Chicago Public Library Database Design

Ronak Desai

I. Introduction

Chicago Public Library is a public library that allows customers to rent books to read. Customers can checkout multiple books each visit, and the library holds multiple copies of all books. Keeping track of the date customers check out and return a book is an important part of the librarian’s job. Customers are responsible for returning the books within the next two weeks after checkout. If the customer fails to make the return by the recorded due date, the customer is then fined $1.00 for each day that the return date exceeds the due date for each book checked out. Librarians are also required to record the customer’s fine amount along with the customer’s contact info to inform them of the fine. Customers are given a onetime option to renew books which extends the due date by an additional two weeks.

II. Project Definition

A. Problem with the current design that will be addressed

The current dataset: Library(Title, Customer, Address, Phone,EmailAddress,Zip,CheckOutDate,ReturnedDate,DueDate,ExtendedDueDate,FineAmount)

Modification problems exist with the current dataset.

Update

If a customer is fined for multiple late returned books the librarians then need to update the spreadsheet with the fine amount and because each book checked out is a separate entry, multiple rows need to be updated. The reason this is a problem is because having to update multiple rows can become very time consuming with a large database. If a customer’s address or phone changes then multiple rows also need to be updated and this is a problem because it can also be very time consuming.

Delete

When a customer pays off all of the fines, they owed the librarians then need to delete multiple rows if there are multiple fines. This is a problem because it can also be very time consuming with a large database to delete multiple rows in the FineAmount column. Deleting the amount that a customer was fined when a partial or full payment was made can make the customer’s current fine amount accurate. However, this is a problem because the history of what the customer paid and previously owed is now lost and only what the customer now currently owes is kept.

Insert

Inserting data is another problem for the librarians because if two people checkout the same copy of a book on the same date the librarians only insert the book name.

No other information about the book being inserted is a problem because when the book is returned librarians cannot tell which customers’ copy was returned. If two customers have the same name inserted into the spreadsheet it is a problem because there is not a way of knowing the two customers apart due to the customers not being given a unique identifier or a library card.

B. Examples of the current approach

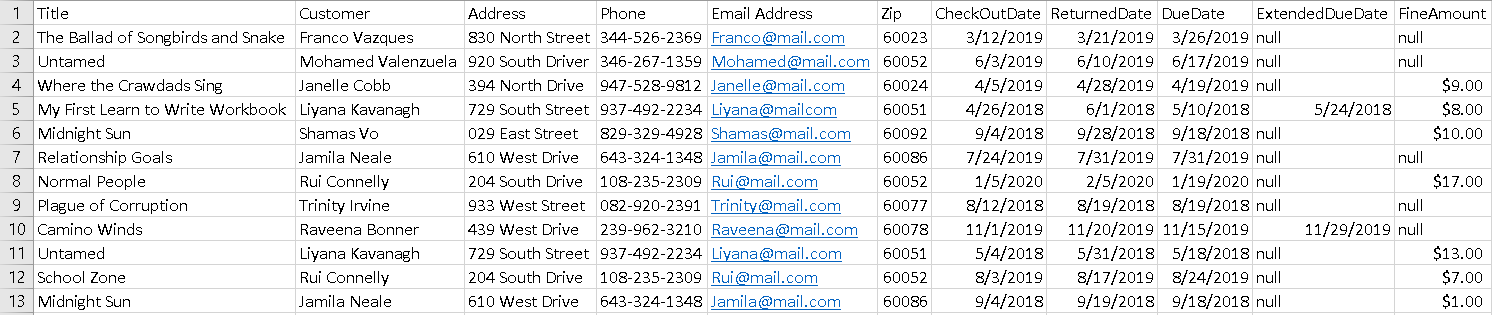
The current data:

Library(Title, Customer, Address, Phone,EmailAddress,Zip,CheckOutDate,ReturnedDate,DueDate,ExtendedDueDate,FineAmount )

Sample Form:

|  |
| --- |
| Title: The Ballad of Songbirds and Snakes  Customer: Franco Vasquez  Address: 830 North Street  Phone: 344-536-2369  Email Address: Franco@mail.com  Zip: 60023  Check Out Date: 3/12/2019  Returned Date: 3/21/2019  Due Date: 3/26/2019  Extended Due Date: null  Fine Amount: null |

Spreadsheet with data from all books that were checked out:



C. The new approach

The intuitive model consists of three tables, with the one keeping track of the customer’s information. Another table keeps track of the date books are checked out and due. The day the books are returned is also tracked. The last table keeps track of the books the library owns. All these tables will be linked together so that we have no problem with modifying any of the tables.

Intuitive model:

CUSTOMER(LastName, FirstName,­­ ­­­­­Address,Phone,EmailAddress,Zip, CurrentFineAmount,TotalAmountFined,AmountPaid)

CheckOut(CheckOutDate,ReturnedDate,DueDate)

BOOK(Dewey Decimal Classification, Title)

The following model would address the issues presented in Part II.A. above.

Deletion: When a customer no longer owes anything the CurrentFineAmount value can be deleted without losing information because there is also a TotalAmountFined column. This solves the problem of multiple rows in a spreadsheet holding multiple fine amounts and having to delete them all when a customer no longer owes anything. It also solves the problem of losing what the customer paid and previously owed because it is kept in the AmountPaid and TotalAmountFined column.

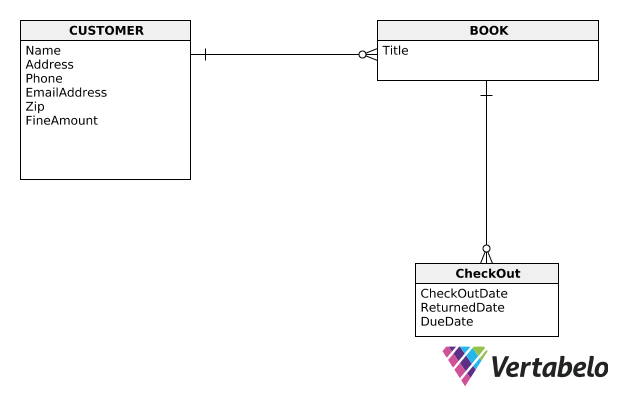
Update: When a customer is fined again, the customer’s info about fines can be updated in the CUSTOMER table. This solves the problem of having to update multiple rows and only requires one row to be updated. When a customer’s phone number or other information changes the customer’s row in the CUSTOMER table can be updated. This also solves the problem of having to update multiple rows and saves time due to only one row having to be updated.

Insertion: The CUSTOMER table will contain information about only the customers and will prevent data from being missing which will help distinguish between customers with the same name when the customer’s info is first inserted into the database. The CheckOut table will contain all information about the books that a customer has checked out and will prevent data from being missing which will be used to prevent problems of knowing the difference between which customer checked out which copy of a book.

III. Entity-Relationship/Data Modeling  
A. Conceptual model

In this model three tables are being used to keep track of the customer, the books in the library, and the books that are checked out.

1. The model



ii. SQL for the model

-- Created by Vertabelo (http://vertabelo.com)

-- Last modification date: 2020-05-29 05:18:57.549

-- tables

-- Table: BOOK

CREATE TABLE BOOK (

Title NOT NULL

);

-- Table: CUSTOMER

CREATE TABLE CUSTOMER (

Name NOT NULL,

Address NOT NULL,

Phone NOT NULL,

EmailAddress NOT NULL,

Zip NOT NULL,

FineAmount NOT NULL

);

-- Table: CheckOut

CREATE TABLE CheckOut (

CheckOutDate NOT NULL,

ReturnedDate NOT NULL,

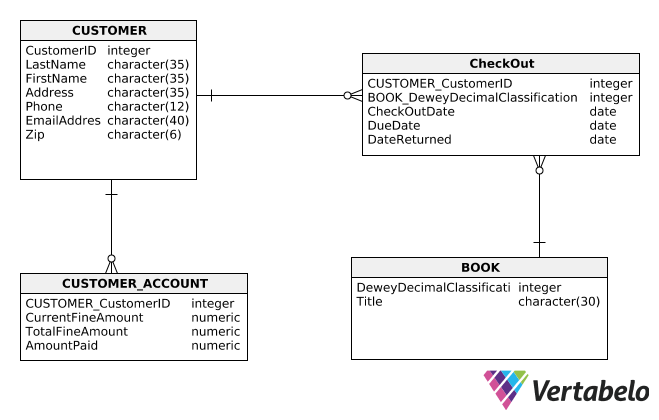
DueDate NOT NULL

);

-- End of file.

B. Logical model

In this model the primary and foreign keys have not been identified yet. A CUSTOMER\_ACCOUNT and a FINES table have been added to separate the customers fines and book history from customers contact information.  
 i. The model



ii. SQL for the model

-- Created by Vertabelo (http://vertabelo.com)

-- Last modification date: 2020-06-02 19:24:25.093

-- tables

-- Table: BOOK

CREATE TABLE BOOK (

DeweyDecimalClassification integer NOT NULL,

Title character(30) NOT NULL

);

-- Table: CUSTOMER

CREATE TABLE CUSTOMER (

CustomerID integer NOT NULL,

LastName character(35) NOT NULL,

FirstName character(35) NOT NULL,

Address character(35) NOT NULL,

Phone character(12) NOT NULL,

EmailAddress character(40) NOT NULL,

Zip character(6) NOT NULL

);

-- Table: CUSTOMER\_ACCOUNT

CREATE TABLE CUSTOMER\_ACCOUNT (

CUSTOMER\_CustomerID integer NOT NULL,

CurrentFineAmount numeric NOT NULL,

TotalFineAmount numeric NOT NULL,

AmountPaid numeric NOT NULL

);

-- Table: CheckOut

CREATE TABLE CheckOut (

CUSTOMER\_CustomerID integer NOT NULL,

BOOK\_DeweyDecimalClassification integer NOT NULL,

CheckOutDate date NOT NULL,

DueDate date NOT NULL,

DateReturned date NOT NULL

);

-- End of file.

C. Physical model for the implementation in Access

This model will normalize the data to make it less difficult to modify data. The tables are linked together using primary and foreign keys and minimum and maximum cardinality is shown.

1. Function dependencies and the normalized tables

CUSTOMER 🡪(LastName,FirstName,Address,Phone,EmailAddress,Phone)  
CUSTOMER🡪 (CheckOutDate,DateReturned,DeweyDecimalClassification)  
CUSTOMER🡪(CurrentFineAmount,TotalFineAmount,AmoundPaid)

BOOK🡪(DeweyDecimalClassification,Title)

CHECKOUT 🡪(CheckOutDate,DueDate,DeweyDecimalClassification,Customer)

CUSTOMER(CustomerID,LastName,FirstName,Address,Phone,EmailAddress,Zip)  
CUSTOMER\_ACCOUNT(*CUSTOMER\_CustomerID*,CheckOutDate,DueDate,DateReturned,*BOOK\_DeweyDecimalClassification*)  
FINES(*CUSTOMER\_CustomerID*,CurrentFineAmount,TotalFineAmount,AmountPaid)  
BOOK(DeweyDecimalClassification,Title)  
CheckOut(*CUSTOMER\_CustomerID*,*BOOK\_DeweyDecimalClassification*,CheckOutDate,DueDate)

ii. Constraints, assumptions, and relations

CONSTRAINTS:  
CUSTOMER\_CustomerID in CUSTOMER\_ACCOUNT must exist in CustomerID in CUSTOMER

BOOK\_DeweyDecimalClassification in CUSTOMER\_ACCOUNT must exist in DeweyDecimalClassification in BOOK  
CUSTOMER\_CustomerID in FINES must exist in CustomerID in CUSTOMER  
CUSTOMER\_CUSTOMERID in CheckOut must exist in CustomerID in CUSTOMER  
BOOK\_DeweyDecimalClassification in CheckOut must exist in DeweyDecimalClassification in BOOK

Assumptions:

CustomerID in the CUSTOMER table represents a customer’s library card  
The CheckOut table will show the history of all books that a user has checkout and the date it was returned along with the due date. This is used to calculate fines and then put into the FINES table.

An ExtendedDueDate column is not need in CheckOut table because DueDate can be updated when a book is renewed.

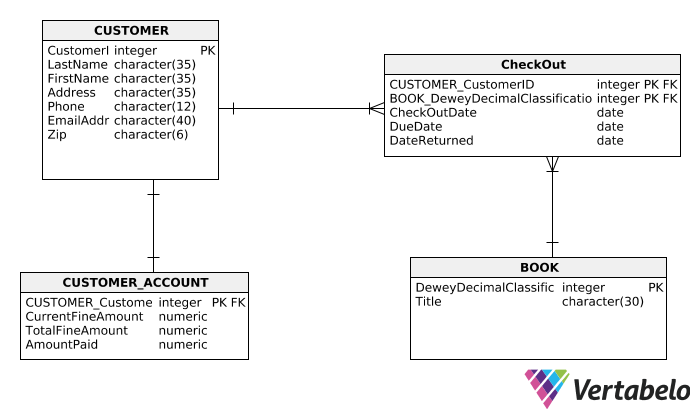
The CustomerAccount table has a CurrentFineAmount which can have its value be set to 0 when a customer no longer owes anything and the TotalAmountFined keeps track of the user’s fine history  
The AmountPaid in the FINES table can be calculated by subtracting the values from the TotalAmountFined and the CurrentFineAmount  
A customer’s ID will show up in the CustomerAccount table when they get a library card even if they have not checked out any books and never been fined. All values besides CustomerID in the FINES table will be set to 0 if no fines have occurred yet. The table will be updated when a customer is fined.  
A customer cannot checkout the same copy of a book after they return it.

CustomerID start at 1, increment by 1  
DeweyDecimalClassification start at 1, increment by 1

Relations:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parent | Child | RelationshipType | MaxCardinality | MinCardinality |
| CUSTOMER | CheckOut | ID-Dependent | 1:N | M:O |
| CUSTOMER | CUSTOMER\_ACCOUNT | ID-Dependent | 1:1 | M-M |
| BOOK | CheckOut | ID-Dependent | 1:N | M-O |

iii. Physical model and SQL



-- Created by Vertabelo (http://vertabelo.com)

-- Last modification date: 2020-06-02 18:50:58.066

-- tables

-- Table: BOOK

CREATE TABLE BOOK (

DeweyDecimalClassification integer NOT NULL CONSTRAINT BOOK\_pk PRIMARY KEY,

Title character(30) NOT NULL

);

-- Table: CUSTOMER

CREATE TABLE CUSTOMER (

CustomerID integer NOT NULL CONSTRAINT CUSTOMER\_pk PRIMARY KEY,

LastName character(35) NOT NULL,

FirstName character(35) NOT NULL,

Address character(35) NOT NULL,

Phone character(12) NOT NULL,

EmailAddress character(40) NOT NULL,

Zip character(6) NOT NULL

);

-- Table: CUSTOMER\_ACCOUNT

CREATE TABLE CUSTOMER\_ACCOUNT (

CUSTOMER\_CustomerID integer NOT NULL CONSTRAINT CUSTOMER\_ACCOUNT\_pk PRIMARY KEY,

CurrentFineAmount numeric NOT NULL,

TotalFineAmount numeric NOT NULL,

AmountPaid numeric NOT NULL,

CONSTRAINT CUSTOMER\_ACCOUNT\_CUSTOMER FOREIGN KEY (CUSTOMER\_CustomerID)

REFERENCES CUSTOMER (CustomerID)

);

-- Table: CheckOut

CREATE TABLE CheckOut (

CUSTOMER\_CustomerID integer NOT NULL,

BOOK\_DeweyDecimalClassification integer NOT NULL,

CheckOutDate date NOT NULL,

DueDate date NOT NULL,

DateReturned date NOT NULL,

CONSTRAINT CheckOut\_pk PRIMARY KEY (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification),

CONSTRAINT CheckOut\_BOOK FOREIGN KEY (BOOK\_DeweyDecimalClassification)

REFERENCES BOOK (DeweyDecimalClassification),

CONSTRAINT CheckOut\_CUSTOMER FOREIGN KEY (CUSTOMER\_CustomerID)

REFERENCES CUSTOMER (CustomerID)

);

-- End of file.

# IV. Database Prototype

**A.**

CREATE TABLE BOOK

(

DeweyDecimalClassification Int NOT NULL,

Title char(30) NOT NULL,

CONSTRAINT BOOK\_PK Primary Key(DeweyDecimalClassification)

);

CREATE TABLE CUSTOMER

(

CustomerID Int NOT NULL,

LastName char(35) NOT NULL,

FirstName char(35) NOT NULL,

Address char(35) NOT NULL,

Phone char(12) NOT NULL,

EmailAddress char(40) NOT NULL,

Zip char(6) NOT NULL,

CONSTRAINT CUSTOMER\_PK PRIMARY KEY(CustomerID)

);

CREATE TABLE CUSTOMER\_ACCOUNT

(

CUSTOMER\_CustomerID Int NOT NULL,

CurrentFineAmount numeric NOT NULL,

TotalFineAmount numeric NOT NULL,

AmountPaid numeric NOT NULL,

CONSTRAINT CUSTOMER\_ACCOUNT\_CUSTOMER\_FK FOREIGN KEY (CUSTOMER\_CustomerID) REFERENCES CUSTOMER (CustomerID)

);

CREATE TABLE CHECKOUT (

CUSTOMER\_CustomerID Int NOT NULL,

BOOK\_DeweyDecimalClassification Int NOT NULL,

CheckOutDate date NOT NULL,

DueDate date NOT NULL,

DateReturned date NOT NULL,

CONSTRAINT CHECKOUT\_PK PRIMARY KEY (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification),

CONSTRAINT CheckOut\_BOOK\_FK FOREIGN KEY (BOOK\_DeweyDecimalClassification)

REFERENCES BOOK (DeweyDecimalClassification),

CONSTRAINT CheckOut\_CUSTOMER\_FK FOREIGN KEY (CUSTOMER\_CustomerID)

REFERENCES CUSTOMER (CustomerID)

);

**B.**

INSERT INTO CUSTOMER(LastName,FirstName,Address,Phone,EmailAddress,Zip)

VALUES('Vazques','Franco','830 North Street','344-536-2369','Franco@mail.com','60023');

INSERT INTO CUSTOMER(LastName,FirstName,Address,Phone,EmailAddress,Zip)

VALUES('Valenzuela','Mohamed','920 South Driver','346-267-1359','Mohamed@mail.com','60052');

INSERT INTO CUSTOMER(LastName,FirstName,Address,Phone,EmailAddress,Zip)

VALUES('Cobb','Janelle','394 North Drive','947-528-9812','Janelle@mail.com','60024');

INSERT INTO CUSTOMER(LastName,FirstName,Address,Phone,EmailAddress,Zip)

VALUES('Kavanagh','Liyana','729 South Street','937-492-2234','Liyana@mail.com','60051');

INSERT INTO CUSTOMER(LastName,FirstName,Address,Phone,EmailAddress,Zip)

VALUES('Shamas','Vo','029 East Street','829-329-4928','Shamas@mail.com','60092');

INSERT INTO CUSTOMER(LastName,FirstName,Address,Phone,EmailAddress,Zip)

VALUES('Jamila','Neale','610 West Drive','643-324-1348','Jamila@mail.com','60086');

INSERT INTO CUSTOMER(LastName,FirstName,Address,Phone,EmailAddress,Zip)

VALUES('Connelly','Rui','204 South Drive','108-235-2309','Rui@mail.com','60052');

INSERT INTO CUSTOMER(LastName,FirstName,Address,Phone,EmailAddress,Zip)

VALUES('Irvine','Trinity','933 West Street','082-920-2391','Trinity@mail.com','60077');

INSERT INTO CUSTOMER(LastName,FirstName,Address,Phone,EmailAddress,Zip)

VALUES('Bonner','Raveena','439 West Drive','239-962-3210','Raveena@mail.com','60078');

INSERT INTO BOOK(Title)

VALUES('The Ballad of Songbirds and Snake’);

INSERT INTO BOOK(Title)

VALUES('Untamed');

INSERT INTO BOOK(Title)

VALUES('Where the Crawdads Sing');

INSERT INTO BOOK(Title)

VALUES('My First Learn to Write Workbook');

INSERT INTO BOOK(Title)

VALUES('Midnight Sun');

INSERT INTO BOOK(Title)

VALUES('Relationship Goals');

INSERT INTO BOOK(Title)

VALUES('Normal People');

INSERT INTO BOOK(Title)

VALUES('Plague of Corruption');

INSERT INTO BOOK(Title)

VALUES('Camino Winds');

INSERT INTO BOOK(Title)

VALUES('School Zone');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(1,1,'3/12/2019','3/26/2019','3/21/2019');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(2,2,'6/03/2019','6/17/2019','6/10/2019');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(3,3,'4/05/2019','4/19/2019','4/28/2019');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(4,4,'4/26/2018','5/10/2018','6/1/2018');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(5,5,'9/04/2018','9/18/2018','9/28/2018');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(6,6,'7/17/2019','7/31/2019','7/31/2019');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(7,7,'1/05/2020','1/19/2020','2/05/2020');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(8,8,'8/12/2018','8/26/2018','8/19/2018');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(9,9,'11/01/2019','11/15/2019','11/20/2019');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(4,2,'5/04/2018','5/18/2018','5/31/2018');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(7,10,'8/03/2019','8/17/2019','8/24/2019');

INSERT INTO CHECKOUT (CUSTOMER\_CustomerID,BOOK\_DeweyDecimalClassification,CheckOutDate,DueDate,DateReturned)

VALUES(6,5,'10/04/2018','10/18/2018','10/19/2018');

INSERT INTO CUSTOMER\_ACCOUNT

(CUSTOMER\_CustomerID,CurrentFineAmount,TotalFineAmount,AmountPaid)

VALUES(1,0,0,0);

INSERT INTO CUSTOMER\_ACCOUNT

(CUSTOMER\_CustomerID,CurrentFineAmount,TotalFineAmount,AmountPaid)

VALUES(2,0,0,0);

INSERT INTO CUSTOMER\_ACCOUNT

(CUSTOMER\_CustomerID,CurrentFineAmount,TotalFineAmount,AmountPaid)

VALUES(3,0,0,0);

INSERT INTO CUSTOMER\_ACCOUNT

(CUSTOMER\_CustomerID,CurrentFineAmount,TotalFineAmount,AmountPaid)

VALUES(4,0,0,0);

INSERT INTO CUSTOMER\_ACCOUNT

(CUSTOMER\_CustomerID,CurrentFineAmount,TotalFineAmount,AmountPaid)

VALUES(5,0,0,0);

INSERT INTO CUSTOMER\_ACCOUNT

(CUSTOMER\_CustomerID,CurrentFineAmount,TotalFineAmount,AmountPaid)

VALUES(6,0,0,0);

INSERT INTO CUSTOMER\_ACCOUNT

(CUSTOMER\_CustomerID,CurrentFineAmount,TotalFineAmount,AmountPaid)

VALUES(7,0,0,0);

INSERT INTO CUSTOMER\_ACCOUNT

(CUSTOMER\_CustomerID,CurrentFineAmount,TotalFineAmount,AmountPaid)

VALUES(8,0,0,0);

INSERT INTO CUSTOMER\_ACCOUNT

(CUSTOMER\_CustomerID,CurrentFineAmount,TotalFineAmount,AmountPaid)

VALUES(9,0,0,0);

ALTER TABLE CUSTOMER\_ACCOUNT

DROP CONSTRAINT CUSTOMER\_ACCOUNT\_CUSTOMER\_FK;

UPDATE CUSTOMER\_ACCOUNT SET CurrentFineAmount = CurrentFineAmonut + 9,TotalFineAmount = TotalFineAmount + 9

WHERE CUSTOMER\_CustomerID = 3;

UPDATE CUSTOMER\_ACCOUNT SET CurrentFineAmount = CurrentFineAmount + 13,TotalFineAmount = TotalFineAmount + 13

WHERE CUSTOMER\_CustomerID = 4;

UPDATE CUSTOMER\_ACCOUNT SET CurrentFineAmount = CurrentFineAmount + 22,TotalFineAmount = TotalFineAmount + 22

WHERE CUSTOMER\_CustomerID = 4;

UPDATE CUSTOMER\_ACCOUNT SET CurrentFineAmount = CurrentFineAmount + 10,TotalFineAmount = TotalFineAmount + 10

WHERE CUSTOMER\_CustomerID = 5;

UPDATE CUSTOMER\_ACCOUNT SET CurrentFineAmount = CurrentFineAmount + 1,TotalFineAmount = TotalFineAmount + 11

WHERE CUSTOMER\_CustomerID = 6;

UPDATE CUSTOMER\_ACCOUNT SET CurrentFineAmount = CurrentFineAmount + 17,TotalFineAmount = TotalFineAmount + 17

WHERE CUSTOMER\_CustomerID = 7;

UPDATE CUSTOMER\_ACCOUNT SET CurrentFineAmount = CurrentFineAmount + 7,TotalFineAmount = TotalFineAmount + 7

WHERE CUSTOMER\_CustomerID = 7;

UPDATE CUSTOMER\_ACCOUNT SET CurrentFineAmount = CurrentFineAmount + 5,TotalFineAmount = TotalFineAmount + 5

WHERE CUSTOMER\_CustomerID = 9;

**C.**

The prototype is for a library to keep track of when a customer registers with the library by entering the customer information into the CUSTOMER table. The prototype can also be used to keep track of fines on a customer. The book the customer checks out and the due date is also kept track of in the database prototype. This prototype is one where customers have not paid any fines.

EQUI-JOIN:

SELECT \*

FROM CUSTOMER INNER JOIN CUSTOMER\_ACCOUNT ON CUSTOMER.CustomerID = CUSTOMER\_ACCOUNT.CUSTOMER\_CustomerID;

| **EQUI-JOIN** | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** | **LastName** | **FirstName** | **Address** | **Phone** | **EmailAddress** | **Zip** | **CUSTOMER\_CustomerID** | **CurrentFineAmount** | **TotalFineAmount** | **AmountPaid** |
| 1 | Vazques | Franco | 830 North Street | 344-536-2369 | Franco@mail.com | 60023 | 1 | 0 | 0 | 0 |
| 2 | Valenzuela | Mohamed | 920 South Driver | 346-267-1359 | Mohamed@mail.com | 60052 | 2 | 0 | 0 | 0 |
| 3 | Cobb | Janelle | 394 North Drive | 947-528-9812 | Janelle@mail.com | 60024 | 3 | 9 | 9 | 0 |
| 4 | Kavanagh | Liyana | 729 South Street | 937-492-2234 | Liyana@mail.com | 60051 | 4 | 35 | 35 | 0 |
| 5 | Shamas | Vo | 029 East Street | 829-329-4928 | Shamas@mail.com | 60092 | 5 | 10 | 10 | 0 |
| 6 | Jamila | Neale | 610 West Drive | 643-324-1348 | Jamila@mail.com | 60086 | 6 | 1 | 11 | 0 |
| 7 | Connelly | Rui | 204 South Drive | 108-235-2309 | Rui@mail.com | 60052 | 7 | 7 | 7 | 0 |
| 8 | Irvine | Trinity | 933 West Street | 082-920-2391 | Trinity@mail.com | 60077 | 8 | 0 | 0 | 0 |
| 9 | Bonner | Raveena | 439 West Drive | 239-962-3210 | Raveena@mail.com | 60078 | 9 | 5 | 5 | 0 |

INNER-JOIN:

SELECT \*

FROM CHECKOUT, CUSTOMER

WHERE CUSTOMER.CustomerID = CHECKOUT.CUSTOMER\_CustomerID;

| **INNER-JOIN** | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CUSTOMER\_CustomerID** | **BOOK\_DeweyDecimalClassification** | **CheckOutDate** | **DueDate** | **DateReturned** | **CustomerID** | **LastName** | **FirstName** | **Address** | **Phone** | **EmailAddress** | **Zip** |
| 1 | 1 | 3/12/2019 | 3/26/2019 | 3/21/2019 | 1 | Vazques | Franco | 830 North Street | 344-536-2369 | Franco@mail.com | 60023 |
| 2 | 2 | 6/3/2019 | 6/17/2019 | 6/10/2019 | 2 | Valenzuela | Mohamed | 920 South Driver | 346-267-1359 | Mohamed@mail.com | 60052 |
| 3 | 3 | 4/5/2019 | 4/19/2019 | 4/28/2019 | 3 | Cobb | Janelle | 394 North Drive | 947-528-9812 | Janelle@mail.com | 60024 |
| 4 | 4 | 4/26/2018 | 5/10/2018 | 6/1/2018 | 4 | Kavanagh | Liyana | 729 South Street | 937-492-2234 | Liyana@mail.com | 60051 |
| 4 | 2 | 5/4/2018 | 5/18/2018 | 5/31/2018 | 4 | Kavanagh | Liyana | 729 South Street | 937-492-2234 | Liyana@mail.com | 60051 |
| 5 | 5 | 9/4/2018 | 9/18/2018 | 9/28/2018 | 5 | Shamas | Vo | 029 East Street | 829-329-4928 | Shamas@mail.com | 60092 |
| 6 | 6 | 7/17/2019 | 7/31/2019 | 7/31/2019 | 6 | Jamila | Neale | 610 West Drive | 643-324-1348 | Jamila@mail.com | 60086 |
| 6 | 5 | 10/4/2018 | 10/18/2018 | 10/19/2018 | 6 | Jamila | Neale | 610 West Drive | 643-324-1348 | Jamila@mail.com | 60086 |
| 7 | 7 | 1/5/2020 | 1/19/2020 | 2/5/2020 | 7 | Connelly | Rui | 204 South Drive | 108-235-2309 | Rui@mail.com | 60052 |
| 7 | 10 | 8/3/2019 | 8/17/2019 | 8/24/2019 | 7 | Connelly | Rui | 204 South Drive | 108-235-2309 | Rui@mail.com | 60052 |
| 8 | 8 | 8/12/2018 | 8/26/2018 | 8/19/2018 | 8 | Irvine | Trinity | 933 West Street | 082-920-2391 | Trinity@mail.com | 60077 |
| 9 | 9 | 11/1/2019 | 11/15/2019 | 11/20/2019 | 9 | Bonner | Raveena | 439 West Drive | 239-962-3210 | Raveena@mail.com | 60078 |

CUSTOMERS WHO RETURNED BOOK ON TIME:

SELECT CheckOutDate,DueDate,DateReturned,LastName,FirstName

FROM CHECKOUT INNER JOIN CUSTOMER ON CHECKOUT.CUSTOMER\_CustomerID = CUSTOMER.CustomerID

WHERE DueDate >= DateReturned;

| **CUSTOMERS-WHO-RETURNED-BOOK-ON-TIME** | | | | |
| --- | --- | --- | --- | --- |
| **CheckOutDate** | **DueDate** | **DateReturned** | **LastName** | **FirstName** |
| 3/12/2019 | 3/26/2019 | 3/21/2019 | Vazques | Franco |
| 6/3/2019 | 6/17/2019 | 6/10/2019 | Valenzuela | Mohamed |
| 7/17/2019 | 7/31/2019 | 7/31/2019 | Jamila | Neale |
| 8/12/2018 | 8/26/2018 | 8/19/2018 | Irvine | Trinity |

CUSTOMERS WHO HAVE BEEN FINED:

SELECT LastName, FirstName

FROM CUSTOMER INNER JOIN CUSTOMER\_ACCOUNT ON CUSTOMER.CustomerID = CUSTOMER\_ACCOUNT.CUSTOMER\_CustomerID

WHERE TotalFineAmount > 0;

| **CUSTOMERS-WHO-HAVE-BEEN-FINED** | |
| --- | --- |
| **LastName** | **FirstName** |
| Cobb | Janelle |
| Kavanagh | Liyana |
| Shamas | Vo |
| Jamila | Neale |
| Connelly | Rui |
| Bonner | Raveena |

Title OF BOOK CUSTOMERS CHECKED OUT:

SELECT LastName,FirstName,Title

FROM (CUSTOMER INNER JOIN CHECKOUT ON CUSTOMER.CustomerID = CHECKOUT.CUSTOMER\_CustomerID)

INNER JOIN BOOK ON CHECKOUT.BOOK\_DeweyDecimalClassification = BOOK.DeweyDecimalClassification;

| **TITLE-OF-BOOK-CUSTOMERS-CHECKEDOUT** | | |
| --- | --- | --- |
| **LastName** | **FirstName** | **Title** |
| Vazques | Franco | The Ballad of Songbirds and Sn |
| Valenzuela | Mohamed | Untamed |
| Cobb | Janelle | Where the Crawdads Sing |
| Kavanagh | Liyana | My First Learn to Write Workbo |
| Shamas | Vo | Midnight Sun |
| Jamila | Neale | Relationship Goals |
| Connelly | Rui | Normal People |
| Irvine | Trinity | Plague of Corruption |
| Bonner | Raveena | Camino Winds |
| Kavanagh | Liyana | Untamed |
| Connelly | Rui | School Zone |
| Jamila | Neale | Midnight Sun |

V. Summary: Results and Recommendations

Overall, the project results were acceptable by the end. However, the project did have its problems, which makes it useable only in a limited situation and requires it to be improved if it was to be used by a library.

A. The DB Prototype

The modeling did work to keep track of customers who checked out one copy of a book, one time. The created database prototype could be useful; however, it’s use is limited because it only allows customers to check out a copy of a book once. The prototype also does not keep track of other items that libraries currently allow customers to rent such as DVDs. Along with that the prototype does not show in its BOOK table that the library can own more than one copy of a book because each TITLE in the BOOK table is unique on its own. The database prototype is only useful in its current situation if a library were okay with these problems.

B. Issues with the model   
What went wrong when the logical model and physical model were being designed is the CHECKOUT table and CUSTOMER\_ACCOUNT table both kept track of the books that a customer checked out. The initial thought process was to have the CHECKOUT table be used to keep track of books that were currently checked out and have the data moved to the CUSTOMER\_ACCOUNT when it was returned. The data would then be deleted from the CHECKOUT table and kept in the CUSTOMER\_ACCOUNT for the long term. This turned out to be unnecessary and the CHECKOUT table can be used for both purposes. The FINES table was then deleted, and the data was moved to the CUSTOMER\_ACCOUNT table. The CUSTOMER\_ACCOUNT was changed to keep track of the customer’s fine history instead of the checkout history. The model was fixed before the database prototype was created.  
C. Next Steps

The next step would be to redesign the database prototype to allow a customer to check out the same copy of a book more than once. This could be done by making the CheckOutDate column in the CHECKOUT table a primary key along with CUSTOMER\_CustomerID and BOOK\_DeweyDecimalClassification columns. If this is done then the CUSTOMER\_CustomerID could be made into only a foreign key because the same copy of a book cannot be checked out on the same date, therefore the composite primary key in the CHECKOUT table won’t have any duplicates. Another step would be to redesign the database prototype to allow customers to checkout DVDs, magazines, and other items. This could be done by adding a TYPE column to the checkout table and a separate table for DVDs, magazines, and other items. The TYPE along with the DeweyDecimalClassification would then be the composite primary key in the CHECKOUT table. The BOOK table also needs to show the same title of a book more than once but with a different DeweyDecimalClassification to show that the library holds more than one copy of a book.