Abstract - Making of AirBnB Database

Project Overview

This project consisted of three phases, in the conception phase I researched and analyzed the database by inspecting the website itself, the functionalities it provides as well as other implementations that are close in nature to the desired outcome. During the concept, I also invested a lot of time into researching what the requirements for such a database, and pitfalls often cause trouble during implementation.

After creating a solid plan and receiving feedback on the ER diagram creating during the initial conception phase, I set out to implement the tables. During implementation, I iteratively worked on the overall table structure (and ER diagram), the test cases and relevant test data to periodically test the database during implementation.

This has helped me produce a solid database while avoiding major time sinks that may have halted development. More about this during the “Learnings” section of this abstract.

As I received positive feedback after the implementation phase, I mainly focused on improving the overall structure of the submission by refactoring the folder structure of the GitHub repository and making said repository public. I also added and improved readme files for each phase making it as easy as possible to navigate.

Although already tested during implementation, while creating the instruction manual and documentation I noticed that I had no test cases displaying how to delete data without damaging data integrity. While this strongly depends on the use case of the individual application I wanted to provide an example. Hence, I added said example to showcase how easy it is to delete data from the database.

Lastly I also carried out finalization tests to make sure the provided sql files run without issues and made a few quality-of-life improvements.

Functionality

As initially mentioned in the concept during phase 1, this database’s main goal is to provide structure which is sensible and easy to use. The idea is almost akin to a modular kit that can be used to build off of.

It should be very user friendly and easy to use in a variety of applications. I tried to achieve this by reducing the number of necessary queries to a minimum and creating template transactions that can be adjusted to fit ones needs. This is the case for both adding and deleting data entries from the database. More about this in the “Learnings section” below.

Another goal was to offer easy deletion or expansion to the system. This is exemplified by how easy it was to delete the triphistory table during the implementation phase.

The actual functionality of the existing tables is talked about in the presentation provided during phase 2. To summarize the overall functionality: The database allows creation of users, the User class is then categorized into the two subclasses Guest & Host.

Data such as images or addresses are created to be usable by many different tables. Addresses are shared by many tables without relying on any of them, the reason for this is that they can be altered easily without having to adjust other tables. The relation between these tables is stored via FK in the respectively linked table, so a user or a property listing would for example have a reference to the address table which can be used when altering the address table.

Images are contained in their own table that stores a bunch of relevant data, the images themselves are stored using a simple URL link. This is because saving images as BLOPs uses a lot of space without providing any real advantages, other than maybe convenience, when compared to storing the images in, for example, a cloud storage and then accessing said file system via the URL stored in the database.

Other important features include saving relevant booking data via multiple N:M relationships, making it easy to adjust available property categories, amenities, house rules, etc.

Meta data

The database consists of 27 tables and 12 views, resulting in a total of 39 entities in the database. The views are used for test cases while the tables form the actual structure of the database.

Running the SQL statements provided in the "metadata.sql" file you can see that the database is also quite small in size, taking up only 1 MB in size. The individual table size also does not exceed 80 KB with the booking table being the largest. This was achieved by researching the minimum necessary size of each individual attribute in the database.

All tables have 20 data entries, some may have more, such as address or user which are required by other tables, resulting in a total of 672 data entries of test data across all tables. I have appended a table showing the individual table sizes and number of data entries below for convenience.

Learnings

I am overall very happy with how the project went, but in this section, I'd like to quickly summarize the mistakes I made during this project, how I fixed them and what I learned from them.

During the conception phase, I made the mistake of not adding a data dictionary right away and having ambiguous naming conventions. In the future I'd like to develop a solid naming convention from the beginning of a project to reduce mistakes such as this, as well as implementing and maintaining a data dictionary right away.

After one of the test phases I noticed that the triphistory table, although helpful may be unnecessary. This resulted in me overhauling some of the dependencies in the system to improve the overall structure, although I am aware that the process of developing such a project is an iterative one, I can't help but feel that I could have accounted for this error during the conception phase with some more in-depth planning.

Reflecting on the workflow, the balance between error-prevention and error-correction may have been a little too one-sided towards prevention, spending too time testing for example.

All in all, I am happy with what I have achieved.

Table Sizes:

Table	Size (KB)	Data entries*
booking	80.00	20
propertylisting	64.00	20
transaction	64.00	20
bankinformation	48.00	20
chat	48.00	20
emergencycontact	48.00	20
guest	48.00	20
host	48.00	20
property_amenity	48.00	20
property_category	48.00	20
property_houserule	48.00	20
propertyreview	48.00	20
userreview	48.00	20
wishlist_propertylisting	48.00	20
creditcardinformation	32.00	20
giftcard	32.00	20
image	32.00	85
message	32.00	20
wishlist	32.00	20
address	16.00	80
amenity	16.00	20
category	16.00	20
currency	16.00	20
houserule	16.00	20
language	16.00	20
propertytype	16.00	4
user	16.00	40

*Data entries are irrelevant for size measurement.