CSCC43: Introduction To Databases

Database Design Project (MyBnB)

Amy Yao, Vishal Deb Sahoo

Purpose

To design and implement a database that supports the operations of a platform such as the popular home sharing service AirBnB. Such a platform would function as a marketplace for various housing and rental locations. Users are able to register as a renter or a host to either view and rent the various listings on the platform or to post their own listings for others to rent. Various functionality would be provided to support both renters and hosts. For renters, the ability to search and filter the listings available for rent, book a listing, view and modify their bookings, and leave a review on hosts or listings. For hosts, the ability to add and modify listings, add and modify availabilities for their listings, view and modify their bookings, and leave a review on renters who have rented for them. In addition, we provide various reports for a user to see information on the data stored in the system such as the bookings, cancellations, number of listings, and more.

Conceptual Problems Encountered And Solutions

Deletes

An important conceptual problem was handling deletes for listings and users. We had two options: delete data from the database or keep the data in the database and restrict actions in the system.

While deleting data from the database would keep our database clean and uncluttered, we would lose very vital information about bookings, reviews, and listings which could further lead to inaccurate reports. As a result, we decided to implement deleting by flagging the relations in the database instead of deleting them entirely. Our system checks for these flags where necessary and restricts actions (a deleted user cannot log in, a booking cannot be created with a deleted listing, and more). As a result, we are able to keep all the information and history, run accurate reports, and make better recommendations to the users. This can further help the business make better decisions and improve user experience.

Renters vs Hosts
We had to make a decision about representing renters and hosts in our system and establish any differences between them. Should renters be allowed to create listings? Should hosts be able to rent listings? We thought that not differentiating between their options and actions could make our system cluttered, complicated, and difficult to use since most users are presumably either renters or hosts. However, we wanted to allow a host to use their email to rent listings and access other functionality for renters. As a result, we decided to differentiate the roles in the database and the system, but accommodated cross functionality by allowing hosts and renters to explicitly sign up as renters and hosts respectively. This additional sign up does not create a new user in the database, simply adds a new role to the existing user, giving them access to the corresponding functionality in the system.

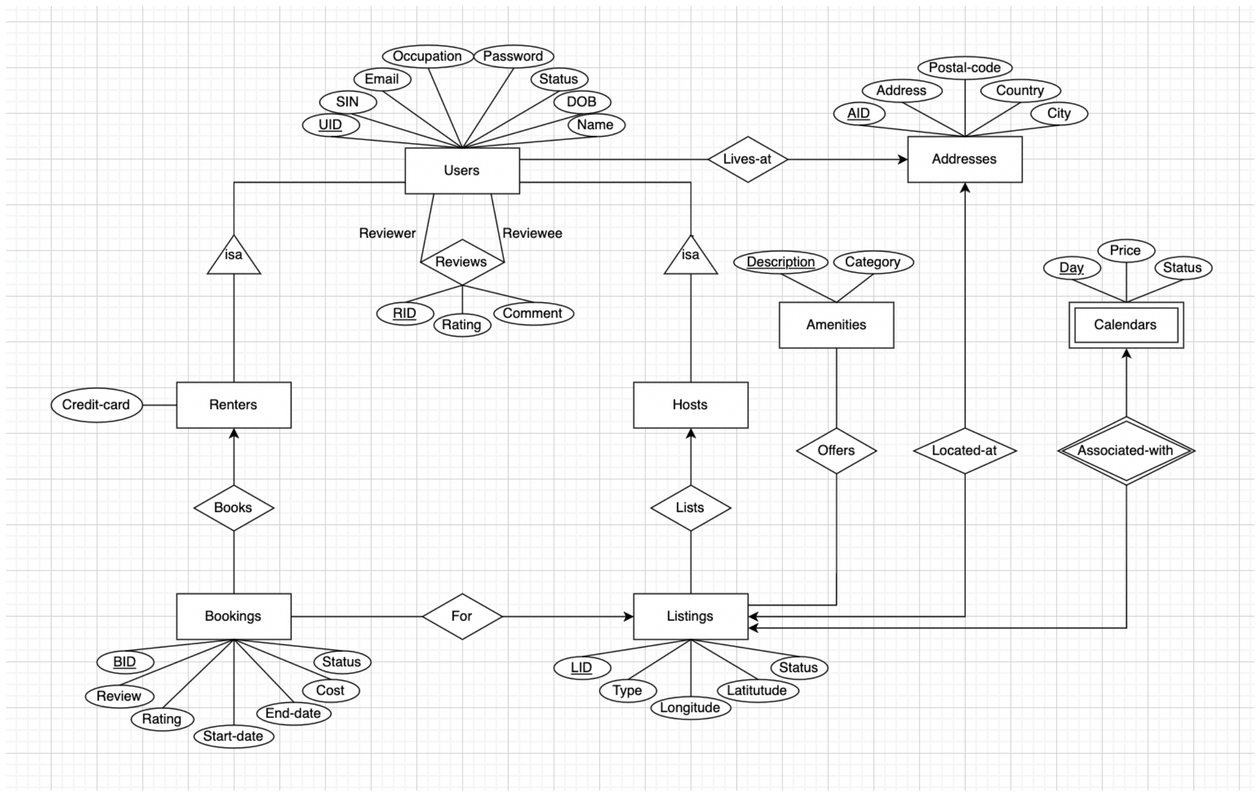
Filters
Another conceptual hurdle was accommodating several filters while searching for listings. We had to think of a solution that took into account all possible combinations of selections. We wanted to do filtering in the database and not in the backend since that would be inefficient. We tried building a query incrementally as a user selected filters; however, this was very difficult to maintain and scale. Finally, we decided to use views: we create a new view for the result of a filter and the next filter filters the tuples in the previous view. This way we have a chain of views and the actual work is done only at the end after all the filters have been chosen.

Assumptions

1. Renters have only 1 credit card, and it is only necessary to store the credit card number.
2. Hosts can only access host functionality unless they also sign up as a renter
    a. When a host signs up as a renter with the same email they used as a host, no new user is created in the database.
3. Renters can only access renter functionality unless they also sign up as a host
    a. When a renter signs up as a host with the same email they used as a renter, no new user is created in the database.
4. Postal codes are at least 3 characters long and do not contain spaces.
5. A renter can leave multiple reviews for a host that they’ve rented from.
6. A host can leave multiple reviews for a renter that has rented from them.
7. A renter can only leave one review for each of their past bookings, and this review cannot be edited later.
8. Both text based comment and a rating are required for a review.

9. Listings which do not have prices (such as those that have been created but do not have any availability) cannot be ordered by price.
10. There can be at most 1 listing at an address.
11. A listing belongs to one host.
12. A booking belongs to one renter and is for one listing.

ER Diagram



Relation Schema

Users(UID, SIN, Name, Email, Password, DOB, Occupation, AID, Status)
Listings(LID, UID, Type, Latitude, Longitude, AID, Status)
Bookings(BID, RID, LID, StartDate, EndDate, Cost, Status, Review, Rating)
Renters(UID, CreditCard)
Hosts(UID)
Reviews(RID, Reviewer, Reviewee, Comment, Rating)
Offers(LID, Description)
Amenities(Description, Category)
Addresses(AID, Address, City, Country, PostalCode)

Calendars(LID, Day, Price, Status)

Renters(UID), Hosts(UID) are subsets of Users(UID)
Users(AID), Listings(AID) are subsets of Addresses(AID)
Listings(HID) is a subset of Hosts(UID)
Bookings(RID) is a subset of Renters(UID)
Reviews(Reviewer), Reviews(Reviewee) are subsets of Users(UID)
Offers(LID) is a subset of Listings(LID)
Offers(Desc) is a subset of Amenities(Desc)
Calendars(LID) is a subset of Listings(LID)

* Note that we collapsed our relational schema to make it simpler. For example, instead of creating three relations from the entities Renters, Bookings, and the relationship Books between them, we create two relations Renters and Bookings where Bookings has a foreign key that refers to the primary key of Renters. This is possible in many to one or one to one relationships. For many to many relationships, we create three relations as usual (for example Listings, Amenities, and Offers). This enables for fewer relations and reduces the need for joining without adding data redundancy as the relationships have no extra attributes.

User Manual

The application is command line based and can be started by compiling the Java project and running Driver.java. When run, the program will display a menu with options available to the user. These can be selected by inputting the number corresponding to the menu option.

The application is menu based: a menu that maps numbers to options is displayed where there are multiple options to choose from. The user is prompted for the number corresponding to the option they want to continue with.

For operations where additional information is required, the user is prompted with questions with valid answers presented in parentheses (separated by commas) beside the question.

When prompted to select a listing (to add availability as a host or to book as a renter), booking (to cancel upcoming bookings or leave a review on a past booking), or user (to leave a review), enter the corresponding number displayed. Users may enter -1 to exit and go back to the previous menu.

Dates should be entered in YYYY-MM-DD format. Date ranges are inclusive and the start date should be entered first, followed by the end date (YYYY-MM-DD YYYY-MM-DD). Prices and coordinates are of type double.

Price ranges are inclusive and the minimum price should be entered first, followed by the maximum price. The valid coordinate range is presented in parentheses.

Ratings are of type integer with valid range (0-5 inclusive) presented in parenthesis. Reviews can have spaces and any number of characters, but they may not have new line characters.

The values for email, password, credit card, SIN, postal code should not have spaces in them. Address, city, country, and occupation may have spaces in them. The format for address is:

1) house/guesthouse: 123 Street name
2) apartment: 123 Street name, unit 123
3) hotel: 123 Street name, room 123

Sign Up

The signup is split into renter and host sign up. Although users cannot sign up as both a renter and host at the same time, users that first sign up as renters can later sign up as a host with the same email and vice versa. The program will prompt the user for the information needed.

Log In

Users can select whether they want to log in as a renter or a host and then enter their registered email and the corresponding password (both must be strings without spaces). The application will display whether the login was successful or unsuccessful. In the former case, it will display the menu option corresponding to the role the user is logged in as. As in the first menu, the user can enter the number corresponding to the option they want to continue with.

View Listings (renter)

The view listings option displays listings in the system. Users have the option to search for listings and filter based on several criteria. They may also add ordering by price. For filter by amenities, the user should enter a comma separated list with no spaces in between. Users can select a listing by entering the corresponding number displayed, view its availabilities in a date range, and create a booking in a specified date range.

Listing Management (host)
Hosts can create and modify their listings through the various menu options. Creating a listing can be done through the “Create a listing” menu option. Most of the listing management is done through the “View your listings” menu, where a host can either view availabilities for a listing, add or remove availabilities, or modify availability prices. In order to add, remove, or modify availabilities, the user needs to enter the “Update a listing” submenu, where they will then enter which listing they wish to modify. The listing number is displayed beside the corresponding listing in the menu. Upon selecting the desired option, the program will prompt for required information such as dates or prices.

Booking Management
Users can view their upcoming bookings and select a booking to cancel. Users can also view their past and canceled bookings. Renters can leave a review by selecting a past booking.

Review Management
Renters can view hosts they rented from in “Leave A Review About A Host” and select a host to review. Hosts can view renters they rented to in “Leave A Review About A Renter” and select a renter to review.

Delete Account
Users can choose to delete their account, this will restrict them from logging in. For hosts, this will delete their listings and cancel any upcoming bookings. For renters, this will cancel their upcoming bookings.

System Limitations

- If a user signs up as both a host and renter then deactivates their account, both accounts will be deactivated.
- If you reactivate your account, both will be activated.
- A listing that was deleted cannot be added again.
- Hosts that reactivate their account will need to re-add their listings.
- The system does not do intensive error handling and input sanitization.

Possibilities For Improvement

- Option to routinely clean up the database and hard delete information no longer wanted; for example, we may want to have the option to clear inactive listings or users from the database.

- Option to reactivate a previously deleted listing.
- More input sanitization and error handling.
- We can denote whether a booking was canceled by the renter or the host, and generate possibly more informative reports.
- Provide some way for a user to know if their booking has been canceled by the other party.

Host Toolkit

Algorithm for recommending the price of a listing:
We take the type, current amenities offered, and address (country, city, postal code) of the newly created listing and get the average price of the listings that are of the same type, offer at least the amenities the new listing offers, and are in the same location. We first check the country, city, and postal code. If there are no listings that meet our criteria, then we relax our amenity requirement. If there are still no listings, we relax the location and check for country and city. We continue like this until we find at least one listing that meets our criteria with a final check of country and no amenity requirement. We recommend the new listing price as the average price of the matched listings. If we find none, we do not recommend a price since we do not have enough data.

Justification:
We believe that the price of a listing is influenced greatly by its location. Hence, we try to find the listings that are closest to the newly created listing to recommend its price. The price is also dependent on the type of the listing and amenities offered which is why we also search for listings that offer at least the same amenities. Finally, we believe that, if there is no other listing in the same city, a hotel in another city is more influential to the price of a hotel listing than a house in the same city; hence, we relax the location in our search and not the type.

Improvement:
A possible improvement is relaxing amenities offered one by one to find listings that are close but do not offer all amenities the new listing does instead of completely dropping the amenity requirement as we currently do.

Algorithm for recommending amenities and the corresponding price increase:
We rank the amenities in the essentials or safety category in descending order of prevalence (get the most offered essentials/safety) and get the first five. We rank the amenities in the features category in ascending order of prevalence (get the least offered features that are offered at least once) and get the first five. We recommend the

two essentials and features with the highest price increase. Let the amenities already offered by the listing be A. For every amenity we recommend, we get the average price of listings of the same type that offer A plus this amenity (using algorithm from listing price recommendation but never dropping the amenity requirement) and subtract from it the average price of listings of the same type that offer A. This difference is our estimated price increase for adding the amenity.

Justification:
If some essentials or safety amenity is offered by many listings, the host should consider adding this amenity to their listing as this amenity would be expected by renters looking to rent in this area. Without these, renters may choose to rent other listings with those essentials instead. On the contrary, if some feature amenity is not offered by many listings, the host should consider adding this amenity as a delighter that attracts renters to their listing and allows their listing to stand out. Subtracting the average price of listings that offer amenities A from that of listings that offer A plus B gives an estimate of the price increase due to amenity B.

Improvement:
Reduce the interference of other amenities when calculating price increase due to an amenity. Since we check if listings have at least a set of amenities, there may be price increases due to other amenities that listings offer that may not be captured by our algorithm.