

MINI-KULibrary

Ali Oktay, Betül Demirtaş, Doğa Ege İnhanlı, Murat Güç, Pınar Erbil

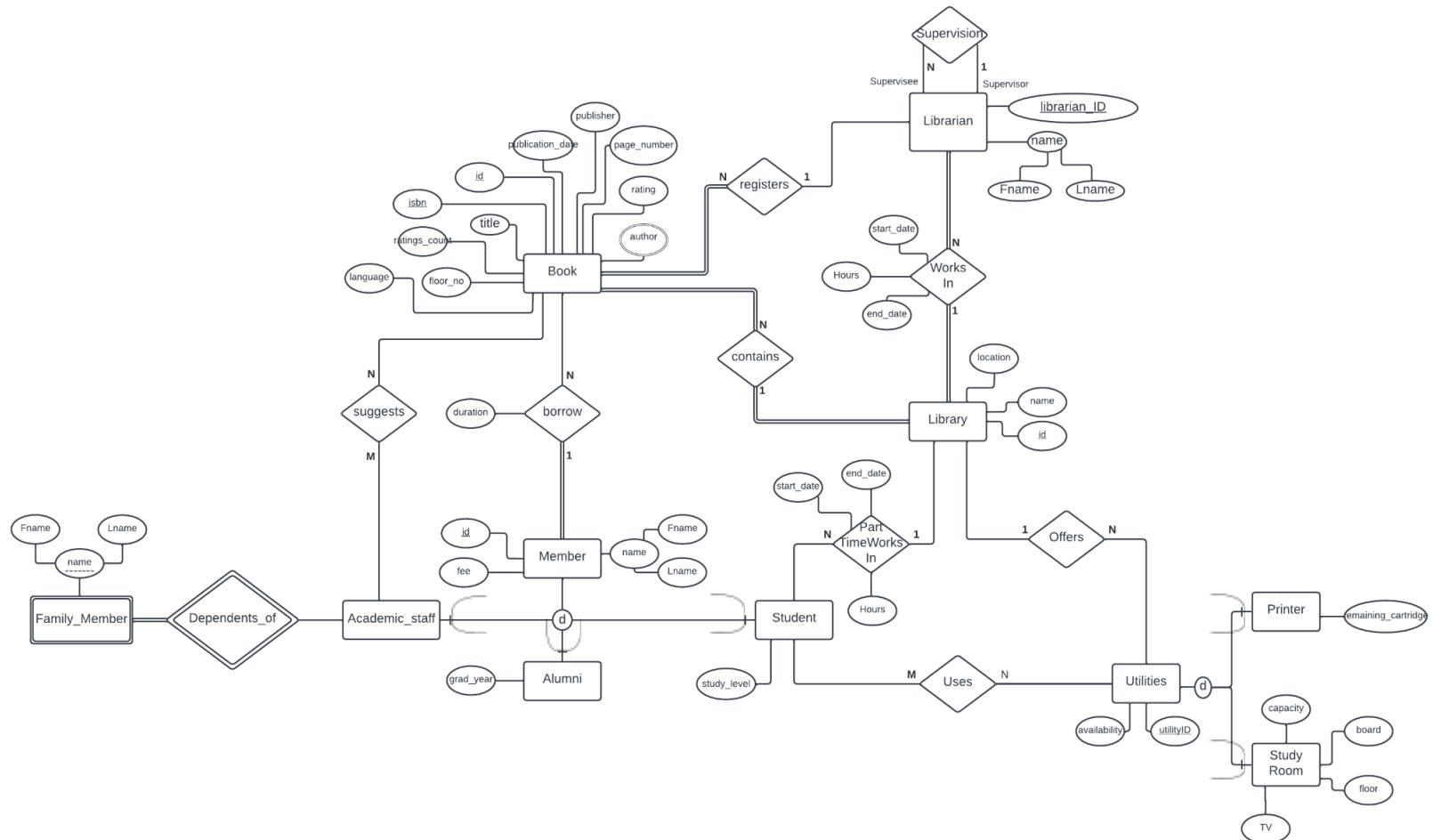
Project Description

We created a mini-library management system. Using our system, librarians and library members (students, alumni, and academic staff) will be able to do many activities such as:

Librarians can register, delete and update a book. They can also hire and fire working students and calculate the debt of any student. Moreover, they can see the students with the most working time each year. Lastly, they can see the usage frequency of each utility in the libraries.

Members can check the availability of a book, and borrow and return a book. They can also check books suggested by academic members. If they are students, they can also use the utilities in the library. If they are academic staff they can input suggested books.

Entity-Relationship Diagram



Relational Database Design

```
CREATE TABLE Book (
    book_ID : INTEGER,
    member_ID : INTEGER,
    librarian_ID : INTEGER,
    library_ID : INTEGER,
    isbn : INTEGER,
    title : CHAR(50),
    floor_no : INTEGER,
    language : CHAR(20),
    ratings_count : INTEGER,
    publication_date : Date,
    publisher : CHAR(50),
    page_number : INTEGER,
    rating : DOUBLE,
    borrow_duration : INTEGER,
    PRIMARY KEY(book_ID, isbn),
    FOREIGN KEY librarian_ID REFERENCES Librarian,
    FOREIGN KEY member_ID REFERENCES Member,
    FOREIGN KEY library_ID REFERENCES Library
)
```

```
CREATE TABLE Utilities (
    utility_ID : INTEGER,
    availability : CHAR(30),
    PRIMARY KEY (utility_ID),
    library_ID: INTEGER,
    FOREIGN KEY library_ID REFERENCES Library
)
```

```
CREATE TABLE Printer (
    utility_ID : INTEGER,
    remaining_cartage : INTEGER,
    PRIMARY KEY (utility_ID),
    FOREIGN KEY utility_ID REFERENCES Utilities
)
```

```
CREATE TABLE Member (
    member_ID : INTEGER,
    fee : INTEGER,
    fname : CHAR(50),
    lname : CHAR(50),
    PRIMARY KEY (member_ID)
)
```

```
CREATE TABLE StudyRoom (
    utility_ID : INTEGER,
    floor : INTEGER,
    capacity : INTEGER,
    tv : BOOLEAN,
    board : BOOLEAN,
    PRIMARY KEY (utility_ID),
    FOREIGN KEY utility_ID REFERENCES Utilities
)
```

```
CREATE TABLE Student (
    member_ID : INTEGER,
    study_level : CHAR(50),
    library_ID : INTEGER,
    end_date : DATE,
    start_date : DATE,
    work_hours : INTEGER,
    PRIMARY KEY (member_ID),
    FOREIGN KEY member_ID REFERENCES Member,
    FOREIGN KEY library_ID REFERENCES Library
)
```

```
CREATE TABLE Alumni (
    member_ID : INTEGER,
    grad_year : DATE,
    PRIMARY KEY (member_ID),
    FOREIGN KEY member_ID REFERENCES Member
)
```

```
CREATE TABLE Academic_staff (
    member_ID : INTEGER,
    PRIMARY KEY (member_ID),
    FOREIGN KEY member_ID REFERENCES Member
)
```

```
CREATE TABLE Library (
    library_ID : INTEGER,
    location : CHAR(50),
    library_name: CHAR(50),
    PRIMARY KEY (library_ID),
)
```

```
CREATE TABLE Family_Member (
    member_ID : INTEGER,
    fname : CHAR(30),
    lname : CHAR(30),
    PRIMARY KEY (fname, lname, member_ID),
    FOREIGN KEY member_ID REFERENCES Academic_staff
)
```

```
CREATE TABLE Uses (
    student_ID : INTEGER,
    utility_ID : INTEGER,
    PRIMARY KEY (student_ID, utility_ID),
    FOREIGN KEY student_ID REFERENCES Student,
    FOREIGN KEY utility_ID REFERENCES Utilities
)
```

```
CREATE TABLE Librarian (
    librarian_ID: INTEGER,
    fname: CHAR(30),
    lname: CHAR(30),
    library_ID : INTEGER,
    supervisor_ID : INTEGER,
    start_date: DATE,
    end_date: DATE,
    hours: INTEGER,
    PRIMARY KEY (librarian_ID),
    FOREIGN KEY supervisor_ID REFERENCES Librarian,
    FOREIGN KEY library_ID REFERENCES Library
)
```

```
CREATE TABLE Suggests (
    book_ID: INTEGER,
    academic_staff_ID: INTEGER,
    FOREIGN KEY book_ID REFERENCES Book,
    FOREIGN KEY academic_staff_ID REFERENCES Member,
    PRIMARY KEY (book_ID, academic_staff_ID)
)
```

```
CREATE TABLE Author (
    book_ID : INTEGER,
    authorName : CHAR(50),
    PRIMARY KEY (book_ID, authorName),
    FOREIGN KEY book_ID REFERENCES Book
)
```

Data Sources

The book's data is from the Goodreads database. We downloaded the data from here:

<https://www.kaggle.com/datasets/jealousleopard/goodreadsbooks?resource=download>

All other data of people (members, family members, librarians) are from here:

<https://eforexcel.com/wp/downloads-16-sample-csv-files-data-sets-for-testing/>. Some of the information is used for students, some for alumni and so on.

Other small datasets are generated by hand and by randomizing numbers.

Missing column information on books, librarians, and member tables are generated by hand and randomizing numbers.

Complex SQL Queries

1) **SELECT M2.fname AS Name, M2.lname AS Last_Name,**
A.Paycheck_After_Fee_Payment AS Debt
FROM (SELECT M.fname, M.lname, M.member_id AS MIDD, ST.work_hours * 10 AS
Total_Paycheck, M.fee AS Total_Fee, ST.work_hours * 10 - M.fee AS
Paycheck_After_Fee_Payment
FROM student as ST, member as M
WHERE ST.member_id = M.member_id and ST.work_hours > 0) AS A, member AS M2
WHERE M2.member_id = A.midd and A.Paycheck_After_Fee_Payment < 0 AND
m2.member_id = '\$student_id'
ORDER BY A.Paycheck_After_Fee_Payment

SELECT M2.fname AS Name, M2.lname AS Last_Name,
A.Paycheck_After_Fee_Payment AS Debt
FROM (SELECT M.fname, M.lname, M.member_id AS MIDD, ST.work_hours * 10 AS
Total_Paycheck, M.fee AS Total_Fee, ST.work_hours * 10 - M.fee AS
Paycheck_After_Fee_Payment
FROM student as ST, member as M
WHERE ST.member_id = M.member_id and ST.work_hours > 0) AS A, member AS M2
WHERE M2.member_id = A.midd and A.Paycheck_After_Fee_Payment < 0
ORDER BY A.Paycheck_After_Fee_Payment

There are two versions of this query. The first one takes a student id as input and shows the dept of the current student. And the second version prints out all of the students with a debt. The debt calculation is performed by first calculating the paycheck of each student who participates in the work&study program. Then it checks whether they have to pay a fine to the library or not. If so, it deducted it from their paycheck. If the total at the end is smaller than 0 than this means that the student is in debt.

2) **SELECT RANK() OVER (ORDER BY COUNT(*) DESC) AS Usage_Rank, SR.utility_ID AS Room_ID, COUNT(*) AS Times_Used**
FROM study_room AS sr, uses AS u, member AS m
WHERE U.s_member_ID = M.member_ID AND SR.utility_ID = U.utility_ID
GROUP BY SR.utility_ID

SELECT RANK() OVER (ORDER BY COUNT(*) DESC) AS Usage_Rank, p.utility_ID AS Printer_ID, count(*) AS Times_Used
FROM printer AS p, uses AS u, member AS m
WHERE u.s_member_ID = m.member_ID AND p.utility_ID = u.utility_ID
GROUP BY p.utility_ID

There are two versions of this query. The first one checks the information about study rooms and the second version checks printers. The queries gives a popularity ranking to the utilities according to their usage history. The more utility used, the higher it will rank.

3) **UPDATE librarian**
SET hours = hours - 1
WHERE supervisor_id = \$s_ID AND librarian_id IN (SELECT librarian_id
FROM (SELECT l.librarian_id
FROM librarian AS l, book AS b, suggests AS s
WHERE l.librarian_id = b.librarian_id AND b.book_id = s.book_id
GROUP BY b.book_id
HAVING COUNT(s.book_id) > 4) AS A);

It takes the ID of a supervisor as input and searches for all the supervisees of the supervisor. If a book, which is registered by one of these supervisees, is recommended by 4 people from the academic staff, the working hours of the supervisee is decreased by one as an incentive move.

4) **SELECT DISTINCT s.study_level AS Year, m.fname AS FirstName, m.lname AS LastName, m.member_ID AS ID, s.work_hours AS Hours**
FROM member AS m, student AS s
WHERE m.member_ID = s.member_ID AND s.end_date IS NOT NULL AND
(s.work_hours, s.study_level) IN
(SELECT MAX(st.work_hours), st.study_level
FROM student AS st
WHERE st.end_date IS NOT NULL
GROUP BY st.study_level)
ORDER BY
CASE
WHEN Year = 'Freshman' THEN 0
WHEN Year = 'Sophomore' THEN 1

```
WHEN Year = 'Junior' THEN 2  
WHEN Year = 'Senior' THEN 3  
ELSE 4 END;
```

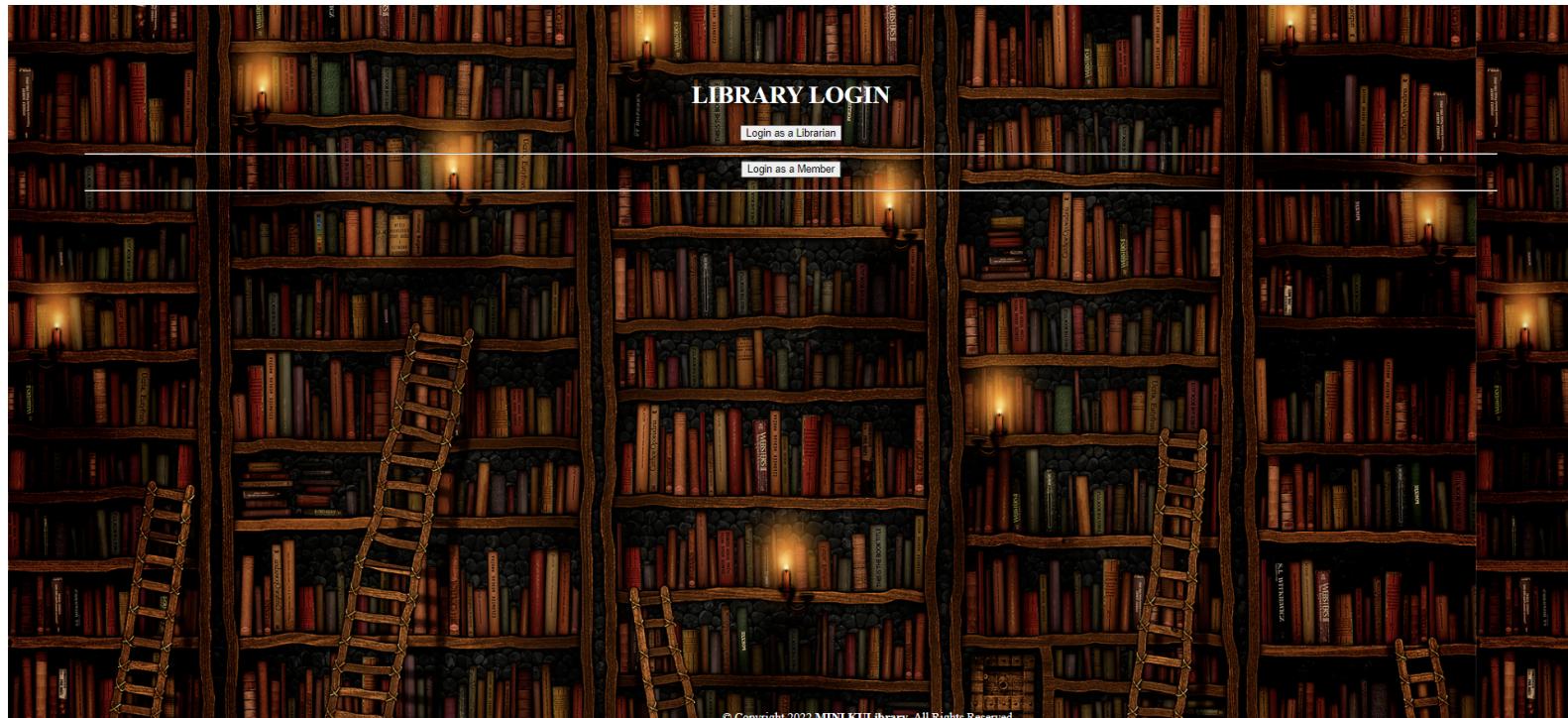
It shows the names, surnames, student IDs, working hours and study level of a student, who is working the most by considering the working hours among the same study level students. It displays the most successful students by ordering them according to their study levels.

5) **SELECT IF ((SELECT COUNT(*)
 FROM Book B, Library L
 WHERE B.library_ID = L.library_ID
 AND B.title = '\$title'
 AND L.library_ID = '\$libraryid') > 0,
 (SELECT B.book_ID
 FROM Book B, Library L
 WHERE B.library_ID = L.library_ID
 AND B.title = '\$title'
 AND L.library_ID = '\$libraryid'),
 (SELECT B.book_ID
 FROM Book B, Library L
 WHERE B.library_ID = L.library_ID
 AND B.title = '\$title'
 AND L.library_ID != '\$libraryid'))**

The query takes two inputs: Book title and Library id. It searches whether the given book is in the given library. If so, it prints the id of the book. If not, it searches for the book in the other libraries and gives out this information.

Screenshots

1- Login Screen:



2-Librarian Screen:

A screenshot of the librarian dashboard. The background is the same library scene as the login screen. The dashboard contains several functional modules: 1) "REGISTER A BOOK" with fields for Librarian ID, Library ID, Book ID, ISBN, Author, Language, Page Number, Publisher, Publication Date (with a date picker), Ratings Count, and Floor No, followed by an "Insert" button. 2) "DELETE A BOOK" with a Book ID input field and a "Delete" button. 3) "UPDATE A BOOK" with fields for Choose Attribute (Franchise ID dropdown), Book ID, Change By (dropdown), and Update Book button. 4) "HIRE/TIRE A STUDENT" with fields for Operation (Hire dropdown), Student ID, Date (date picker), and Approve/Reject buttons. 5) "DEBT of STUDENT" with a Student ID input field and a Calculate button. 6) "STUDENTS with the MOST WORK TIME" with a Show button. 7) "UTILITIES USAGE RANK" with buttons for Printer and Study Room. 8) "REWORLD YOUR SUPERVISEE" with a Supervisor ID (Your ID) input field and a Reward button. At the bottom left, a copyright notice reads "© Copyright 2022 MINI KUL library. All Rights Reserved".

3-Member Screen:

CHECK BOOK INFO

Returns the book you selected in the library. If the book is not in the library, it returns where you can find it.

Choose an library to search: 1 ▾

Book Name: Check

SUGGESTED BOOKS INFO

Academic Staff ID: Check

BORROW A BOOK

Member ID:
Book ID:
Duration: Borrow

RETURN A BOOK

Book ID: Return

SUGGEST A BOOK - only allowed to academic staff

Member ID:
Book ID: Suggested

CHECK OUT UTILITY

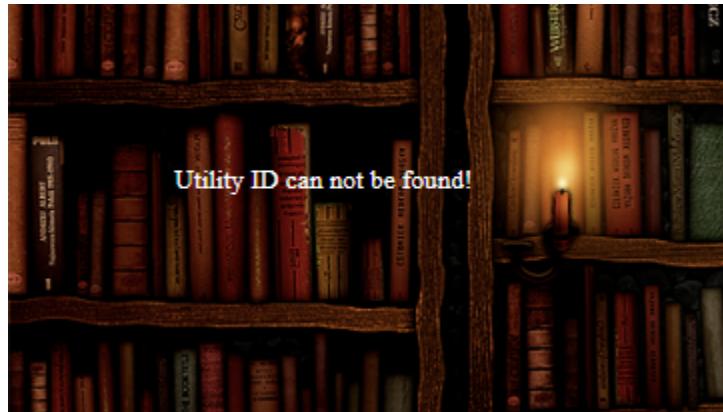
Choose an utility to use: Printer ▾
Get Available Utilities

LEAVE UTILITY

Utility ID: Leave

© Copyright 2022 MINI KULibrary. All Rights Reserved

4-Example of Providing a Wrong Input:



5- Example of a Correctly Executed Complex Query:

REWARD YOUR SUPERVISEE!

Supervisor ID: (Your ID)

Working hours decreased by one hour! Congratz

librarian_id	fname	lname
202	Lucienne	Dawson
203	Tiffi	Wilson
208	Ninon	Anderson