DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Databases Project – Spring 2019

Team No: 32

Names: Sophie Ammann, Samuel Chassot and Daniel Filipe Nunes Silva

Contents

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Deliverable 1

Assumptions

We use MySQL syntax for this project.

Listings all have a bed_type, property_type, room_type, cancellation_policy, a host and a neighborhood.

A review must be written by a reviewer about a given listing.

A calendar must have a day and a listing, otherwise it should not exist.

A host must be in a neighborhood, which must be itself in a city, which must be itself in a country.

Entity Relationship Schema

Schema

See Database/*ER-Schema.jpg*

Description

Most of the attributes related to a listing are grouped in one single table *Listing*.

The *Amenity*, Bed*_type*, *Property_type*, *Room_Type*, *Cancellation_policy*, *Country*, *City*, *Day* and *Host_verification* have been normalized to reduce redundancy.

Foreign key relationships are hardly connected to preserve integrity even if it implies to drop a lot of data after deletions (*ON DELETE CASCADE*). For example, if we delete a *Country*, we will delete every *City* in this *Country* and therefore delete every *Neighborhood* in the *Cities* and so forth.

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Relational Schema

ER schema to Relational schema

The translation of our ER schema to our Relation schema was more or less straightforward. Nevertheless, the type were not always obvious. We also decided to add a unique *id* in our tables.

DDL

See Database/*Create.sql*

General Comments

For this first work, we thought it was important to work the three together to understand the database correctly. We designed the basis of the ER model, and modified it until the three of us were satisfied.

Deliverable 2

Assumptions

We assume that all listings' city corresponds to the filename in which they are.

We put new ids for all listings and hosts to not have duplicates between cities.

We remove reviewers if they have the same id but not same name to keep only one.

We remove comments in reviews that consist of only one character (often a quote or a comma).

After parsing and cleaning all the data, we end up with simple csv files, whose name corresponds to our tables' names and columns are the same as specified in our schema. See *Datasets/Final/*

Data Loading

Query Implementation

Note:

Queries 1, 2, 3, 5 are functional.

Queries 4, 6, 7, 8, 9, 10 do not work but are implemented.

Query 1:

The query finds the average price for a listing with a specified number of bedrooms. We use 8 bedrooms for the example. Since the table Listing has an attribute for the number of bedrooms and one for the price, the query is direct.

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Query 2:

The query finds the average cleaning review score for the listings with TV. We suppose that these listings include TV's and Smart TV's. The implementation requires the Listing and Amenity tables, and the relation that maps both of them (Listing_amenity_map).

Query 3:

The query finds the hosts that have an available listing between two dates (03.2019 and 09.2019). We suppose that it suffices that the listing is available just one day in this interval. We need the Host table for the name, the Listing, the Day and the Calendar tables for the implementation.

Query 4:

The query finds how many listings exist that are posted by two different hosts but the hosts have the same name.

Query 5:

The query finds the dates that a specified host (we use 'Viajes Eco') has available accommodations for rent. The Day, Calendar (listing-calendar relation), Listing (shows which listings Viajes Eco owns) and Host tables are necessary for the implementation.

Query 6:

The query finds all the pairs (host_ids, host_names) that only have one listing online. We only need Listing and Host table to implement this query. The COUNT syntax is necessary to implement the constraint of "exactly one".

Query 7:

The query computes the difference of price (average) between listings with or without Wifi. We suppose that the listing with Wifi have either 'Wifi' or 'Pocket wifi' in their amenities. For the implementation we need the many-to-many Listing_amenity_map to link the wifi information (Amenity table) and the listings (Listing table).

Query 8:

The query computes the difference of price in a room with 8 beds in Berlin compared to Madrid. We need the Listing, Neighbourhood and City tables to implement the query, since the Neighbourhood table links the Listing and the City by our normalization.

Query 9:

The query finds the top-10 host (host_ids, host_names) in terms of number of listings per host in Spain.

Query 10:

The query finds the top-10 listings (review_score_rating) in terms of review_score_rating apartments in Barcelona. For the implementation, we used the TOP synthax which seems to not work.

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

SQL statements

See Database/Queries.sql

Interface

Design logic Description

We used a local MySQL server, Python 3.7 with Tkinter and mysql-connector to link the two.

See ../README.md for configuration.

When launching the application, the database is automatically created and populated (It takes ~20min.).

You can see the status of the connection and of the database on the top of the application. If any problem occurs, you can the delete DB and connect DB buttons to repeat the operations again. If the problem persists, close the application, go to the MySQL shell and execute *DROP DATABASE Airbnb;*.

The application is separated in Tabs, for now, you can search listings, hosts and neighborhoods using the available fields, each of them is initially set to the less restricted input or execute the predefined queries.

After searching or executing, a new window shows up to display the executed statement with the parameterized values as well as the results of this query.

Some operations may take some time (executing the queries and populating the tables) and these are blocking so we have to wait until it finishes to continue.

Screenshots

See Interface-Screenshots/

General Comments

We worked the three together to make the suggested changes after Milestone 1. Then, Sophie worked on the queries as well as on redesigning the new ER schema, Samuel worked on parsing and cleaning the data and Daniel worked on the interface.

For the next Milestone, we will improve the interface by including loading animations during blocking operations and work harder on the predefined queries before going further.