CS 4400
Library Management System
Phase II

Group 36

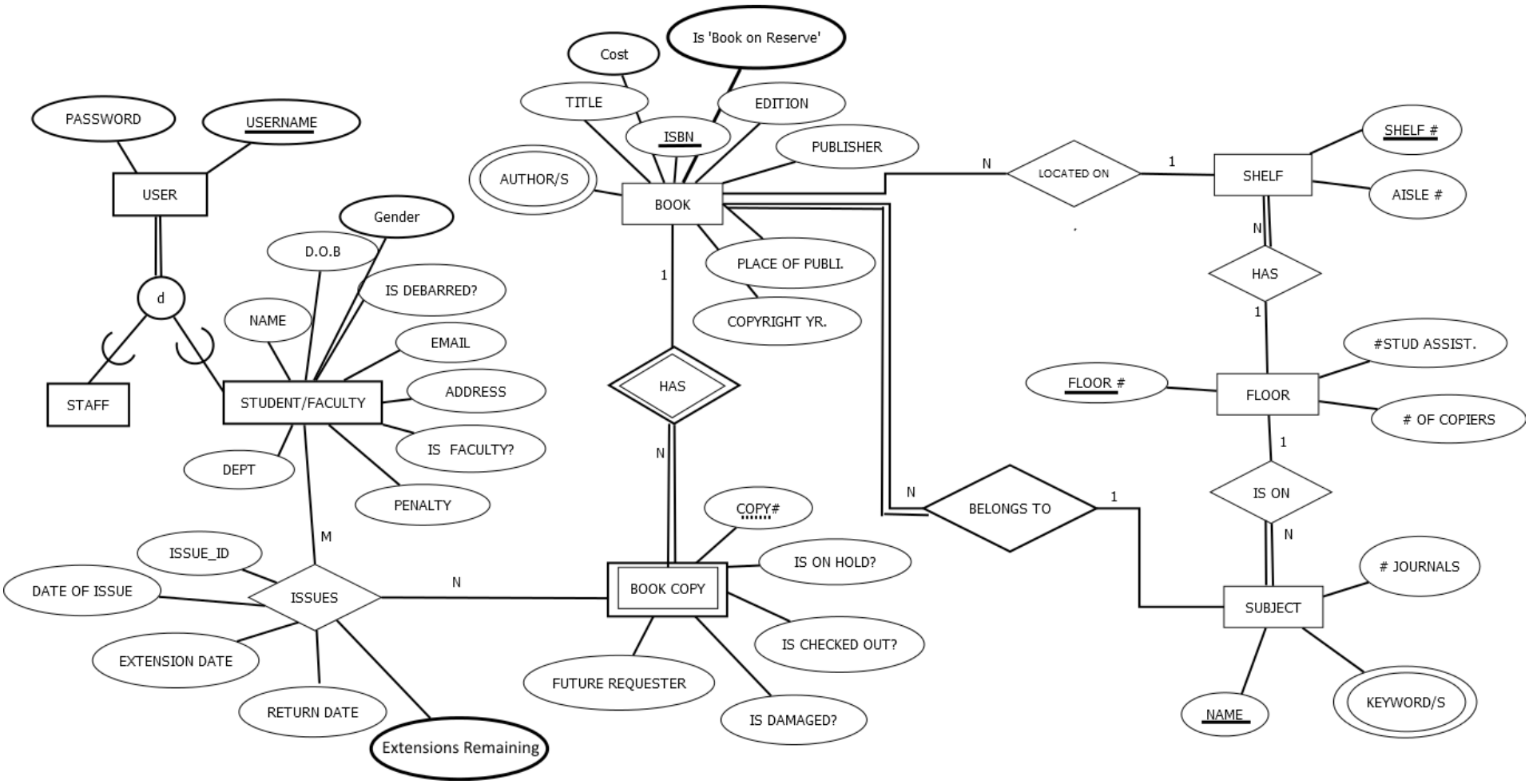
Amit Lift • alift3@gatech.edu • alift3 • section A

Paras Jain • pjain67@gatech.edu • pjain67 • section A

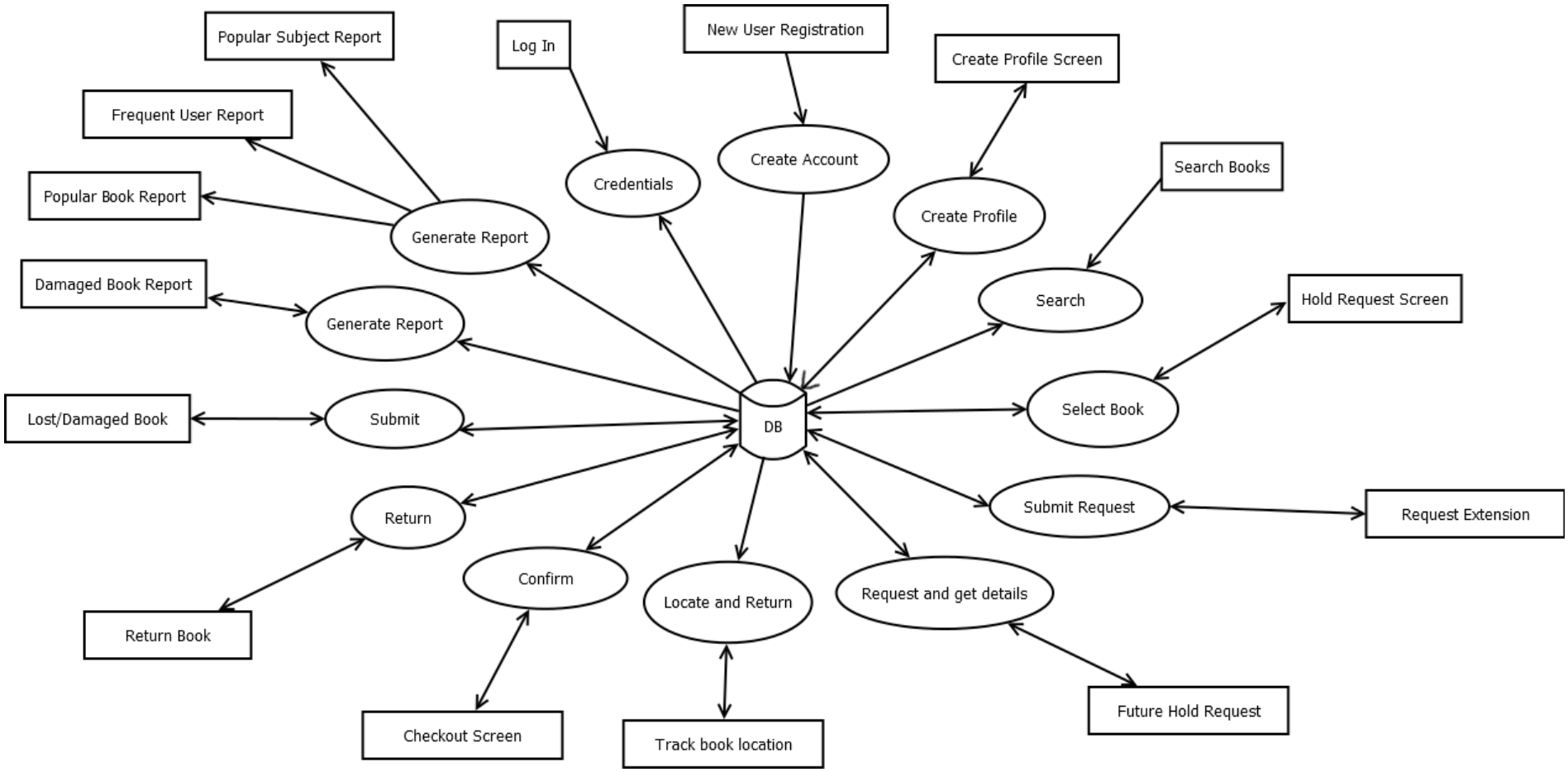
Andrew Smith • asmith379@gatech.edu • asmith379 • section A

Kenneth Thompson • kthompson62@mail.gatech.edu • kthompson62 • section A

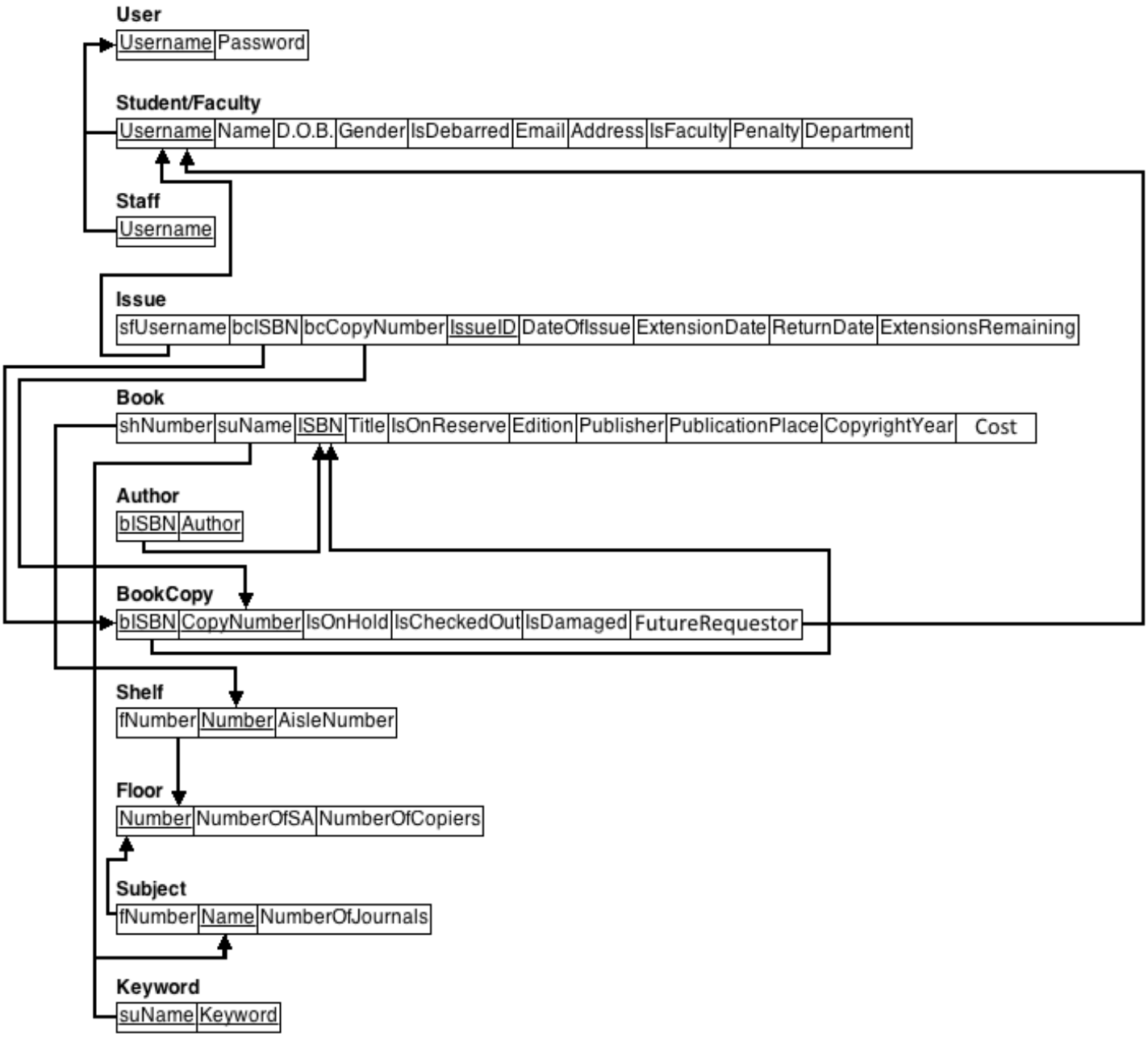
ER DIAGRAM


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

INFORMATION FLOW DIAGRAM



RELATIONAL SCHEMA DIAGRAM



CREATE TABLE STATEMENTS

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

CREATE TABLE StudentFaculty (
     Username varchar(50) NOT NULL,
     Name varchar(50) NOT NULL,
     DOB datetime NULL,
     Gender char(1) NULL,
     IsDebarred bool NOT NULL DEFAULT '0',
     Email varchar(50) NOT NULL,
     Address varchar(200) NULL,
     IsFaculty bool NOT NULL DEFAULT '0',
     Penalty numeric(7,2) NULL DEFAULT '0',
     Department varchar(50) NULL,
     PRIMARY KEY(Username),
     FOREIGN KEY(Username)
          References `User` (Username)
          ON DELETE CASCADE
          ON UPDATE CASCADE);

CREATE TABLE Staff (
     Username varchar(50) NOT NULL,
     PRIMARY KEY(Username),
     FOREIGN KEY(Username)
          References `User` (Username)
          ON DELETE CASCADE
          ON UPDATE CASCADE);

CREATE TABLE Issue (
     sfUsername varchar(50) NOT NULL,
     bcISBN varchar(13) NOT NULL,
     bcCopyNumber INT NOT NULL,
     IssueID BIGINT NOT NULL AUTO_INCREMENT,
     DateOfIssue datetime NOT NULL,
     ExtensionDate datetime NOT NULL,
     ReturnDate datetime DEFAULT NULL,
     ExtensionsRemaining int(1) NOT NULL DEFAULT '2',
     PRIMARY KEY(IssueID),
     FOREIGN KEY(sfUsername)
          References `User` (Username),
     FOREIGN KEY(bcISBN, bcCopyNumber)
          References BookCopy (bISBN, CopyNumber));
```

```
CREATE TABLE Book (
     shNumber MEDIUMINT NOT NULL,
     suName varchar(50) NOT NULL,
     ISBN varchar(13) NOT NULL,
     Title varchar(200) NOT NULL,
     IsOnReserve bool NOT NULL DEFAULT '0',
     Edition INT NOT NULL DEFAULT '1',
     Publisher varchar(50) NOT NULL,
     PublicationPlace varchar(50) NOT NULL,
     CopyrightYear INT NOT NULL,
     Cost INT NOT NULL DEFAULT '10.00',
     PRIMARY KEY(ISBN),
     FOREIGN KEY(shNumber)
          References Shelf (Number),
     FOREIGN KEY(suName)
          References Subject (Name),

     CHECK(LEN(ISBN) = 10 OR LEN(ISBN) = 13));

CREATE TABLE Author (
     bISBN varchar(13) NOT NULL,
     Author varchar(50) NOT NULL,
     PRIMARY KEY(bISBN, Author),
     FOREIGN KEY(bISBN)
          References Book (ISBN)
          ON DELETE CASCADE);

CREATE TABLE BookCopy (
     bISBN varchar(13) NOT NULL,
     CopyNumber INT NOT NULL AUTO_INCREMENT,
     IsOnHold bool NOT NULL DEFAULT '0',
     IsCheckedOut bool NOT NULL DEFAULT '0',
     IsDamaged bool NOT NULL DEFAULT '0',
     FutureRequestor varchar(50) NULL,
     PRIMARY KEY(bISBN, CopyNumber),
     FOREIGN KEY(bISBN)
          References Book (ISBN)
          ON DELETE CASCADE
          ON UPDATE CASCADE);
```

```
CREATE TABLE Shelf (
     fNumber INT NOT NULL,
     Number INT NOT NULL,
     AisleNumber INT NOT NULL,
     PRIMARY KEY(Number),
     FOREIGN KEY(fNumber)
          References `Floor` (Number));

CREATE TABLE `Floor` (
     Number INT NOT NULL,
     NumberOfSA INT NOT NULL DEFAULT '0',
     NumberOfCopiers INT NOT NULL DEFAULT '0',
     PRIMARY KEY(Number));

CREATE TABLE Subject (
     fNumber INT NOT NULL,
     Name varchar(50) NOT NULL,
     NumberOfJournals INT NOT NULL DEFAULT '0',
     PRIMARY KEY(Name),
     FOREIGN KEY(fNumber)
          References `Floor` (Number));

CREATE TABLE Keyword (
     suName varchar(50) NOT NULL,
     Keyword varchar(50) NOT NULL,
     PRIMARY KEY(suName, Keyword),
     FOREIGN KEY(suName)
          References Subject (Name));
```

SQL TASKS

Get Credentials (Log in)

```
--$Username and $Password will be provided by program

SELECT *
FROM User
WHERE Username = '$Username' AND Password = '$Password';
```

Create Account

```
--$Username and $Password will be provided by program

INSERT INTO `User` (Username, Password) VALUES ('$Username', '$Password');
```

Create Profile

Initialize Create Profile Screen

```
--first, return list of possible departments, for program to
--create a dropdown list

SELECT DISTINCT Department
FROM StudentFaculty
WHERE Department IS NOT NULL
```

User Clicks 'Submit'

```
--variables preceded with '$' provided by user input

INSERT INTO     StudentFaculty (Username, Name, DOB, Gender, IsDisbarred,
                Email, Address, IsFaculty, Penalty, Department)
VALUES     ('$Username', '$Name', '$Date', '$Gender', 0, '$Email',
           '$Address', '$IsFaculty', 0.00, $Department);
```

Book Search

User Clicks 'Search'

```
--All 5 fields must be searched for. All variables preceded by $
--are provided by the user.

--@ISBN MIGHT NOT EQUAL $ISBN (no book found)
--Use @ISBN as the book that is found

SET @ISBN := (
SELECT ISBN
FROM Book INNER JOIN Author
ON Book.ISBN = Author.bISBN
WHERE  (ISBN = '$ISBN')
        AND (Title = '$Title')
        AND (Author.Author = '$Author')
        AND (Publisher = '$Publisher')
        AND (Edition = '$Edition')
);
```

Book Is Available

```
--Return a table that says whether book is on reserve or not,
--and how many copies are available (table will only have 1 row)

SELECT *
FROM
(
    SELECT IsOnReserve FROM Book WHERE ISBN = @ISBN
) AS table1
CROSS JOIN
(
    SELECT COUNT(*) FROM BookCopy
        WHERE bISBN = @ISBN AND IsCheckedOut = 0
        AND IsOnHold = 0 AND IsDamaged = 0
) AS table2;
```

Book Is Not Available

```
--return the expected available date

SELECT ReturnDate
FROM Issue, BookCopy
WHERE Issue.bcISBN = BookCopy.bISBN AND Issue.bcCopyNumber = CopyNumber
AND bcISBN = @ISBN AND Issue.ReturnDate IN (
      SELECT MIN(ReturnDate) FROM Issue INNER JOIN BookCopy
      ON bcISBN = bISBN AND bcCopyNumber = CopyNumber
      WHERE Issue.bcISBN = @ISBN AND IsCheckedOut = 1 AND IsDamaged = 0);
```

FUTURE HOLD REQUEST

```
--user provides $ISBN and $Username

--get the copy number of the next available book
--store this in @CopyNumber

SET @CopyNumber :=
(
      SELECT BookCopy.CopyNumber
      FROM Issue, BookCopy
      WHERE Issue.bcISBN = BookCopy.bISBN
      AND Issue.bcCopyNumber = CopyNumber AND bcISBN = '$ISBN'
      AND IsDamaged = 0 AND FutureRequestor IS NULL
      AND Issue.ReturnDate IN (
            SELECT MIN(ReturnDate) FROM (Issue INNER JOIN BookCopy
                  ON (bcISBN = bISBN AND bcCopyNumber = CopyNumber))
            WHERE Issue.bcISBN = '$ISBN' AND IsCheckedOut = 1
            AND IsDamaged = 0 AND FutureRequestor IS NULL)
);



--store the requestor's name against the next available copy

UPDATE BookCopy
SET FutureRequestor = '$Username'
WHERE bISBN = '$ISBN' AND CopyNumber = @CopyNumber
AND FutureRequestor IS NULL;



--select Copy Number and Expected return date for program output

SELECT BookCopy.CopyNumber, Issue.ReturnDate
FROM Issue, BookCopy
WHERE Issue.bcISBN = BookCopy.bISBN AND Issue.bcCopyNumber = CopyNumber
AND bcISBN = '$ISBN' AND IsDamaged = 0 AND FutureRequestor = '$Username'
AND Issue.ReturnDate IN (
      SELECT MIN(ReturnDate) FROM (Issue INNER JOIN BookCopy
            ON (bcISBN = bISBN AND bcCopyNumber = CopyNumber))
     WHERE Issue.bcISBN = '$ISBN' AND IsCheckedOut = 1 AND IsDamaged = 0
      AND FutureRequestor = '$Username');
```

TRACK BOOK LOCATION

```
--$ISBN is provided by program

SELECT fNumber, Shelf.Number, AisleNumber, suName
FROM Book, Shelf
WHERE ISBN = '$ISBN' AND Shelf.Number = shNumber
```

CHECKOUT BOOK

```
-- user provides $ISBN, $CopyNumber, and $Username

-- update the status flags for the book copy being checked out

UPDATE BookCopy
SET IsOnHold = 0, IsCheckedOut = 1
WHERE bISBN = '$ISBN' AND CopyNumber = '$CopyNumber';


-- query whether the user checking out the book is a student or faculty.
-- store this result in @IsFaculty
-- $Username is provided by program

SET @IsFaculty := (
SELECT      IsFaculty
FROM        StudentFaculty
WHERE       Username = '$Username'
);


--query whether the book being checked out is on reserve
--store result in @IsOnReserve
--$ISBN is provided by the program

SET @IsOnReserve := (
    SELECT IsOnReserve
    FROM   Book
    WHERE  ISBN = '$ISBN'
);
```

--cont. on next page...

```
-- if the checkout is not from a hold being fulfilled
-- (null ReturnDate means it is a hold being fulfilled)
-- then insert an entirely new issue into the table
-- normally mysql doesn't allow a WHERE in an insertion clause, but we
-- used a workaround by inserting the results of a selection, and tacking
-- the WHERE clause onto the selection
-- in other words, when it IS a hold being fulfilled, it inserts an EMPTY
-- selection

INSERT INTO Issue (sfUsername, bcISBN, bcCopyNumber, DateOfIssue,
ExtensionDate, ReturnDate, ExtensionsRemaining)
SELECT * FROM (
                SELECT '$Username', '$ISBN', '$CopyNumber', CURDATE() as
                IssDate, CURDATE() as extDate,
                            (CASE @IsOnReserve
                                WHEN 0 THEN CURDATE() + INTERVAL 14 DAY
                                ELSE CURDATE()
                            END) as retDate,
                            (CASE @IsFaculty
                                WHEN 0 THEN 2
                                ELSE 5
                            END) AS extRemaining
                ) AS selectionBeingInserted
                WHERE
                (SELECT IsDebarred FROM
                     StudentFaculty WHERE Username = '$Username') = 0
                AND NOT EXISTS(
                     SELECT i.IssueID
                     FROM (SELECT * FROM Issue) AS i
                     WHERE i.ReturnDate IS NULL AND i.bcISBN = '$ISBN'
                     AND i.bcCopyNumber = '$CopyNumber'
);

-- if the checkout IS fulfilling a hold, update the hold issue
-- no need to do case statement for IsFaculty (extensions remaining would
-- be properly set when the hold was issued) or IsOnReserve (reserved books
-- can't be put on hold in the first place).
-- we have to use workaround to reference issue in the where clause of an
-- update statement for issue (namely, copying issue into a new table i,
-- and using THAT table in the where clause)
UPDATE Issue
SET DateOfIssue = CURDATE(), ExtensionDate = CURDATE(),
     ReturnDate = CURDATE() + INTERVAL 14 DAY
WHERE ReturnDate IS NULL AND bcISBN = '$ISBN'
     AND bcCopyNumber = '$CopyNumber' AND EXISTS(
          SELECT i.IssueID FROM (SELECT * FROM Issue) AS i
          WHERE i.ReturnDate IS NULL AND i.bcISBN = '$ISBN'
          AND i.bcCopyNumber = '$CopyNumber'
);


-- NOTE: EITHER the update or the insert will affect the table. It is
-- impossible for both to occur. The one that will occur depends on the
-- result of the WHERE NOT EXISTS / WHERE EXISTS clauses.
```

RETURN BOOK

```
-- user provides $ISBN, $CopyNumber, and $Username

-- update status flags

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

-- return the expected return date for the issue of this copy with the
-- highest issue id (it's possible the same user checked out the same copy
-- more than once, at different times, so the issue with the largest issue
-- id is used to determine which issue is the one they currently hold)
-- store this as @ReturnDate
SET @ReturnDate := (
     SELECT      ReturnDate
     FROM        Issue
     WHERE      sfUsername = '$Username' AND bcISBN = '$ISBN'
                AND bcCopyNumber = '$CopyNumber'
                AND IssueID IN (
                     SELECT MAX(IssueID) FROM Issue as I
                     WHERE i.sfUsername = '$Username'
                     AND i.bcISBN = '$ISBN'
                     AND i.bcCopyNumber = '$CopyNumber')
);

-- get the price of the book being returned late
-- (so max penalty can be calculated)
-- store in variable @Cost
SET @Cost := (
SELECT      Cost
FROM        Book
WHERE       ISBN = '$ISBN'
);

-- determine whether late fee should be charged, and
-- what the charge should be
UPDATE      StudentFaculty
SET         Penalty = Penalty +
                (CASE
                WHEN @Cost / 2.0 > 0.50 * DATEDIFF(CURDATE(), @ReturnDate)
                     THEN 0.50 * DATEDIFF(CURDATE(), @ReturnDate)
                ELSE
                     @Cost / 2.0
                END)
WHERE Username = '$Username' AND DATEDIFF(CURDATE(), @ReturnDate) > 0;

--debar the user if total penalty is >= $100
UPDATE      StudentFaculty
SET         IsDebarred = 1
WHERE       Username = '$Username' AND Penalty >= 100;
```

--cont. on next page...

```
--set ReturnDate of the issue to today's date
--look for max issue id again
--have to use workaround to reference issue in the where clause of an
--update statement for issue (namely, copying issue into a new table i,
--and using THAT table in the where clause)
UPDATE      Issue
SET         ReturnDate = CURDATE()
WHERE       sfUsername = '$Username' AND bcISBN = '$ISBN'
                AND bcCopyNumber = '$CopyNumber'
                AND IssueID IN (
                      SELECT MAX(i.IssueID) FROM (SELECT * FROM Issue) AS i
                           WHERE i.sfUsername = '$Username'
                           AND i.bcISBN = '$ISBN'
                           AND i.bcCopyNumber = '$CopyNumber');
```

REPORT BOOK AS LOST / DAMAGED

```
--Return current time, to be displayed in the program
--Although, because we order IssueID's chronologically, we don't actually
--have to use the current time

SELECT NOW();

--Find last user for given bookcopy(max issue id = most recent issue)
--store them as @MostRecentUser
--also, return them to the php code for display
--user supplies $ISBN and $CopyNumber

SET @MostRecentUser := (
SELECT sfUsername
FROM Issue INNER JOIN BookCopy
ON Issue.bcCopyNumber = CopyNumber AND Issue.bcISBN = BookCopy.bISBN
WHERE bcISBN = '$ISBN' AND CopyNumber = '$CopyNumber'
     AND IssueID IN (
          SELECT MAX(i.IssueID) FROM Issue AS i
          WHERE i.bcISBN = '$ISBN' AND i.bcCopyNumber = '$CopyNumber')
);

--charge specified amount to user
--$ChargeAmount provided by staff making the charge

UPDATE StudentFaculty
SET Penalty = Penalty + '$ChargeAmount'
WHERE Username = @MostRecentUser;

--debar the user if total penalty is >= $100

UPDATE      StudentFaculty
SET         IsDebarred = 1
WHERE       Username = @MostRecentUser AND Penalty >= 100;

--set book as lost/damaged, remove checkout/hold/future hold status

UPDATE      BookCopy
SET         IsDamaged = 1, IsCheckedOut = 0, IsOnHold = 0,
                FutureRequestor = NULL
WHERE       bISBN = '$ISBN' AND CopyNumber = '$CopyNumber';
```

GENERATE LOST / DAMAGED BOOKS REPORT

```
--user supplies $Month and $Subject1, $Subject2, $Subject3
--max IssueID = latest issue

--subject 1

SELECT COUNT(*)
FROM (
           (
           BookCopy
                INNER JOIN
           Issue
                ON Issue.bcCopyNumber = CopyNumber AND Issue.bcISBN = bISBN
           )
                INNER JOIN
           Book
                ON Book.ISBN = BookCopy.bISBN
     )
WHERE Book.suName = '$Subject1' AND MONTH(Issue.DateOfIssue) = '$Month'
AND BookCopy.IsDamaged = 1 AND Issue.IssueID IN (
           SELECT MAX(i.IssueID) FROM Issue AS i, BookCopy AS bc
           WHERE i.bcISBN = bc.bISBN
           AND i.bcCopyNumber = bc.CopyNumber
           GROUP BY CopyNumber, bISBN);

--subject 2

SELECT COUNT(*)
FROM (
           (
           BookCopy
                INNER JOIN
           Issue
                ON Issue.bcCopyNumber = CopyNumber AND Issue.bcISBN = bISBN
           )
                INNER JOIN
           Book
                ON Book.ISBN = BookCopy.bISBN
     )
WHERE Book.suName = '$Subject2' AND MONTH(Issue.DateOfIssue) = '$Month'
AND BookCopy.IsDamaged = 1 AND Issue.IssueID IN (
           SELECT MAX(i.IssueID) FROM Issue AS i, BookCopy AS bc
           WHERE i.bcISBN = bc.bISBN
           AND i.bcCopyNumber = bc.CopyNumber
           GROUP BY CopyNumber, bISBN);



--cont. on next page...
```

```
--subject 3

SELECT COUNT(*)
FROM (
           (
           BookCopy
                INNER JOIN
           Issue
                ON Issue.bcCopyNumber = CopyNumber AND Issue.bcISBN = bISBN
           )
                INNER JOIN
           Book
                ON Book.ISBN = BookCopy.bISBN
     )
WHERE Book.suName = '$Subject3' AND MONTH(Issue.DateOfIssue) = '$Month'
AND BookCopy.IsDamaged = 1 AND Issue.IssueID IN (
           SELECT MAX(i.IssueID) FROM Issue AS i, BookCopy AS bc
           WHERE i.bcISBN = bc.bISBN
           AND i.bcCopyNumber = bc.CopyNumber
           GROUP BY CopyNumber, bISBN);
```

GENERATE POPULAR BOOK REPORT

```
--January

SELECT Title, COUNT(*)
FROM Issue INNER JOIN Book ON Issue.bcISBN = Book.ISBN
WHERE MONTH(DateOfIssue) = 1
GROUP BY Title
ORDER BY COUNT(*) DESC
LIMIT 3

--February

SELECT Title, COUNT(*)
FROM Issue INNER JOIN Book ON Issue.bcISBN = Book.ISBN
WHERE MONTH(DateOfIssue) = 2
GROUP BY Title
ORDER BY COUNT(*) DESC
LIMIT 3
```

GENERATE FREQUENT USER REPORT

```
--JAN

SELECT *
FROM(
    SELECT Username, COUNT(*) AS numOfCheckouts
    FROM Issue INNER JOIN StudentFaculty ON sfUsername = Username
    WHERE MONTH(DateOfIssue) = 1
    GROUP BY Username
    ORDER BY COUNT(*) DESC
    LIMIT 5
) AS top5
WHERE numOfCheckouts >= 10;

--FEB

SELECT *
FROM(
    SELECT Username, COUNT(*) AS numOfCheckouts
    FROM Issue INNER JOIN StudentFaculty ON sfUsername = Username
    WHERE MONTH(DateOfIssue) = 2
    GROUP BY Username
    ORDER BY COUNT(*) DESC
    LIMIT 5
) AS top5
WHERE numOfCheckouts >= 10;
```

GENERATE POPULAR SUBJECT REPORT

```
--JAN

SELECT SuName, COUNT(*)
FROM (Issue INNER JOIN Book ON Issue.bcISBN = Book.ISBN)
WHERE MONTH(DateOfIssue) = 1
GROUP BY SuName
ORDER BY COUNT(*) DESC
LIMIT 3;

--FEB

SELECT SuName, COUNT(*)
FROM (Issue INNER JOIN Book ON Issue.bcISBN = Book.ISBN)
WHERE MONTH(DateOfIssue) = 2
GROUP BY SuName
ORDER BY COUNT(*) DESC
LIMIT 3;
```