Lucas Beal
CS – 3140
Rod Library Database Implementation Assignment
12/10/2019

As stated in the requirements for this assignment, I will only illustrate the setters and getters for a few of the relations, primarily Patron, as these are very simplistic in nature. Otherwise, I will be detailing the methods discussed in my design documents.

I have also left the table creation SQL statements within the document for easy access, as well as 1 or so insertion statements that I used to insert data into my database.

Please note, that I went back to my SQL tables and edited them to have the Primary Keys I had originally forgot to add, therefore some Foreign Constraints under SHOW CREATE TABLE might be one off in their …_ibfk_1 part.

Lastly, for some reason, Putty/MySQL may not always take my commas well from Word. Sometimes it will and sometimes it will throw an error.

For some of the methods, I have attached them to the relations they directly effect, instead of the methods they necessarily "came from", as I didn't want to clutter the page with several methods originating from one table.

PATRON:

```
CREATE TABLE Patron(
	first_name		VARCHAR (255) NOT NULL,
	last_name		VARCHAR (255) NOT NULL,
	fine_total		DECIMAL(5,2) DEFAULT '0.00',
	catID			INT,
	PRIMARY KEY (catID);
);
============
```

Data Insertion:
People entering users into the system must first create a tuple of that individual within the Patron relation. Trying to create a guest or student etc., tuple first will rightfully cause an error, as the foreign key for catID was not already established within the Patron relation.

```
INSERT INTO Patron (first_name, last_name, catID) VALUES ('Lucas', 'Baal', '555111');
INSERT INTO Patron (first_name, last_name, catID) VALUES ('Andrew','Berns','981324');
INSERT INTO Patron (first_name, last_name, catID) VALUES ('Carl', 'Eugen', '888222');
INSERT INTO Patron (first_name, last_name, catID) VALUES ('Khan', 'Singh', '777444');
=================
```

RequestViewingAccess()
Someone wants to access the online text for the book Fahrenheit 451.

```
SELECT link_to_text FROM Online_Book WHERE title = 'Fahrenheit 451';
=================
```

Simple Setter and Getter:
UPDATE Patron SET last_name = ‘Beal’ WHERE catID = ‘555111’;
SELECT first_name FROM Patron WHERE catID = ‘777444’;
================
addToFineTotal()
Upon the unfortunate event of someone not returning their loaned items in time, their fine total is increased by whatever the fine rate is for the particular item they had loaned out. Guests, Students, and others have a different flat fine rate for books, DVDs, etc, but rental equipment have very different fees associated. For this, we will look at Carl who rented a laptop and forgot to return it in time.

```
UPDATE Patron SET fine_total = fine_total + (
SELECT laptop_loan_fine_rate FROM Laptop, Rent_Item WHERE Laptop.barcode_number =
Rent_Item.barcode_number AND Rent_Item.patron_ID = ‘888222’
) WHERE catID = ‘888222’;
```

===============
Scan() and Print() are just insertions into the relevant Scan_Queue and Print_Queue tables.

===============

GUEST:

```
CREATE TABLE Guest(
        guest_permissions               INT DEFAULT 1,
        guest_book_checkout_period      INT DEFAULT 7,
        guest_media_checkout_period     INT DEFAULT 7,
        guest_fine_rate                 DECIMAL(5,2) DEFAULT ‘1.25’,
        guest_max_books_allowed_out     INT DEFAULT 10,
        guest_max_media_allowed_out     INT DEFAULT 10,
        catID                           INT,
        PRIMAY KEY (catID),
        FOREIGN KEY (catID) REFERENCES Patron (catID)
);
```

=====
Putting Lucas into the Guest Relation
INSERT INTO Guest (catID) VALUES (‘555111’);
Checking to Ensure was Added Correctly
SELECT * FROM Guest;
=====
STUDENT:

```
CREATE TABLE Student(
          student_permissions                 INT DEFAULT 2,
          student_book_checkout_period        INT DEFAULT 14,
          student_media_checkout_period       INT DEFAULT 7,
          student_fine_rate                   DECIMAL(5.2) DEFAULT ‘1.25’,
          student_max_books_allowed_out       INT DEFAULT 15,
          student_max_media_allowed_out       INT DEFAULT 15,
          catID                               INT,
          PRIMARY KEY (catID),
          FOREIGN KEY catID REFERENCES Patron (catID)
);
========
```

Putting Carl into the Student Relation

```
INSERT INTO Student (catID) VALUES (‘888222’);

SELECT * FROM Student;
========
```

FACULTY:

```
CREATE TABLE Faculty(
          faculty_permissions                 INT DEFAULT 3,
          faculty_book_checkout_period        INT DEFAULT 30,
          faculty_media_checkout_period       INT DEFAULT 30,
          faculty_fine_rate                   DECIMAL(5,2) DEFAULT ‘2.50’,
          faculty_max_books_allowed_out       INT DEFAULT 50,
          faculty_max_media_allowed_out       INT DEFAULT 50,
          catID                               INT,
          PRIMARY KEY (catID),
          FOREIGN KEY catID REFERENCES Patron (catID)
};
=======
```

Putting Andrew Berns in the Faculty Relation

```
INSERT INTO Faculty (catID) VALUES (’981324’);

SELECT * FROM Faculty;
=======
```

Librarian:

```
CREATE TABLE Librarian(
          librarian_permissions               INT DEFAULT 4,
          librarian_book_checkout_period      INT DEFAULT 14,
          librarian_media_checkout_period     INT DEFAULT 7,
          librarian_fine_rate                 DECIMAL(5.2) DEFAULT ‘3.00’ ,
          librarian_max_books_allowed_out     INT DEFAULT 25,
	librarian_max_media_allowed_out	INT DEFAULT 25,
	catID					INT,
	PRIMARY KEY(catID),
	FOREIGN KEY (catID) REFERENCES Patron (catID)
);
```

==========
Putting Khan into the Librarian Relation
INSERT INTO Librarian (catID) VALUES (‘777444’);
==========
The librarian does have several ‘ADD’ methods of adding books, DVD’s etc. These are merely insertions. As an example, if Khan wanted to add a CD, he would first have to add the CD’s barcode in Barcodes, then add the relevant information to Disc_Media, and then finally into the CD relation.

```
INSERT INTO Barcodes (barcode_number) VALUES ('123');

INSERT INTO Disc_Media (title, barcode_number, description) VALUES ('Imagine', '123', 'John Lennons Imagine Cd with tracks ...');

INSERT INTO CD (artist, music_style, barcode_number) VALUES ('John Lennon', 'Rock','123');
```

==========
ONLINE BOOK:

Note: I had to change the VARCHAR limits to primary_author and link_to_text, as cannot exceed 767 bytes for a key, which translates to 191 in this case. I also removed title as part of the primary key, as was not necessary, with the author and link being unique values already.

```
CREATE TABLE Online_Book(
	primary_author		VARCHAR(100),
	secondary_author	VARCHAR(255),
	isbn			BIGINT(8),
	publisher		VARCHAR(255),
	genre			VARCHAR(255) NOT NULL,
	title			VARCHAR(255),
	description		VARCHAR(255) NOT NULL,
	link_to_text		VARCHAR(90),
	PRIMARY KEY (primary_author, link_to_text)
);
```

=================
addOnlineBook()
When a Librarian is tasked with adding an additional resource into the database to be freely accessible by others. For this scenario, no secondary author is known.

```
INSERT INTO Online_Book (primary_author, isbn, publisher, genre, title, description,
link_to_text) VALUES (‘Ray Bradbury’ ,’ 9788429772456 ‘, ‘Simon & Schuster’, ‘Fiction’,
‘Fahrenheit 451’, ‘Guy Montag is a fireman…’, ‘cool url here’);
```

=================

accessOnlineMaterial()
When someone who is a registered user from the library, wants to gain access to an online book, they will need the url from the book to go to the book.

```
SELECT link_to_text FROM Online_Book WHERE title = ‘Fahrenheit 451’;
```

=================

BARCODES:

```
CREATE TABLE Barcodes(
	barcode_number	INT,
	item_status		VARCHAR(255) DEFAULT ‘Available’,
	PRIMARY KEY (barcode_number)
);
```

PRINT BOOK:

```
CREATE TABLE Print_Book(
	primary_author		VARCHAR(255) NOT NULL,
	secondary_author	VARCHAR(255),
	isbn			INT,
	barcode_number		INT,
	publisher		VARCHAR(255),
	genre			VARCHAR(255) NOT NULL,
	title			VARCHAR(255) NOT NULL,
	dewey_decimal		VARCHAR(255),
	description		VARCHAR(255) NOT NULL,
	PRIMARY KEY (barcode_number),
	FOREIGN KEY (barcode_number) REFERENCES Barcodes (barcode_number)
);
```

DISC MEDIA:

```
CREATE TABLE Disc_Media(
	title			VARCHAR(255) NOT NULL,
	barcode_number		INT,
	description		VARCHAR(255) NOT NULL,
	PRIMARY KEY (barcode_number),
	FOREIGN KEY (barcode_number) REFERENCES Barcodes (barcode_number)
);
```

CD:

```
CREATE TABLE CD(
	artist			VARCHAR(255) NOT NULL,
	secondary_artist	VARCHAR(255),
	music_style		VARCHAR(255) NOT NULL,
	barcode_number		INT,
	PRIMARY KEY (barcode_number),
	FOREIGN KEY (barcode_number) REFERENCES Disc_Media (barcode_number)
);
```

DVD:

```
CREATE TABLE DVD(
	main_director		VARCHAR(255) NOT NULL,
	genre			VARCHAR(255) NOT NULL,
	publisher		VARCHAR(255) NOT NULL,
	barcode_number		INT,
	PRIMARY KEY (barcode_number),
	FOREIGN KEY (barcode_number) REFERENCES Disc_Media (barcode_number)
);
```

ARTICLE:

```
CREATE TABLE Article(
	primary_author			VARCHAR(255) NOT NULL,
	secondary_author		VARCHAR(255),
	issn				INT,
	title				VARCHAR(255) NOT NULL,
	description			VARCHAR(255) NOT NULL,
	PRIMARY KEY (issn)
);
```

PRINT ARTICLE:

```
CREATE TABLE Print_Article(
	barcode_number		INT,
	issn			INT,
	PRIMARY KEY(barcode_number),
	FOREIGN KEY (issn) REFERENCES Article (issn),
	FOREIGN KEY (barcode_number) REFERENCES Barcodes (barcode_number)
);
```

DIGITAL ARTICLE:

```
CREATE TABLE Digital_Article(
        online_source           VARCHAR(255) NOT NULL,
        issn                    INT,
        PRIMARY KEY (issn),
        FOREIGN KEY (issn) REFERENCES Article (issn)
);
```

STUDENT STUDY ROOM:

```
CREATE TABLE Student_Study_Room(
        room_number                     INT,
        room_computer                   VARCHAR(255),
        wall_whiteboard                 VARCHAR(255),
        recommended_seating_limit INT NOT NULL,
        PRIMARY KEY (room_number)
);
```

SCANNER:

```
CREATE TABLE Scanner(
        scanner_id                      VARCHAR(100),
        scanner_location                VARCHAR(255) NOT NULL,
        scanner_operating_status        VARCHAR(255) NOT NULL,
        PRIMARY KEY (scanner_id)
);
```

PRINTER:

```
CREATE TABLE Printer(
        printer_id                      VARCHAR(100),
        printer_location                VARCHAR(255) NOT NULL,
        printer_ink_level               INT NOT NULL,
        printer_paper_level             INT NOT NULL,
        printer_operating_status        VARCHAR(255) NOT NULL,
        PRIMARY KEY (printer_id)
);
```

RENT-ABLE ITEMS:

```
CREATE TABLE Rentable_Items(
        item_name               VARCHAR(255) NOT NULL,
	barcode_number	INT,
	PRIMARY KEY (barcode_number)
);
==========
```

Insertion to allow Carl to rent an item

```
INSERT INTO Rentable_Items(item_name,barcode_number) VALUES ('Laptop1','774455');
==========
```

CAMERA:

```
CREATE TABLE Camera(
	camera_loan_duration		INT DEFAULT 7,
	camera_loan_fine_rate		DECIMAL(5.2) DEFAULT ’10.00’,
	camera_availability_status	VARCHAR(255) NOT NULL,
	barcode_number			INT,
	PRIMARY KEY (barcode_number),
	FOREIGN KEY (barcode_number) REFERENCES Rentable_Items (barcode_number)
);
```

CALCULATOR:

```
CREATE TABLE Calculator(
	calculator_loan_duration	INT DEFAULT 14,
	calculator_loan_fine_rate	DECIMAL(5,2) DEFAULT ‘2.50’
	calculator_availability_status VARHCAR(255) NOT NULL,
	barcode_number			INT,
	FOREIGN KEY (barcode_number) REFERENCES Rentable_Items (barcode_number)
);
```

LAPTOP:

```
CREATE TABLE Laptop(
	laptop_loan_duration		INT DEFAULT 1,
	laptop_loan_fine_rate		DECIMAL(5.2) DEFAULT ’25.00’,
	laptop_availability_status	VARCHAR(255) NOT NULL,
	barcode_number			INT,
	FOREIGN KEY (barcode_number) REFERNCES Rentable_Items (barcode_number)
);
=======
```

Insertion statement so Carl can rent a laptop

```
INSERT INTO Laptop (laptop_availability_status, barcode_number) VALUES
('Available','774455');
=======
```

WHITEBOARD AND MARKERS:

```
CREATE TABLE Whiteboard_Markers(
        whiteboard_markers_loan_duration          INT DEFAULT 1,
        whiteboard_markers_loan_fine_rate         DECIMAL(5,2) DEFAULT ‘1.50’,
        whiteboard_markers_availability_status    VARCHAR(255) NOT NULL,
        barcode_number                            INT,
        PRIMARKY KEY (barcode_number),
        FOREIGN KEY (barcode_number) REFERENCES Rentable_Items (barcode_number)
);
```

CHECK OUT:

Note: I had to deviate away from my design, as I found out that having multiple different check_out tables for each different type of item the library offered would be tedious to maintain. So, I created a Barcodes relation, that holds all the barcodes of anything you can get on loan from the Library that isn’t a Rentable_Item, like a laptop. With this in mind, we only need to communicate with the Barcodes Relation, which also has an item_status attribute we can use to communicate that an item is out on loan.

```
CREATE TABLE Check_Out(
        loan_ID             INT,
        lib_ID              INT,
        barcode_number      INT,
        patron_ID           INT,
        PRIMARY KEY (loan_ID),
        FOREIGN KEY (barcode_number) REFERENCES Barcodes (barcode_number),
        FOREIGN KEY (lib_ID) REFERENCES Librarian(catID),
        FOREIGN KEY (patron_ID) REFERENCES Patron (catID)
);
```

=======

CheckOut()
Luke decides that he wants to check something out. So, we need to perform a tuple insertion into the Check_Out relation, with the loan_ID, lib_ID, barcode_number, and patron_ID all required. With them being foreign keys, it ensures that someone is not checking out an item that does not exist, they are actually in the system, and that a valid Librarian is conducting the transaction.

```
INSERT INTO Check_Out (loan_ID, lib_ID, barcode_number, patron_ID) VALUES
(‘411’,’777444’,’888’,’555111’);
```

After an Insertion, we must also modify the Item_Status in the Barcodes table to ensure that online it would say that it was not available.

UPDATE Barcodes SET item_status = “Loaned Out” WHERE barcode_number = ‘888’;
============
Return()

Upon returning an item, the tuple from the Check_Out relation must be deleted, and the availability changed in Barcodes.

DELETE FROM Check_Out WHERE loan_ID = ‘411’;

UPDATE Barcodes SET item_status = “Available” WHERE barcode_number = ‘888’;
============

RENT ITEM:

```
CREATE TABLE Rent_Item(
        loan_ID             INT,
        lib_ID              INT,
        barcode_number      INT,
        patron_ID           INT,
        PRIMARY KEY (loan_ID),
        FOREIGN KEY (barcode_number) REFERENCES Rentable_Items (barcode_number),
        FOREIGN KEY (lib_ID) REFERENCES Librarian(catID),
        FOREIGN KEY (patron_ID) REFERENCES Patron (catID)
);
```
===========
rentCheckOut()
User Carl rents a laptop with Khan as the librarian authorizing the rent application.

INSERT INTO Rent_Item (loan_ID, lib_ID, barcode_number, patron_ID) VALUES (‘0001’,’777444’,’774455’,’888222’);

Since the laptop was just rented out, the laptop_availability_status must also be set to indicate that it is not available.

UPDATE Laptop SET laptop_availability_status = 'Loaned Out' WHERE barcode_number = 774455;

Return()
Merely delete the tuple from the Rent Item table, and set laptop to Available Status

DELETE FROM Rent_item WHERE loanID = ‘0001’;

```
UPDATE Laptop SET laptop_availability_status = "Available" WHERE barcode_number =
'774455';
===========
```

ROOM RESERVATION:

```
CREATE TABLE Room_Reservation(
	reservation_number	INT,
	student_ID		INT,
	room_number		INT,
	PRIMARY KEY (reservation_number, room_number),
	FOREIGN KEY (student_ID) REFERENCES Student (catID),
	FOREIGN KEY (room_number) REFERENCES Student_Study_Room(room_number)
);

=========
```

ReserveRoom()
Carl wants to reserve a room for study purposes. An insertion is made in the room reservation relation.

```
INSERT INTO Room_Reservation (reservation_number, student_ID, room_number) VALUES
(‘878’, ‘888222,’2’);
==========
```

SCAN QUEUE:

```
CREATE TABLE Scan_Queue(
	position	INT,
	document	VARCHAR(255) NOT NULL,
	patronID	INT,
	PRIMARY KEY(position, patronID),
	FOREIGN KEY (patronID) REFERENCES Patron (catID)
);
```

PRINT QUEUE:

```
CREATE TABLE Print_Queue(
	position	INT,
	document	VARCHAR(255) NOT NULL,
	patronID	INT,
	PRIMARY KEY(position, patronID),
	FOREIGN KEY (patronID) REFERENCES Patron (catID)
);
```