (repeated verbatim, including tables):

Tracking and reporting usage of electronic materials is equivalent to tracking and reporting circulation and reference statistics. Proposed baseline usage measures and their nonelectronic equivalents include:

- Number of log-ins, sessions, or library visits
- Number of searches or reference questions (self-service)
- Number of items examined (viewed, downloaded, or printed) circulation

The usage measures suggested are consistent with the measures identified by the JSTOR Web Statistics Task Force and the ICOLC. Since many libraries depend on vendors to capture and transmit these statistics, the library profession should agree on a baseline set of measures, regardless of the type or size of library. Each segment of the profession may inter-

pret and report the statistics differently, as appropriate for individual audiences, but the vendors can only be expected to provide one common set of data elements.

A report for baseline usage data could look like this:

	# of logins/sessions
	# of searches
	# of items examined viewed, downloaded, or printed
	TOTAL

Expanded usage data for specified periods of time (week/month/year) could look like this:

	In-library use	Off-site use	Total use
Usage counts by database			
# of logins/sessions			
# of searches			
# of items examined (viewed, downloaded, or printed)			
# of turnaways			
Usage counts by time			
# of logins/sessions per hour			
# of searches per hour			
# of items examined (viewed, downloaded, or printed) per hour			
# of turnaways per hour			
Peak # of simultaneous users			
Usage counts by title			
# of items examined (viewed, downloaded, printed) per title			

You can see a remarkable degree of consistency among these three efforts. If information providers were to provide use statistics for each database or full-text resource according to these specifications, a large part of the process of measuring the use of electronic resources could be routinely collected and organized by each library.

Although most vendors will likely eventually satisfy this demand for consistent and thorough statistics, this information is not yet routinely available.

In October 2000, Judy Luther reported: "Providing usage data is a new role for publishers and aggregators—one that requires not only much learning but also a financial investment. Although the data appears to be as useful to publishers as to librarians, publishers must first develop the capability to serve their own purposes and then provide additional analyses and support to present the data so that librarians can use them.

"Less than half of the publishers who offer journals in electronic form today are able to provide statistics on the usage of these journals. What is available varies widely among publishers, and librarians are often unclear about what to ask for and how they will use the data. Guidelines for compiling statistics are just emerging and have not been widely adopted."

***Source:** Luther, Judy. White paper on "Electronic Journal Usage Statistics. Council on Library and Information Resources." October 2000. www.clir.org/pubs/reports/pub94/introduction.html#issues*

Only limited progress has been made since this report was published. Likewise, Charles McClure states:

"The issue of non-existent or inconsistent, incomparable usage statistics provided by external information content providers (vendors) was identified as a major stumbling block for libraries to gauge rapidly increasing use of electronic resources by research library users and thus making it difficult to use the data as sources for establishing library outcomes."

***Source:** McClure, Charles. "Identifying and Measuring Library Activities/services Related to Academic Institutional Outcomes,"2002, p. 2.*

From the vendors' perspective, the task of providing statistical information to their library customers is not a trivial duty they can immediately fulfill. In some cases, their systems may not have the technical capability to gather and report statistics in the way libraries have specified the information be delivered.

Not all libraries want exactly the same thing, making the task for vendors even more complex. Although some consistency exists in what some of the larger organizations have specified for use statistics, individual libraries may present vendors with additional requests.

Companies in the business of providing information resources have limited resources. Allocating programming staff to reporting use statistics may be a lower priority than implementing other requested new features. Some information providers are reluctant to provide user statistics because they believe usage data are proprietary information and that releasing that data puts them at a competitive disadvantage.

The reporting of use statistics is also complicated when products are licensed through contracts by library consortia. In such cases, two levels of reporting may be necessary, one for the consortium as a whole and another for each library within the consortium.

More vendors will supply use statistics when that function is part of the business contract negotiated with the library. If the level of use data provided becomes a decision point as libraries choose between competing information products, vendors will be highly motivated to offer enhanced reports. Libraries also may want to propose stipulations regarding the provision of use statistics when they negotiate contracts for new products and services.

The industry has a few years before libraries can rely on vendors to provide an adequate set of statistics that measures how each library's patron accesses licensed resources. Libraries today cannot count on receiving comprehensive use statistics from vendors for all the information products licensed.

Library-collected use statistics

Without comprehensive use statistics provided by the vendors of the information products, libraries must monitor use and collect statistics locally. Even if all vendors provided use statistics, libraries would still likely need to gather local information to verify that the numbers provided by the vendors are correct, especially if the level of use drives up the cost of the product.

This section describes three alternative approaches for collecting statistics locally for external Web-based resources. These resources are typically databases and e-journals licensed by libraries.

This section also describes ways to implement a moderately complex system and lists some simple computer programs. Nontechnical readers may gain some insight to the process and learn a little about the way that Web servers and scripts work. More technical readers, such as systems librarians, server administrators, or programmers may find an approach suitable to their environment and be able to modify and employ the examples provided, making whatever adjustments are necessary for the local hardware and software environment.

Capturing information about the use of electronic resources by the library is challenging. Counting the use of items on a locally maintained Web server is a straightforward process. Only under unusual conditions, mostly associated with Web caches, can activity go unrecorded. Recording the use of a resource when it resides on a Web server outside your local environment is difficult.

Consider a typical example: a library provides access to an electronic journal through a link on a library's Web page. The page might be a listing of e-journals in a particular subject area. The figure below is a portion of what might be a typical example of a page on a library Web server that lists the electronic resources in a particular discipline.

Electronic Journals related to the subject: Philosophy

American Journal of Philology *Vanderbilt users only* **[Project Muse]** [Info]

Analysis *Vanderbilt users only* **[Ingenta]** [Info]

Animals' Agenda, The *Vanderbilt users only* **[ProQuest]** [Info]

Animus: A Philosophical Journal for Our Time [Info]

This Web page fragment illustrates a typical listing of e-journals.

When a library user loads the Web page with the subject listing, the page request is recorded in the log files of the library's Web server. But clicking the link goes unnoticed by the local Web server. The page loaded by clicking on the link is on an external server, and any pages viewed by the user appear in the external server's log files and are not recorded by the local server. Recording the use of local pages that are jumping-off points to external resources is easy, but recording use of the external resources themselves takes some creative maneuvering.

The sections that follow offer approaches that enable the library to measure when users access external resources.

Intermediate HTML pages

One simple approach to creating a record of when a library user accesses an external Web-based resource is to create an intermediate page for each resource and channel the users' navigation path so they load this page to gain access to the resource. With the subject list of e-journals, for example, you create a link to a particular e-journal coded with the URL of a page that might have a brief description of the resource. The page has a link that connects to the resource itself. In this example, the link on the subject list page of *Electronic Journals for Philosophy* for the title "British Journal for the Philosophy of Science" is coded something like this:

```
<a href="bjps.html">british Journal for the Philosophy
of Science</a>
```

The page "bjps.html" provides any explanatory information that might be helpful to the user and presents a link to the resource. Since most users travel through the page bjps.html as they access the *British Journal of the Philosophy of Science,* the number of times this page was viewed is a strong indicator of the use of that particular electronic journal.

The library examines its Web log analysis report to determine the counts for each of the intermediate pages that lead to electronic resources. The measurement is not exact since cases exist where a user might view the intermediate page, see that the description doesn't match their interests, and then use the Back button to go elsewhere. Users who find this title to be especially interesting might create a bookmark in their browser so they can return to the title without having to go through the intermediate page (or the library's Web site altogether). Some of the user actions that lead to discrepancies overreport the use, and others will underreport, canceling each other out so the access count of this page more or less reflects the use of the resource itself.

This technique for capturing use of resources may not be popular with library users, because it forces them through an extra step for every electronic journal they use on your site. In the realm of Web page usability, the economy of clicks to reach a resource is paramount. This strategy trades off a degree of usability as it increases the library's ability to gather statistics.

The advantage of this method lies in its simplicity. Anyone who has the ability to create Web pages can accomplish it, unlike some of the more sophisticated approaches described later that require installing scripts on the server.

Although simple, this technique of using intermediate HTML pages for gathering statistics may be unwieldy for monitoring a large number of resources. You would have to transfer a huge number of statistics from the Web log file analysis system into a spreadsheet or other application used for managing journal statistics. This strategy is especially onerous for academic libraries, which often subscribe to several thousand electronic journals.

Pass-through scripts

Libraries with the ability to run scripts on their Web server can implement a system that records the use of external resources without introducing an intermediate Web page. Such a system is one step higher in difficulty than the simple intermediate page method described above yet is within the grasp of most libraries that operate their own Web server.

Author Disclaimer: *In describing this method, my main intent lies in providing the general concept, not in providing a complete system. Although I have tested and implemented scripts like these on both Unix and Windows-based environments, libraries that decide to follow such an approach likely will follow their own local preferences. The examples I provide are written in Perl, since this is the language with which I'm most familiar. Almost any other programming language could be used instead. Many security-related and error-checking components have been left out of the script listings for the sake of simplicity and clarity.*

To use scripts, your Web server must be configured to allow executable programs. You also need someone with at least a minimal level of computer programming ability, and server administrators need to provide access to the directories in the server designated for such programs. For libraries that manage their own Web servers, this level of access is usually possible. Libraries that rely on externally hosted Web servers may have more difficulty in getting their server administrators to provide scripting support. In today's environment Web scripting is commonplace, and most servers are configured to support scripts.

Scripts are programs that carry out a set of commands or instructions on the Web server. They can be simple, with just a few lines of program instructions that perform a small task, or they can be large and complex. A complete search-and-retrieval environment or an e-commerce system can be created using scripts. Many programming languages are available for the creation of scripts. Some of the most common languages include:

- ASP, commonly used on Microsoft Windows-based Web servers
- Perl, a versatile programming language, popular on Unix-based systems but also available for Windows-based systems, Macintosh servers, and most others
- PHP, a scripting language developed especially for Web-based systems; mostly used in Unix-based systems

A basic system for recording the use of external Web-based resources can be created through a simple Web script. The job of the script is to record any given title into a log file when a library user clicks its link. For this system to work, each link is coded to invoke the script, which creates a log file entry indicating the date and time and the name of the resource involved, and then redirects the user's Web browser to that resource. This system creates its own log file, separate from the one used by the Web server itself. From the end-user's perspective, the process is transparent: they clicked on the link and their Web browser went directly to the associated resource.

Here's a look at the process one step at a time.

In a standard Web page, a link to the ALA TechSource Web site would be coded in this way:

```
<a href=http://techsource.ala.org/>ALA TechSource</a>
```

In a scripted pass-through environment, the same link might be coded as follows:

```
<a href="http://www.library.Vanderbilt.edu/
scrpts?URL=http://techsource.ala.org/">ALA TechSource</
a>
```

Note that the URL now points to the location of the pass-through script, not to that of the destination resource. Since the pass-through script needs this information to do its work, the URL of the destination is passed as a parameter. The syntax for posting additional information on a URL is to use a question mark separating the URL from its additional parameters, followed by one or more sets of name or value pairs. In this case, the name of the parameter is the URL and its value is the URL of the destination resource.

The link is still presented on the HTML page as "ALA TechSource," so users of the page do not see anything different. If users look at the status bar at the bottom of their Web browser, they would see that the link points to a local server. If desired, adding a few lines of JavaScript could mask that artifact, and make the link look entirely normal:

```
<a href="http://www.library.Vanderbilt.edu/
scrpts?URL=techsource.ala.org/"
onMouseOver ="self.status='http://techsource.ala.org';
return true"
onClick     ="self.status='http://techsource.ala.org';
return true"
onMouseOut  ="self.status=' '; return true"
>ALA TechSource</a>
```

The script itself performs two tasks:

1. Write an entry in a log file in a way that is invisible to the user.
2. Redirect the user's browser to the resource.

Now let's look at an example, written in the Perl programming language, that accomplishes these two steps.

```
# Get the current date and time
($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)=localtime(time);
$fullyear        = 1900 + $year;
%monthname = ("0","Jan","1","Feb","2","Mar","3","Apr",
         "4","May","5","Jun","6","Jul","7","Aug","8","Sep",
              "9","Oct","10","Nov","11","Dec");
$month           = $monthname{$mon};
# Get the parameters passed to the script
$ip=$ENV{'REMOTE_ADDR'};
@pairs=split(/&/,$temp);
foreach $item(@pairs) {
     ($key,$content)=split (/=/,$item,2);
     $content=~tr/+/ /;
     $content=~ s/%(..)/pack("c",hex($1))/ge;
     $fields{$key}=$content;
}
# Step 1: Record the entry into the log file:
$logdirectory     = "/logfiles/";
$logfilename      = "ERES-$fullyear-$month.log";
open (ACCESSLOG, ">>$logdirectory\\$logfilename");
print ACCESSLOG "$date|$ip|$fields{'URL'}|\n";
close (ACCESSLOG);
```

```
# Step 2: Redirect user's browser to the URL:
print "Location: $fields{'URL'}\n\n";
```

One of the technical requirements for a script like this one to operate involves having the ability to create and update files. The log files should be placed in a separate directory from the script that generates them. Basic security concerns dictate that scripts always reside in a directory that is read-only. Otherwise, hackers might be able to place their own programs in that directory, which would give them the ability to gain full control of your system.

With the pass-through script in place, an entry appears in a log file each time a user clicks one of the specially coded links to gain access to an electronic resource. In the example above, a new log file will be started each month. Each month's log file then can be analyzed to determine how many times each resource was used. You might, for example, write a script to count each unique entry, or the file might be imported into a spreadsheet or database application for counting, summarizing, and analyzing.

Great flexibility is possible in implementing a pass-through script, as shown in the example above. For example, the script doesn't need to run on the same server as the Web server that hosts the library's Web pages. Many organizations, especially those with complex Web environments, choose to run scripts and database-driven applications on a separate server from their primary Web server.

Although this approach has some advantages over the one using intermediate pages, it also has some limitations:

- All the links to external electronic resources must be re-coded in HTML to the form required by the script, a laborious process if many resources are involved.
- The system does not record access to resources if the resources are accessed directly or through links pages without the special coding.
- Analysis of the log files can be difficult.
- Access to the resources may be blocked if problems occur with the script. Thoroughly test the script before putting it into production use. The server on which the script runs should be fast and reliable.

Despite these limitations, the system does accomplish the task of creating a local record each time an electronic resource is accessed through the library's Web pages.

Using the library's online catalog with pass-through scripts

Many libraries provide access to their electronic resources through their Web-based online catalog, so each of these resources is cataloged. USMARC includes an 856 field, with a subfield "u" designated for the URL of a Web-based resource. Most Web OPACs present the 856 as a clickable link that takes the user to that resource. But just like links presented on other Web pages, no record is left on the local system when users click the link to access the resource.

You can incorporate the use of a pass-through script within the library's Web OPAC to capture use. How, or if, this capture can be accomplished depends on the integrated library system (ILS) used and its degree of customizability. Two alternative approaches include:

- Changing the contents of the 856 field itself to accommodate the structure of the pass-through script. This approach has some significant disadvantages. Changing the 856 introduces nonstandard data into the catalog record, which goes against most libraries' cataloging practices. Changing all the records involved is a laborious process. Should anything change about the pass-through script, or should the library decide to quit using it, all the records would have to be changed yet again.
- Customizing the Web OPAC to construct the URLs it presents in the form required by the pass-through script. Some Web OPACs give the library liberal control over how the HTML is generated as the displays catalog records. With a little tweaking, the HTML generated can be modified so the clickable links associated with the 856 field are constructed in the new form when the records display, following the same model as shown above for static HTML pages.

Libraries should, of course, proceed carefully before making changes to their online catalog. Consult your ILS vendor. The vendor may be able to advise you on how to implement a pass-through script such as the one described here or may provide alternative means of counting the use of electronic resources as they are accessed through the vendor's system.

Database-driven access systems

Next, consider how the use of a database can be implemented both as a more sophisticated approach to providing access to electronic resources but also as a more efficient way to record and report how resources are used.

First, review the basic concept of a Web-enabled database, especially as it might be applied to managing the library's collection of electronic resources. Such a system can be built as a gateway or finding aid for the library's electronic resources, providing users with the ability to find resources by browsing through dynamically generated lists of titles, by browsing subject categories, or by performing a search. Such a system also can measure use by adding counters to each record that are incremented each time they are accessed. With the use information stored in the database, creating reports that present use statistics associated with each resource becomes simple.

As libraries build collections of hundreds or thousands of electronic resources, providing access to them through basic Web pages becomes a laborious proposition. These Web pages are static—they are built in advance and do not change their appearance unless someone changes them manually. Not only is maintaining a page with links to hundreds of items difficult and tedious, but the effort must be doubled or tripled to provide multiple ways of finding the links. If one set of pages lists resources alphabetically and another provides access by subject, then each link must be coded on at least two separate HTML pages. When titles are added, removed, or if the URL for the link changes, the Web editor must make the same changes on each of these multiple pages. If the library decides to change the look of its Web pages, then the process of revamping these listings can be difficult.

An alternative to creating static Web pages to manage electronic resources involves the use of a database that can dynamically create HTML pages based on requests made by library users. The task requires cataloging all the electronic resources into a database and creating scripts that retrieve and sort records, adding HTML code to make them look nice, and delivering

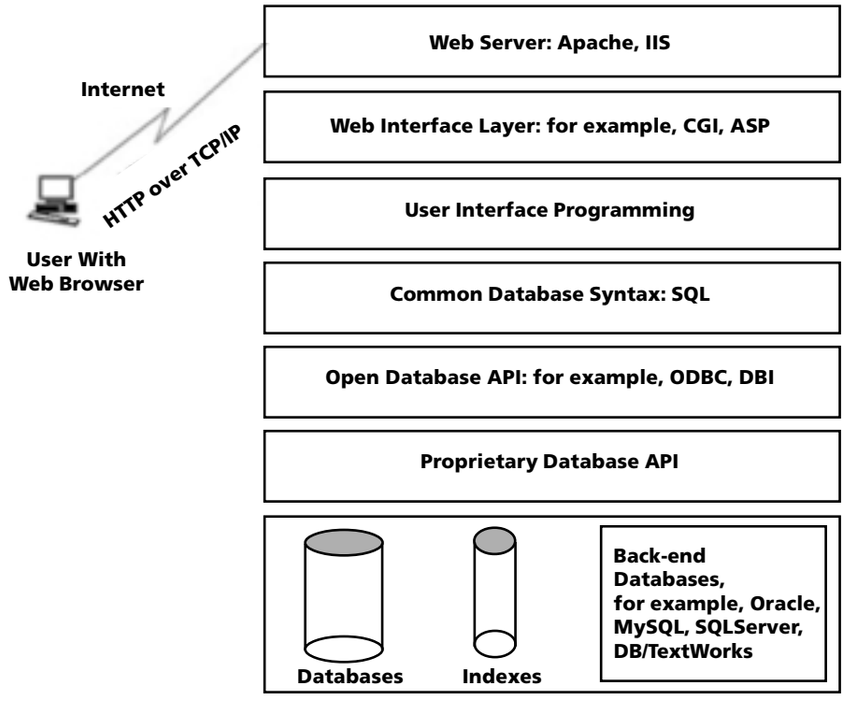
them to the user's Web browser. This model also provides some good infrastructure for counting the use of the resources. Before going into that model, however, first consider the approach in detail.

Web-enabled databases are not necessarily difficult or costly to create. These databases can be made using commercial products such as MacroMedia's ColdFusion or can be developed using scripting languages such as PHP or Perl in conjunction with an Open Source, no-cost database such as MySQL. The figure below illustrates the basic model that underlies all Web-enabled databases. This figure shows that many components, separated into functional layers, work together to provide access to the information held in the database to users on the Web. Although this section isn't intended to be a tutorial on creating Web-enabled databases, some basic understanding is necessary to learn how these databases can be used to assist in gathering and reporting use statistics.

Regardless of what software tools you use to create the database, the same basic concepts apply. Information about the each resource is stored in a database. At a minimum, each record contains its title and URL, but most likely you include a description, publisher, dates of coverage, and other details. A page is then created in HTML that provides access to the records. Such a page might have links for each of the major subject areas represented in the database. The link is coded in such a way that, instead of simply displaying a static Web page, it invokes a script that gathers the records in the database that match the request. The script also takes care of sorting the records and generating a Web page that displays the results of the request.

From the end-users' perspective, this dynamically generated page might look just like the static lists, but through a more efficient and manageable process for the library. Through this database-driven approach, providing multiple ways to find resources is easy. A search box can be created to find resources by keywords in their titles or descriptions, links can be provided to subject categories, or letter-by-letter alphabetical lists can be generated.

Generalized Model for Web-enabled Databases



One approach to measuring electronic use counts items as they are clicked from within the library's Web environment or online catalog through the use of a pass-through script. This method simply creates a log file, which then must be processed and analyzed as a separate process.

Through the use of a database, though, you can create a counting mechanism that indicates the number of times each resource is accessed in any given period. Such a database can serve many functions, such as: general management of electronic resources, creating an access system or finding aid for library users, and creating a consistent display of resources without laborious HTML coding.

As an example of using a Web-enabled database to facilitate the process of recording and reporting use statistics for electronic resources, look at the system at Vanderbilt University. The system provides many components for the management of electronic resources, providing interfaces to end-users to help them find resources and measuring the level of use of each resource.

See one of the interfaces library patrons use to access electronic resources through Vanderbilt University's system at www.library.vanderbilt.edu/eresources.shtml.

The system measures the use of each electronic resource through the use of a series of counters. A field is created for each month, and the system automatically increments the counter for the current month. All previous monthly counters are preserved for studying patterns of how items are used.

The system provides an interface to assist users in the process of finding electronic resources. All the Web pages the user sees are created dynamically, presenting lists of resources that can be accessed by clicking the links provided.

In a more basic form, the links provided would take the user directly to the resource. But since the system controls the underlying HTML coding for the links, presenting them in a way that takes advantage of a pass-through script using the same techniques as described above is easy. To reiterate, the general process creates a link to a resource that passes through an intermediate script before delivering the user's Web browser to the destination the link represents.

To better understand how the system works, the following steps show the way the system builds the links seen by the user. Remember that the HTML is generated dynamically by the scripts that are part of the Web-enabled database application and are not manually coded through an HTML editor.

If the system were to generate a link that simply connects the user to the resource, the link would take the form of a typical link coded in HTML:

```
<a href=http://www.worldcat.oclc.org>OCLC WorldCat</a>
```

To capture use statistics, the system must be coded to use a pass-through script. In this example, the script is written in Perl and resides on a server called libxx.library.Vanderbilt.edu, and is named go2.pl.

For each link presented on a dynamically generated HTML page, links are constructed according to the following example. Note that the URL provided in the link is one for the pass-through script and that two parameters are provided: the URL for the destination resource and the corresponding record number in the database to which it corresponds. Remember that the link is being created from a database record, so it is passing along the record number for the record it is currently processing.

```
<a href="http://libxx.library.vanderbilt.edu/diglib/
go2.pl?http://www.worldcat.oclc.org&RC=23453">OCLC
WorldCat</a>
```

To avoid end-user confusion, also add some JavaScript commands to mask the intermediate script in much the same way as the previous example did in a static HTML environment:

```
<a href=http://libxx.library.vanderbilt.edu/diglib/
go2.pl?http://www.worldcat.oclc.org&RC=23453'
onMouseOver =self.status=' http://www.worldcat.oclc.org
'; return true
onClick     ="self.status='http://www.worldcat.oclc.org
'; return true"
onMouseOut  ="self.status=' '; return true"
>OCLC WorldCat</a>
```

These links are constructed by the scripts that underlie the interface of the Web-enabled database system for providing access to electronic resources. To provide a small example of how this action is accomplished, the following code shows how such a link is created using Perl. Note that this code is just a small part of a larger program that dynamically creates a Web page based on records returned from a request to the database:

```
$goto="http://libxx.library.vanderbilt.edu/diglib/
go2.pl?URL=";
print " <a href=\"$goto$data{'URL'}",
       "&RC=$data{'RecordNumber'}\"\n",
       " onMouseOver =\"self.status=\'$data{'URL'}\';
return true\"\n",
       " onClick     =\"self.status=\'$data{'URL'}\';
return true\"\n",
       " onMouseOut  =\"self.status=\' \'; return
true\"\n",
       "><b>$data{'Title'}</b></a>\n";
```

Finally, look at the pass-through itself. It works much like the example above, except that instead of just writing a log entry, it updates a counter in the database record that corresponds to the resource requested. This example is taken from a Windows 2000 server running ActivePerl from ActiveState and requires the Win32:ODBC Perl mod from Roth Consulting. Minor modifications may be necessary to customize the script for other operating systems or database connectivity environments.

The script itself performs tasks:

1. Retrieves the database record associated with the URL
2. Reads the current value of the current month's counter
3. Increments the counter
4. Updates the current month's counter with its new value
5. Redirects the user's browser to the resource

Steps 1 to 4 happen in a way that is invisible to the end-user. From the user's perspective, he or she clicked on the URL and simply viewed the page in the Web browser.

```
# Script for providing logged access to Web resources
# Created October 9, 2000 by Marshall Breeding
# Last modified October 11, 2000
#
# This script takes two arguments:
#  1.) the URL to be launched
#  2.) the RecordNumber for this record in the "eres"
electronic resources database
```

```
#
# This script:
#  1. Uses a SQL statement to access the eres database
through ODBC to fetch
#     the current value of the counter for the current
month.
#  2. Increments the counter
#  3. Uses SQL to update the record with the
incremented counter
#  4. Write the following line of information to the
current month's log file:
#        - Date/Time stamp
#        - IP address of use
#        - URL (proxy component removed)
#        - eres record key
#  5. Performs redirection to the resource by printing
the URL into the location box of the user's browser.
#
#
# first we need to accept the raw string from the form
$temp=$ENV{'QUERY_STRING'};
# get the ip address while we're at it
$ip=$ENV{'REMOTE_ADDR'};
$host=$ENV{'REMOTE_HOST'};
($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)=localtime(time);
if (length($min) ==1) {$min = '0'.$min};
if (length($sec) ==1) {$sec = '0'.$sec};
%monthname = ("0","Jan","1","Feb","2","Mar","3","Apr",
          "4","May","5","Jun","6","Jul","7","Aug","8","Sep",
               "9","Oct","10","Nov","11","Dec");
#
$fullyear = 1900 + $year;
$month=$monthname{$mon};
++$mon;
$date="$mon/$mday/$fullyear, $hour:$min";
@pairs=split(/&/,$temp);
foreach $item(@pairs) {
     ($key,$content)=split (/=/,$item,2);
     $content=~tr/+/ /;
     $content=~ s/%(..)/pack("c",hex($1))/ge;
     $fields{$key}=$content;
}
use Win32::ODBC;
my($db) = new Win32::ODBC('DIGLIB');
$currentcounter=$month.$fullyear;
$SqlStatement="SELECT * FROM eres WHERE RecordNumber =
\'$fields{'RC'}\'";
if ($db->Sql($SqlStatement)) {
   print "Content-type: text/html\n\n";
   print "Error: " . $db->Error() . "<br />\n";
   $db->Close();
   exit;
}
while ($db->FetchRow()) {
     my(%data) = $db->DataHash();
```

```
     if (length($data{$currentcounter}) > 0) {
       $counternow=$data{$currentcounter};
     } else {
         $counternow=0;
     }
}
$newcounter = $counternow+1;
$filestring="eres-$fullyear-$month.log";
# Let's not show the proxy string in the logs:
$loggedURL=$fields{'URL'};
$loggedURL=~s/http:\/\/proxy.library.vanderbilt.edu\/
login\?url=//;
open (DATAFILE, ">>e:\\diglib\\logs\\".$filestring);
print DATAFILE "$date|$ip|$loggedURL|$fields{'RC'}\n";
close (DATAFILE);
$SqlStatement="UPDATE eres SET $currentcounter =
\'$newcounter\' WHERE RecordNumber=\'$fields{'RC'}\'";
if ($db->Sql($SqlStatement)) {
     print "Content-type: text/html\n\n";
     print "<b>Error:</b> " . $db->Error() .
"<br>\n\n";
     print "<b>SqlStatement:</b> $SqlStatement<br />
>\n\n";
     $db->Close();
    exit;
}
$db->Close();
print "Location: $fields{'URL'}\n\n";
```

Although techniques such as these may be helpful in collecting data about the way patrons use remotely hosted resources, these techniques do have limitations. The principal problem is that these scripts work only when library users gain access to these resources through library-provided gateways rather than direct access.

Proxy servers

Information providers limit product access to their licensed customers. In the current networked environment, the most common method employed for restricting access to authorized users involves IP address filters. When a library signs a license with a vendor for a product, the library provides a list of IP addresses that represent that organization's network. Although a college, university, school, or corporation might be able to provide a range of IP addresses that identify their local networks, they cannot provide such a range for remote users.

When a person attempts to access a restricted resource, the system checks the user's IP address. If the address falls within the range of addresses associated with one of their customer libraries, access is allowed. Although this arrangement works well for those people who access the resource within the organization's network, it fails to provide access to home users and others on a computer with an IP address associated with an independent Internet service provider (ISP).

The most popular method for authenticating remote users employs a proxy server. Proxy servers provide an institutional IP address to remote users

whose native IP address isn't accepted. The proxy server resides at the library and does its own authentication to ensure the remote user is affiliated with the library—usually through a username and password. Once the user has been authenticated by the proxy server, the access is allowed.

Several styles of proxy servers are available. Some require the users to make changes in the configuration of their browser. The type of proxy server most commonly used in libraries, called a reverse URL-rewriting proxy, does not require users to make changes in their browsers, but it does involve presenting URLs in a different way, much like the technique above for the pass-through script. URLs are coded to link to the proxy server, with the URL of the destination resource passed along as a parameter. Just like the modified URLs for the pass-through script, the changes for the proxy server can usually be implemented in some global or automatic fashion.

Implementing a proxy server also can be a valuable source of use statistics. Proxy servers usually can be set to record every instance of use into log files. The logs of the proxy server then can be mined to create statistical reports that show the use of each of the library's restricted resources by remote users. A proxy server also can log use by users within the local network. In these cases, the user doesn't need to be prompted for their username and password, but the server inserts the instance of use into the log files. Given that many local users may bypass the institution's proxy server by connecting directly to the resource, as noted in the section on unmediated access, the server still cannot serve as a comprehensive gateway for gathering use statistics but does provide another option.