Source Code + Schema + Sample Data

The relevant source files can be found in this repository:
https://github.com/garychen2002/CSCC43-2023

- CSCC43Driver.java contains the main Java code and implementation of queries/reports
- c43schema.ddl contains the definitions for each table and attributes
- c43sampledata.sql contains the sample data loaded in for testing and demo

Purpose

The purpose of the project is to simulate the operations of a home sharing service like AirBNB, with the data defined and stored in a MySQL database and interacted with using an interface implemented by a Java application.

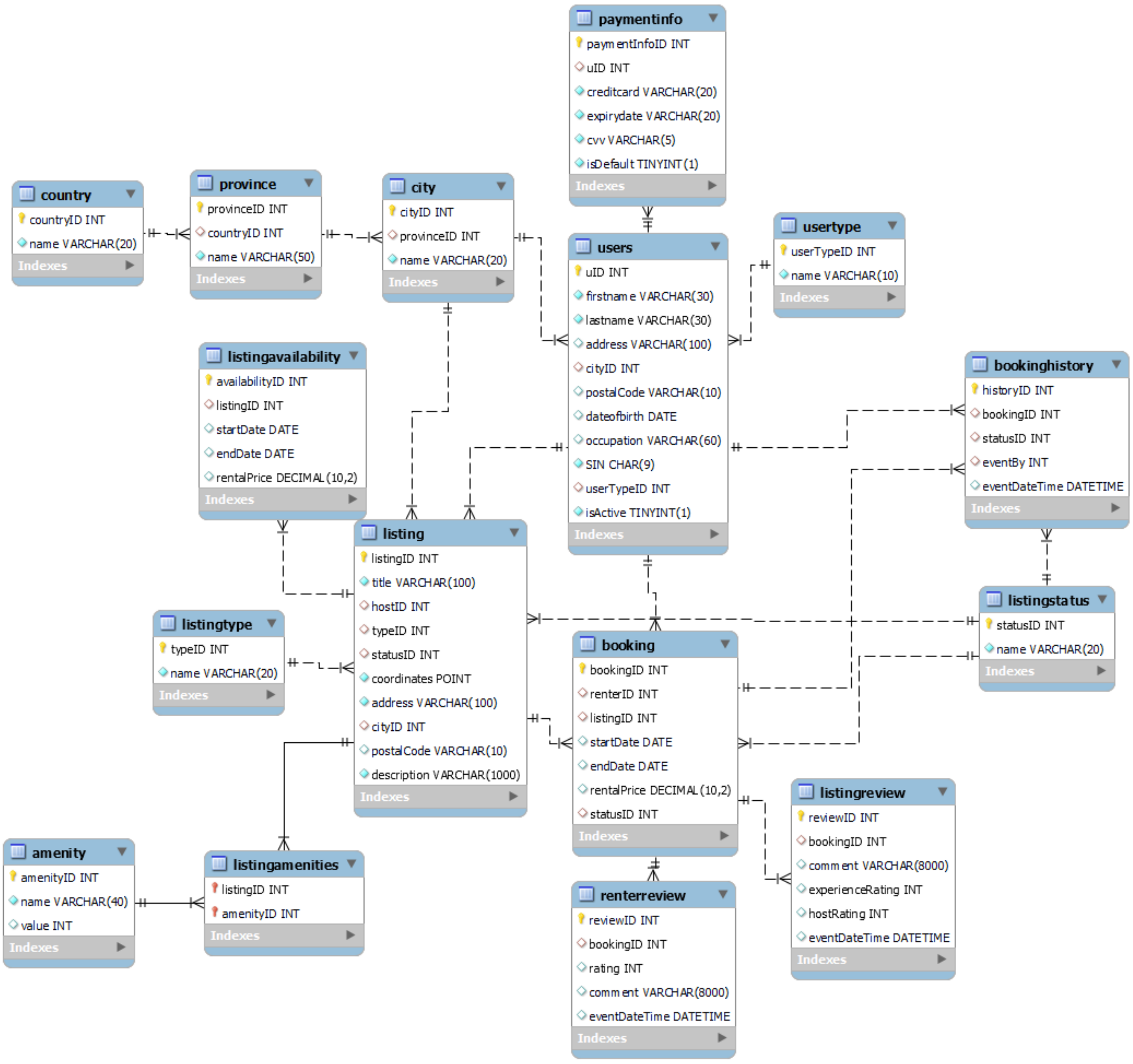
Assumptions + Problems and Solutions

- I added province to the listings in addition to city and country because it made sense to divide the cities.
- Assuming that a user can only be one of a host or renter and not both, based on the real AirBNB.
- For deleting a user, I implemented it as setting a flag to be marked as inactive and not being able to log in anymore, in case the user wants to reactivate their account later and so all the user data doesn't have to be deleted from the database which would cause problems when the uID is used as a foreign key.
- Assuming that adjacent postal codes share the first 3 characters, so they are at least in the same region as a way to identify them.
- Assuming that the amenity suggestions can just be a hardcoded list of popular amenities shown to the user, and don't need to be customized for each listing. The host toolkit price suggestions are also similarly hardcoded, but individually mapped per type of listing.
- Assuming that only the title and description and amenities of a listing can be updated and not its physical location because the real airbnb makes you contact them to edit it since it should not be changing.
- Assuming the available dates for listings are stored in the form of date ranges, and can be updated and deleted appropriately when bookings are made.
- Assuming date ranges can start and end on the same day.
- Since availabilities are deleted when bookings are created for that date range, hosts will not be able to update or delete those availabilities themselves and they will have to cancel the booking as per the requirements.
- Assuming that rental prices are tied to availabilities as well, since the documentation says "A host should be able to update the price of a listing, but only if the listing is available for rent in the specific date range the change is to be made." implying that hosts are changing prices for certain date ranges.

- Assuming that when you cancel a booking, the available date range will automatically be freed back up if the current date is before the end date. If the current date is past the start date, then the new availability will start from the current day.
- Based on the instructions saying “For example you cannot comment on a listing if you haven’t rented it recently”, I assumed that comments for hosts should be based on the listings rather than the host accounts. To ensure that renters and hosts can only comment if they have rented or been rented by the other user, comments/reviews will be tied to the booking IDs to make sure that both IDs exist and a booking was made between both of the users. I assumed that users can review any bookings they were involved in regardless of status, because they could have had bad communications with the owner beforehand that they may want to leave a bad review about.
- For viewing comments for a listing, the current listing is automatically viewed to simulate going on a listing page and looking at the reviews on it. For viewing comments of users, I decided to let the user enter any user ID to view comments about them (renter reviews for renters and listing reviews for hosts), since it would make sense that the host would be able to see the reviews of a renter before allowing them in and airbnb has public profiles so renters could see reviews that a host had. The real airbnb doesn’t let users see what ratings were given on each review, but I added that information to show that it is present for the demo.
- An assumption made for the “same and adjacent postal code” functionality for finding adjacent postal codes is just searching based on the first 3 digits of the postal code being the same since I know that means they are within the same region to a degree, but don’t know exactly how to identify proximity based on postal codes otherwise.
- The default search distance is 15 kilometers.
- Searching listings and ordering by price will show all different prices across all availabilities if a date is not specified, since each availability has its own individual rental price.
- Assuming that searching by price range is inclusive.
- Assuming that the reports for total bookings can have bookings of any status.
- Assuming that the postal code reports will search for the exact postal code, since it didn’t mention adjacent like it did for the search earlier.
- Assuming that the “total bookings per postal code within a city” search requires the user to provide the city name initially, which also requires the country and province to get exact results.
- Assuming that ranking hosts by total listings overall per country automatically just takes all countries in account and ranks each host/country combination individually.
- Assuming that the report “We would also like to rank the renters by the number of bookings in a specific time period as well as rank them by number of bookings in a specific time period per city. For the later report, we are only interested in ranking those renters that have made at least two bookings in the year.” means that the renters should have at least 2 bookings within the past year in general to represent them being an active user (using a nested query), but the bookings themselves are still searched for in the specified time period, so you could have results with renters having only 1 booking in 1 city but they are still ranked because they made 2 bookings across different cities. To find bookings in the past year, I also assumed just looking for booking with a start date within the past year.
- Assuming that the report for “We also wish to report the hosts and renters with the largest number of cancellations within a year” automatically uses the time period of the past year.

- For the most popular noun phrases, I decided to count the top 5 words and added a list of reasonable stopwords to exclude from the count that weren't meaningful like "a", "the", "I", and "this".

ER Diagram



Relational Schema and Keys

I made several lookup tables with IDs that would be joined together to get string names to promote normalization.
Country(<u>countryID</u>, name)

Province(provinceID, countryID, name) (countryID is a foreign key from Country)
City(cityID, provinceID, name) (provinceID is a foreign key from Province)
ListingType(typeID, name)
Amenity(amenityID, name, value)
UserType(userTypeID, name)
ListingStatus(statusID, name)
Users(uID, cityID, userTypeID, firstname, lastname, address, postalCode, dateOfBirth, occupation, SIN, isActive)

- (cityID, userTypeID are foreign keys from City, UserType)

Listing(listingID, hostID, typeID, statusID, cityID, title, coordinates, address, postalCode, description)

- hostID, typeID, statusID, cityID are foreign keys from Users, ListingType, ListingStatus, City

PaymentInfo(paymentInfoID, uID, creditcard, expirydate, cvv, isDefault)

- uID is a foreign key from User

ListingAmenities(listingID, amenityID)

- listingID and amenityID are foreign keys from Listing and Amenity

ListingAvailability(availabilityID, listingID, startDate, endDate, rentalPrice)

- listingID is a foreign key from listing

Booking(bookingID, renterID, listingID, startDate, endDate, rentalPrice, statusID)

- renterID, listingID, statusID are foreign keys from User, Listing, ListingStatus

BookingHistory(historyID, bookingID, statusID, eventBy, eventDateTime)

- bookingID, statusID, eventBy are foreign keys from Booking, ListingStatus, User (uID)

ListingReview(reviewID, bookingID, comment, experienceRating, hostRating, eventDateTime)

- bookingID is a foreign key from Booking

RenterReview(reviewID, bookingID, rating, comment, eventDateTime)

- bookingID is a foreign key from Booking

DDL Schema

TA Daren said this would be good to include.

```
use mybnb;
DROP TABLE IF EXISTS ListingAmenities, ListingAvailability, Listing, BookingHistory, Booking,
ListingReview, RenterReview, PaymentInfo, Users,
ListingStatus, ListingType, UserType, Amenity, City, Province, Country;

create table Country (
        countryID integer AUTO_INCREMENT primary key,
    name varchar(20) not null
);

create table Province (
        provinceID integer AUTO_INCREMENT primary key,
        countryID integer,
        foreign key (countryID) references Country (countryID),
    name varchar(50) not null
);

create table City (
        cityID integer AUTO_INCREMENT primary key,
        provinceID integer,
        foreign key (provinceID) references Province (provinceID),
    name varchar(20) not null
```

```
);

create table ListingType (
       typeID integer AUTO_INCREMENT primary key,
    name varchar(20) not null
    );

create table Amenity (
       amenityID integer AUTO_INCREMENT primary key,
    name varchar(40) not null,
    value integer default 10
    );

create table UserType (
       userTypeID integer AUTO_INCREMENT primary key,
    name varchar(10) not null
    );

create table ListingStatus (
       statusID integer AUTO_INCREMENT primary key,
    name varchar(20) not null
);

create table Users (
       uID integer AUTO_INCREMENT primary key,
    firstname varchar(30) not null,
       lastname varchar(30) not null,
    address varchar(100),
       cityID int,
    foreign key (cityID) references City(cityID),
       postalCode varchar(10),
    dateofbirth date,
    occupation varchar(60),
    SIN char(9) not null,
    userTypeID integer,
    foreign key (userTypeID) references UserType(userTypeID),
       isActive boolean not null default 1
    );

create table Listing (
       listingID integer AUTO_INCREMENT primary key,
       title varchar(100) not null,
    hostID integer,
    typeID integer,
    statusID integer,
    foreign key (hostID) references Users (uID),
    foreign key (typeID) references ListingType(typeID),
       foreign key (statusID) references ListingStatus (statusID),
    coordinates point not null,
    address varchar(100) not null,
       cityID int,
    foreign key (cityID) references City(cityID),
       postalCode varchar(10),
    description varchar(1000) not null
    );

create table PaymentInfo (
       paymentInfoID integer AUTO_INCREMENT primary key,
    uID integer,
       foreign key (uID) references Users (uID),
    creditcard varchar(20) not null,
    expirydate varchar(20) not null,
    cvv varchar(5) not null,
    isDefault boolean not null
    );

create table ListingAmenities (
       listingID integer,
    amenityID integer,
       foreign key (listingID) references Listing (listingID),
    foreign key (amenityID) references Amenity (amenityiD),
    primary key (listingID, amenityID)
    );

create table ListingAvailability (
       availabilityID integer AUTO_INCREMENT primary key,
    listingID integer,
       foreign key (listingID) references Listing (listingID),
    startDate date,
    endDate date,
    rentalPrice decimal(10,2)
);

create table Booking (
       bookingID integer auto_increment primary key,
    renterID integer,
    listingID integer,
    foreign key (renterID) references Users (uID),
    foreign key (listingID) references Listing (listingID),
    startDate date,
    endDate date,
    rentalPrice decimal(10,2),
       statusID integer,
       foreign key (statusID) references ListingStatus (statusID)
);

create table BookingHistory (
       historyID integer auto_increment primary key,
       bookingID integer,
    foreign key (bookingID) references Booking (bookingID),
       statusID integer,
       foreign key (statusID) references ListingStatus (statusID),
       eventBy integer,
       foreign key (eventBy) references Users (uID),
       eventDateTime datetime default now()
);

create table ListingReview (
       reviewID integer auto_increment primary key,
    bookingID integer,
    foreign key (bookingID) references Booking (bookingID),
    comment varchar(8000),
    experienceRating integer,
    hostRating integer,
    CHECK (experienceRating >= 0 and experienceRating <= 5),
       CHECK (hostRating >= 0 and hostRating <= 5),
       eventDateTime datetime default now()
    );

create table RenterReview (
       reviewID integer auto_increment primary key,
    bookingID integer,
    foreign key (bookingID) references Booking (bookingID),
    rating integer,
    comment varchar(8000),
    CHECK (rating >= 0 and rating <= 5),
       eventDateTime datetime default now()
);
```

User Manual / Explanations / Examples

Note: This Java program requires the mysql connector jar file for Java to connect to the MySQL database.

The program acts as a text based interface on the terminal screen where you can input various commands for the possible operations. Each command is explained briefly below and each query requested in the project documentation is implemented, as well as some additional actions for testing and ease of use. Examples are provided based on the sample data.

Actions:

- You can look at all the lookup table mappings with their names (userType, listingType, listingStatus, country, province, city, amenity) to see possible values (for testing).
- “listings” lets you see all listings, and “availabilities” will let you see all slots for all listings.
- “create host” allows you to create a new host account. It will ask for first name, last name, address, country, province, city (will be validated), postal code, date of birth (must be 18+ to register), occupation, and SIN.
- “create renter” allows you to create a new renter account. It will ask for the same information as the host, plus payment information.
- “users” allows you to see a list of all users, ordered by uID. I used joins to show the names for user type and city/province/country rather than just their IDs.
    - “current user” lets you see the current user.
- “switch user” lets you switch to another uID as long as it is active.
- “delete user” lets you set a user to be marked as deleted/inactive, and “undelete user” lets you undo this action.
- “switch listing” lets you choose the current listing ID to be working on.
    - “current listing” lets you see the current listing.
- “create listing” lets you create a listing, only if you are a host user.
    - It asks for title, type, coordinates, address, country/province/city, postal code, description, and adding amenities one at a time.
    - Latitude and longitude coordinates are added using MySQL’s POINT spatial data type ST_GeomFromText(POINT(latitude,longitude), 4326). This data type lets us calculate distances for search later.
    - Host Toolkit: A price will be suggested based on the listing type selected.
    - A static list of amenities and revenue values is also provided when adding amenities, and the list will automatically adjust based on the amenities selected.
    - These functions are both hardcoded for simplicity, but are based on airbnb’s suggestions of popular trends to follow.
- “list listings” lets you see listings owned by the current user, by selecting and joining on Users where the listing’s hostID = user hostID and user hostID is equal to the current userID passed in as a variable.
- “create availability” lets you create a new availability for the current listing, if you are a host and own the listing.
    - It asks for start date, end date, and price.
- “update availability” lets you update the above variables for a certain availability ID.

    - Since bookings delete the availability ID they are based on, hosts cannot edit the details of a booked stay unless they cancel the booking.
- "delete availability" will delete the row from ListingAvailability
- "delete listing" will set the Listing's status to 5 (deleted) internally.
- "update listing" will allow you to update metadata like title and description, but not physical location, since on the real airbnb you have to contact the staff to change a location since it usually should not change unless it is a mistake.
- "list amenity" lists the amenities for the current listing
- "update amenity" makes you enter a new list of amenities you want the listing to have.
- "book listing" lets you make a booking for the current listing, if you are a renter. It will show availability slots, ask which you want to book, then ask what dates you want to stay for. If you stay less than the entire length, the program automatically generates new availabilities for the listing to fill in the gaps, and updates the booking history.
- "cancel booking" lets you cancel a booking that you have booked, and updates the history.
- "occupy booking" lets you occupy a booking, and updates the booking history.
- "list bookings listing" lets you view all the bookings for the current listing.
- "list bookings user" lets you view all the bookings for the current user (renter).
- "add comment" will let a renter and host choose a recent booking to leave a comment on about their respective experiences. The renter can rate both their experience and the host while the host has a general experience rating.
- "view comments user" allows you to view comments for a user. If it is a renter then you will see comments made for them, and if it is a host you can see comments made for all their listings.
- "view comments listing" lets you view the comments for the current listing.
- "Search" lets you search for listings with various options and filters (adjacent postal codes, set of amenities, date range, price range, exact address, latitude/longitude distance, rank by distance/price ascending/descending).
- Filtering was implemented by saving the IDs found in the initial search and passing them in an array to another search function which would add an additional check where the IDs for the next search would have to be in the same set of IDs.

Search examples for demo:

- Distance Search: -79 lat, 43 long, default distance (15 km)
- Postal search: M4M
- Exact address: 25 River Street
- Combo search for listing 6:
    - Postal Code: M4M
    - Date: 2023-10-01 to 2024-01-01
    - Price: $500 to $1500
    - Amenities: bedding, pool, gym, wi-fi, coffee, snacks, etc.
    - Address: 630 Gerrard St. E.

Reports:

- The “report total bookings date city” lets you see the total number of bookings in a specific date range by city. I selected from listing, booking, and city to get all the info I needed (listing to get the city ID and city for city name, counting the unique bookingID instances, grouping by city name, ordering by count descending to show from highest to lowest).
- The “report total bookings date city postal code” lets you run the same report for postal codes, which requires you to specify a city first, as well as province and country because I need those to get a unique cityID from just a name. I group by postalCode instead.
- The “report total listings country” lets you see the total number of listings per country, grouping by country name and counting the unique listingIDs. Since we join by Listing and Listing only has a cityID stored, we still need to join with city and province to get the country back.
- The “report total listings country city” lets you see the total number of listings per country and per city. Similarly, “report total listings country city postal code” lets you see the counts for postal codes as well.
- The “report host rank country” lets you see how many listings each host has per country, grouped by user name and country and sorted by highest to lowest in the same way as the first report. “report host rank city” divides by city instead.
- The “report commercial hosts” shows for which cities and countries which hosts have over 10% of the listings, calculated by getting the value of the cityID and count first and then adding another statement to select hosts that have more listingIDs than 10% of the count.
- “report renters rank bookings” lets you rank renters by number of bookings in a certain time period, and “report renters rank bookings city” lets you see it per city. The latter will only show renters that have made at least 2 bookings in the prior year in general.
- For cancellations, you can use “report cancellations host” and “report cancellations renter” to see who are the hosts and renters that have cancelled the most bookings, tracked in BookingHistory.
- The “report phrases” action allows you to view a set of the top 5 noun phrases used in comments for each listing, obtained by going through each ListingID and then selecting every comment for it. To prevent the phrases from being dominated by conjunctions and prepositions like ‘the' or ‘is’ or ‘a’, I made an array of such stopwords to exclude when finding the top words.

System Limitations and Improvements

You have to switch to the relevant user and listing ID before making use of the functions each time.

Entering an invalid country/province/city name will send you back to the main command input section instead of allowing you to input another name. I did most invalid inputs this way since the alternative would be putting in a loop but I wanted to give the user a chance to cancel out instead of getting stuck.

Entering a value that violates the constraints of the relational schema will cause the whole application to close, like entering a value that goes beyond the character limits. These could be improved by imposing additional checks in the Java code and allowing you to have another chance to make the correct inputs, but for the purposes of the demo not all of these were checked for since I wanted to focus on showing the existing functionality is correct and less about error checking.

Updating amenities requires you to re-enter all the amenities every time. Since this is in a terminal format I couldn’t make it as easy as removing or adding list elements from a GUI, but an alternative way could be to split up the functions into appending and removing amenities.

There is no explicit functionality for marking when you leave the booking since it can be assumed that if the date is past the end date then you have left.. It can be assumed that if you leave early you should cancel.

The system currently allows writing multiple reviews for the same booking, while on the real airBNB you can only make one review per booking, but this was not mentioned as a requirement in the project PDF. This could be improved by not allowing multiple reviews and/or allowing you to edit a previous review.

Currently you cannot see the reviews you yourself have written which is a feature of the real airBNB, but this was not a requirement mentioned in the project PDF so it was not prioritized.

The search results will not automatically filter by listings available in the current date range unless the date range option is picked.
If there is one listing with multiple available date ranges, all will be shown when searching for overlapping date ranges.

You have to name the country, province, and city for the report for the postal code within a city, because there could potentially be multiple cities with the same name.

Currently the reports display all results for the demo, while in a real application you would probably just show the top 10 or some other amount you are interested in.

The country/city/postal code listing could be more organized by having the right cities and postal codes below their home countries/cities.