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/

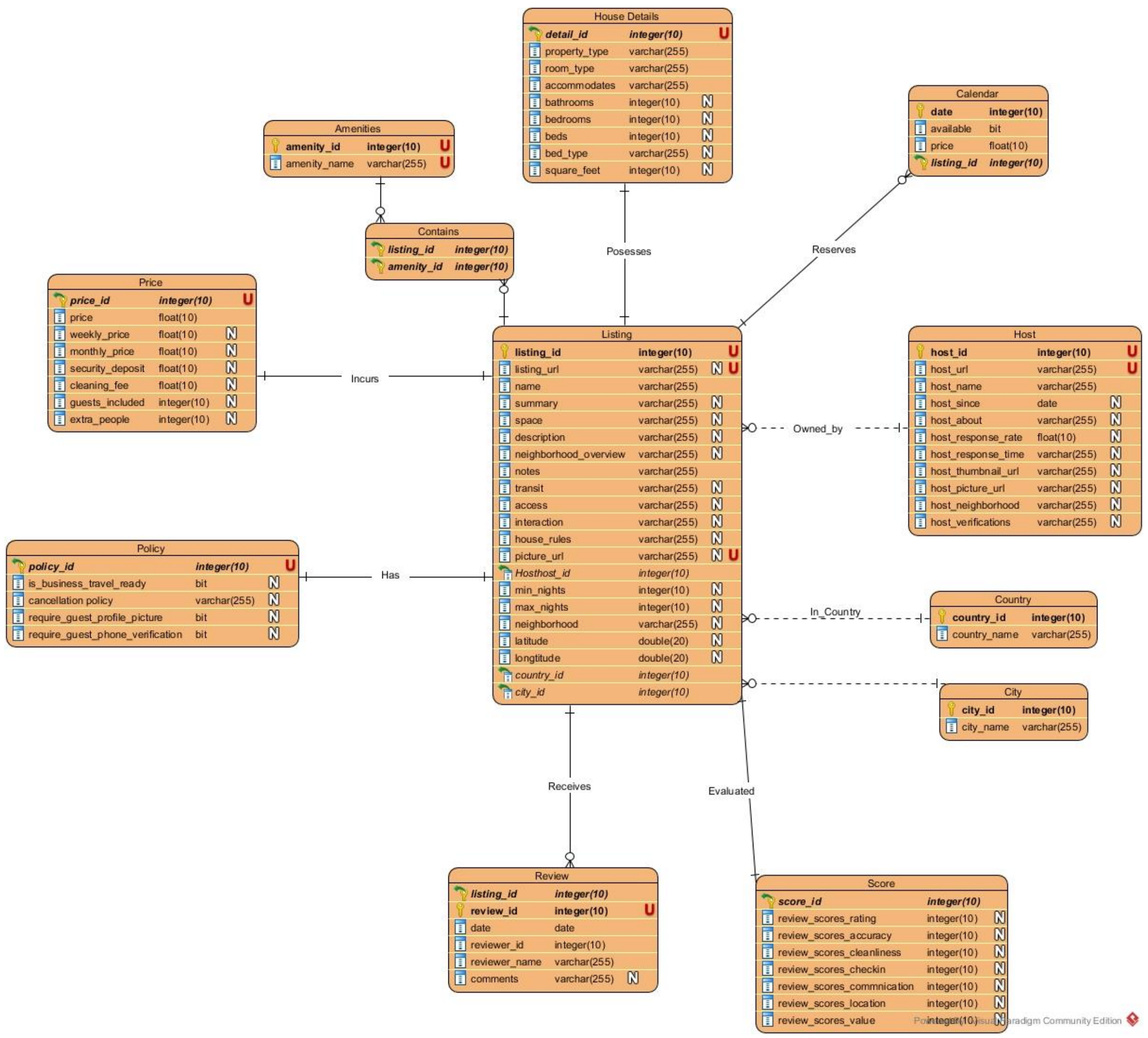
Databases Project – Spring 2019

Name: Yong Hao, Zeng Yanxi, Zhang Yuehan Team No: 49

Deliverable 1

Entity Relationship Schema

Schema



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/

Description

<Describe all the choices you made for Entities and Relationships>

Justification of the design choices & Description of the data constraints

Relationships:

- Possesses: an association among two entities *Listing* and *House Details.* It represents that house(House refers to listing item for all following contents) possesses its house details. *Listing* and *House Details* both have a key constraint and total participation, i.e. exactly one relationship. Every house (listing) possesses exactly one set of house details, and every set of house details is possessed by exactly one house.

- Owned by: an association among entities *Listing* and *Host*. This relationship represents that host owns house(s). *Listing* has a key constraint and total participation, i.e. exactly one relationship, while *Host* has total participation, which makes *Host* and *Listing* a one-to-many relationship. This means every house must have and only can have one host, and every host must have at least one house to make them a host in this system.

- City_in: an association among entities *Listing* and *City*. It represents that house locates at certain city. *Listing* has a key constraint as well as total participation, i.e. exactly one relationship. Every house locates at exactly one city, which means house must and can only reside in one city, which is obviously true. *City* has no constraints such that a city can be not located at by any house, or be located at by one to many houses. These make *City* and *Listing* a one-to-many relationship.

- Country_in: an association among entities *Listing* and *Country*. It represents that house locates at certain country. *Listing* has a key constraint as well as total participation, i.e. exactly one relationship. Every house locates at exactly one country, which means house must and can only reside in one country. *Country* has no constraints such that a country can be not located at by any house, or be located at by one to many houses. These make *Country* and *Listing* a one-to-many relationship.

- Incurs: an association among entities *Listing* and *Price*. This represents a relationship that listing(house) incurs price in this Airbnb system. *Listing* and *Price* both have a key constraint and total participation, i.e. exactly one relationship. Every house must incur and can only have one price, and a certain price must and can only be incurred by one house.

- Reserves: an association among entities *Listing* and *Calendar*. This represents the relationship that house is reserved on certain date shown on calendar. *Listing* and *Calendar* both have a total participation, while Calendar also has a key constraint, which makes *Listing* and *Calendar* a one-to-many relationship, and each instance in calendar corresponds to exactly one instance in listing. Every house must have information about its availability and price on at least one date on the calendar. Every instance in Calendar must indicate the availability and price for one corresponding Listing instance for the purpose of reservation.

- Receives: an association among entities *Listing* and *Review*. It represents that house receives review from tenants. *Review* has a key constraint as well as total participation, i.e. exactly one relationship, while *Listing* has no constraint. This makes *House* and *Review* a one-to-many relationship. Every house can attain no review, or can attains one to many reviews. Since review is a weak entity of house, if there exists a review, there must have a house and only one house for it to review.

- Has: an association among entities *Listing* and *Policy*. This represents a relationship that house has policy for tenants to obey. *Listing* and *Policy* both have a key constraint and total participation, i.e. exactly one relationship. Every house has exactly one policy for its tenant, and every policy must and only can be owned by one house.

- Evaluated: an association among entities *Listing* and *Score*. It represents that house be evaluated by tenants and receives a score. *Listing* and *Score* both have a key constraint and total participation, i.e. exactly one relationship. Every house is evaluated exactly once to achieve a score. Every score must and can only be given to one house after tenants' evaluation.

- Contains: an association among entities *Listing* and *Amenity*. This relationship represents that house contains amenities inside. There is no constraint for both entities, which makes *Listing* and *Amenity* a many-to-many relationship. Every house can contain no amenity at all, also can have one or many amenities inside. Every kind of amenity can be not contained by any house or can be contained by one to many houses.

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

Please refer to the description under the corresponding DDL code.

DDL

```
CREATE TABLE Listing
            (
            listing_id INTEGER(10),
             listing_url VARCHAR(255) NOT NULL,
             name   VARCHAR(255),
             summary VARCHAR(255),
             space VARCHAR(255),
             description VARCHAR(255),
             neighborhood_overview VARCHAR(255),
             notes VARCHAR(255),
             transit VARCHAR(255),
             access VARCHAR(255),
             interaction VARCHAR(255),
             house_rules VARCHAR(255),
             pricture_url VARCHAR(255),
            minimum_nights INTEGER(10),
            maximum_nights INTEGER(10),
            host_id INTEGER(10) NOT NULL,
             neighborhood VARCHAR(255) ,
             city_id INTEGER(10) NOT NULL,
             country_id INTEGER(10) NOT NULL,
             latitude DOUBLE(20),
             longtitude DOUBLE(20),
             PRIMARY KEY(listing_id),
             FOREIGN KEY(host_id) REFERENCES Host(host_id),
            FOREIGN KEY(city_id) REFERENCES Venue(city_id),
            FOREIGN KEY(country_id) REFERENCES Venue(country_id),
            UNIQUE(listing_url)
            )
```

The Listing Entity is translated into a table with one primary key, listing_id, and three foreign keys, host_id, city_id and country_id. With the three foreign keys, we combined the relationship Owned_by, Located_in, Located_at with Listing since each listing will have one unique host, one city and one country. As a result, these columns have NOT NULL constraint on their entry values.

Also, the listing_url is set to be unique for all instances to ensure proper display of the listing items on the website.

```
CREATE TABLE Host
        (
        host_id INTEGER(10),
         host_url VARCHAR(255) NOT NULL,
         host_name VARCHAR(255) NOT NULL,
         host_since DATE,
         host_about VARCHAR(255),
         host_response_rate FLOAT(10),
         host_response_time VARCHAR(255),
         host_thumbnail_url VARCHAR(255),
         host_neighborhood VARCHAR(255),
         host_verifications VARCHAR(255),
         PRIMARY KEY(host_id),
         UNIQUE(host_url),
        )


CREATE TABLE City
        (
        city_id INTEGER(10),
        city_name VARCHAR(255),
         PRIMARY KEY(city_id)
        )


CREATE TABLE Country
        (
        country_id INTEGER(10),
        country_name VARCHAR(255),
         PRIMARY KEY(country_id)
        )

CREATE TABLE Review
        (
        review_id INTERGER(10),
        listing_id INTEGER(10) NOT NULL,
        date DATE NOT NULL,
        reviewer_id INTERGER(10),
        reviewer_name VARCHAR(255),
        comments VARCHAR(255),
        PRIMARY KEY(review_id, listing_id),
        FOREIGN KEY (listing_id) REFERENCES Listing(listing_id)
        ON DELETE CASCADE
        )
```

The relationship Receives is combined with the Reviews table as each Reviews corresponds to a unique listing item. As Reviews is designed to be a weak entity of Listing, when one instance of Listing is deleted, the corresponding Reviews instances will be deleted as well.

```
CREATE TABLE Score
           (
           score_id INTERGER(10),
           listing_id INTEGER(10) NOT NULL,
           review_scores_accuracy INTEGER(10),
           review_scores_clean INTEGER(10),
           reciew_scores_checkin INTERGER(10),
           review_scores_communication INTERGER(10),
           review_scores_location INTERGER(10),
           review_scores_value INTERGER(10),
           PRIMARY KEY(score_id),
           FOREIGN KEY (listing_id) REFERENCES Listing(listing_id)
           )
```

The relationship Evaluated is combined with the Score table as each Score corresponds to a unique listing item.

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/

```
CREATE TABLE Policy
        (
        policy_id INTEGER(10),
        is_business_travel_ready BIT,
        cancellation_policy VARCHAR(255),
        require_guest_profile_picture BIT,
        require_guest_phone_verification BIT,
        listing_id INTEGER(10) NOT NULL,

        PRIMARY KEY(policy_id),
        FOREIGN KEY (listing_id) REFERENCES Listing(listing_id)
        )
```

The relationship Has is combined with the Policy table as each Policy instance corresponds to a unique Listing instance.

```
CREATE TABLE Price
        (
        price_id INTEGER(10),
        price FLOAT(10),
        weekly_price FLOAT(10),
        monthly_price FLOAT(10),
        security_deposit FLOAT(10),
        cleaning_fee FLOAT(10),
        guests_included INTEGER(10),
        extra_people INTERGER(10),
        listing_id INTEGER(10) NOT NULL,
        PRIMARY KEY(price_id),
        FOREIGN KEY (listing_id) REFERENCES Listing(listing_id)
        )
```

The relationship Incurs is combined with the Price table as each Price instance corresponds to a unique Listing instance.

```
CREATE TABLE House_Details
        (
        detail_id INTEGER(10),
        property_type VARCHAR(255),
        room-type VARCHAR(255),
        accommodates VARCHAR(255),
        bathrooms INTEGER(10),
        bedrooms INTEGER(10),
        beds INTERGER(10),
        bed_type VARCHAR(255),
        amenities VARCHAR(255),
        square_feet INTEGER(10),
        listing_id INTEGER(10) NOT NULL,
        PRIMARY KEY(detail_id),
        FOREIGN KEY (listing_id) REFERENCES Listing(listing_id)
        )
```

The relationship Pocesses is combined with the House_Details table as each House_Details instance corresponds to a unique Listing instance.

```
CREATE TABLE Amenities
        (
        amenity_id INTERGER(10),
        amenity_name varchar(255),
        PRIMARY KEY(amenity_id)
        )

CREATE TABLE Contains
        (
        listing_id INTEGER(10),
        amenitty_id INTEGER(10),
        PRIMARY KEY(amenity_id),
        FOREIGN KEY(listing_id)  REFERENCES Listing(listing_id)
        )
```

The table Contains is created to store the many-to-many relationship between Amenities and Listing.

```
CREATE TABLE Calender

        {
        listing_id INTEGER(10),
        date DATE,
        available BIT,
        price FLOAT(10),
        PRIMARY KEY(listing_id, date),
        FOREIGN KEY(listing_id)  REFERENCES Listing(listing_id)
        }
```

The relationship Reserves is combined with the Calendar table as each Calendar instance corresponds to exactly one Listing instance.