Justin Phillips
John Walbert
CS340 fall 2019
10/19/19

Project Step 3 (Final Version)

Draft Feedback

1. Based on the current structure of the database, it is impossible to present many-to-many relationships between checkout and books and checkout and movies. You need to add an associative entity.

   Fixed by adding the relationship tables customer_books and customer_movies.

2. The datatype of phone number should be varchar. The disadvantages of INT here are: you cannot store a dash and the leading zero may be missed.

   Fixed by changing the data type of the phone number attribute to varchar.

3. Please specify the number of characters for all varchars.

   Fixed by specifying varchar(255) for all attributes using the varchar datatype.

Peer Reviews and Graders Feedback

David Swope

Looks good, nice and simple but seems to meet all reqs. I like it. One thing, you don't need it at all but it does kind of bug me you don't have "ISBN" as an attribute of books.

Although in a real library an ISBN would probably be useful, for simplicity purposes it will not be used in this DB.

Brandon Mai

This looks pretty good. Something that you could add would be a checkout date as an attribute for the checkout entity to go along with the due date, I think most libraries keep track of that.

Checkout date was added as part of the transaction entity.

Frannie Richert

Love the library theme - libraries are my happy place! Overall, good ideas and I like that you have kept it simple enough to actually be implemented. Something that my partner and I found is we over-complicated a lot at first, and had to pare down. This feels like a manageable set of tables. One item I think needs more clarification is a many to many relation b/t books and checkout and movies and checkout. In my eyes, in a true library setup, you would have distinct copies of books for a given book name. For example, you may have one book called "The Giver" but have 5 different copies of it. In this case, there wouldn't be a many to many relationship b/t books and checkout. Two customers cannot checkout the same copy of a book at a time. I think how you're setting it up, you're seeing "The Giver" as just book name, not individual copy, and therefore many people could have "The Giver" checked out. I would suggest making this more explicit in the outline. It also leads to some weirdness (for lack of a better word) with having "Checked Out" be a column in the Book/Move tables. I'm still not 100% convinced there is a many to many relationship there, but I think if you explain more what you're thinking in the outline for the final version especially in the relationship section, it would be more clear.

Although an individual book or movie cannot be checked out by more than one person at the same time, more than 1 book or movie can be checked out at any given time as well as there will be multiple checkout transactions. Additionally, the checkout history of movies / books will be stored for empirical data purposes.

Eric Sund

Really like the look of your project so far. Another interesting aspect that you could add would be something in the way of fines/penalties for those who fail to return their product by the due date.

While fines would be used in a real library setting, to keep from making the database unnecessarily complicated they will not be incorporated.

Haochong Pan

Hi Justin and John,

I really like the functionality your database has. Also, you guys did an excellent job of describing the details of each entity, relationships, and constraints. One thing I would recommend adding is use customer_id as a foreign key, so it can refer back to the person who rent the items.

The cust_movie and cust_book relationship tables contain the transaction number where the book or movie was checked out. From this transaction the customer can be determined as cardNum is a foreign key in the checkOut table.

Herbert Diaz

Hey Justin and John! Here's what I was able to catch after a brief read-through:

- ERD
    - This project requires at least 4 relationships - I see 3.

      Employee entity added to incorporate a one-to-many relationship with checkout entity.

- Schema
    - Use another color besides light gray - makes it hard to see with a white background.

      Schema lines updated to be blue.

    - Follow the notation described in the Week 3 Learn Module (where arrows point from foreign keys to primary keys).

      Schema lines updated to incorporate arrows per notation requirements.

- Additional Notes

    - I notice some redundancy which goes against database normalization.
        - For example, I see libMovie and cust_movie both have a "checked_out" attribute. This attribute shouldn't be stored in both.

      Checked_out attribute removed from both cust_book and cust_movie to remove redundancy. Additionally, removed cardNum from cust_book and cust_movie as that data is present in the checkout transaction.

    - Aside from that I wasn't able to spot anything else in my quick overview - be sure to double check that your entities, attributes, and relationships match your database outline.

Updated Outline

Overview

The project that we propose is a database that will handle the checkout process for a library. It will allow the library patron to view the available titles, check items out if they are available, and provide a due date for return.

Entities

The first entity that we will have is ***customers***. Their attributes will include library card number (a unique identifier), first name, last name and a telephone number. The second entity will be the ***checkout***. It is the transaction itself, where the customer borrows books or movies. Its attributes will include transaction ID (a unique identifier), checkout date and due date for return. The third entity will be ***books***. Their attributes will include author of the book, the title of the book, a unique ID and if it is already checked out or not. The fourth entity is ***movies***. Similar to books, their attributes are title, the year the movie was released, a unique ID and if it is checked out. The last entity is ***Employee***. The attributes for it will be employee ID, first name and last name. To support the many to many relationship between books / movies and checkout, the associative entities cust_movie and cust_books will be used.

Relationships

The relationships between these entities are defined as follows. For the first entity, the customer, it will have a one to many relationship with the checkout entity. An individual checkout transaction can only be performed by one customer, but an individual customer can have many

checkout transactions. The checkout entity will have a many to many relationship with both the books and movies entities. A checkout transaction can contain multiple books and/or movies, as well as each book or movie being associated with multiple checkout transactions. The customer will not have any direct relationship with the movies or the books. The employee entity will have a one to many relationship with the checkout entity. Many employees can perform checkouts but an individual checkout will only be associated with one employee.

Constraints

Each entity attribute will have defined data types. For customers the library card number will be an auto-incrementing integer, the first and last names as well as phone number will be varchar(255). The primary key for the customer will be the library card number this will provide a unique identifier for each customer. For the checkout entity, the transaction ID will be an auto-incrementing integer and also the primary key for that entity. It will also server as a foreign key between the checkout entity and the customer to allow tracking of checkout transaction to a particular customer. The due date and check out date will be will be stored as date type. For books and movies, the title will be varchar(255), checked out will be bool data type, and ID will be an auto-incrementing integer (also the primary key for both of those entities). Specifically for movies, the year of release will be stored as an int, and for books the author will be stored as varchar(255). Using these unique attributes the customer will be able to go select the movies and books they wish to checkout from a list that shows what is currently in stock. These items will be added to the checkout then will be assigned a due date and transaction id. After the customer finished the transaction, it will change the checked out attribute for the items that were borrowed from false to true, and will subsequently reflect this on additional searches. There will

be an Employee entity that will be linked to check out and handle the check out process. The employee id will be an auto-incrementing integer. The first and last names will be varchar(255) .

Entities : *Attributes*

Books : *Title, Author, ID, Checked Out*

Check Out : *Transaction Num, Due Date, Check Out Date, Card Num, Employee ID*

Customer : *Library Card Number, First Name, Last Name, Phone Number*

Movies : *Title, Year, ID, Checked Out*

Employee : *Employee ID, First Name, Last Name*

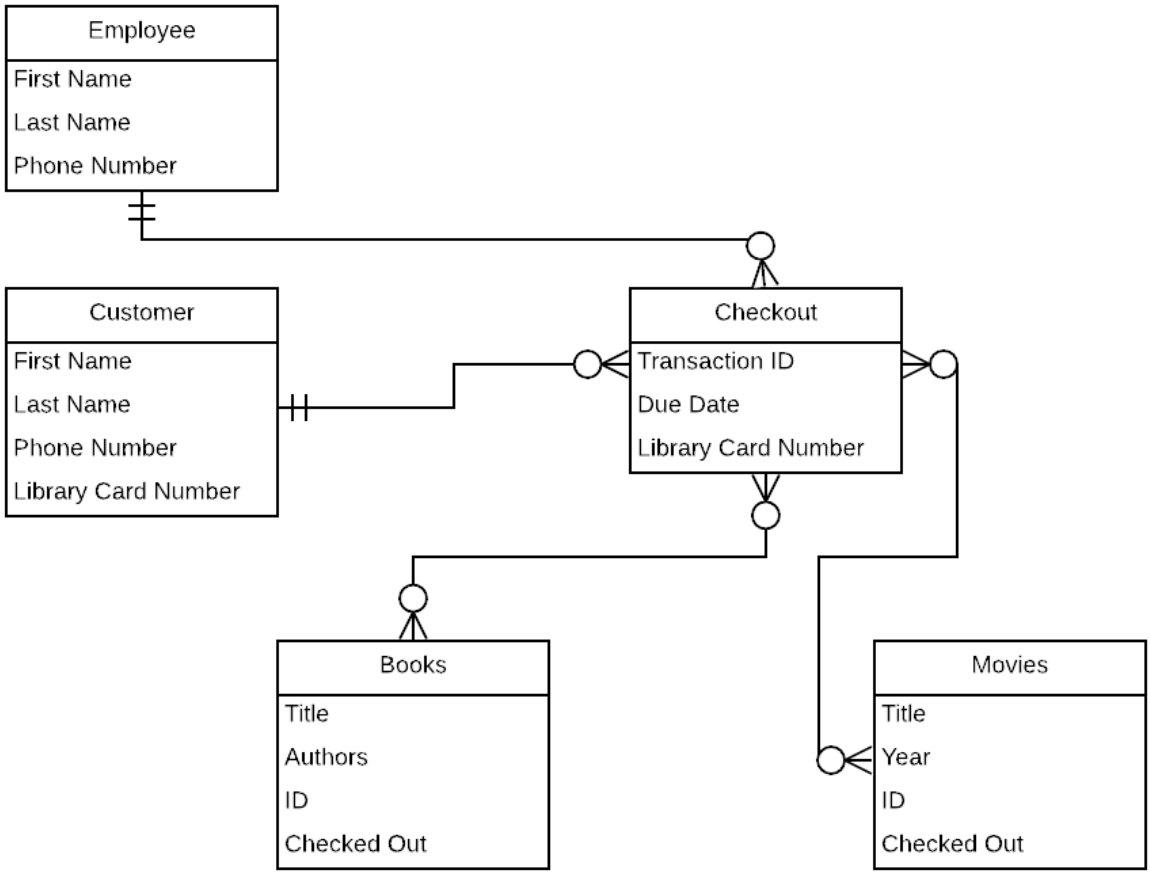
Constraints : *Attributes*

Primary key : *library card number, transaction ID, ID (books / movies), Employee ID*

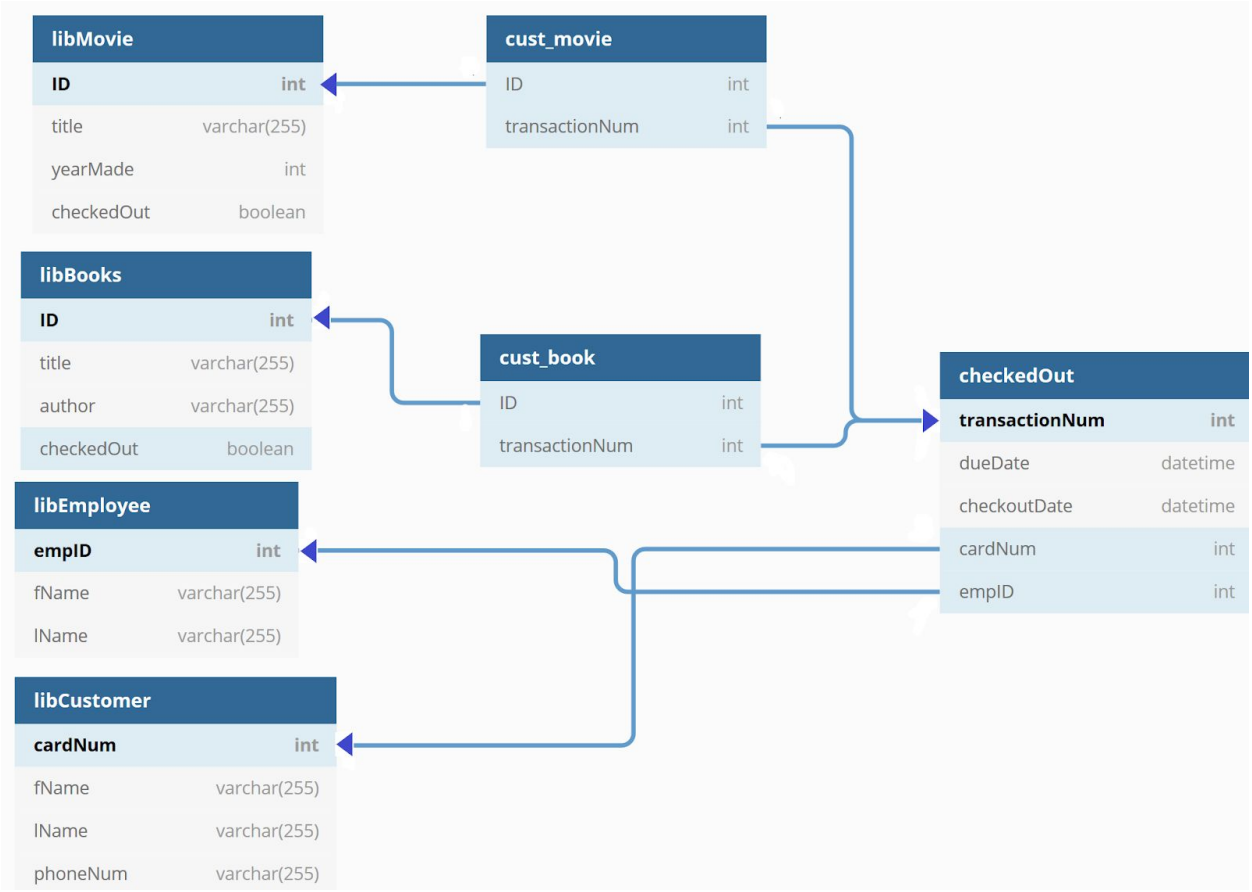
Not Null : *All attributes*

Foreign key : *transaction ID, Employee ID, ID (books), ID (movies)*

ER diagram


Employee
First Name
Last Name
Phone Number
Customer
First Name
Last Name
Phone Number
Library Card Number
Checkout
Transaction ID
Due Date
Library Card Number
Books
Title
Authors
ID
Checked Out
Movies
Title
Year
ID
Checked Out

Schema:


libMovie
ID int
title varchar(255)
yearMade int
checkedOut boolean
cust_movie
ID int
transactionNum int
libBooks
ID int
title varchar(255)
author varchar(255)
checkedOut boolean
cust_book
ID int
transactionNum int
checkedOut
transactionNum int
dueDate datetime
checkoutDate datetime
cardNum int
empID int
libEmployee
empID int
fName varchar(255)
lName varchar(255)
libCustomer
cardNum int
fName varchar(255)
lName varchar(255)
phoneNum varchar(255)