Harvesting Information from a Library Data Warehouse

Siew-Phek T. Su and
Ashwin Needamangala

Data warehousing technology has been defined by John Ladley as "a set of methods, techniques, and tools that are leveraged together and used to produce a vehicle that delivers data to end users on an integrated platform."[1] This concept has been applied increasingly by industries worldwide to develop data warehouses for decision support and knowledge discovery. In the academic sector, several universities have developed data warehouses containing the universities' financial, payroll, personnel, budget, and student data.[2] These data warehouses across all industries and academia have met with varying degrees of success. Data warehousing technology and its related issues have been widely discussed and published.[3] Little has been done, however, on the application of this cutting edge technology in the library environment using library data.

Motivation of Project

Daniel Boorstin, the former Librarian of Congress, mentions that "for most of Western history, interpretation has far outrun data."[4] However, he points out "that modern tendency is quite the contrary, as we see data outrun meaning." His insights tie directly to many large organizations that long have been rich in data but poor in information and knowledge. Library managers are increasingly finding the importance of obtaining a comprehensive and integrated view of the library operations and the services it provides. This view is helpful for the purpose of making decisions on the current operations and for their improvement. Due to financial and human constraints for library support, library managers increasingly encounter the need to justify everything they do—for example, the library's operation budget. The most frustrating problem they face is knowing that the information needed is available somewhere in the ocean of data but there is no easy way to obtain it. For example, it is not easy to ascertain whether the materials of a certain subject area, which consumed a lot of financial resources for their acquisition and processing, are either frequently used (i.e., high rate of circulation), seldom used, or not used at all. Or, whether they satisfy users' needs. Another example, an analysis of the methods of acquisition (firm order vs. approval plan) together with the circulation rate could be used as a factor in deciding the best method of acquiring certain types of material. Such information can play a pivotal role in performing collection development and library management more efficiently and effectively. Unfortunately, the data needed to make these types of decisions are often scattered in different files maintained by a large centralized system, such as NOTIS, that does not provide a general querying facility or by different file/data management or application systems. This situation makes it very difficult and time-consuming to extract useful information. This is precisely where data warehousing technology comes in.

The goal of this research and development work is to apply data warehousing and data mining technologies in the development of a Library Decision Support System (LDSS) to aid the library management's decision making. The first phase of this work is to establish a data warehouse by importing selected data from separately maintained files presently used in the George A. Smathers Libraries of the University of Florida into a relational database system (Microsoft Access). Data stored in the existing files were extracted, cleansed, aggregated, and transformed into the relational representation suitable for processing by the relational database management system. A graphical user interface (GUI) is developed to allow decision makers to query for the data warehouse's contents using either some predefined queries or ad hoc queries. The second phase is to apply data mining techniques on the library data warehouse for knowledge discovery. This paper covers the first phase of this research and development work. Our goal is to develop a general methodology and inexpensive software tools, which can be used by different functional units of a library to import data from different data sources and to tailor different warehouses to meet their local decision needs. For meeting this objective, we do not have to use a very large centralized database management system to establish a single very large data warehouse to support different uses.

Local Environment

The University of Florida Libraries has a collection of more than two million titles, comprising over three million volumes. It shares a NOTIS-based integrated system with nine other State University System (SUS) libraries for acquiring, processing, circulating, and accessing its collection. All ten SUS libraries are under the consortium umbrella of the Florida Center for Library Automation (FCLA).

Siew-Phek T. Su (pheksu@mail.uflib.ufl.edu) is Associate Chair of the Central Bibliographic Services Section, Resource Services Department, University of Florida Libraries, and **Ashwin Needamangala** (nsashwin@grove.ufl.edu) is a graduate student at the Electrical and Computer Engineering Department, University of Florida.

Library Data Sources

The University of Florida Libraries' online database, LUIS, stores a wealth of data, such as bibliographic data (author, title, subject, publisher information), acquisitions data (price, order information, fund assignment), circulation data (charge out and browse information, withdrawn and inventory information), and owning location data (where item is shelved). These voluminous data are stored in separate files. The NOTIS system as used by the University of Florida does not provide a general querying facility for accessing data across different files. Extracting any information needed by a decision maker has to be done by writing an application program to access and manipulate these files. This is a tedious task since many application programs would have to be written to meet the different information needs. The challenge of this project is to develop a general methodology and tools for extracting useful data and metadata from these disjointed files, and to bring them into a warehouse that is maintained by a database management system such as Microsoft Access. The selection of Access and PC hardware for this project is motivated by cost consideration. We envision that multiple special purpose warehouses be established on multiple PC systems to provide decision support to different library units.

The Library Decision Support System (LDSS) is developed with the capability of handling and analyzing an established data warehouse. For testing our methodology and software system, we established a warehouse based on twenty thousand monograph titles acquired from our major monograph vendor. These titles were published by domestic U.S. publishers and have a high percentage of DLC/DLC records (titles cataloged by the Library of Congress). They were acquired by firm order and approval plan. The publication coverage is the calendar year 1996–1997. Analysis is only on the first item record (future project will include all copy holdings). Although the size of the test data used is small, it is sufficient to test our general methodology and the functionality of our software system.

FCLA DB2 Tables and Key List

Most of the data from the twenty-thousand-title domain that go into the LDSS warehouse are obtained from the DB2 tables maintained by FCLA. FCLA developed and maintains the database of a system called Ad Hoc Report Request Over the Web (ARROW) to facilitate querying and generating reports on acquisitions activities. The data are stored in DB2 tables.[5]

For our research and development purpose, we needed DB2 tables for only the twenty-thousand titles that we identified as our initial project domain. These titles all have an identifiable 035 field in the bibliographic records (zybp1996, zybcip1996, zybp1997 or zybpcip1997). We used the BatchBAM program developed by Gary Strawn of Northwestern University Library to extract and list the unique bibliographic record numbers in separate files for FCLA to pick up.[6] Using the unique bibliographic record numbers, FCLA extracted the DB2 tables from the ARROW database and exported the data to text files. These text files then were transferred to our system using the file transfer protocol (FTP) and inserted as tables into the LDSS warehouse.

Bibliographic and Item Records Extraction

FCLA collects and stores complete acquisitions data from the order records as DB2 tables. However, only brief bibliographic data and no item record data are available. Bibliographic and item record data are essential for inclusion in the LDSS warehouse in order to create a viable integrated system capable of performing cross-file analysis and querying for the relationships among different types of data. Because these required data do not exist in any computer readable form, we designed a method to obtain them. Using the identical NOTIS key lists to extract the targeted twenty-thousand bibliographic and item records, we applied a screen scraping technique to scrape the data from the screen and saved them in a flat file. We then wrote a program in Microsoft Visual Basic to clean the scraped data and saved them as text-delimited files that are suitable for importing into the LDSS warehouse.

Screen Scraping Concept

Screen scraping is a process used to capture data from a host application. It is conventionally a three-part process:

- Displaying the host screen or data to be scraped.
- Finding the data to be captured.
- Capturing the data to a PC or host file, or using it in another Windows application.

In other words, we can capture particular data on the screen by providing the corresponding screen coordinates to the screen scraping program. Numerous commercial applications for screen scraping are available on the market. However, we used an approach slightly different from the conventional one. Although we had to capture only certain fields from the NOTIS screen, there were other factors that we had to take into consideration. They are:

- The location of the various fields with respect to the screen coordinates changes from record to record. This makes it impossible for us to lock a particular field with a corresponding screen coordinate.

- The data present on the screen are dynamic because we are working on a "live" database where data are frequently modified. For accurate query results, all the data, especially the item record data where the circulation transactions are housed, need to be captured within a specified time interval so that the data are uniform. This makes the time taken for capturing the data extremely important.
- Most of the fields present on the screen needed to be captured.

Taking the above factors into consideration, it was decided to capture the entire screen instead of scraping only certain parts of the screen. This made the process both simpler and faster. The unnecessary fields were filtered out during the cleanup process.

System Architecture

The architecture of the LDSS system is shown in figure 1 and is followed by a discussion on its components' functions.

NOTIS

NOTIS (Northwestern Online Totally Integrated System) was developed at the Northwestern University Library and introduced in 1970. Since its inception, NOTIS has undergone many versions. University of Florida Libraries is one of the earliest users of NOTIS. FCLA has made many local modifications of the NOTIS system since UF Libraries started using it. As a result, the UF NOTIS is different from the rest of the NOTIS world in many respects. NOTIS can be broken down into four subsystems:

- acquisitions
- cataloging
- circulation
- online public access catalog (OPAC)

At the University of Florida Libraries, the NOTIS system runs on an IBM 370 main frame computer that runs the OS/390 operating system.

Host Explorer

Host Explorer is a software program that provides a TCP/IP link to the main frame computer. It is a terminal emulation program supporting the IBM main frame, AS/400, and VAX hosts. Host Explorer delivers an enhanced user environment for all Windows NT platforms, Windows 95 and Windows 3.x desktops. Exact TN3270E, TN5250, VT420/320/220/101/100/52, WYSE 50/60 and ANSI-BBS display is extended to leverage the wealth of the Windows desktop. It also supports all

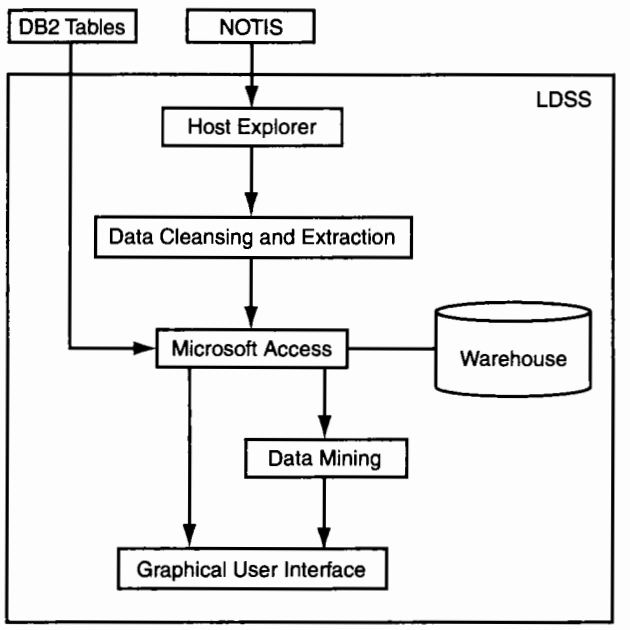

Figure 1. LDSS Architecture and Its Components

TCP/IP based TN3270 and TN3270E gateways.

The Host Explorer program is used as the terminal emulation program in LDSS. It also provides VBA compatible BASIC scripting tools for complete desktop macro development. Users can run these macros directly or attach them to keyboard keys, toolbar buttons, and screen hotspots for additional productivity.

The function of Host Explorer in the LDSS is very simple. It has to "visit" all screens in the NOTIS system corresponding to each NOTIS number present in the BatchBam file, and capture all the data on the screens. In order to do this, we wrote a macro that read the NOTIS number one at a time from the BatchBam file and input the number into the command string of Host Explorer. The macro essentially performed the following functions:

- Read the NOTIS numbers from the BatchBam file.
- Inserted the NOTIS number into the command string of Host Explorer.
- Toggled the Screen Capture option in Host Explorer so that data are scraped from the screen only at necessary times.
- Saved all the scraped data into a flat file.

After the macro has been executed, all the data scraped from the NOTIS screen reside in a flat file. The data present

in this file have to be cleansed in order to make them suitable for insertion into the Library Warehouse. A Visual Basic program is written to perform this function. The details of this program will be given in the next section.

Data Cleansing and Extraction

This component of the LDSS is written in the Visual Basic programming language. Its main function is to cleanse the data that have been scraped from the NOTIS screen. The Visual Basic code saves the cleansed data in a text-delimited format that is recognized by Microsoft Access. This file is then imported into the Library Warehouse maintained by Microsoft Access. The detailed working of the code that performs the cleansing operation is discussed below.

The NOTIS screen that comes up for each NOTIS number has several parts that are critical to the working of the code. They are:

- NOTIS number present in the top-right of the screen (in this case, AKR9234)
- Field numbers that have to be extracted. *Example*: 010::, 035::
- Delimiters. The "|" symbol is used as the delimiter throughout this code. For example, in the 260 field of a bibliographic record, "|a" delimits the place of publication, "|b" the name of the publisher and, "|c" the date of publication.

We shall now go step by step through the cleansing process. Initially we have the flat file containing all the data that have been scraped from the NOTIS screens.

- The entire list of NOTIS numbers from the BatchBam file is read into an array called *Bam_Number*$.
- The file containing the data that have been scraped is read into a single string called *BibRecord*$.
- This string is then parsed using the NOTIS numbers from the *Bam_Number*$ array.
- We now have a string that contains a single NOTIS record. This string is called *Single_Record$*.
- The program runs in a loop till all the records have been read.
- Each string is now broken down into several smaller strings based on the field numbers. Each of these smaller strings contains data pertaining to the corresponding field number.
- A considerable amount of the data present on the NOTIS screen is unnecessary from the point of view of our project. We need only certain fields from the NOTIS screen. But even from these fields we need the data only from certain delimiters. Therefore, we now scan each of these smaller strings for a certain set of delimiters, which was predefined for each individual field. The data present in the other delimiters are discarded.
- The data collected from the various fields and their corresponding delimiters are assigned to corresponding variables. Some variables contain data from more than one delimiter concatenated together. The reason for this can be explained as follows. There are certain fields, which are present in the database only for informational purposes and will not be used as a criteria field in any query. Since these fields will never be queried upon, they do not need to be cleansed as rigorously as the other fields and therefore, we can afford to leave the data of these fields as concatenated strings. *Example*: The Catalog_source field which has data from "|a" and "|c" is of the form "|a DLC |c DLC" while the Lang code field which has data from "|a" and "|h" is of the form "|a eng |h rus." But we split this into two fields: Lang_code_1 containing "eng" and Lang_code_2 containing "rus."
- The data collected from the various fields are saved in a flat file in the text-delimited format. Microsoft Access recognizes this format.

A screen dump of the text-delimited file, which is the end result of the cleansing operation, is shown in figure 2. The flat file, which we now have, can be imported into the Library Warehouse.

Graphical User Interface

In order to ease the tasks of the user (i.e., the decision maker) to create the library warehouse and to query and analyze its contents, a graphical user interface tool has been developed. Through the GUI, the user can enact the following processes or operations through a main menu:

- connection to NOTIS
- screen scraping
- data cleansing and extracting
- importing data
- viewing collected data
- querying
- report generating

The first option opens HostExplorer and provides a connection to NOTIS. It provides a shortcut to closing or minimizing LDSS and opening HostExplorer. The Screen Scraping option activates the data scraping process. The Data Cleansing and Extracting option filters out the unnecessary data fields and saves the cleansed data in a text-delimited format. The Importing Data option imports the data in the text-delimited format into the warehouse. The Viewing Collected Data option allows the user to view the contents of a selected relational table stored in

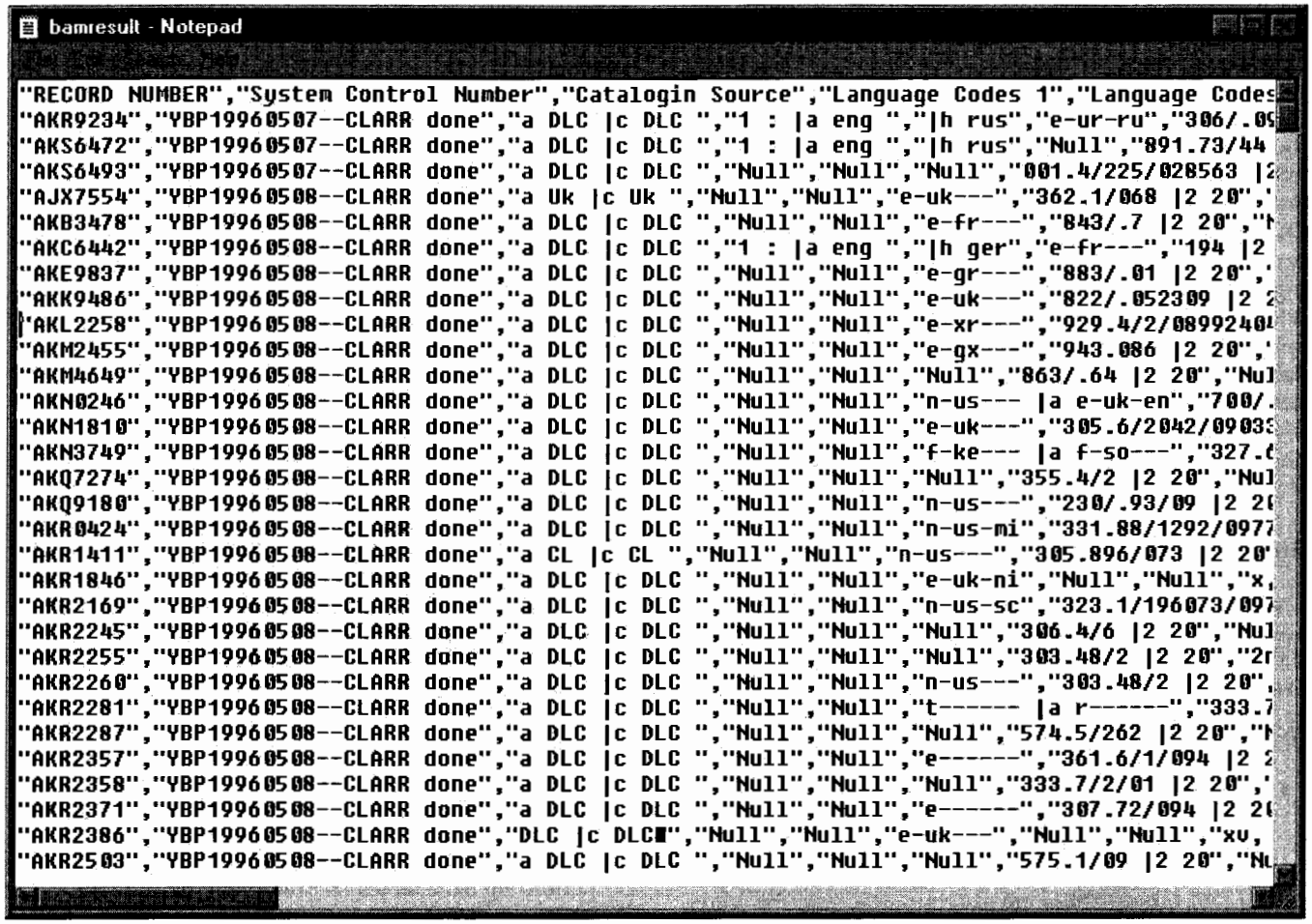

Figure 2. A Text-Delimited File

the warehouse. The Querying option activates LDSS's querying facility that provides wizards to guide the formulations of different types of queries, as discussed later in this article. The last option, Report Generating, is for the user to specify the report to be generated.

Data Mining Tool

A very important component of LDSS is the data mining tool for discovering association rules that specify the interrelationships of data stored in the warehouse. Many data mining tools are now available in the commercial world. For our project, we are investigating the use of a neural-network-based data mining tool developed by Limin Fu of the University of Florida.[7] The tool allows the discovery of association rules based on a set of training data provided to the tool. This part of our research and development work is still in progress. The existing GUI and report generation facilities will be expanded to include the use of this mining tool.

Library Warehouse

FCLA exports the data existing in the DB2 tables into text files. As a first step towards creating the database, these text files are transferred using FTP and form separate relational tables in the Library Warehouse. The data that

are scraped from the bibliographic and item record screens result in the formation of two more tables.

Characteristics

Data in the warehouse are snapshots of the original data files. Only a subset of the data contents in these files are extracted for querying and analysis since not all the data are useful for a particular decision-making situation. Data are filtered as they pass from the operational environment to the data warehouse environment. This filtering process is necessary particularly when a PC system, which has limited secondary storage and main memory space, is used. Once extracted and stored in the warehouse, data are not updateable. They form a read-only database. However, different snapshots of the original files can be imported into the warehouse for querying and analysis. The results of the analyses of different snapshots can then be compared.

Structure

Data warehouses have a distinct structure. There are summarization and detail structures that demarcate a data warehouse. The structure of the Library Data Warehouse is shown in figure 3.

The different components of the Library Data Warehouse as shown in figure 3 are:

- **NOTIS and DB2 Tables.** Bibliographic and circulation data are obtained from NOTIS through the screen scraping process and imported into the warehouse. FCLA maintains acquisitions data in the form of DB2 tables. These are also imported into the warehouse after conversion to a suitable format.
- **Warehouse.** The warehouse consists of several relational tables that are connected by means of relationships. The universal relation approach could have been used to implement the warehouse by using a single table. The argument for using the universal relation approach would be that all the collected data fall under the same domain. But let us examine why this approach would not have been suitable. The different data collected for import into the warehouse were bibliographic data, circulation data, order data, and pay data. Now, if all these data were incorporated into one single table with many attributes, it would not be of any exceptional use since each set of attributes have their own unique meaning when grouped together as bibliographic table, circulation table, and so on. For example, if we group the circulation data and the pay data together in a single table, it would not make sense. However, the pay data and the circulation data are related through the Bib_key. Hence, our use of the conventional approach of having several tables connected by means of relationships is more appropriate.
- **Views.** A view in SQL terminology is a single table that is derived from other tables. These other tables could be base tables or previously defined views. A view does not necessarily exist in physical form; it is considered a virtual table, in contrast to base tables whose tables are actually stored in the database. In the context of the LDSS, views can be implemented by means of the AdHoc Query Wizard. The user can define a query/view using the Wizard and save it for future use. The user can then define a query on this query/view.
- **Summarization.** The process of implementing views falls under the process of summarization. Summarization provides the user with views, which make it easier for users to query on the data of their interests.

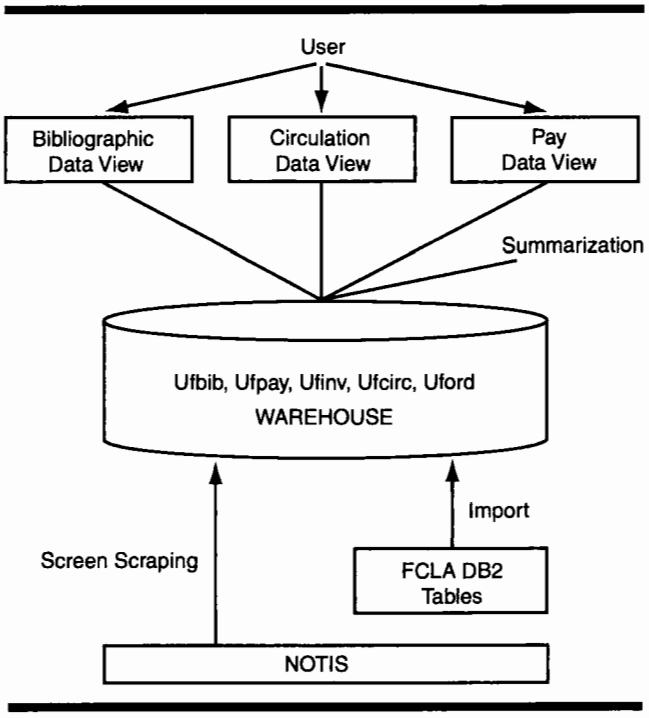

Figure 3. Structure of the Library Data Warehouse

As explained above, the specific warehouse we established consists of five tables. Table names including "_WH" indicates that it contains current detailed data of the warehouse. Current detailed data represents the most recent snapshot of data that has been taken from the NOTIS system. The summarized views are derived from the current detailed data of the warehouse. Since current detailed data of the warehouse are the basic data of the

application, only the current detailed data tables are shown in appendix A.

Decision Support by Querying the Warehouse

The warehouse contains a set of integrated relational tables whose contents are linked by the common primary key, the Bib_key (Biblio_key). The data stored across these tables can be traversed by matching the key values associated with their tuples or records. Decision makers can issue all sorts of SQL-type queries to retrieve useful information from the warehouse. Two general types of queries can be distinguished: predefined queries and ad hoc queries. The former type refers to queries that are frequently used by decision makers for accessing information from different snapshots of data imported into the warehouse. The latter type refers to queries that are exploratory in nature. A decision maker suspects that there is some relationship between different types of data and issues a query to verify the existence of such a relationship. Alternatively, data mining tools can be applied to analyze the data contents of the warehouse and discover rules of their relationships (or associations).

Predefined Queries

Below are some sample queries posted in English. Their corresponding SQL queries can be processed using LDSS.

1. Number and percentage of approval titles circulated and noncirculated.
2. Number and percentage of firm order titles circulated and noncirculated.
3. Amount of financial resources spent on acquiring noncirculated titles.
4. Number and percentage of DLC/DLC cataloging records in circulated and noncirculated titles.
5. Number and percentage of "shared" cataloging records in circulated and noncirculated titles.
6. Numbers of original and "shared" cataloging records of noncirculated titles.
7. Identify the broad subject areas of circulated and noncirculated titles.
8. Identify titles that have been circulated "n" number of times and by subjects.
9. Number of circulated titles without the 505 field.

Each of the above English queries can be realized by a number of SQL queries. We shall use the first two English queries and their corresponding SQL queries to explain how the data warehouse contents and the querying facility of Microsoft Access can be used to support decision making. The results of SQL queries also are given. The first English query can be divided into two parts (see figure 4), each realized by a number of SQL queries as shown below.

Sample Query Outputs

Query 1: Number and percentage of approval titles circulated and noncirculated

Result:	*Total approval titles*	*1172*	
	Circulated	*980*	*83.76%*
	Noncirculated	*192*	*16.24%*

Similar to the above SQL queries, we can translate the second English query into a number of SQL queries and the result is given below:

Query 2: Number and percentage of firm order titles circulated and noncirculated

Result:	*Total firm order titles*	*1829*	
	Circulated	*1302*	*71.18%*
	Noncirculated	*527*	*28.82%*

Report Generation

The results of the two predefined English queries can be presented to users in the form of a report.

Total titles	*3001*	
Approval	*1172*	*39%*
Circulated	*980*	*83.76%*
Noncirculated	*192*	*16.24%*
Firm Order	*1829*	*61%*
Circulated	*1302*	*71.18%*
Noncirculated	*527*	*28.82%*

From the above report, we can ascertain that, though 39 percent of the titles were purchased through the approval plan and 61 percent through firm orders, the approval titles have a higher rate of circulation, 83.76 percent, as compared to firm order titles of 71.18 percent. It is important to note that the result of the above queries is taken from only one snapshot of the circulation data. Analysis from several snapshots is needed in order to compare the results and arrive with reliable information.

We now present a report on the financial resources spent on acquiring and processing noncirculated titles. In order to generate this report, we need the output of queries four and five listed earlier in this article. The corresponding outputs are shown below.

Query 4: Number and percentage of DLC/DLC cataloging records in circulated and noncirculated titles.

Result:	*Total DLC/DLC records*	*2852*	
	Circulated	*2179*	*76.40%*
	Noncirculated	*673*	*23.60%*

Query 5: Number and percentage of "shared" cataloging records in circulated and noncirculated titles.

Result:	*Total "shared" records*	*149*	
	Circulated	*100*	*67.11%*
	Noncirculated	*49*	*32.89%*

In order to come up with the financial resources, we need to consider several factors, which contribute to the amount of financial resources spent. For the sake of simplicity, we consider only the following factors:

1. the cost of cataloging each item with DLC/DLC record
2. the cost of cataloging each item with shared record
3. the average price of noncirculated books
4. the average pages of noncirculated books
5. the value of shelf space per centimeter

Because the value of the above factors differs from institution to institution and might change according to more efficient workflow and better equipment used, users are required to fill in the value for factors 1, 2, and 5. LDSS can compute factors 3 and 4. The financial report, taking into consideration the value of the above factors, could be as shown below.

Processing cost of each DLC Title = $10.00
673 x $10.00 = $ 6,730.00
Processing cost of each Shared Title = $20.00

Approval Titles Circulated

SQL query to retrieve the distinct bibliographic keys of all the approval titles:

```
SELECT DISTINCT  BibScreen.Bib_key
FROM  BibScreen RIGHT JOIN pay1 ON BibScreen.Bib_key = pay1.BIB_NUM WHERE
(((pay1.FUND_KEY) Like "*07*"));
```

SQL query to count the number of approval titles that have been circulated:

```
SELECT     Count (Appr_Title.Bib_key) AS CountOfBib_key
FROM       (BibScreen INNER JOIN Appr_Title ON BibScreen.Bib_key = Appr_Title.Bib_key) INNER JOIN
  ItemScreen ON BibScreen.Bib_key = ItemScreen.Biblio_key
WHERE      (((ItemScreen.CHARGES)>0))
ORDER BY  Count(Appr_Title.Bib_key);
```

SQL query to calculate the percentage:

```
SELECT     Cnt_Appr_Title_Circ.CountOfBib_key,
  Int(([Cnt_Appr_Title_Circ]![CountOfBib_key])*100/Count([BibScreen]![Bib_key])) AS Percent_apprcirc
FROM       BibScreen, Cnt_Appr_Title_Circ
GROUP BY  Cnt_Appr_Title_Circ.CountOfBib_key;
```

Approval Titles Noncirculated

SQL query for counting the number of approval titles that have not been circulated:

```
SELECT DISTINCT  Count(Appr_Title.Bib_key) AS CountOfBib_key
FROM       (Appr_Title INNER JOIN BibScreen ON Appr_Title.Bib_key = BibScreen.Bib_key) INNER JOIN
  ItemScreen ON BibScreen.Bib_key = ItemScreen.Biblio_key
WHERE      (((ItemScreen.CHARGES)=0));
```

SQL query to calculate the percentage:

```
SELECT     Cnt_Appr_Title_Noncirc.CountOfBib_key,  Int(([Cnt_Appr_Title_Noncirc]![CountOfBib_key])*100/
  Count([BibScreen]![Bib_key]))) AS Percent_appr_noncirc
FROM                BibScreen, Cnt_Appr_Title_Noncirc
GROUP BY  Cnt_Appr_Title_Noncirc.CountOfBib_key;
```

Figure 4. Example of an English Query Divided into Two Parts

49 x $20.00 = $ 980.00
Average price paid per noncirculated item = $48.00
722 x $48.00 = $34,656.00
Average size of book = 288 pages = 3 cm
Average cost of 1 cm of shelf space = $0.10
722 x $0.30 = $216.60
Grand Total = $42,582.60

Again it is important to point out that several snapshots of the circulation data have to be taken to track and compare the different analyses before deriving the reliable information.

Ad Hoc Queries

Alternately, if the user wishes to issue a query that has not been predefined, the Ad Hoc Query Wizard can be used. The following example illustrates the use of the Ad Hoc Query Wizard. Assume the sample query is:
How many circulated titles in the English subject area cost more than $35?

We now take you on a walk-through of the AdHoc Query Wizard starting from the first step till the output is obtained.

Figure 4 depicts Step 1 of the Ad Hoc Query Wizard. The sample query mentioned above requires the following fields:

- Biblio_key for a count of all the titles which satisfy the given condition.
- Charges to specify the criteria of "circulated title".
- Fund_Key to specify all titles under the "English" subject area.
- Paid_Amt to specify all titles which cost more than $35.

Step 2 of the Ad Hoc Query Wizard (figure 5) allows the user to specify criteria and thereby narrow the search domain. Step 3 (figure 6) allows the user to specify any mathematical operations or aggregation functions to be performed. Step 4 (figure 7) displays the user-defined query in SQL form and allows the user to save the query for future reuse. The output of the query is shown below in figure 8. The figure shows the number of circulated titles in the English subject area that cost more than $35. Alternatively, the user might wish to obtain a listing of these 33 titles. Figure 9 shows the listing.

Figure 4. Step 1: Ad Hoc Query Wizard

Figure 5. Step 2: Ad Hoc Query Wizard

Conclusion

In this article, we presented the design and development of a library decision support system based on data warehousing and data mining concepts and techniques. We described the functions of the components of LDSS. The screen scraping and data cleansing and extraction

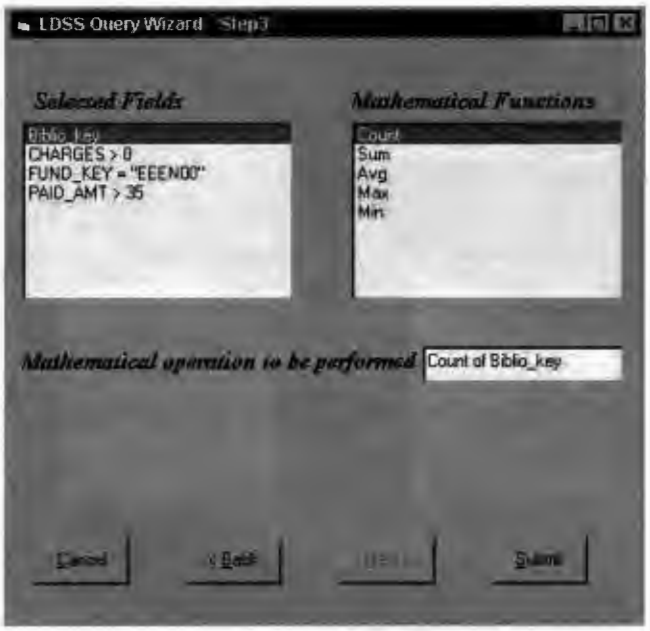

Figure 6. Step Three: Ad Hoc Query Wizard

Figure 7. Step Four: Ad Hoc Query Wizard

Figure 8. Query Output

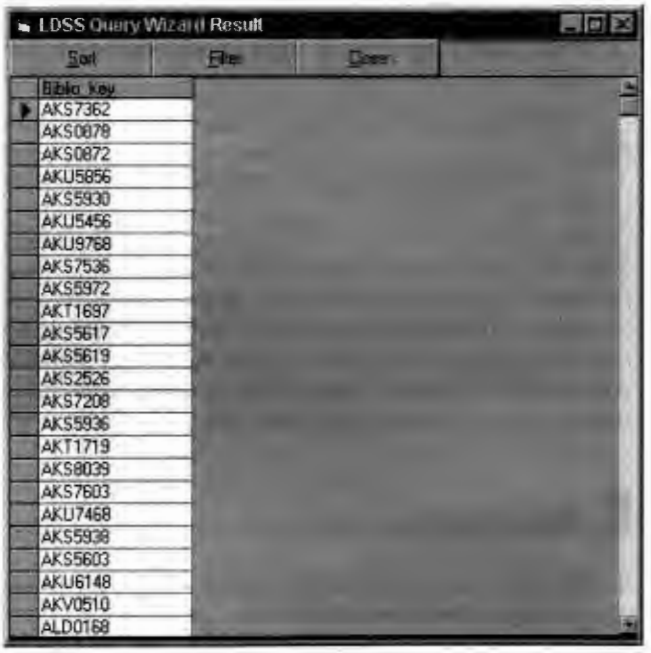

Figure 9. Listing of Query Output

processes were described in detail. The process of importing data stored in LUIS as separate data files into the library data warehouse was also described. The data contents of the warehouse can provide a very rich information source to aid the library management in decision making. Using the implemented system, a decision maker can use the GUI to establish the warehouse, and to activate the querying facility provided by Microsoft Access to explore the warehouse contents. Many types of queries can be formulated and issued against the database. Experimental results indicate that the system is effective and can provide pertinent information for aiding the library management in making decisions. We have fully tested the implemented system using a small sample database. Our on going work includes the expansion of the database size and the inclusion of a data mining component for association rule discovery. Extensions of the existing GUI and report generation facilities to accommodate data mining needs are expected.

Acknowledgments

We would like to thank Professor Stanley Su for his support and advice on the technical aspect of this project. We would also like to thank Donna Alsbury for providing us with the DB2 data, Daniel Cromwell for loading the DB2 files and along with Nancy Williams and Tim Hartigan for their helpful comments and valuable discussions on this project.

References and Notes

1. John Ladley, "Operational Data Stores: Building an Effective Strategy," *Data Warehouse: Practical Advice from the Experts* (Englewood Cliffs, N.J.: Prentice Hall, 1997).

2. Information on Harvard University's ADAPT project. Accessed March 8, 2000, www.adapt.harvard.edu/; Information on the Arizona State University Data Administration and Institutional Analysis warehouse. Accessed March 8, 2000, www.asu.edu/Data_Admin/WH-1.html; Information on the University of Minnesota CLARITY project. Accessed March 8, 2000, www.clarity. umn.edu/; Information on the UC San Diego DARWIN project. Accessed March 8, 2000, www.act.ucsd.edu/dw/darwin.html; Information on University of Wisconsin–Madison InfoAccess. Accessed March 8, 2000, http://wiscinfo.doit.wisc.edu/infoaccess/; Information on the University of Nebraska Data Warehouse-nulook. Accessed March 8, 2000, www.nulook. uneb.edu/.

3. Ramon Barquin and Herbert Edelstein, eds., *Building, Using, and Managing the Data Warehouse* (Englewood Cliffs, N.J.: Prentice Hall, 1997); Ramon Barquin and Herbert Edelstein, eds., *Planning and Designing the Data Warehouse* (Upper Saddle River, N.J.: Prentice Hall, 1996); Joyce Bischoff and Ted Alexander, *Data Warehouse: Practical Advice from the Experts* (Englewood Cliffs, N.J.: Prentice Hall, 1997); Jeff Byard and Donovan Schneider, "The Ins and Outs (and Everything in Between) of Data Warehousing," ACM SIGMOD 1996 Tutorial Notes, May 1996. Accessed March 8, 2000, www.redbrick.com/products/white/pdf/sigmod96.pdf; Surajit Chaudhuri and Umesh Dayal, "An Overview of Data Warehousing and OLAP Technolog," ACM SIGMOD Record 26(1), March 1997. Accessed March 8, 2000, www.acm.org/sigmod/record/issues/9703/chaudhuri.ps; B. Devlin, *Data Warehouse: From Architecture to Implementation* (Reading, Mass.: Addison-Wesley, 1997); U. Fayyad and others, eds., *Advances in Knowledge Discovery and Data Mining* (Cambridge, Mass.: The MIT Pr., 1996); Joachim Hammer, "Data Warehousing Overview, Terminology, and Research Issues." Accessed March 8, 2000, www.cise.ufl.edu/~jhammer/classes/wh-seminar/Overview/index.htm; W. H. Inmon, *Building the Data Warehouse* (New York, N.Y.: John Wiley, 1996); Ralph Kimball, "Dangerous Preconceptions." Accessed March 8, 2000, www.dbmsmag.com/9608d05.html; Ralph Kimball, *The Data Warehouse Toolkit* (New York, N.Y.: John Wiley, 1996); Ralph Kimball, "Mastering Data Extraction," in *DBMS Magazine,* June 1996. (Provides an overview of the process of extracting, cleaning, and loading data.) Accessed March 8, 2000, www.dbmsmag.com/9606d05.html; Alberto Mendelzon, "Bibliography on Data Warehousing and OLAP." Accessed March 8, 2000, www.cs.toronto.edu/~mendel/dwbib.html.

4. Daniel J. Boorstin, "The Age of Negative Discovery," *Cleopatra's Nose: Essays on the Unexpected* (New York: Random House, 1994).

5. Information on the ARROW system. Accessed March 8, 2000, www. fcla.edu/system/intro_arrow.html.

6. Gary Strawn, "BatchBAMing." Accessed March 8, 2000, http://web.uflib.ufl.edu/rs/rsd/batchbam.html.

7. Li-Min Fu, "DOMRUL: Learning the Domain Rules." Accessed March 8, 2000, www.cise.ufl.edu/~fu/domrul.html.

Appendix A Warehouse Data Tables

UFCIRC_WH

Attribute	*Domain*
Bib_key	Text(50)
Status	Text(20)
Enum/Chron	Text(20)
MidSpine	Text(20)
Temp_Locatn	Text(20)
Pieces	Number
Charges	Number
Last_Use	Date/Time
Browses	Number
Value	Text(20)
Invnt_Date	Date/Time
Created	Date/Time

UFINV_WH

Attribute	*Domain*
Inv_Key	Text(20)
Create_Date	Date/Time
Mod_Date	Date/Time
Approv_Stat	Text(20)
Vend_Adr_Code	Text(20)
Vend_Code	Text(20)
Action_Date	Text(20)
Vend_Inv_Date	Date/Time
Approval_Date	Date/Time
Approver_Id	Text(20)
Vend_Inv_Num	Text(20)
Inv_Tot	Number
Calc_Tot_Pymts	Number
Calc_Net_Tot_Pymts	Number
Currency	Text(20)
Discount_Percent	Number
Vouch_Note	Text(20)
Official_Vend	Text(20)
Process_Unit	Text(20)
Internal_Note	Text(20)
DB2_Timestamp	Text(20)

UFORD_WH

Attribute	*Domain*
Id	AutoNumber
Ord_num	Text(20)
Ord_Div	Number
Process_Unit	Text(20)
Bib_num	Text(20)
Order_date	Date/Time
Mod_Date	Date/Time
Vendor_Code	Text(20)
VndAdr_Order	Text(20
VndAdr_Claim	Text(20)
VndAdr_Return	Text(20)
Vend_Title_Num	Text(20)
Ord_Unit	Text(20)
Rcv_Unit	Text(20)
Ord_Scope	Text(20
Pur_Ord_Prod	Text(20)
Action_Int	Number
LibSpec1	Text(20)
LibSpec2	Text(20)
Vend_Note	Text(20)
Ord_Note	Text(20)
Source	Text(20)
Ref	Text(20)
CopyCtl_Num	Number
Medium	Text(20)
Piece_Cnt	Number
Div_Note	Text(20)
Acr_Stat	Text(20)
Rel_Stat	Text(20)
Lst_Date	Date/Time
Action_Date	Text(20)
LibSpec3	Text(20)
LibSpec4	Text(20)
Encumb_Units	Number
Currency	Text(20)
Est_Price	Number
Encumb_outs	Number
Fund_key	Text(20)
Fiscal_Year	Text(20)
Copies	Number
Xpay_Method	Text(20)
Vol_Isu_Date	Text(20)
Title_Author	Text(20)
DB2_Timestamp	Date/Time

UFPAY_WH

Attribute	*Domain*
Inv_key	Text(20)
Ord_num	Text(20)
Ord_div	Number
Process_Unit	Text(20)
Bib_key	Text(20)
Ord_Seq_Num	Number
Inv_Seq_Num	Number
Status	Text(20)
Create_Date	Date/Time
Lst_update	Date/Time
Currency	Text(20)
Paid_amt	Number
USD_amt	Number
Fund_Key	Text(20)
Exp_class	Text(20)
Fiscal_year	Text(20)
Copies	Number
Type_pay	Text(10)
Text	Text(20)
DB2_TimeStamp	Date/Time

UFBIB_WH

Attribute	*Domain*
Bib_key	Text(20)
System_Control_Num	Text(80)
Catalog_Source	Text(20)
Lang_Code_1	Text(20)
Lang_Code_2	Text(20)
Geo_Code	Text(20)
Dewey_Num	Text(20)
Edition	Text(20)
Pagination	Text(20)
Size	Text(20)
Series_440	Text(20)
Series_490	Text(20)
Content	Text(20)
Subject_1	Text(20)
Subject_2	Text(20)
Subject_3	Text(20)
Authors_1	Text(20)
Authors_2	Text(20)
Authors_3	Text(20)
Series	Text(20)