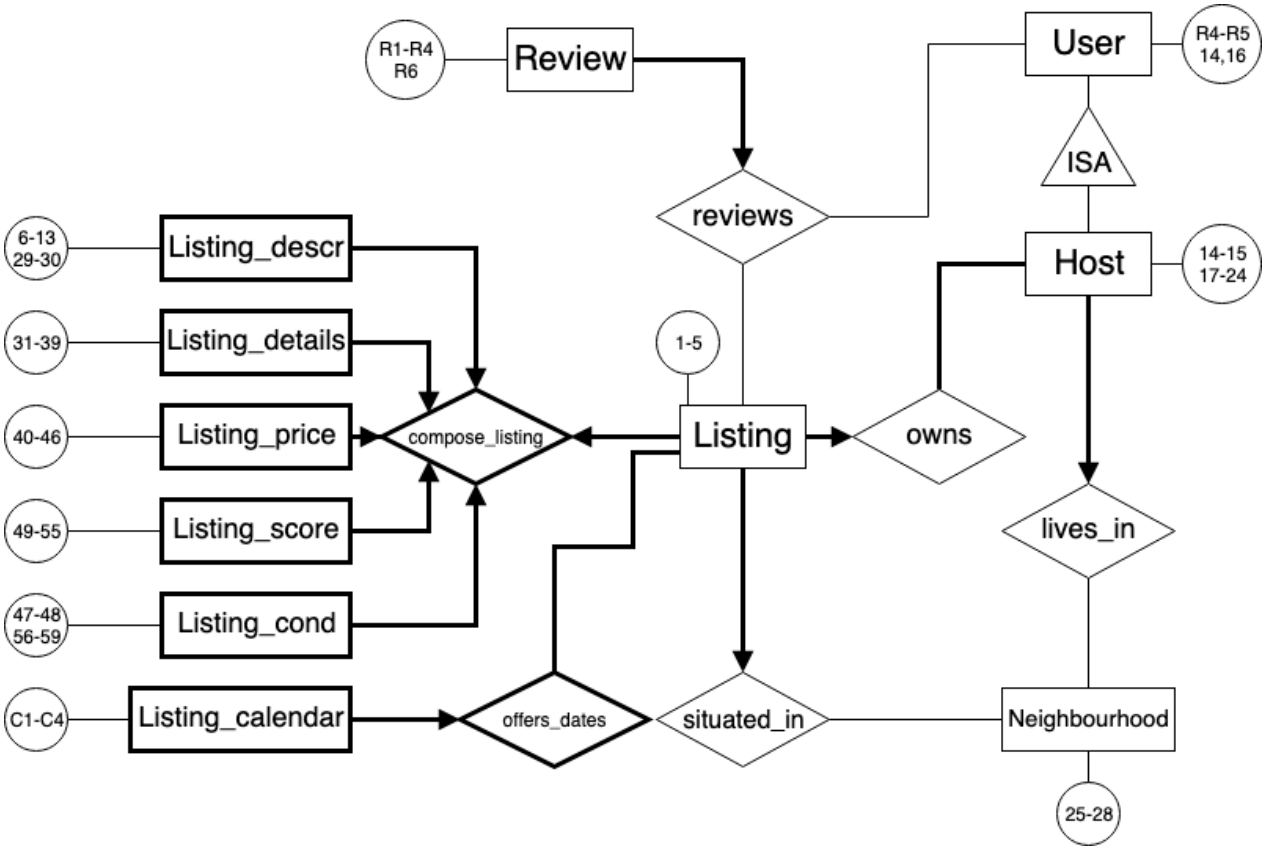
Introduction to Database Systems

Projet: Deliverable 1 - Group 11

Assumptions

All the assumptions we made can be seen in our ER model. We made sensible assumptions about listings which will be discussed in the next section. Also, these assumptions have been made after having studied the data.

Entity Relationship Schema



In the ER-model, the attribute numbers are given in Appendix.

First of all, it was pretty clear that the central entity would be the listing.

Since the listing has a significant number of attributes, we regrouped them in a logical way to split them in different tables for better performances. These entities represent Listing_descr, Listing_details, Listing_price, Listing_score, Listing_cond, Listing_calendar; they are weak entities because without a listing, they have no meaning. Also, these specific attributes can only refer to a single listing, and symmetrically, a listing can only have one listing price, one description and so on and this has been enforced in our ER model.

The weak entity Listing_calendar has to be considered separately since a listing can have more than

one calendar and, naturally, a listing needs to have at least one (also verified with a python script).

About the reviews, each listing may have reviews or may not, if it is a new one for example. On the other hand, a review must be related to exactly one listing. A review is written by a user which may or may not be a host.

A host must own at least one listing and must live in one neighbourhood.

And finally, a listing must be situated in exactly one neighbourhood.

A neighbourhood can be an independent entity, we decided to create this entity since there was multiple hosts and listings which referred to the same neighbourhood.

Relational Schema

Instead of CHAR fields, we put VARCHAR ones for better performance. Moreover, a script in Python has been written, length_finder.ipynb, which computes the maximum length of each field because we did not want to reserve a space of 1000 characters if the maximum length of this field was only of, for example, 300 characters.

About the code to create sql tables, we naturally did our best to enforce all conditions of our ER model. First, we had to add a neighbourhood ID, nid, which is not present in the original data. Indeed, since a neighbourhood can be a total independent entity, it is sensible that it has an ID on its own.

Moreover, for all weak entities which compose the listing as stated in the ER model figure, we added the condition „ON DELETE CASCADE“ because all these weak entities must be deleted in case the listing they compose is deleted. The option „NOT NULL“ has been added for the same purpose if the referenced entity was not the primary key. Otherwise, there is nothing else special to mention; all primary keys have been specified and foreign keys have been referenced.

SQL Code

```
CREATE TABLE Neighbourhood(
  nid INTEGER,
  -- not in the data provided: to add ourselves somehow.
  neighbourhood VARCHAR2(40),
  city VARCHAR2(40),
  country_code CHAR(2),
  country VARCHAR2(7),
  PRIMARY KEY(nid)
);


CREATE TABLE Listing (
    id INTEGER,
    nid INTEGER,
    -- nid: not originally provided
```

```
    listing_url VARCHAR2(40),
    name VARCHAR2(150),
    summary VARCHAR2(1000),
    space VARCHAR2(1000),
    PRIMARY KEY (id),
    FOREIGN KEY (nid) REFERENCES Neighbourhood (nid)
);

CREATE TABLE Listing_descr(
    id INTEGER,
    description VARCHAR2(1000),
    neighborhood_overview VARCHAR2(1000),
    notes VARCHAR2(1000),
    transit VARCHAR2(1000),
    l_access VARCHAR2(1000),
    interaction VARCHAR2(1000),
    house_rules VARCHAR2(1000),
    picture_url VARCHAR2(120),
    latitude FLOAT,
    longitude FLOAT,
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES Listing (id) ON DELETE CASCADE
);

CREATE TABLE Listing_details(
    id INTEGER,
    property_type VARCHAR2(22),
    room_type VARCHAR2(15),
    accommodates INTEGER,
    bathrooms INTEGER,
    bedrooms INTEGER,
    beds INTEGER,
    bed_type VARCHAR2(13),
    amenities VARCHAR2(1400),
    square_feet INTEGER,
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES Listing (id) ON DELETE CASCADE
);

CREATE TABLE Listing_price(
    id INTEGER,
    price INTEGER,
    weekly_price INTEGER,
    monthly_price INTEGER,
    security_deposit INTEGER,
    cleaning_fee INTEGER,
    guest_included INTEGER,
    extra_people INTEGER,
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES Listing (id) ON DELETE CASCADE
);

CREATE TABLE Listing_score(
```

```
  id INTEGER,
  review_scores_rating INTEGER,
  review_scores_accuracy INTEGER,
  review_scores_cleanliness INTEGER,
  review_scores_checkin INTEGER,
  review_scores_communication INTEGER,
  review_scores_location INTEGER,
  review_scores_value INTEGER,
  PRIMARY KEY (id),
  FOREIGN KEY (id) REFERENCES Listing (id) ON DELETE CASCADE
);

CREATE TABLE Listing_cond(
  id INTEGER,
  minimum_nights INTEGER,
  maximum_nights INTEGER,
  is_business_travel_ready CHAR(1),
  cancellation_policy VARCHAR2(27),
  require_guest_profile_picture CHAR(1),
  require_guest_phone_verification CHAR(1),
  PRIMARY KEY (id),
  FOREIGN KEY (id) REFERENCES Listing (id) ON DELETE CASCADE
);

CREATE TABLE Listing_calendar(
  id INTEGER,
  cdate DATE,
  available CHAR(1),
  price FLOAT,
  PRIMARY KEY (id),
  FOREIGN KEY (id) REFERENCES Listing (id) ON DELETE CASCADE
);

CREATE TABLE airbnb_User(
  user_id INTEGER,
  uname VARCHAR2(40),
  PRIMARY KEY (user_id)
);

CREATE TABLE Host(
  user_id INTEGER,
  url VARCHAR2(43),
  since DATE,
  about VARCHAR2(4000),
  response_time VARCHAR2(18),
  response_rate INTEGER,
  -- other value for int. percentages? (no decimal point)
  thumbnail_url VARCHAR2(120),
  picture_url VARCHAR2(120),
  nid INTEGER,
  -- references id not name
  verifications VARCHAR2(170),
```

```
  PRIMARY KEY (user_id),
  FOREIGN KEY (user_id) REFERENCES airbnb_User (user_id)
     ON DELETE CASCADE,
  FOREIGN KEY (nid) REFERENCES Neighbourhood
);

CREATE TABLE Review(
  id INTEGER NOT NULL,
  user_id INTEGER,
  rdate DATE,
  rid INTEGER,
  comments VARCHAR2(4000),
  PRIMARY KEY(rid),
  FOREIGN KEY (user_id) REFERENCES Airbnb_user(user_id),
  FOREIGN KEY (id) REFERENCES Listing (id) ON DELETE CASCADE
);
```

General Comments

Work allocation between team members

We naturally began to work on the ER model and what we did is that every team member had to present an ER model. The aim of this was to discuss differences we had and take the best out of the three versions. Then, for the tables, Eric wrote the length_finder script in Python. About the DDL code, the work has been split between Robin and Charline and the code has been mutually improved. Each of us also have contributed to writing this report.

Appendix

Listings

1. Id
The unique listing identifier.

2. listing_url
The URL of the listing.

3. name
The name of the listing.

4. summary
A small description of the listing.

5. space
A small description of the space of the listing.

6. description
A large description of the listing.

7. neighborhood_overview
Description of the neighbourhood of the listing.

8. notes
An extra note about the listing.

9. transit
Description of the transportation to the listing.

10. access
Specification of the accessibilities of household stuff, such as kitchen facilities.

11. interaction
Description of whom/how to interact regarding the listing.

12. house_rules
House rule specifications.

13. picture_url
The URL to the picture of the listing.

14. host_id
The unique host identifier.

15. host_url
The URL of the host.

16. host_name
The name of the host.

17. host_since
The date that the host has started working with Airbnb.

18. host_about
A small description of the host.

19. host_response_time
The amount of time within which the host responses.

20. host_response_rate
The rate at which the host replies the messages.

21. host_thumbnail_url
The URL to a thumbnail profile photo of the host.

22. host_picture_url
The URL to a profile photo of the host.

23. host_neighbourhood
The neighbourhood the host lives in.

24. host_verifications
The way with which the host can be verified.

25. neighbourhood
The neighbourhood where the listing is in.

26. city
The city where the listing is in.

27. country_code
The code of the country where the listing is in.

28. country
The country where the listing is in.

29. latitude
The latitude of the listing.

30. longitude
The longitude of the listing.

31. property_type
The type of the property.

32. room_type
The type of the room.

33. accommodates
The number of people that the listing can accommodate.

34. bathrooms
The number of bathrooms that the listing has.

35. bedrooms
The number of bedrooms that the listing has.

36. beds
The number of beds that the listing has.

37. bed_type
The type of the beds.

38. amenities

The set of amenities that listing features.

39. square_feet
The area of the listings in square feet.

40. price
The daily price of the listing. It is the price for the day when the data is collected. For the price for a
particular date, please see the *_calendar.csv files.

41. weekly_price
The weekly price of the listing.

42. monthly_price
The monthly price of the listing.

43. security_deposit
The amount of money for security deposit.

44. cleaning_fee
The fee for cleaning.

45. guests_included
The number of guests that the daily price covers.

46. extra_people
The additional price to be paid for every extra guest in addition to the number of guests specified by
the guests_included attribute.

47. minimum_nights
The minimum number of nights to rent.

48. maximum_nights
The maximum number of nights to rent.

49. review_scores_rating
The rating score of the listing.

50. review_scores_accuracy
The accuracy score of the listing.

51. review_scores_cleanliness
The cleanliness score of the listing.

52. review_scores_checkin
The checkin score of the listing (to quantify how easy the checkin is).

53. review_scores_communication
The communication score of the host.

54. review_scores_location

The location score of the listing.

55. review_scores_value
The score on the value that the listing provides for the price.

56. is_business_travel_ready
Whether the listing can be used for business travels.

57. cancellation_policy
The cancellation policy of the listing.

58. require_guest_profile_picture
Whether the listing requires a guest profile picture.

59. require_guest_phone_verification
Whether the listing requires guest phone verification.

Reviews

R1. listing_id
The identifier of the listing that is reviewed.

R2. id
The unique review identified.

R3. date
The date that the review has been written.

R4. reviewer_id
The uniquer reviewer identified

R5. reviewer_name
The name of the reviewer

R6. comments
The review.

Calendar

C1. listing_id
The identifier of the listing whose availability and price information is given.

C2. date
The date on which the listing is available or not.

C3. available
Whether the listing is available or not.

C4. price
The price of the listing for the particular date.