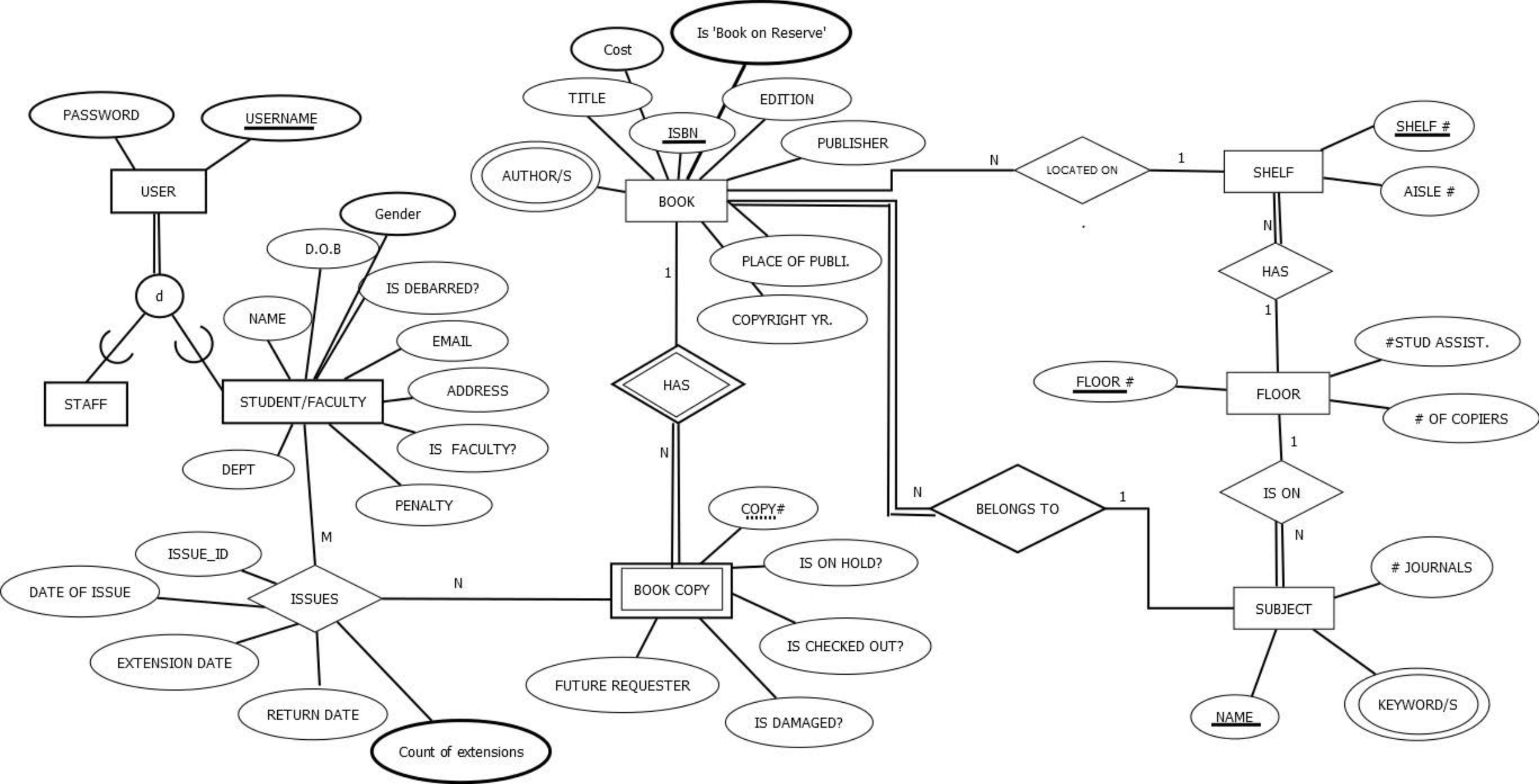
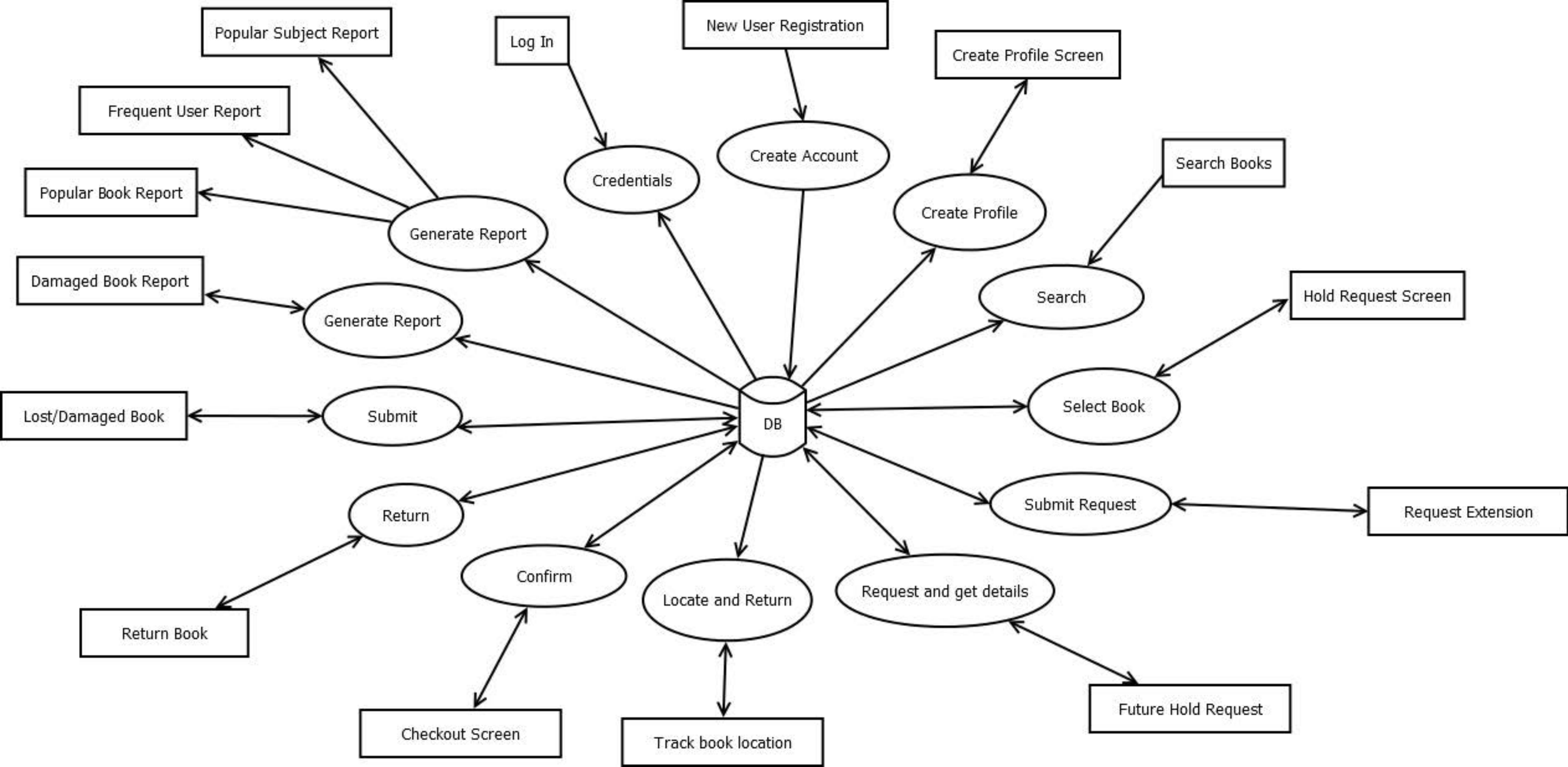
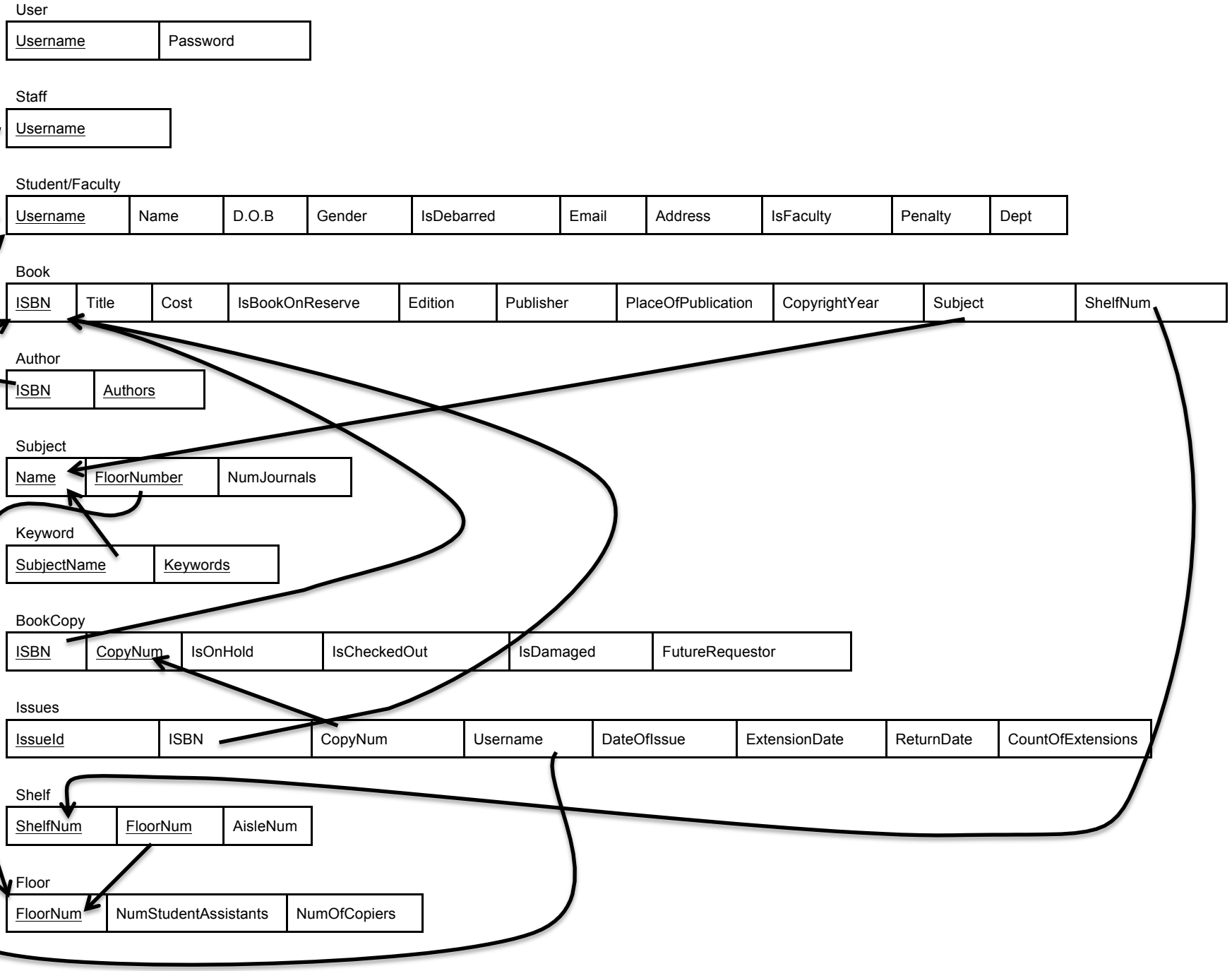
CS 4400 Project
Library Management System
Group Number- 41
Phase 2
3/12/2015

Name	Section	T-square username	Email address
Michael Beyrouti	A	mbeyrouti3	mbeyrouti3@gatech.edu
Avery Dingler	A	adingler6	adingler@gatech.edu
Savannah O'Connor	A	soconnor31	soconnor31@gatech.edu
Vishnu Premsankar	B	vpremsankar3	p.vishnu@gatech.edu


PASSWORD
USERNAME
USER
d
STAFF
Gender
D.O.B
IS DEBARRED?
NAME
EMAIL
ADDRESS
STUDENT/FACULTY
IS FACULTY?
DEPT
PENALTY
Cost
Is 'Book on Reserve'
TITLE
EDITION
ISBN
PUBLISHER
AUTHOR/S
BOOK
PLACE OF PUBLI.
COPYRIGHT YR.
1
HAS
N
N
LOCATED ON
1
SHELF
SHELF #
AISLE #
N
HAS
1
#STUD ASSIST.
FLOOR #
FLOOR
OF COPIERS
1
IS ON
N
N
BELONGS TO
1
M
ISSUE_ID
DATE OF ISSUE
ISSUES
N
EXTENSION DATE
RETURN DATE
Count of extensions
COPY#
IS ON HOLD?
BOOK COPY
IS CHECKED OUT?
FUTURE REQUESTER
IS DAMAGED?
JOURNALS
SUBJECT
NAME
KEYWORD/S


Popular Subject Report
Log In
New User Registration
Create Profile Screen
Frequent User Report
Create Account
Credentials
Search Books
Popular Book Report
Create Profile
Generate Report
Damaged Book Report
Search
Hold Request Screen
Generate Report
Lost/Damaged Book
Submit
DB
Select Book
Return
Submit Request
Request Extension
Confirm
Locate and Return
Request and get details
Return Book
Future Hold Request
Checkout Screen
Track book location

Relational Schema Diagram



Create Table Statements - Group 41

```
CREATE TABLE User(
        Username varchar(50) NOT NULL,
        Password varchar(50) NOT NULL,
        PRIMARY KEY(Username));

CREATE TABLE Staff(
        Username varchar(50) NOT NULL,
        PRIMARY KEY (Username),
        FOREIGN KEY (Username)
                REFERENCES User (Username));

CREATE TABLE StudentFaculty(
        Username VARCHAR(50) NOT NULL,
        Name VARCHAR(50) NOT NULL,
        DOB DATE,
        Gender VARCHAR(50),
        IsDebarred BOOLEAN NOT NULL,
        Email VARCHAR(50),
        Address VARCHAR(100),
        IsFaculty BOOLEAN NOT NULL,
        Penalty DECIMAL(10,2) NOT NULL DEFAULT 0,
        Dept VARCHAR(50) NOT NULL,
        PRIMARY KEY (Username),
        FOREIGN KEY (Username)
                REFERENCES User (Username));
```

SBN, Title, Author/s, Publisher, Place of Publication, Edition, whether it is a book on reserve or not (A reserved book is only available to be read in the library not for checkout), Copyright Year, and cost of the book.

```
CREATE TABLE Book(
        ISBN VARCHAR(20) NOT NULL,
        Title VARCHAR(100) NOT NULL,
        Cost DECIMAL(10,2) NOT NULL,
        IsBookOnReserve BOOLEAN NOT NULL,
        Edition INT(3) NOT NULL DEFAULT 1,
        Publisher VARCHAR(50) NOT NULL,
        PlaceOfPublication VARCHAR(50) NOT NULL,
        CopyrightYear INT(4) NOT NULL,
        Subject VARCHAR(50),
        ShelfNum INT(12),



        PRIMARY KEY (ISBN),
        FOREIGN KEY(Subject)
                RERERENCES Subject (Name),
        FOREIGN KEY(ShelfNum)
                RERERENCES Shelf (ShelfNum));

CREATE TABLE Author(
        ISBN VARCHAR(20) NOT NULL,
        Authors VARCHAR(50) NOT NULL,
        PRIMARY KEY (ISBN, AUTHORS),
        FOREIGN KEY(ISBN)
                REFERENCES Book (ISBN));

CREATE TABLE Subject(
        Name VARCHAR(50) NOT NULL,
        FloorNum INT(5) NOT NULL,
        NumJournals INT(10) NOT NULL,
        PRIMARY KEY(Name, FloorNum),
        FOREIGN KEY(FloorNum)
                REFERENCES FloorNum (Floor));

CREATE TABLE Keyword(
        SubjectName VARCHAR(50) NOT NULL,
        Keywords VARCHAR(50) NOT NULL,
        PRIMARY KEY(SubjectName, Keywords),
        FOREIGN KEY(SubjectName)
                REFERENCES Name (Subject));
```

```
CREATE TABLE BookCopy(
        ISBN VARCHAR(20) NOT NULL,
        CopyNum INT(3) NOT NULL,
        IsOnHold BOOLEAN NOT NULL,
        IsCheckedOut BOOLEAN NOT NULL,
        IsDamaged BOOLEAN NOT NULL,
        FutureRequestor VARCHAR(50),
        PRIMARY KEY(ISBN,CopyNum),
        FOREIGN KEY(ISBN)
                REFERENCES Book (ISBN));
```

```
CREATE TABLE Issues(
        UserName VARCHAR(50) NOT NULL,
        ISBN VARCHAR(20) NOT NULL,
        CopyNum INT(3) NOT NULL,
        IssueID VARCHAR(20) NOT NULL,
        DateOfIssue DATE NOT NULL,
        ExtensionDate DATE,
        ReturnDate DATE,
        CountOfExtension INT(1),
        PRIMARY KEY(IssueID),
        FOREIGN KEY(UserName)
                REFERENCES STUDENT/FACULTY (UserName)
        FOREIGN KEY(ISBN)
                REFERENCES Book(ISBN)
        FOREIGN KEY(CopyNum)
                REFERENCES BookCopy (CopyNum))

CREATE TABLE Shelf(
        ShelfNum INT(10) NOT NULL,
        FloorNum INT(10) NOT NULL,
        AisleNum INT(10) NOT NULL,
        PRIMARY KEY(ShelfNum),
        FOREIGN KEY(FloorNum)
                REFERENCES Floor (FloorNum))
```

```
CREATE TABLE Floor(
        FloorNum INT(10) NOT NULL,
        NumStudentAssistants INT(10) NOT NULL,
        NumOfCopies INT(10) NOT NULL,
        PRIMARY KEY(FloorNum))
```

SQL Statements & Logic – Group 41

Log in:

1. grab user table
2. project user ID and Pass
3. check if ID and Pass exist in the same tuple of studentFaculty
4. return boolean

SELECT Username,Password FROM User
WHERE Username = $username AND Password = $password;
$ = username and password are the input username and password from the user of the system

New User Registration:

1. grab user table, project user ID
2. check if username exists already
3. make sure “password” and “confirm password” are the same **
4. update user, student/faculty relation

SELECT Username FROM StudentFaculty WHERE Username = $username;
If returns empty table perform next SQL Statement
INSERT INTO User (Username, Password) VALUES ($username, $password);
$ = username and password are the inputted username and password from the user of the system

Create Profile Screen:

1. grab fields from screen
2. insert new tuple in Student/Faculty

INSERT INTO StudentFaculty (Username, Name, Dob, Gender, isDebarred, Email, Address, IsFaculty, Penalty, Dept) VALUES ($username, $name, $dob, $gender, $isdebarred, $email, $address, $isfaculty, $penalty, $dept);
$ = username, name, dob, gender, isdebarred, email, address, isfaculty, penalty, dept are all inputted by the user of the system

Search Books:

1. grab book relation
2. select tuple w/ ISBN or Title or Author
3. is a reserved book?
4. grab book copy relation
5. project checked out AND on hold
6. return availability of copies (isCheckedOut, calculated value - next available date)
7. return boolean if available (if true, go to a new screen with checkout options)
8. if false, check the date of issue and extension date for the issue with the ISBN and username
9. return when book is available next based off date of issue or extension date (make sure we don't go over the max of 28 or 56 days)

show available (books you can check out now or put a hold on aka some book already on hold), onReserve

Need to check IsDamaged from book copy!!!! Phase 2 only, all fields required.

SELECT Book.ISBN, Book.Title, Book.Edition, COUNT(BookCopy.ISBN)
FROM Book, BookCopy
WHERE Book.ISBN = BookCopy.ISBN AND Book.ISBN=$isbn AND Book.Title = $title AND Book.Author = $author AND Book.Publisher = $publisher AND Book.Edition = $edition)
GROUP BY ISBN;

$ = author, publisher, edition, are all inputted by the user of the system

Future Hold Request:

1. grab user relation, is user debarred?
2. grab book relation, is book on reserve?
3. with isbn input, look up the copy with the earliest expected available date.
    a. If book is on hold, expected available date is null and cannot be future requested.
    b. If book is checked out, earliest available date will be shown to user.
4. update availability of requested book (updates username as future requestor for copy number)

CREATE VIEW bookToUpdate(ISBN, CopyNum)
AS SELECT BookCopy.ISBN, BookCopy.CopyNum,
FROM BookCopy, Issues
WHERE BookCopy.IsBookOnReserve = 0 AND BookCopy.IsOnHold = 0 AND BookCopy.ISBN = $isbn AND BookCopy.IsCheckedOut = 1 AND BookCopy.ISBN = Issues.ISBN and Issues.ReturnDate = 0 AND Issues.IssueId LIKE "C%"
ORDER BY Issues.ExtensionDate ASC
LIMIT 1;

If a book is available the above query returns the book. The the next sql statement runs. If the table 'book' is empty the statement updates nothing

UPDATE BookCopy SET FutureRequestor = $Username WHERE ISBN = bookToUpdate.ISBN AND CopyNum = bookToUpdate.CopyNum;

Track book location:

1. grab book relation, tuple with ISBN
2. return values shelf number, aisle number, floor, and subject

SELECT Book.ShelfNum, Book.Subject, Shelf.FloorNum, Shelf.AisleNum
FROM Book, Shelf WHERE Book.ISBN = $ISBN AND Book.ShelfNum = Shelf.ShelfNum;

$ = ISBN will be typed in by user

Checkout Screen:

1. grab book copy relation tuple w/ ISBN, CopyNum, and username
2. If user comes to pick up book after the 3 day grace period, throw error message that his hold has been dropped.

a. if book is still physically in library, he can relocate it and check it out
b. if not, he can future request it at earliest available date

3. Check out date (14 days from current date) and estimated return date will be inserted in database. isCheckedout flag updated
4. if there is a hold request placed by the user for that copy number, drop request

If there was a hold request placed by the user on that copy:

SELECT DateOfIssue
FROM Issues
WHERE ISBN=$isbn AND DateOfIssue < DATEADD(day, -17,CURDATE());
Note: If the current date is after the DateOfIssue + 17 returns a 0 tuple

If successfully checked out:
UPDATE BookCopy SET IsCheckedOut = 1 WHERE ISBN=$isbn AND CopyNum = $CopyNumber;

And if it was also on hold run the next two sql statments:
UPDATE BookCopy SET IsOnHold = 0 WHERE ISBN=$isbn AND CopyNum = $CopyNumber;

DELETE FROM Issues WHERE Username = $Username AND ISBN = $isbn AND CopyNum = $CopyNumber AND IssueId LIKE "H%"

insert new checked out book into the Issue table:
INSERT INTO Issues ($Username, $ISBN, $CopyNumber, $IssueId, CURDATE(), CURDATE(), NULL, 0)

We have the constraint where issues for holds begin with 'H'
$ =isbn is the value that was automatically populated when the book was scanned in by the staff

Return Book:

1. check if the book is damaged or not
    a. if damaged, apply penalty charges to user (aka go to next screen)
    b. if not, isCheckedout flag goes to 0
2. if returned after due date- if so, $0.50/day will automatically be charged to user account up to 50% price of book
   IF penalty puts user's cumulative penalty over $100, update user to debarred
3. grab book relation w/ tuple ISBN, copyNum, user
4. update availability

If book is damaged, go to next screen.

If not damaged, run:

UPDATE BookCopy SET IsCheckedOut = 0 WHERE ISBN = $ISBN AND CopyNum = $CopyNumber;

If returned after due date:

UPDATE StudentFaculty SET Penalty = $amount WHERE Username = $username;
* amount is calculated by $0.50 per each day after due date up to 50% price of book

→ If penalty puts user's cumulative penalty over $100:
UPDATE StudentFaculty SET IsDebarred = 1 WHERE Username = $username;

UPDATE BookCopy SET IsCheckedOut = 0 WHERE ISBN = $ISBN AND CopyNum = $CopyNumber;

UPDATE Issues SET ReturnDate = CURDATE() WHERE Username = $username AND ISBN = $ISBN and CopyNum = $CopyNum AND ReturnDate = 0;

Lost/Damaged Book:

1. grab Issues relation w/ ISBN, copyNum, and current time
2. return user tuple with most recent date (specifically the student/faculty name)
3. manually update student/faculty user with new penalty charge from screen
   a. if damaged, 50% of price of book will be charged to user (we can ignore, manual input)
   b. if lost, the user will be charged the price of book (we can ignore, manual input)
   c. **IF penalty puts user's cumulative penalty over $100, update user to debarred**
4. once book is marked damaged, book cannot be borrowed by any user

If book is damaged, run following sql statement:

CREATE VIEW damager(Username, DateOfIssue, ISBN, CopyNum)
AS SELECT Username, max(DateOfIssue), ISBN, CopyNum
FROM Issues
WHERE DateOfIssue < NOW() AND ReturnDate = NULL AND ISBN = $ISBN AND CopyNum = $BookCopy;

UPDATE StudentFaculty Set Penalty = $amount WHERE Username= damager.Username

→ If penalty puts user's cumulative penalty over $100:
UPDATE StudentFaculty SET IsDebarred = 1 WHERE Username = damager.Username

UPDATE BookCopy SET IsDamaged = 1 WHERE ISBN = damager.ISBN and CopyNum = damager.CopyNum;

Damaged Book Report:

SELECT Book.Subject, COUNT(ISBN.BookCopy) AS damagedCount,
MONTH(Issues.ReturnDate) AS Month
FROM Book, BookCopy, Issues
WHERE Book.ISBN = BookCopy.ISBN AND Issues.ISBN = Book.ISBN AND Issues.IsDamaged = 1 AND MONTH(Issues.ReturnDate) = $month
GROUP BY Month, Book.Subject

Popular Book Report:

1. grab Issues table
2. select all tuples in Jan AND Feb where checkout date is not null, group by month, count ISBN, order by DESC
3. return 3 max(count ISBN

This grabs the books for Jan:

SELECT MONTH(Issues.IssueDate) AS Month, COUNT(Issues.ISBN) AS numCheckouts, book.Title FROM Issues, Book WHERE Issues.IssueDate LIKE "C%" AND MONTH(Issue.IssueDate) = "January" AND Issue.ISBN = Book.ISBN
GROUP BY Month
ORDER BY NumCheckouts
LIMIT 3

This grabs the books for Feb:

SELECT MONTH(Issues.IssueDate) AS Month, COUNT(Issues.ISBN) AS numCheckouts, book.Title FROM Issues, Book
WHERE Issues.IssueDate LIKE "C%" AND MONTH(Issue.IssueDate) = "February" AND Issue.ISBN = Book.ISBN
GROUP BY Month
ORDER BY NumCheckouts
LIMIT 3

NOTE: I did IssueDate like "C%" because we are making a constraint where any Issue that corresponds to a checked out books begins with the letter 'C'

Frequent User Report:

1. grab issues table
2. select all tuples for Jan. and Feb where checkout date is not null and count(user) > 10, grouped by month
3. return 5max(count User)

This grabs the user for Jan:

SELECT MONTH(Issue.IssueDate) AS Month, StudentFaculty.Name,
COUNT(Issue.Username) AS numCheckouts
FROM Issues, StudentFaculty
WHERE Issues.IssueDate LIKE "C%" AND MONTH(Issue.IssueDate) = "January" AND Issue.Username = StudentFaculty.Username
GROUP BY Month
HAVING numCheckouts > 10
ORDER BY numCheckouts DESC
LIMIT 5

This grabs the top users for Feb:

SELECT MONTH(Issue.IssueDate) AS Month, StudentFaculty.Name,
COUNT(Issue.Username) AS numCheckouts
FROM Issues, StudentFaculty
WHERE Issues.IssueDate LIKE "C%" AND MONTH(Issue.IssueDate) = "February" AND Issue.Username = StudentFaculty.Username
GROUP BY Month
HAVING numCheckouts > 10
ORDER BY numCheckouts DESC
LIMIT 5

Popular Subject Report:

1. grab issues table
2. select all tuples for last two months where checkout date is not null, group by subject
3. return 3max(count Subject)

SELECT MONTH(Issues.ReturnDate), Book.Subject, COUNT(Issues.IssuesID) AS numCheckouts
FROM Issues, Book
WHERE (MONTH(Issues.ReturnDate) = "January" OR MONTH(Issues.ReturnDate) = "February") AND Issues.ISBN = Book.ISBN
GROUP BY Month, Book.Subject
ORDER BY numCheckouts