LIBRARY MANAGEMENT SYSTEM: DESIGN AND IMPLEMENTATION

Prepared for: Cynthia Xin Zhang, Instructor
ITCS 3160 - 001

Prepared by: **Group 4**
Darren Adams
Sergey Begun
Andrew Fail
Shawn Haigler
Franklin Lee

April 18, 2007

ABSTRACT

This report describes our group's implementation of a library management system. We used the Entity-Relationship model to design a database that will store and organize the library's data. We have created the database using SQL and populated it with some sample data. The system can keep track of library cards, customers, librarians, library locations, books, videos, and the relationships between them. Using Java Server Pages, servlets, and JDBC, we have created an Internet-based graphical user interface that allows customers and librarians to access the system remotely.

LIST OF ILLUSTRATIONS

Figures

Tables

TABLE OF CONTENTS

INTRODUCTION

This report will provide a detailed account of the processes our group used to design and implement a database that can be used to manage a library system. Each subsection of the report corresponds to an important feature of database design.

REQUIREMENT ANALYSIS

A library database needs to store information pertaining to its users (or customers), its workers, the physical locations of its branches, and the media stored in those locations. We have decided to limit the media to two types: books and videos.

The library must keep track of the status of each media item: its location, status, descriptive attributes, and cost for losses and late returns. Books will be identified by their ISBN, and movies by their title and year. In order to allow multiple copies of the same book or video, each media item will have a unique ID number.

Customers will provide their name, address, phone number, and date of birth when signing up for a library card. They will then be assigned a unique user name and ID number, plus a temporary password that will have to be changed. Checkout operations will require a library card, as will requests to put media on hold. Each library card will have its own fines, but active fines on any of a customer's cards will prevent the customer from using the library's services.

The library will have branches in various physical locations. Branches will be identified by name, and each branch will have an address and a phone number associated with it. Additionally, a library branch will store media and have employees.

Employees will work at a specific branch of the library. They receive a paycheck, but they can also have library cards; therefore, the same information that is collected about customers should be collected about employees.

Functions for customers:

- Log in
- Search for media based on one or more of the following criteria:
  - type (book, video, or both)
  - title

    - author or director
    - year
- Access their own account information:
    - Card number(s)
    - Fines
    - Media currently checked out
    - Media on hold
- Put media on hold
- Pay fines for lost or late items
- Update personal information:
    - Phone numbers
    - Addresses
    - Passwords

Functions for librarians are the same as the functions for customers plus the following:

- Add customers
- Add library cards and assign them to customers
- Check out media
- Manage and transfer media that is currently on hold
- Handle returns
- Modify customers' fines
- Add media to the database
- Remove media from the database
- Receive payments from customers and update the customers' fines
- View all customer information except passwords

ER DESIGN

It is clear that the physical objects from the previous section – the customers, employees, cards, media, and library branches – correspond to entities in the Entity-Relationship model, and the operations to be done on those entities – holds, checkouts, and so on – correspond to relationships. However, a good design will minimize redundancy and attempt to store all the required information in as small a space as possible. After some consideration, we have decided on the following design:

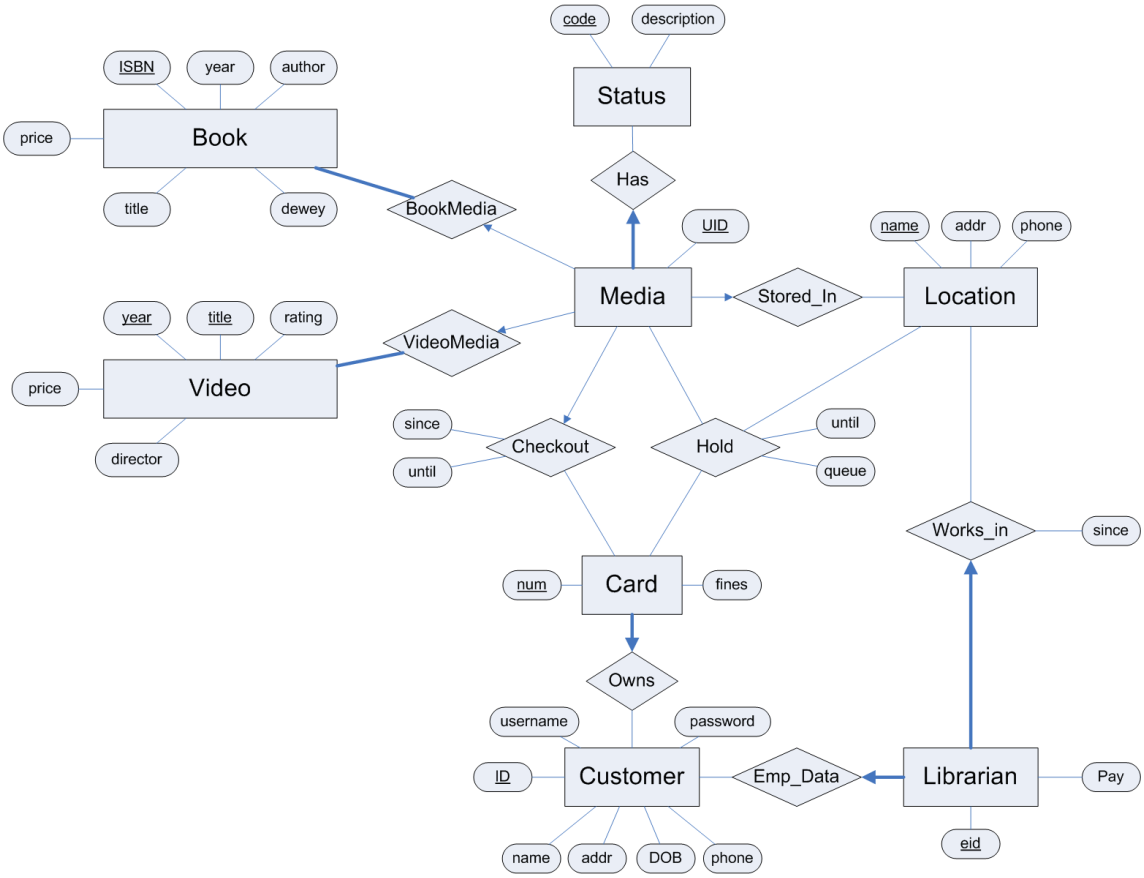


Figure 1: ER Diagram
Refer to Appendix A for a zoomed-in view.

Notice that the information about books and videos has been separated from the Media entity. This allows the database to store multiple copies of the same item without redundancy. The Status entity has also been separated from Media in order to save space. The

Hold relationship stores the entry's place in line (denoted by "queue"); it can be used to create a waiting list of interested customers. The Librarian entity is functionally an extension to Customer, so each Librarian also has a customer associated with it. The librarians will have access to the same features as customers, but they will also perform administrative functions, such as checking media in and out and updating customers' fines.

RELATIONAL DATABASE DESIGN

After coming up with an Entity-Relationship model to describe the library system, we took advantage of the special relationships found in our design, and were able to condense the information to 13 tables. This new design is a database that combines some entities and relationships into common tables.

Table 1: Relational Database Schema

Status	code	desctiption					
Media	media_id	↓ code					
Book	ISBN	title	author	year	dewey	price	
BookMedia	↓ media_id	↓ ISBN					
Customer	ID	name	addr	DOB	phone	username	password
Card	num	fines	↓ ID				
Checkout	↓ media_id	↓ num	since	until			
Location	name	addr	phone				
Hold	↓ media_id	↓ num	↓ name	until	queue		
Stored_In	↓ media_id	↓ name					
Librarian	eid	↓ ID	Pay	↓ name	since		
Video	title	year	director	rating	price		
VideoMedia	↓ media_id	↓ title	↓ year				

Note: the arrows in the diagram represent foreign key constraints.

NORMALIZATION

As stated earlier, the tables in this database are in the Third Normal Form (3 NF.) In order to decompose the relationships into this form, we had to split Status table from the Media table. Each Media object has a status code, and each status code has an associated description. It would be redundant to store both codes and descriptions in the Media object, so we created a dedicated Status table with the code as the primary key.

The other tables were designed with optimization in mind. The Card entity, for instance, was separated from the Customer entity to avoid a functional dependency (since the "num" attribute of the Card entity determines the "fines" attribute.)

PHYSICAL DATABASE DESIGN

The next step was to create the physical database and input some sample data. In order to turn the relational design into a database, we ran the following script in UNCC's Oracle database:

```
CREATE TABLE Status ( code INTEGER, description CHAR(30), PRIMARY KEY (code) );
CREATE TABLE Media( media_id INTEGER, code INTEGER, PRIMARY KEY (media_id),
       FOREIGN KEY (code) REFERENCES Status );
CREATE TABLE Book(ISBNCHAR(14), title CHAR(128), author CHAR(64),
       year INTEGER, dewey INTEGER, price REAL, PRIMARY KEY (ISBN) );
CREATE TABLE BookMedia( media_id INTEGER, ISBN CHAR(14), PRIMARY KEY (media_id),
       FOREIGN KEY (media_id) REFERENCES Media,
       FOREIGN KEY (ISBN) REFERENCES Book);
CREATE TABLE Customer( ID INTEGER, name CHAR(64), addr CHAR(256), DOB CHAR(10),
       phone CHAR(30), username CHAR(16), password CHAR(32), PRIMARY KEY (ID),
       UNIQUE (username) );
CREATE TABLE Card( num INTEGER, fines REAL, ID INTEGER, PRIMARY KEY (num),
       FOREIGN KEY (ID) REFERENCES Customer );
CREATE TABLE Checkout( media_id INTEGER, num INTEGER, since CHAR(10),
       until CHAR(10), PRIMARY KEY (media_id),
       FOREIGN KEY (media_id) REFERENCES Media,
       FOREIGN KEY (num) REFERENCES Card );
CREATE TABLE Location( name CHAR(64), addr CHAR(256), phone CHAR(30),
       PRIMARY KEY (name) );
CREATE TABLE Hold( media_id INTEGER, num INTEGER, name CHAR(64), until CHAR(10),
       queue  INTEGER, PRIMARY KEY (media_id, num),
       FOREIGN KEY (name) REFERENCES Location,
       FOREIGN KEY (num) REFERENCES Card,
       FOREIGN KEY (media_id) REFERENCES Media );
CREATE TABLE Stored_In( media_id INTEGER, name char(64), PRIMARY KEY (media_id),
       FOREIGN KEY (media_id) REFERENCES Media  ON DELETE CASCADE,
       FOREIGN KEY (name) REFERENCES Location      );
CREATE TABLE Librarian( eid INTEGER, ID INTEGER NOT NULL, Pay REAL,
       Loc_name CHAR(64) NOT NULL, PRIMARY KEY (eid),
       FOREIGN KEY (ID) REFERENCES Customer ON DELETE CASCADE,
       FOREIGN KEY (Loc_name) REFERENCES Location(name) );
CREATE TABLE Video( title CHAR(128), year INTEGER, director CHAR(64),
       rating REAL, price REAL, PRIMARY KEY (title, year) );
CREATE TABLE VideoMedia( media_id INTEGER, title CHAR(128), year INTEGER,
       PRIMARY KEY (media_id), FOREIGN KEY (media_id) REFERENCES Media,
       FOREIGN KEY (title, year) REFERENCES Video );
```

The next script populated the database with sample data:

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
       (60201, 'Jason L. Gray', '2087 Timberbrook Lane, Gypsum, CO 81637',
       '09/09/1958', '970-273-9237', 'jlgray', 'password1');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
       (89682, 'Mary L. Prieto', '1465 Marion Drive, Tampa, FL 33602',
       '11/20/1961', '813-487-4873', 'mlprieto', 'password2');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
```

```
        (64937, 'Roger Hurst', '974 Bingamon Branch Rd, Bensenville, IL 60106',
        '08/22/1973', '847-221-4986', 'rhurst', 'password3');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
        (31430, 'Warren V. Woodson', '3022 Lords Way, Parsons, TN 38363',
        '03/07/1945', '731-845-0077', 'wvwoodson', 'password4');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
        (79916, 'Steven Jensen', '93 Sunny Glen Ln, Garfield Heights, OH 44125',
        '12/14/1968', '216-789-6442', 'sjensen', 'password5');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
        (93265, 'David Bain', '4356 Pooh Bear Lane, Travelers Rest, SC 29690',
        '08/10/1947', '864-610-9558', 'dbain', 'password6');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
        (58359, 'Ruth P. Alber', '3842 Willow Oaks Lane, Lafayette, LA 70507',
        '02/18/1976', '337-316-3161', 'rpalber', 'password7');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
        (88564, 'Sally J. Schilling', '1894 Wines Lane, Houston, TX 77002',
        '07/02/1954', '832-366-9035', 'sjschilling', 'password8');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
        (57054, 'John M. Byler', '279 Raver Croft Drive, La Follette, TN 37766',
        '11/27/1954', '423-592-8630', 'jmbyler', 'password9');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
        (49312, 'Kevin Spruell', '1124 Broadcast Drive, Beltsville, VA 20705',
        '03/04/1984', '703-953-1216', 'kspruell', 'password10');

INSERT INTO Card(num, fines, ID) VALUES ( 5767052, 0.0, 60201);
INSERT INTO Card(num, fines, ID) VALUES ( 5532681, 0.0, 60201);
INSERT INTO Card(num, fines, ID) VALUES ( 2197620, 10.0, 89682);
INSERT INTO Card(num, fines, ID) VALUES ( 9780749, 0.0, 64937);
INSERT INTO Card(num, fines, ID) VALUES ( 1521412, 0.0, 31430);
INSERT INTO Card(num, fines, ID) VALUES ( 3920486, 0.0, 79916);
INSERT INTO Card(num, fines, ID) VALUES ( 2323953, 0.0, 93265);
INSERT INTO Card(num, fines, ID) VALUES ( 4387969, 0.0, 58359);
INSERT INTO Card(num, fines, ID) VALUES ( 4444172, 0.0, 88564);
INSERT INTO Card(num, fines, ID) VALUES ( 2645634, 0.0, 57054);
INSERT INTO Card(num, fines, ID) VALUES ( 3688632, 0.0, 49312);

INSERT INTO Location(name, addr, phone) VALUES ('Texas Branch',
        '4832 Deercove Drive, Dallas, TX 75208', '214-948-7102');
INSERT INTO Location(name, addr, phone) VALUES ('Illinois Branch',
        '2888 Oak Avenue, Des Plaines, IL 60016', '847-953-8130');
INSERT INTO Location(name, addr, phone) VALUES ('Louisiana Branch',
        '2063 Washburn Street, Baton Rouge, LA 70802', '225-346-0068');

INSERT INTO Status(code, description) VALUES (1, 'Available');
INSERT INTO Status(code, description) VALUES (2, 'In Transit');
INSERT INTO Status(code, description) VALUES (3, 'Checked Out');
INSERT INTO Status(code, description) VALUES (4, 'On Hold');

INSERT INTO Media( media_id, code) VALUES (8733, 1);
INSERT INTO Media( media_id, code) VALUES (9982, 1);
INSERT INTO Media( media_id, code) VALUES (3725, 1);
INSERT INTO Media( media_id, code) VALUES (2150, 1);
INSERT INTO Media( media_id, code) VALUES (4188, 1);
INSERT INTO Media( media_id, code) VALUES (5271, 2);
INSERT INTO Media( media_id, code) VALUES (2220, 3);
INSERT INTO Media( media_id, code) VALUES (7757, 1);
INSERT INTO Media( media_id, code) VALUES (4589, 1);
INSERT INTO Media( media_id, code) VALUES (5748, 1);
```

```
INSERT INTO Media( media_id, code) VALUES (1734, 1);
INSERT INTO Media( media_id, code) VALUES (5725, 1);
INSERT INTO Media( media_id, code) VALUES (1716, 4);
INSERT INTO Media( media_id, code) VALUES (8388, 1);
INSERT INTO Media( media_id, code) VALUES (8714, 1);

INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
       ('978-0743289412', 'Lisey''s Story', 'Stephen King',
       2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
       ('978-1596912366', 'Restless: A Novel', 'William Boyd',
       2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
       ('978-0312351588', 'Beachglass', 'Wendy Blackburn',
       2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
       ('978-0156031561', 'The Places In Between', 'Rory Stewart',
       2006, 910, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
       ('978-0060583002', 'The Last Season', 'Eric Blehm',
       2006, 902, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
       ('978-0316740401', 'Case Histories: A Novel', 'Kate Atkinson',
       2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
       ('978-0316013949', 'Step on a Crack', 'James Patterson, et al.',
       2007, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
       ('978-0374105235', 'Long Way Gone: Memoirs of a Boy Soldier',
       'Ishmael Beah',       2007, 916, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
       ('978-0385340229', 'Sisters', 'Danielle Steel', 2006, 813, 10.0);

INSERT INTO BookMedia(media_id, ISBN) VALUES (8733, '978-0743289412');
INSERT INTO BookMedia(media_id, ISBN) VALUES (9982, '978-1596912366');
INSERT INTO BookMedia(media_id, ISBN) VALUES (3725, '978-1596912366');
INSERT INTO BookMedia(media_id, ISBN) VALUES (2150, '978-0312351588');
INSERT INTO BookMedia(media_id, ISBN) VALUES (4188, '978-0156031561');
INSERT INTO BookMedia(media_id, ISBN) VALUES (5271, '978-0060583002');
INSERT INTO BookMedia(media_id, ISBN) VALUES (2220, '978-0316740401');
INSERT INTO BookMedia(media_id, ISBN) VALUES (7757, '978-0316013949');
INSERT INTO BookMedia(media_id, ISBN) VALUES (4589, '978-0374105235');
INSERT INTO BookMedia(media_id, ISBN) VALUES (5748, '978-0385340229');

INSERT INTO Checkout(media_id, num, since, until) VALUES
       (2220, 9780749,       '02/15/2007', '03/15/2007');

INSERT INTO Video(title, year, director, rating, price) VALUES
       ('Terminator 2: Judgment Day', 1991, 'James Cameron', 8.3, 20.0);
INSERT INTO Video(title, year, director, rating, price) VALUES
       ('Raiders of the Lost Ark', 1981, 'Steven Spielberg', 8.7, 20.0);
INSERT INTO Video(title, year, director, rating, price) VALUES
       ('Aliens', 1986, 'James Cameron', 8.3, 20.0);
INSERT INTO Video(title, year, director, rating, price) VALUES
       ('Die Hard', 1988, 'John McTiernan', 8.0, 20.0);

INSERT INTO VideoMedia(media_id, title, year) VALUES
       ( 1734, 'Terminator 2: Judgment Day', 1991);
```

```
INSERT INTO VideoMedia(media_id, title, year) VALUES
       ( 5725, 'Raiders of the Lost Ark', 1981);
INSERT INTO VideoMedia(media_id, title, year) VALUES
       ( 1716, 'Aliens', 1986);
INSERT INTO VideoMedia(media_id, title, year) VALUES
       ( 8388, 'Aliens', 1986);
INSERT INTO VideoMedia(media_id, title, year) VALUES
       ( 8714, 'Die Hard', 1988);

INSERT INTO Hold(media_id, num, name, until, queue) VALUES
       (1716, 4444172, 'Texas Branch', '02/20/2008', 1);

INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
       (2591051, 88564, 30000.00, 'Texas Branch');
INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
       (6190164, 64937, 30000.00, 'Illinois Branch');
INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
       (1810386, 58359, 30000.00, 'Louisiana Branch');

INSERT INTO Stored_In(media_id, name) VALUES(8733, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(9982, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(1716, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(1734, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(4589, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(4188, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(5271, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(3725, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(8388, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(5748, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(2150, 'Louisiana Branch');
INSERT INTO Stored_In(media_id, name) VALUES(8714, 'Louisiana Branch');
INSERT INTO Stored_In(media_id, name) VALUES(7757, 'Louisiana Branch');
INSERT INTO Stored_In(media_id, name) VALUES(5725, 'Louisiana Branch');
```

The database was created and filled with 10 Customers, 11 Cards, 3 Locations and 3 Employees, and 15 media items. A Hold relationship and a Checkout relationship were also created.

GUI DESIGN

The first step in designing the GUI was to choose a means of accessing the database. After evaluating various options, we settled on using the JDBC API. The availability of JavaServer Pages on UNCC's servers was an important factor, as it allowed us to develop our application using a three-tier architecture. By using JDBC we could separate the application logic from the DBMS as well as from clients. In addition to simplifying operations on the database, this architecture makes extending the functionality of our system easier. When adding a new feature or improving an existing one, we will not need to change the database; it will only be necessary to modify the Java portion of the code.

Before beginning Java development, however, we needed to define a set of queries that our application would use to communicate with the Oracle database. The queries are presented below. Note that the terms labeled `<user input>` are to be filled in by the application after it receives input from the user and validates it. Note also that complex procedures that require several steps and modify more than one table – such as operations to check out media or put media on hold - will combine several queries into a single transaction, eliminating the possibility of corrupting the database. Finally, some searches (i.e. searches for Book or Video entries) may have a variable number of search parameters, determined at run-time. For example, users will have the option to search for a book by title only, by author only, by year only, by all three fields, or by any combination of two fields. For simplicity's sake, the search queries listed below contain all possible search parameters, but not their possible combinations.

```
/*                                *\
  Functions available to customers
\*                                */

/* User login and authentication */
SELECT C.ID, C.name, C.addr, C.DOB, C.phone, C.username,
       nvl((SELECT 'Librarian'
            FROM   Librarian L
           WHERE  L.ID = C.ID), 'Customer') AS role
```

```
FROM   Customer C
WHERE  C.username = <user input> AND C.password = <user input>;

/* Book search for customers */
SELECT B.ISBN, B.title, B.author, B.year,
       (SELECT COUNT(*)
        FROM   BookMedia BM
        WHERE  BM.ISBN = B.ISBN AND BM.code = 1) AS num_available
FROM   Book B
WHERE  B.title LIKE '%<user input>%' AND B.author LIKE '%<user input>%' AND
       B.year <= <user input> AND B.year >= <user input>;

/* Find all copies of a book (used for placing holds or viewing detailed
   information). */
SELECT BM.media_id, S.description,
       nvl((SELECT SI.name
            FROM   Stored_In SI
           WHERE  SI.media_id = BM.media_id), 'none') AS name
FROM   BookMedia BM, Media M, Status S
WHERE  BM.ISBN = <user input> AND M.media_id = BM.media_id AND S.code = M.code;

/* Video search for customers */
SELECT V.title, V.year, V.director, V.rating
       (SELECT COUNT(*)
        FROM   VideoMedia VM
        WHERE  VM.ID = V.ID AND VM.code = 1) AS num_available
FROM   Video V
WHERE  V.title LIKE '%<user input>%' AND V.year <= <user input> AND V.year <= <user input>
       AND V.director LIKE '%<user input>%' AND V.rating >= <user input>;

/* Find all copies of a video (used for placing holds or viewing detailed
   information). */
SELECT VM.media_id, S.description,
       nvl((SELECT SI.name
            FROM   Stored_In SI
           WHERE  SI.media_id = VM.media_id), 'none') AS name
FROM   VideoMedia VM, Media M, Status S
WHERE  VM.title = <user input> AND VM.year = <user input> AND
       M.media_id = VM.media_id AND S.code = M.code;

/* Find the status of a given media item */
SELECT S.description
FROM   Status S, Media M
WHERE  S.code = M.code AND M.media_id = <user input>;

/* Create a new Hold */
INSERT INTO Hold(media_id, num, name, until, queue) VALUES
(<user input>, <user input>, <user input>, <user input>,
 nvl((SELECT MAX(H.queue)
      FROM   Hold H
      WHERE  H.media_id = <user input>), 0) + 1 );

/* Cancel Hold, Step 1: Remove the entry from hold */
DELETE FROM Hold
WHERE  media_id = <user input> AND num = <user input>

/* Cancel Hold, Step 2: Update queue for this item */
UPDATE Hold
```

```
SET    queue = queue-1
WHERE  media_id = <user input> AND queue > <user input>;

/* Functions needed to view information about a customer */

/* View the customer's card(s) */
SELECT CR.num, CR.fines
FROM   Card CR
WHERE  CR.ID = <user input>;

/* View media checked out on a given card */
SELECT B.title, B.author, B.year, BM.media_id, CO.since, CO.until
FROM   Checkout CO, BookMedia BM, Book B
WHERE  CO.num = <user input> AND CO.media_id = BM.media_id AND B.ISBN = BM.ISBN
UNION
SELECT V.title, V.director, V.year, VM.media_id, CO.since, CO.until
FROM   Checkout CO, VideoMedia VM, Book B
WHERE  CO.num = <user input> AND CO.media_id = VM.media_id AND
       VM.title = V.title AND VM.year = V.year;

/* View media currently on hold for a given card */
SELECT B.title, B.author, B.year, BM.media_id, H.until, H.queue, SI.name
FROM   Hold H, BookMedia BM, Book B, Stored_In SI
WHERE  H.num = <user input> AND H.media_id = BM.media_id AND B.ISBN = BM.ISBN
       AND SI.media_id = H.media_id
UNION
SELECT V.title, V.director, V.year, VM.media_id, H.until, H.queue, SI.name
FROM   Hold H, VideoMedia VM, Book B, Stored_In SI
WHERE  H.num = <user input> AND H.media_id = VM.media_id AND
       VM.title = V.title AND VM.year = V.year AND SI.media_id = H.media_id;

/* View the total amount of fines the customer has to pay */
SELECT SUM(CR.fines)
FROM   Card CR
WHERE  CR.ID = <user input>;

/*                                  *\
  Functions reserved for librarians
\*                                  */

/* Add new customer */
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(<user input>, <user input>, <user input>, <user input>, <user input>,
 <user input>, <user input>, );

/* Find a customer */
SELECT C.ID, C.name, C.addr, C.DOB, C.phone, C.username,
       nvl((SELECT 'Librarian'
            FROM   Librarian L
           WHERE  L.ID = C.ID), 'Customer') AS role
FROM   Customer C
WHERE  C.username = <user input> AND C.name LIKE '%<user input>%';

/* Add new card and assign it to a customer */
INSERT INTO Card(num, fines, ID) VALUES ( <user input>, 0, <user input>);

/* Create an entry in Checkout */
INSERT INTO Checkout(media_id, num, since, until) VALUES
```

```
       (<user input>, <user input>, <user input>, <user input>);

/* Remove the entry for Stored_In */
DELETE FROM Stored_In
WHERE  media_id = <user input>;

/* Change the status code of the media */
UPDATE Media
SET    code = <user input>
WHERE  media_id = <user input>;

/* Remove the entry from Checkout */
DELETE FROM Checkout
WHERE  media_id = <user input>;

/* Create the entry in Stored_In */
INSERT INTO Stored_In(media_id, name) VALUES (<user input>, <user input>);

/* Find the next Hold entry for a given media */
SELECT H.num, H.name, H.until
FROM   Hold H
WHERE  H.queue = 1 AND H.media_id = <user input>;

/* Change the Stored_In entry to the target library branch */
UPDATE Stored_In
SET    name = <user input>
WHERE  media_id = <user input>;

/* Find the customer that should be notified about book arrival */
SELECT C.name, C.phone, CR.num
FROM   Customer C, Card CR, Hold H
WHERE  H.queue = 1 AND H.name = <user input> AND H.media_id = <user input> AND
       CR.num = H.num AND C.ID = CR.ID;

/* Add a new entry into the Book table */
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
(<user input>, <user input>, <user input>, <user input>, <user input>,
 <user input>);

/* Add a new entry into the Video table */
INSERT INTO Video(title, year, director, rating, price) VALUES
(<user input>, <user input>, <user input>, <user input>, <user input>);

/* Add a new Media object */
INSERT INTO Media( media_id, code) VALUES (<user input>, 1);

/* Add a new BookMedia object */
INSERT INTO BookMedia(media_id, ISBN) VALUES (<user input>, <user input>);

/* Add a new VideoMedia object */
INSERT INTO VideoMedia(media_id, title, year) VALUES
(<user input>, <user input>, <user input>);

/* Remove an entry from the BookMedia table */
DELETE FROM BookMedia
WHERE  media_id = <user input>;

/* Remove an entry from the VideoMedia table */
```

```
DELETE FROM VideoMedia
WHERE  media_id = <user input>;

/* Remove an entry from the Media table */
DELETE FROM Media
WHERE  media_id = <user input>;

/* Remove an entry from the Book table */
DELETE FROM Book
WHERE  ISBN = <user input>;

/* Remove an entry from the Video table */
DELETE FROM Video
WHERE  title = <user input> AND year = <user input>;

/* Update the customer's fines */
UPDATE Card
SET    fines = <user input>
WHERE  num = <user input>
```

After learning about the optimizers used by commercial database management systems, we reviewed the above queries for efficiency. They turned out to be simple and efficient enough not to require further optimization.

With the query design and optimization finished, we turned our attention to the GUI itself. Our design is laid out in a fairly traditional manner – a navigation bar on the top, a navigation box on the left side of the screen, and a content box on the right. Upon first entering the website, the user is presented with a log-in prompt. When the user attempts to log in, the system compares entered credentials with those stored in the database and presents the user with a menu.

The menus change based on the user's role: customers have basic functions to search for media, view their account options, fines, and so on, while librarians get an extended menu with administration-related links. It should be noted that there is no special log in for librarians; instead, the system accesses the database to find out whether the user is a librarian and builds the menu dynamically.

We used a combination of tables, cascading style sheets, and Java Server Pages to display the site's graphics. The application performs queries on the database using servlets and JDBC. Refer to Appendix B

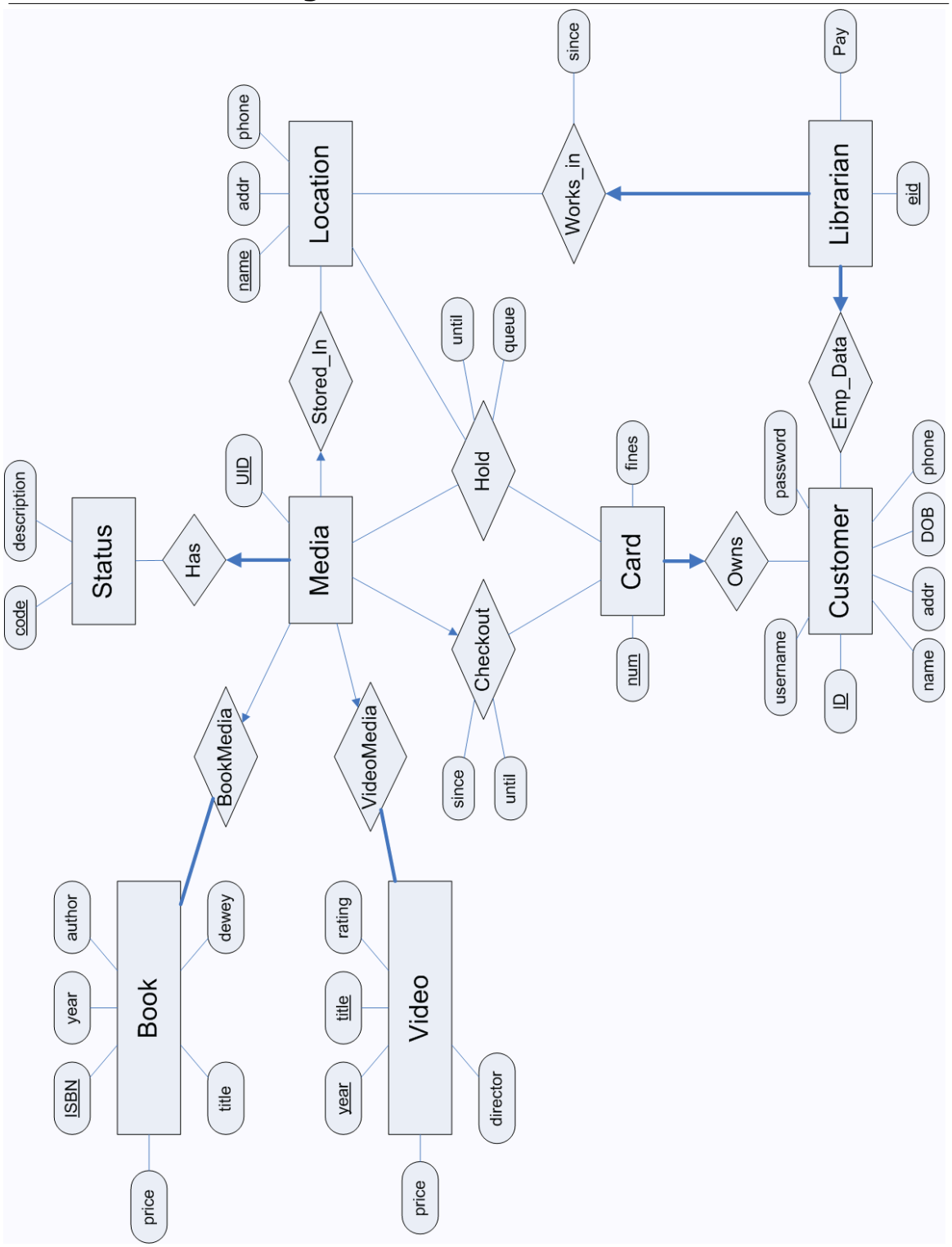
for a hard copy of the website design, or go to http://coit-servlet01.uncc.edu:8080/svbegun/library/layout.jsp to see the site in action.

CONCLUSIONS AND FUTURE WORK

During this semester, our group has designed and implemented a database for managing a library system with multiple locations, the ability to store videos as well as books, and separate functions for customers and librarians. Furthermore, we were able to keep the system's logic abstracted from both the end users and the DBMS by using JDBC to create a third-tier architecture. Finally, we used CSS and JSP to design an Internet-based GUI for remotely accessing the database.

The database design supports more operations than are currently implemented by the GUI. For example, the "hold" relationship can store a waiting list of customers who want to check out a particular media item; there is also the possibility of a customer having multiple library cards. If we were to continue with this project, we would implement the above features, and also improve the account management and search features.

APPENDIX A: ER Diagram



APPENDIX B: Website Layout