Library Database

StackOverflow

Benjamin Beuca
Bradley La
Taha Lahricr
Khuzaima Mushtaq

CSS475 - Databases
University of Washington Bothell
Autumn 2018

Table of Contents

1. Introduction

This document provides the overview of the team StackOverflow and our project the Library Database. Addition iterations of this document will include detailed description of the database and our progress on this project. This document will help our team stay on track with everything and will also help us not fall behind.

2. Divided Tasks

As of now, we have decided to work on all aspects of this project collectively in order to enhance our learning and progress. There may be some tasks that will be divided amongst us, and we will update this as soon as we run into them.

- Design
  - Entity Relationship Diagram
    - Benjamin
    - Taha
    - Bradley
    - Khuzaima
  - Relational Model
    - Benjamin
    - Taha
    - Bradley
    - Khuzaima
- Documentation
  - Benjamin
  - Taha
  - Bradley
  - Khuzaima
- Implementation
  - Creating Database
    - Benjamin
  - Inserting Data
    - Bradley
    - Khuzaima
  - Constraints
    - Benjamin
    - Khuzaima
  - User Interface

        - Benjamin
- Presentation
    - Benjamin
    - Taha
    - Bradley
    - Khuzaima

3. Database Specifications

Our proposed database is a Library Database. This database will be designed to store the many aspects a library circulates, The many users/Patrons of the Library and their interactions with the library materials. As such we will have the following aspects in our database. (Look at User Requirements for more details)

4. User Requirements

- Library System
    - Has Libraries
    - Name, County/City,
- Library
    - Library will have a <u>Location</u>, <u>Name</u>, Phone Number,
    - Holds Inventory
    - Staff lends Inventory to Patrons
    - Has Staff
- Inventory (Weak Entity)
    - How many copies of Media does Library have?
    - We will have multiple inventories. Depending on weather the user wants to buy a monthly subscription or not.
- Author
    - Has books
    - Will have Name(First Name, Last Name), Bio, Date of Birth, SSN
- Book
    - Has <u>Author</u>, Publisher, Date Released, <u>Name</u>, <u>Edition,</u>Synopsis
- DvD's
    - Has Authors, Publishers, Actors, Date Released, Name,
- Patrons (customers?)
    - Borrowers/ users/ etc.
    - Ratings for Inventory
    - Borrower Credit

- Staff
    - Name (First Name, Last Name), SSN, Phone Number, WorkID
    - Works at Library
    - Can set Borrowers status (allowed to borrow/ Fines/ Overdue etc)

5. Searching Abilities

- Searching for all materials published by an author
- Searching for materials specified by genre
- Searching for materials specified by name
- Searching for materials with ID
- Materials checked out by patrons
- How many of a specific Material are available in the library?

6. Implementation

This section outlines our planned progress for this project in regards to the deliverables and due dates. Our general approach to progressing on the project is to have weekly goals based on upcoming due dates as well as the incorporation of in class materials as we learn them.

Progress

- Iteration 0 (Done)
- Proposal (Done)
- Tooling (Done)
- Design Documentation (Done)
    - ER
    - RM
- Create Database (Done)
- Insert Data (Done)
- Finish UI (Done)
- Poster Presentation (Done)

Plan

- Creates/ Inserts (by 11/27)
- Constraint Checks (by 11/27)
- SQL queries (In-Progress)
- Iteration 3 by (11/27)
- Iteration 4 by (12/12)
- Poster Presentation by (12/12)

7. Entity Relationship Diagram

This is a updated version of our entity relationship diagram. It may change in the future when we start the implementation process. We noticed that we ended up having around 15 tables, so we might trim some things that are not needed in our database. Please see Table 1.1.

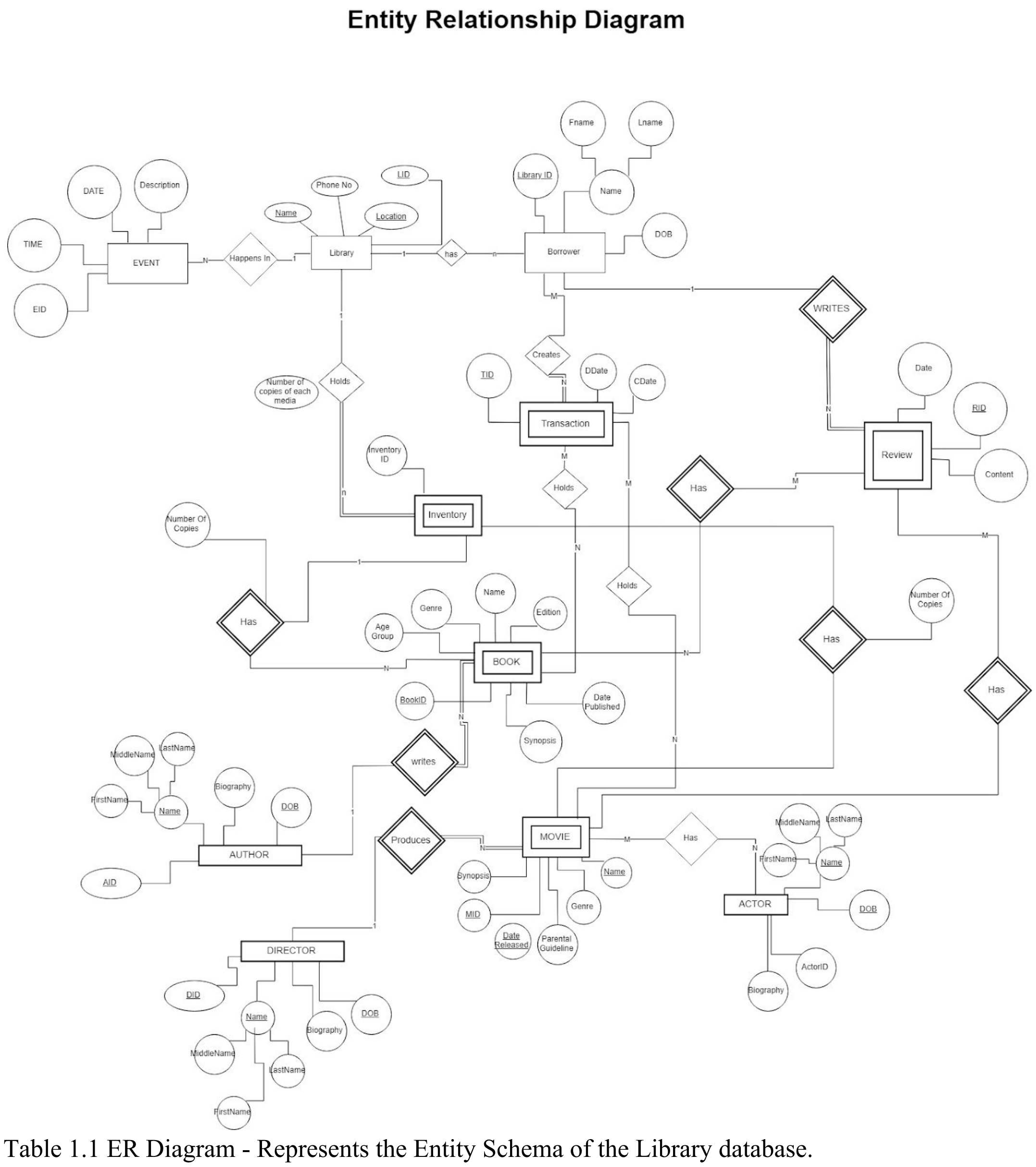
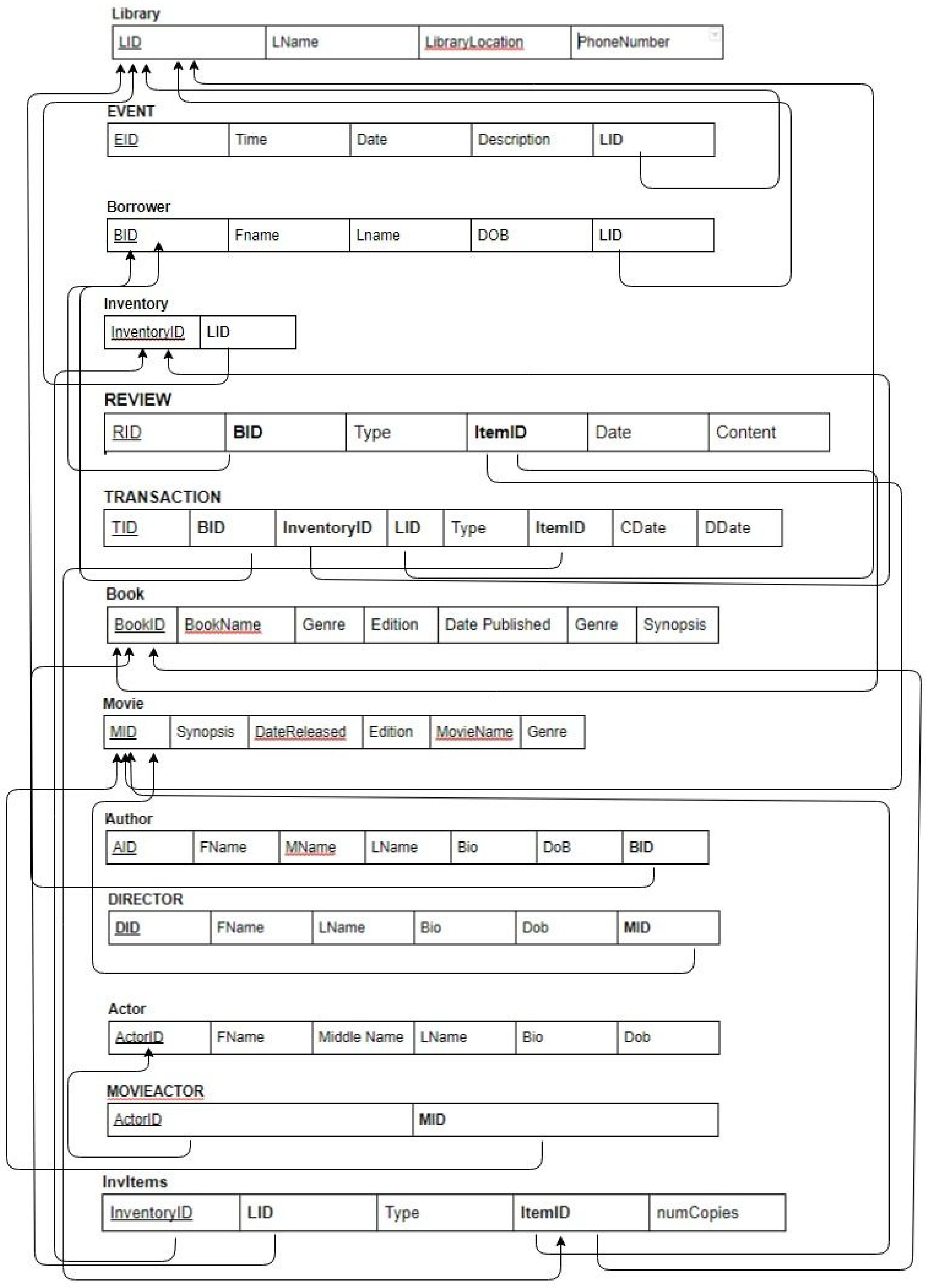


Table 1.1 ER Diagram - Represents the Entity Schema of the Library database.

8. Relational Model

This is our relational model which was built off of our Entity relational model. We ended up having many tables and we noticed that somethings are not needed in our design. We removed BOOKREVIEW, MOVIEREVIEW, BOOKTRANSACTION, MOVIETRANSACTION and added a Transaction table and a review table that is used by both Movie and Book.We also add a InvItems table. Please see Table 1.2.

RELATIONAL MODEL


Library
LID | LName | LibraryLocation | PhoneNumber
EVENT
EID | Time | Date | Description | LID
Borrower
BID | Fname | Lname | DOB | LID
Inventory
InventoryID | LID
REVIEW
RID | BID | Type | ItemID | Date | Content
TRANSACTION
TID | BID | InventoryID | LID | Type | ItemID | CDate | DDate
Book
BookID | BookName | Genre | Edition | Date Published | Genre | Synopsis
Movie
MID | Synopsis | DateReleased | Edition | MovieName | Genre
Author
AID | FName | MName | LName | Bio | DoB | BID
DIRECTOR
DID | FName | LName | Bio | Dob | MID
Actor
ActorID | FName | Middle Name | LName | Bio | Dob
MOVIEACTOR
ActorID | MID
InvItems
InventoryID | LID | Type | ItemID | numCopies

9. Normalization

Library

- LID -> LName
- LID -> Library Location
- LID->PhoneNumber
- LibraryLocation → PhoneNumber

Library

LID	LName	LibraryLocation	PhoneNumber

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Assuming that PhoneNumber is dependent on LibraryLocation, this would not be in 3NF because a non prime attribute is dependent on another non prime attribute. This is how you fix it.

Library

LID	**LName**	**LibraryLocation**

Location

LibraryLocation	**PhoneNumber**

Now, it is in 3NF

Event

- EID -> Date
- EID -> Description
- EID->LID

Event_s

EID	Date	Description	**LID**

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

Borrower

- BID -> Fname
- BID -> Lname
- BID ->DOB
- BID ->LID

Borrower

<u>BID</u>	Fname	Lname	DOB	**LID**

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

Inventory

- InventoryID -> LID

Inventory

<u>InventoryID</u>	**LID**

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

InvItems

- InventoryID -> LID
- InventoryID ->Type
- InventoryID -> ItemID
- InventoryID -> numcopies

InvItems

InventoryID	**LID**	Type	**ItemID**	numCopies

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

Reviews

- RID->BID
- RID ->Type
- RID> ItemID
- RID-> Date
- RID-> Content

REVIEW

RID	**BID**	Type	**ItemID**	Date	Content

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

Transactions

- TID->BID
- TID->InventoryID
- TID->LID
- TID->Type
- TID->ItemID
- TID->CDate
- TID->DDate

TRANSACTION

TID	**BID**	**InventoryID**	**LID**	Type	**ItemID**	CDate	DDate

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

Author

- AID->FName
- AID->MName
- AID->LName
- AID->Bio
- AID->DoB
- AID->BID

Author

<u>AID</u>	FName	MName	LName	Bio	DoB	**BID**

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

Book

- BookID->BookName
- BookID->Genre
- BookID->Edition
- BookID->DatePublished
- BookID->Synopsis

Book

<u>BookID</u>	BookName	Genre	Edition	Date Published	Synopsis

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

Director

- DID->FName
- DID->LName
- DID->Bio
- DID->Dob
- DID->MID
- **DIRECTOR**

DID	FName	LName	Bio	Dob	**MID**

-
- 1NF
- Yes, because it has only single atomic attributes
- 2NF
- Yes, because every non prime attribute is dependent on the primary key.
- 3NF
- Yes, because no non prime attributes are dependent on other non prime attributes

Movie

- MID->Synopsis
- MID->DateReleased
- MID->Edition
- MID->MovieName
- MID->Genre

Movie

MID	Synopsis	DateReleased	Edition	MovieName	Genre

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

Actor

- ActorID->FName
- ActorID->MName
- ActorID->LName
- ActorID->Bio
- ActorID->Dob

Actor

<u>ActorID</u>	FName	Middle Name	LName	Bio	Dob

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

MovieActor

- ActorID->MID

MOVIEACTOR

<u>ActorID</u>	**MID**

1NF

Yes, because it has only single atomic attributes

2NF

Yes, because every non prime attribute is dependent on the primary key.

3NF

Yes, because no non prime attributes are dependent on other non prime attributes

Table 1.2 RM Diagram - Represents the Relational Schema of the Library database.

10. Constraints

- Library_B
  - LID should be a unique integer and not a Null value
  - Library Name is a string of maximum 50 characters, and is not nullable
  - Library Location is a string with a maximum of 50 characters.
  - Phone number is a 10 digit integer that is non nullable and cannot be negative.

- Inventory
  - InventoryID can only be a unique integer that is non nullable
  - LID should be a unique integer and not a Null value
  - LID must be an integer and must exist in Library_B, FK constraint
- Book
  - BookID must be an Integer
  - Genre can only be a String

    - Name can only be a string
    - Edition can only be an integer, positive and non nullable
    - Date published is using the date data type and is non nullable
    - Synopsis can only be a string that can be nullable
- Author
    - Fname is a string that cannot be null and has a max of 50 characters.
    - Lname is a string that cannot be null and has a max of 50 characters.
    - Mname is a string that can be null and has a max of 50 characters.
    - Bio is a string that can be null.
    - DOB is using the date data type and is non nullable
    - BookID must be an integer and must exist in Book, FK constraint
    - AID (AuthorID) must be unique and cannot be null
    -
- Movie
    - Synopsis should be a string and may be nullable
    - Date Released is using the date data type and is non nullable
    - Edition should only be a positive integer and may be nullable
    - Name can only be a string and non nullable
- Actor
    - Fname is a string that cannot be null and has a max of 50 characters.
    - Lname is a string that cannot be null and has a max of 50 characters.
    - Mname is a string that cannot be null and has a max of 50 characters.
    - Bio is a string that can be null.
    - DOB is a using the date data type non nullable.
- Borrower
    - Borrower ID is unique a non negative integer that can not be null
    - Fname is a string that cannot be null and has a max of 50 characters.
    - Lname is a string that cannot be null and has a max of 50 characters.
    - DOB is using the date data type and is non nullable
    - LID must be an integer and must exist in Library_B, FK constraint
- Director
    - DID should be a unique integer and not a Null value
    - FName should be a string, not NULL and a max of 50 characters
    - LName should be a string, not NULL and a max of 50 characters
    - Bio should be a string, not NULL and could contain the maximum amount of characters
    - DOB is using the date data type and is non nullable
    - MID should be a unique integer and not a Null value
    - MID must be an integer and must exist in Movie, FK constraint

- MovieActor
  - ActorID must exist in Actor, FK constraint
  - MID must exist in Movie, FK constraint
- Event_S
  - EID should be a unique integer and not a Null value
  - Date_Time is using the DATETIME data type and is non nullable
  - Description should be a string, not NULL and could contain the maximum amount of characters
  - LibID must be an integer and must exist in Library_B, FK constraint
- InvItems
  - InventoryID must be an integer and must exist in Inventory, FK constraint
  - LID must be an integer and must exist in Library_B, FK constraint
  - Type should be either 0 or 1 (0 = Book, 1 = Movie)
  - ItemID must be an integer and must exist in Movie and Book, FK constraint
- Transaction
  - TID (Transaction ID) must be unique and the Primary Key
  - Type is an integer that can only be either 0 or 1 (0 = Book, 1 = Movie)
  - ItemID is an integer that must exist in either Book or Movie, FK constraints
  - CDate (check out date) is a date and cannot be null
  - DDate (Due Date) is a date, cannot be null, and must be after CDate
  - BID (Borrower ID) in an integer and must exist in Borrower, FK constraint
- Review
  - RID should be a unique integer that is not null
  - Type should be either 0 or 1 (0 = Book, 1 = Movie)
  - ItemID must be an integer and must exist in Movie, FK constraint
  - ItemID must be an integer and must exist in Book, FK constraint
  - Date is using the DATE data type and is non nullable
  - Content should be a string and could be the maximum number of characters and should not be null
  - BID (Borrower ID) must be an integer and must exist in Borrower, FK constraint

11. Tooling Assessment

- DBMS
    - SQL Server - Free and allows UI
        - We changed from MYSQL because we learned more about SQL Server during our research and it made more sense for use to use SQL Server. We also decided to make our UI on a local host so we learned more about it.
- User Interface
    - C#
    - Windows Forms
    - As we said earlier that we might change the way we use our User Interface. We all did more research in attempting to finish our project and we found that Windows Forms worked better for us. We decided to do everything through Visual Studios and Use SQL Server and C# to fully finish our Library database with a local UI.
    -
- Host
    - Local through SQL Server and Visual Studios
- Additional Tools
    - Draw.io - Diagram Creation
    - Discord - Communication
    - GITHUB - Version Control
    - Visual Studio - IDE
        - As we said earlier that we might change to Visual Studio, we did end up going with Visual Studio. We kept our options open and as we started to implement our project, we felt more comfortable going the Visual Studio route using C#, Windows Forms, and SQL Server to complete our project.

12. Tables Created in Project

The tables that we have created and have create statements for are the following. Please see the SQL file to see the code for each table.

- Actor
- Author
- Book
- Borrower
- Director
- Event_S
- Inventory

- Library_B
- Movie
- MovieActor
- Review
- Transaction

13. Methodology to Create Test Data

- Our initial thought when creating data was inputting random data, but upon further deliberation, in order to get results for even the simplest of queries we must have correlating data between the tables.
- http://www.generatedata.com/#generator
    - We found this website online to generate our data. This worked really well and was really convenient in creating data for us to test.
    - When generating data we have an easy to use interface much like visual studios that will generate code for us. It gave us options such as MySQL and SQL server. We were able to generate as much as 100 samples at a time. To generate more than 100 we would have to refresh the page and generate an additional 100. This was a fixed number due to the website wanting us to donate to have full access to its features. For certain data types such as date it had multiple types of formats ready to use.
    - There was no True and false statements so we just used 1 and 0 to represent booleans.
    - Different options were given to us for characters, strings and integers such as name, address, zip code, address, and custom lists have a select words for genre.

14. Inserts Showing Constraint Violation

- **INSERT INTO InvItems**([InventoryID],[LID],[Type],[ItemID],[NumCopies]) **VALUES**(1,1,5,1,1);
    - This returns an error because the third integer should only be a 0 or a 1 for type
- **INSERT INTO** Transactions([TID],[**TYPE**],[ItemID],[CDate],[DDate],[BID]) **VALUES**('Hello',1,1,'11/18/2019','09/01/2018',1);
    - This returns an error because a string was placed in the TID location where TID should only be an integer

- **INSERT INTO** Transactions([TID],[**TYPE**],[ItemID],[CDate],[DDate],[BID]) **VALUES**(1,1,1,'11/18/2019','Hello',1);
    - This would return an error because Hello is not a date
- **INSERT INTO** MovieActor([ActorID],[**MID**]) **VALUES**(1,21);
    - This returns an error because there is no MID that is 21
- **INSERT INTO** Review([RID],[**Type**],[ItemID],[Date],[**Content**],[BID]) **VALUES**(1,0,1,'02/07/2018', 7,1);
    - This would return an error because Content should be a paragraph not a number
- **INSERT INTO Book([BookID],[BookName],[Genre],[Edition],[DatePublished],[Synopsis]) VALUES(1,'Hiroko',NULL,6,'09/18/2018','Praesent luctus. Curabitur egestas nunc sed libero. Proin sed turpis');**
    - This returns an error because genre can not be NULL

15. Source Code And Instructions to Run

Our source code is uploaded in a separate zipped Visual Studio file. There will be a zipped folder Library1.zip within a zipped folder Project Iteration 4.zip. It runs on a local server. In order to run the code, you need to unzip the file and have visual studio on you computer to open it. Click on Project Iteration 4.zip → Library1.zip→ Library1.sln and it should open the project in Visual Studio. Once you open the project in Visual Studio, you need to press start and it should create tables, insert data into tables, and run our User Interface. In the User Interface, you have 3 buttons. Add, remove, and update. You can see the LibraryID and Library name in 2 list boxes on the left. You can add, remove, and update using the Name, and ID text boxes to input data. The other 4 List boxes are more complicated queries that are hard coded do to difficult computations. frmMain.cs is our c# code for the User Interface, and SQLQuery1.sql is our cql code.

16. Queries

SQL Statement	**Purpose**
SELECT ACTOR.First FROM Actor FULL JOIN MOVIEACTOR ON	Shows All actors that worked on a given movie by name

ACTOR.ActorID = MOVIEACTOR.ActorID
WHERE MOVIEACTOR.MID IN
(SELECT MOVIE.MID FROM MOVIE
WHERE MOVIE.MovieName like 'mi');
	Uses 3 Relations: ACTOR, MOVIEACTOR and MOVIE.
	Full Join on MovieActor and nested query on Movie

SELECT Book.BookName FROM Book
WHERE Book.BookID IN(SELECT ItemID
FROM Transactions
FULL JOIN Borrower
ON Borrower.BID = Transactions.BID
WHERE Borrower.Fname like 'Halla' AND
transactions.Type77 = 0);
	Shows all the books checked out by Borrower Named "x"
	Uses 3 Relations: Book, Transactions, Borrower
	Full Join on Borrower and nested query on Transactions

select m.MovieName from Movie m where Genre = 'Action'	Retrieves All the movies by genre "x" in this case its Action
select b.BookName from Review r, Book b where r.ItemID = b.BookID and r.Rating > 7	Gets Higher rated books from the database. Rating is from 1-10

SELECT DISTINCT Book. BookName
FROM LIBRARY, INVITEMS, BOOK
Where LIBRARY. LID == INVITEMS.LID
AND BOOK.ItemID == INVITEMS.ITemID
AND INVITEMS.Type == 0;
	How many of a specific Material are available in the library?

SELECT Book.BookName
FROM Book, Author
WHERE Book.BookID == Author.BookID
AND Author.AID == 12;
	Shows all the books published by author, with AuthorID == "x" ;

SELECT AVG(REVIEW.Rating) as
AverageRating
FROM Review
Where ItemID == 'x'
GROUP BY Review.Rating;
	Average Reviews for Item X

17. Project evaluation

- **Successes**
  - Getting complex and simple queries to work with our database. We were able to make the database by learning how to use C# thru Microsoft Visual Studios. The RM and ER had to be redone several times during each iteration and as we worked on the database and got feedback the final version came out working quite well.
- **Issues**
  - Using our RM and moving it to SQL in visual studios. There was many foreign key errors and we had to change our RM several times. When writing queries we had to create several FK constraints.
  - One of the issues that we ran into was the scale of our database. With 14 different relations and many complex relationships between them our time was mostly spent getting the database working correctly and we ran out of time on the UI and some other aspects of this project.
- **What we could have done to improve.**
  - If I had one more week, I would work more on our User Interface and trying to connect it to the cloud using Azure. This would be a great to know. We actually plan to continue our project and try to figure out how to connect to Azure because we have always wanted to learn it.
- **Time Spent**
  - We spent quite a bit of time on our ER and RM diagrams because without having the correct diagrams, we cannot continue to create our project. What we did do well was the data creation and the queries for our UI. This ended up becoming more successful than anticipated. When generating data there were several cases of trial and error and we had to reset a few times are each table. The process took several hours.
- **What we would do better**
  - Since we started the er and rm when we had only first learnt it I think if we were to do it again we would use the experience from this time to really knock the design out of the park and make it perfect including normalization from the start before any sql.

    - We would host the database on azure so that we can all have remote access and implement a more visually appealing UI
    - With better understanding of constraints and how to make sql tables from an RM we would ensure that all constraint requirements are met.
- **Testing**
    - Using our data generation that we described above we tested the data base to make sure that no constraints are being violated.
    - We generated random data because for our testing purposes it does not matter if a books name is a real name or just a random string of characters
    - We insured correctness by using our sql queries to verify correct behaviors

18. References

"Generatedata.com." *Generatedata.com*, Generatedata, 30 July 2017, www.generatedata.com/#t2.