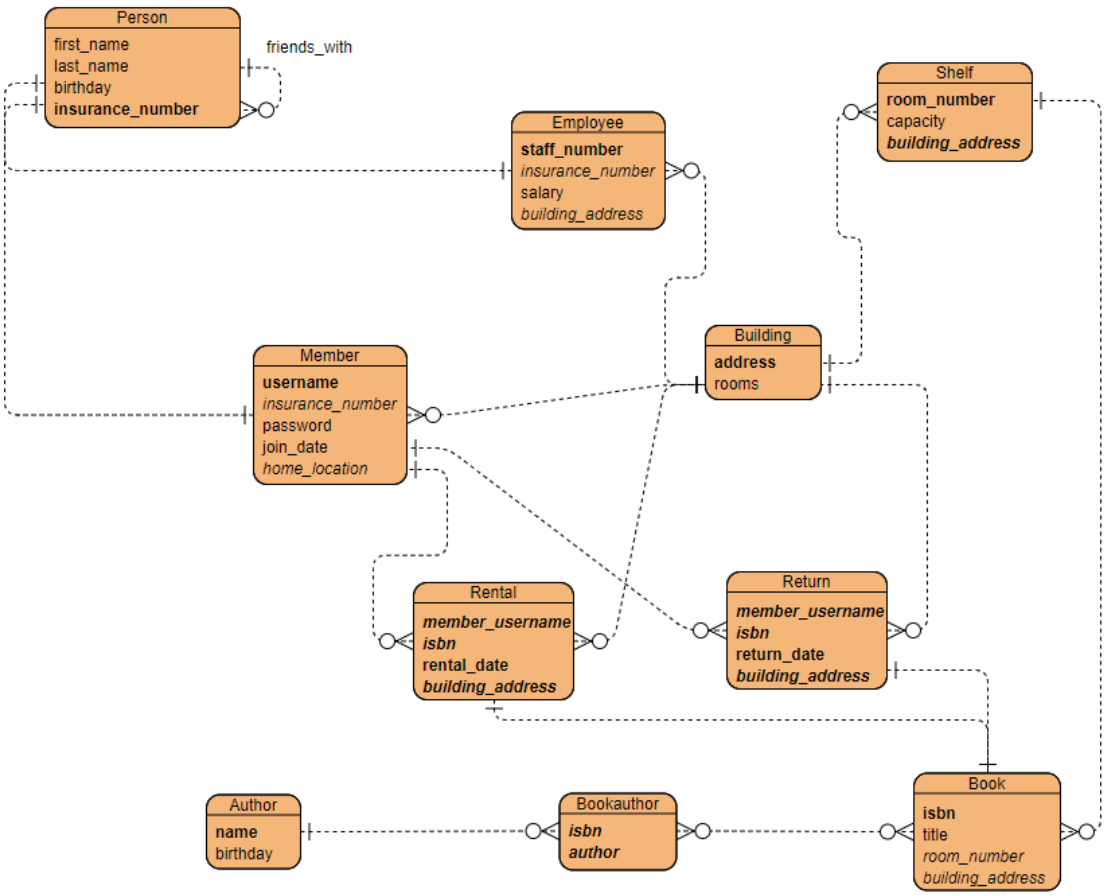
IMSE M2 Report

Dusan Popovic

01651258

Updated ER Diagram



Based on the feedback and due to changes, that I have made during the implementation of M2, I've updated the ER Diagram here, so it's easier to see the changes. Primary keys are shown in bold and foreign keys in italic. Note, the building address and room number columns in Book are only there so that I can fulfil my use case description of showing a member where a book is located within the library, they serve no other purpose (I have not implemented anything that keeps track of how many books are on each shelf etc.)

Technology Stack

My M2 implementation consists of a python Flask application being served by Nginx, which also supports a secure web server. The database itself is running on a MySQL docker image, and it is being filled via a python script utilizing the MySQL python module. All four parts of the application are deployed in separate docker containers and the MySQL, python and nginx images have been pulled from the docker hub.

Technical notes and remarks

- From the root directory, the application can be initialized with docker-compose up --build -d
- All the containers start up in order after the above command has been executed
- Once up and running, the website can be reached at https://localhost
- The database filling script can be found in src/filler/filler.py
- The flask app script can be found in src/app/app.py
- In the above-mentioned folder, all the templates can also be found as well
- The certificate and key for the secure web server are located in src/certs
- Initial values are located in the docker compose file (PERSONS =100, AUTHORS=10, BOOKS = 50, BUILDINGS=4) and can be changed there. All other values are calculated from these.

REMARKS – I kept running into one specific problem while developing this, and no matter what I tried, I could not solve it. On all the systems that I've done tests, if the docker stack hasn't been built before, i.e. the volume does not exist yet, about 50% of the time the MySQL image would refuse all connections for about 5-10 seconds after it starts up. This leads to the whole application crashing, as both the app and the filler need to connect to it and the nginx container requires the app container to be running. The only way to fix this at the moment is to wait a little bit after the MySQL container starts and then start the other three containers (filler, app and nginx) in order (this can be done by simply running the docker-compose up --build -d command again).

- I am not too experienced with Flask and app routes, so some problems arose when making the employee dashboard and related functionalities. For an error-free user experience, I recommend logging out or going back after adding a new member or searching for a rental. Doing multiple searches after each other can break the URL, leading to a 404.

Use Cases

- Login page – here one can select between a member and an employee login, member login is selected by default (login details given on the page). Even though the member entity has a password, I didn't include this check, as to not make two templates for the login, i.e. a member logs in using only his/her username (a login check is still being done)
- I implemented all my use cases, as I was a one-person team and did not want to take any chances on the number of use cases, hopefully this is fine.

Main use case – renting/returning a book: when clicking on the rent book button, the user is presented with a list of books available at their current location. Clicking on one of the books saves the rental and going to dashboard – return books and clicking on one will execute the return.

SQL for renting: SELECT isbn FROM books WHERE title = <desired title>

INSERT INTO rentals VALUES (<username>, <isbn>, <today's date>, <address>)

SQL for returning: INSERT INTO returns VALUES ('<username>, <isbn>, <today's date>, <address>)

Use case – Rental History: Displays all the rentals made by all members, searching based on a member's username is also possible. The table is shown on the employee dashboard (after logging in as an employee)

SQL: SELECT first_name, last_name, members.username, title, rental_date FROM books NATURAL JOIN rentals NATURAL JOIN members NATURAL JOIN persons

Use case – Adding a new member. This is done at the employee dashboard. Only the first and last name are needed, as well as an insurance number, the home library location can be selected by clicking on one of the addresses from the list shown. After registering a member, the randomly generated username and password are shown in the label below. These can later be changed from the update information section on the member dashboard.

SQL: INSERT INTO persons (first_name, last_name, insurance_number) VALUES (<first name>, <last name>, <insurance>)

INSERT INTO members VALUES (<random username>, <insurance>, <random password>, <today's date>, <address>)

Use case – Update personal info: This can be done from the update information section on the member dashboard. All the data is loaded into text boxes, where it can be edited, and then saved. Note, if a member was recently added, their birthday will be “None”, so if they go to change their info, a birthday must also be entered.

SQL: UPDATE persons SET first_name = '{}', last_name = '{}', birthday = '{}' WHERE insurance_number = '{}

UPDATE members SET username = '{}', password = '{}' WHERE insurance_number = '{}'

Elaborate/Reporting Use Case – The members that returned a book to a different library, and their home location: This is the table in the middle of the employee dashboard, it simply lists all the members that satisfy this condition.

SQL: SELECT first_name, last_name, username, home_location FROM (SELECT returns.member_username FROM rentals INNER JOIN returns ON rentals.building_address <> returns.building_address) AS t1 NATURAL JOIN members NATURAL JOIN persons GROUP BY username

Work Protocol

- setting up db connections - 2h (including troubleshooting)
- filler error fixes - 1h
- filler time taken - 3 hours
- nginx basic setup - 40 minutes
- html/css and flask – 16 hours

Total about 22-24 hours