GROUP-36
Dummy Library Management-COL362

Md Asif Anwar (2017TT10922)
Minhaj Shakeel (2016CS10354)
Mohammad Zia Kamran (2017TT10925)

Due date: March 1, 2020, 11:55pm IST

1 Section 1

We have developed a dummy library management system for the project.The motivation behind to choose this idea was the clarity of the specifications and design of the project.
In this system there's two kind of users
(1) Student
(2) Admin
Both the users have login and logout system. If the new user(student) wants to login then user has to first sign-up and then wait for confirmation by admin. After confirmation they can login and search for the book.

1.1 Student

Student need to login first to see the book availability. Student can search the book if they remember a keyword, title, bibliography number etc of the book. They have profile section where they can see their checkout history, return history and their profile. Our system keep track of user data, and which user is login or not.
We have categorised student in 3 category(or type_account)
(1) UG Student has fine limit of 20, can issue at most 5 book at a time and for 7 days max.
(2) M.tech Student has fine limit of 40, can issue at most 9 book at a time and for 14 days max.
(3) PhD Student has fine limit of 60, can issue at most 15 book at a time and for 21 days max.
Our system keeps track of user statistics, including every single checkout, checkin history, fine, currently number of book issued by user, and this is made available for that user and admin only.

1.2 Admin

Admin also need to login first to access all the features of dummy library. After login admin can see their profile where admin can see their checkout and check in history, option to return the book by entry number, see the history of user i.e. number of book issued to that user, and option to issue a new book.Only admin have access give the user access.
Our dummy library already handle all the fine issues, number of book a student can take at a time etc. If users exceeds the fine or max number of book then it will automatically suggest that user has exceeded the limit.

Table	No of Tuples	Raw Datasize	Clean Datasize
Library_Collection	1,37,389	1 gb	46.1 mb
userdetails	5484	400 kb	400 kb
checkin_data	1098	46 kb	46 kb
checkouts_data	3512	270 kb	270 kb
pending	3	∼1 kb	∼1 kb
admindetails	4	∼1 kb	∼1 kb
booklimit	3	∼1 kb	∼1 kb

Table 2. Data Statistics

2 Section 2

Entity	Attributes
ADMIN-DETAILS	Username,Name, Email, Passwords, Contact
CHECKINDATA	UserId , BookId, issue_date, admin_issued, due_date, return date, admin_returned
CHECKOUTS-DATA	UserId, BookId, issue_date, admin_issued, due_date
BOOKLIMIT	type, Number Of Books, Maximum Fine, Maximum Days
LIBRARYCOLLECTION	id, bibnum, title, author, isbn, publicationyear, publisher, subject, itemtype, itemcollection, floatingItem, itemlocation, reportdate, itemcount
PENDING	UserId, Name, Password, Contact, Account Type, Books Issued, Fine
USERDETAILS	userId, Name, Password, Contact, Account Type, Books Issued, Fine

Table 1. **List Of all Entity and Attributes**

2.1

Our primary source of data is Seattle library checkout data present on kaggle link to the dataset is:-
https://www.kaggle.com/seattle-public-library/seattle-library-checkout-records#Library_Collection_Inventory.csv

2.2

We downloaded the library inventory list from this dataset and cleaned it up in order to remove null values. We used hostel students for user accounts. We generated randomly the user checkout data because the checkout data present in the above described dataset was insufficient.

2.3

Removed null values from the inventory dataset.

3 Section 3

3.1 User's View of the system

- **Welcome Page**
  - The home page contains 3 major options; **login, signup and Admin login**. An admin can directly login on this page (which leads to the admin profile home page). Student can click on either of the signup/login links.
  - On pressing login, a page open where user can login
  - On pressing signup a page open where user enter their details to signup for the library.
- **Login Page**

- The login page asks the users to enter their login details and when the user press the login button. Then we have first cross checked it from database that user account is registered or not, If user is not registered then we will show the warning message that User is not avaialbe please sign up. If user is available and password not correct then we will show the warning messages that password is not correct. If both are correct then we will login the user to system.

- **Sign-Up Page**
  - The sign-up page asks the user to enter account details - username, password and email address,and other information. On successfully signing up, the user account approval is sent to the admin, if admin confirm his account then the user account will get registered and he will be able to login in the system.

- **Admin Login**
  - Admin has to enter their login details. After successfully login they can see their profile and user checkout history

Every Login and Sign-Up page has a switch type of navigation button on right side, where user can freely move to another page or go to homepage.

- **Search Page**
  - Here Student or Admin can search for books. Books can be searched by using the title, author, ISBN, Publisher, Bibliography Number. Users need not enter the full name or title or author or ISBN, etc. Our system handles all the pattern matching in the background. On this page, initially, we show some trending books based on new checkouts. When a user searches any book, then it shows the results. It has a title, author name, ISBN, and book Id in a column. And then users can click on the book and see the full data of the book. There's an issue button beside every book. Only admin can issue the book.We were also planning to implement the functionality that any account holder can issue book himself but it was not completed.

- **Profile Page**
  - Students and admin both have Profile Page. Students Profile Page contains all the **checkout**, **checkin**, **fine**, Number of books issued, Account Type etc of that user. Where as Admin Profile page contain details about admin, recent checkout-checkin and a libraray book issue section which contains **issue books**, **returns books** and **Students history**.

- **Issue Book**
  - In search results section there is issue button. Only admin has access to this button. If a student want to issue book then he has to go to admin desk. Admin enter userid in issue book section and then issue the book. If user hasn't exceeded the fine or no of books limit then the book issued. All the query of checking fine and book limit are handled.

- **Return Book**
  - Only admin can return the book. User's have to provide their userid to admin, then admin can see all the book name which is issued to the user. It also shows fine and due_date and has button to return book. When admin click on return book the book will return and we handled all the query.

- **User History**
  - Admin can see checkout, check in and recent books issued history of any students.

3.2 Special Functionality

- **Indexes**
  - Library Book Collection log table has over 1 lacks rows, so we indexed the table on bookid using a B-Tree index for fast lookups on book details using bookId which is unique. In addition, we also indexed the userdetails table (also using a B-Tree index), to allow us to search userid quick at login and wherever needed.
- **Constraints**
  - Aside from primary key constraints on every table and setting all the values to not null in table, there are constraints on tables like in checkout tables we set the constraint that book issued data should be less than books due dates and item count must be greater than 0. There is also a referential constraint on checkout and checkin tables i.e. if the book is returned then that tuple must not present in checkout tables and it must present in checkin table. There's default constraint on some of the attributes.
- **Views**
  - We created view by joining Library collection and checkin data by using join on checkin bookid to library collection bookid, and then used this view to search bookname, title, bibnum of that user to show on their profile. Similarly, we created view with checkout table and shows the data on user profile as well as admin profile.
- **Access Privileges**
  - Only admin has acces to insert data in the table and see everyone's data from the table
  - User has only access to their data and book inventory

3.3 List of Queries

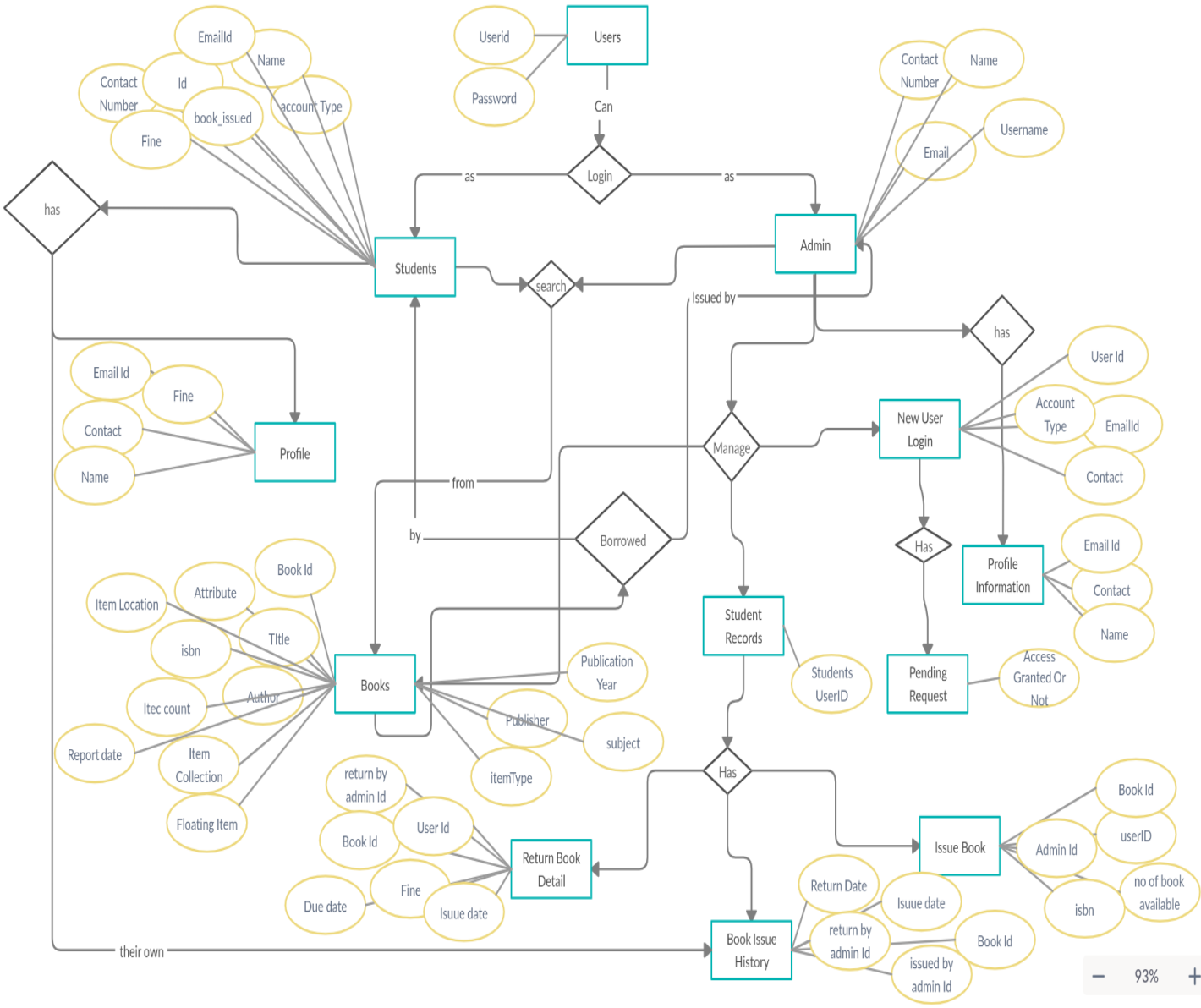
- **Login Page-Students**
  i. **create unique index** userAvailablityCheck **on** userdetails (id)
  ii. **select** id **from** userdetails **where** id = $1;
  iii. **drop index** userAvailablityCheck;
- **Login Page-Admin**
  i. **create unique index** admindetails_username **on** admindetails (username);
  ii. **select** * **from** admindetails **where** username = $1 **and** password = $2 ;
  iii. **drop index** admindetails_username ;
- **SignUP Page**
  i. **create unique index** userAvailablityCheck **on** userdetails (id);
  ii. **select** id **from** userdetails **where** id = $1 ;
  iii. **drop index** userAvailablityCheck ;
  iv. **select** id **from** pending **where** id = $1 ;
  v. **insert into** pending **values** ($1 , $2 , $3 , $4 , $5 , $6 , $7) ;
- **Search Book Page**
  i. **select** * **from** library_collection **limit** 10;
  ii. **select** * **from** library_collection **where** $1 like $2 **limit** 20;
- **Profile Page-Students**
  i. **create unique index** userAvailablityCheck **on** userdetails (id) ;
  ii. **select * from** userdetails **where** id = $1 ;
  iii. **drop index** userAvailablityCheck ;
  iv. **create view** userchekin **as** (**select * from** library_collection **as** b **join** checkin_data **as** a **on** a.bookid = b.id);
  v. **select** id , title , issue_date , due_date , return_date **from** userchekin **as** k **where** k.userid =

$1 ;
vi. **drop view** userchekin ;
vii. **create view** usercheckouts **as** (**select * from** library_collection **as** b **join** checkouts_data **as** a **on** a.bookid = b.id) ;
viii. **select** id , title , issue_date , due_date **from** usercheckouts **as** k **where** k.userid = $1 ;
ix. **drop view** usercheckouts ;

- **Remaining Other Queries**

i. **select * from** admindetails **where** username = $1 ;
ii. **select * from** pending **limit** 20;
iii. **select * from** pending **limit** 20;
iv. **select * from** checkouts_data **where** admin_issued = $1 **order by** issue_date **desc limit** 10;
v. **select * from** checkin_data **where** admin_issued = $1 **order by** return_date **limit** 10;
book
vi. **select * from** pending **where** id = $1 ;
vii. **insert into** userdetails **values** ($1 , $2 , $3 , $4, $5, $6, $7)
viii. **delete from** pending **where** id = $1
ix. **create unique index** userdetails_id **on** userdetails (id);
x. **select * from** userdetails **where** id = $1 ;
xi. **drop index** userdetails_id ;
xii. **create view** userchekin **as** (**select * from** library_collection **as** b **join** checkin_data **as** a **on** a.bookid = b.id) ;
xiii. **select** id , title , issue_date , due_date , return_date **from** userchekin **as** k **where** k.userid = $1 ;
xiv. **drop view** userchekin ;
xv. **create view** usercheckouts **as** (**select * from** library_collection **as** b **join** checkouts_data **as** a **on** a.bookid = b.id) ;
xvi. **select** id , title , issue_date , due_date **from** usercheckouts **as** k **where** k.userid = $1 ;
xvii. **drop view** usercheckouts ;
return book
xviii. **select * from** checkouts_data **where** userid = $1 ;
xix. **select** date(current_timestamp);
xx. **select** $1 ¡ $2 ;
xxi. **SELECT** DATE_PART('day', '$1 00:00:00'::timestamp - '$2 00:00:00'::timestamp);
xxii. **select * from** checkouts_data **where** userid = $1 **and** bookid = $2 ;
xxiii. **select** date(current_timestamp) ;
xxiv. **insert into** checkin_data **values** ($1 , $2 , $3 ::date, $4 , $5 ::date, $6::date, $6) ;
xxv. **delete from** checkouts_data **where** userid = $1 and bookid = $2
xxvi. **update** userdetails **set** books_issued = books_issued - 1, fine = fine + $1 **where** id = $2 ;
xxvii. **update** library_collection **set** itemcount = itemcount + 1 **where** id = $1 ;
book issue
xxvii. **create unique index** BookSearch **on** library_collection (id) ;
xxviii. **select * from** library_collection **where** id = $1 ;
xxix. **drop index** BookSearch ;
xxx.**select** books_issued, fine **from** userdetails **where** id = $1 ;
xxxi. **select** acc_type **from** userdetails **where** id = $1 ;
xxxii. **select * from** booklimit **where** type = $1 ;
xxxiii. **insert into** checkouts_data **values** ($1 , $2 , date(current_timestamp), $3 , date(current_timestamp + interval '$4 days'));
xxxiv. **update** library_collection **set** itemcount = itemcount - 1 **where** id = $1
xxxv. **update** userdetails **set** books_issued = books_issued + 1 **where** id = $1 ;

4 Section 4



ER Diagram For Dummy library system

Query Number	Avergage Running Time(ms)
1 (i)	268.199
1 (ii)	0.516
1 (iii)	0.516
2 (i)	78.66
2 (ii)	1.508
2 (iii)	0.412
3 (i)	268.199
3 (ii)	2.136
3 (iii)	0.254
3 (iv)	0.809
3 (v)	31.674
4 (i)	28.220
4 (ii)	783.718
5 (i)	268.199
5 (ii)	0.516
5 (iii)	0.516
5 (iv)	77.438
5 (v)	33.733
5 (vi)	26.287
5 (vii)	60.546
5 (viii)	31.543
5 (ix)	28.654

Table 3. **Average Query Running Time**