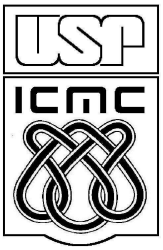

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação

Segundo Trabalho Prático

Normalização dos CVs - Airbnb

Prof. Dr. Caetano Traina Júnior
PAEs: Mariana Aya Suzuki Uchida, Erica Peters do Carmo

Bernardo Rodrigues Tameirão Santos - 12733212 - bernardotameirao@usp.br
Bruno Berndt Lima - 12542550 - brunolima674@usp.br
Vinicius Kazuo Fujikawa Noguti - 11803121 - vinicius.noguti@usp.br
Vitor da Silveira Paula - 10689651 - vitor.silveira98@usp.br

São Carlos

No presente trabalho iremos abordar as normatizações que foram realizadas nas tabelas criadas a partir dos 3 CVs informados no exercício 2.2. Para melhor entendimento as normatizações serão separadas por CSV e no fim será toda as tabelas geradas pela normalização.

1. Tabela Reviews (Reviews.csv)

A tabela criado pelo CSV é a seguinte:

```
CREATE TABLE IF NOT EXISTS Reviews (
    id BIGINT PRIMARY KEY,
    listing_id BIGINT,
    date DATE,
    reviewer_id BIGINT,
    reviewer_name VARCHAR(255),
    comments TEXT
);
```

É possível notar nesta tabela que com base no *reviewer_id* eu consigo saber o *reviewer_name*, então *reviewer_name* depende de *reviewer_id* e isso fere a terceira forma nominal. Para resolver este problema é possível retirar o *reviewer_name* da tabela e criar uma nova tabela chamada *reviewers* em que contém *reviewer_name* e *reviewer_id* sendo o *reviewer_id* a chave primária.

Então, ambas tabelas ficam da seguinte forma:

```
CREATE TABLE IF NOT EXISTS Reviews (
    id BIGINT PRIMARY KEY,
    listing_id BIGINT,
    date DATE,
    reviewer_id BIGINT,
    comments TEXT,
    FOREIGN KEY (reviewer_id) REFERENCES reviewers(reviewer_id)
);

CREATE TABLE IF NOT EXISTS reviewers  (
    reviewer_id BIGINT PRIMARY KEY,
    reviewer_name VARCHAR(255),
);
```

2. Tabela Listings (listings.csv)

A tabela criada pelo CSV é a seguinte:

```
CREATE TABLE IF NOT EXISTS Listings (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    host_id INT,
    latitude DECIMAL(10, 6),
    longitude DECIMAL(10, 6),
    room_type VARCHAR(100),
    price DECIMAL(10, 2),
    minimum_nights INT,
    number_of_reviews INT,
    last_review DATE,
    reviews_per_month DECIMAL(6, 2),
    availability_365 INT,
    number_of_reviews_ltm INT,
    license VARCHAR(100)
);
```

É possível notar nesta tabela que com base no *latitude* e *longitude* determinam os campos *neighbourhood* e *neighbourhood_group*, isso fere a terceira forma nominal e o mesmo ocorre com os campos *host_id* consegue determinar os valores *host_name* e *calculated_host_listings_count* . Então, é possível separar essa tabela em 3 tabelas.

```
CREATE TABLE IF NOT EXISTS Listings (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    host_id INT,
    latitude DECIMAL(10, 6),
    longitude DECIMAL(10, 6),
    room_type VARCHAR(100),
    price DECIMAL(10, 2),
    minimum_nights INT,
    number_of_reviews INT,
    last_review DATE,
    reviews_per_month DECIMAL(6, 2),
    availability_365 INT,
    number_of_reviews_ltm INT,
    license VARCHAR(100),
    FOREIGN KEY (host_id) REFERENCES host(host_id),
```

```
    FOREIGN KEY (latitude, longitude) REFERENCES locations(latitude,
longitude)
);

CREATE TABLE IF NOT EXISTS host (
    host_id INT PRIMARY KEY,
    host_name VARCHAR(255),
    calculated_host_listings_count INT,
);

CREATE TABLE IF NOT EXISTS locations (
    neighbourhood_group VARCHAR(100),
    neighbourhood VARCHAR(100),
    latitude DECIMAL(10, 6),
    longitude DECIMAL(10, 6),
    PRIMARY KEY(latitude, longitude)
);
```

3. Tabela Calendar (calendar.csv)

A tabela criada pelo CSV é a seguinte:

```
CREATE TABLE IF NOT EXISTS Calendar (
    listing_id INT,
    date DATE,
    available BOOLEAN,
    price DECIMAL(10, 2),
    adjusted_price DECIMAL(10, 2),
    minimum_nights INT,
    maximum_nights INT,
    PRIMARY KEY (listing_id, date)
);
```

Essa tabela possui uma chave primária composta, ou seja, dois campos definem a chave primária, mas três campos dessa tabela dependem apenas de um dos campos da chave primária. Ou seja, os campos *price*, *minimum_nights* e *maximum_nights* depende apenas do *listing_id* e isso fere a segunda forma nominal e para resolver isso, basta juntar esses campos com a tabela listing após a normalização, tendo as seguintes tabelas.

```
CREATE TABLE IF NOT EXISTS Calendar (
    listing_id INT,
    date DATE,
    available BOOLEAN,
    adjusted_price DECIMAL(10, 2),
    PRIMARY KEY (listing_id, date),
```

```
    FOREIGN KEY (listing_id) REFERENCES Listings(id)
);

CREATE TABLE IF NOT EXISTS Listings (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    host_id INT,
    latitude DECIMAL(10, 6),
    longitude DECIMAL(10, 6),
    room_type VARCHAR(100),
    price DECIMAL(10, 2),
    minimum_nights INT,
    maximum_nights INT,
    number_of_reviews INT,
    last_review DATE,
    reviews_per_month DECIMAL(6, 2),
    availability_365 INT,
    number_of_reviews_ltm INT,
    license VARCHAR(100)
);
```

4. Mapeamento Final:

```
CREATE TABLE IF NOT EXISTS Calendar (
    listing_id INT,
    date DATE,
    available BOOLEAN,
    adjusted_price DECIMAL(10, 2),
    PRIMARY KEY (listing_id, date),
    FOREIGN KEY (listing_id) REFERENCES Listings(id)
);

CREATE TABLE IF NOT EXISTS Listings (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    host_id INT,
    latitude DECIMAL(10, 6),
    longitude DECIMAL(10, 6),
    room_type VARCHAR(100),
    price DECIMAL(10, 2),
    minimum_nights INT,
    maximum_nights INT,
    number_of_reviews INT,
    last_review DATE,
    reviews_per_month DECIMAL(6, 2),
    availability_365 INT,
    number_of_reviews_ltm INT,
```

```
    license VARCHAR(100)
);

CREATE TABLE IF NOT EXISTS host (
    host_id INT PRIMARY KEY,
    host_name VARCHAR(255),
    calculated_host_listings_count INT,
);

CREATE TABLE IF NOT EXISTS locations (
    neighbourhood_group VARCHAR(100),
    neighbourhood VARCHAR(100),
    latitude DECIMAL(10, 6),
    longitude DECIMAL(10, 6),
    PRIMARY KEY(latitude, longitude)
);

CREATE TABLE IF NOT EXISTS Reviews (
    id BIGINT PRIMARY KEY,
    listing_id BIGINT,
    date DATE,
    reviewer_id BIGINT,
    comments TEXT,
    FOREIGN KEY (reviewer_id) REFERENCES reviewers(reviewer_id)
);

CREATE TABLE IF NOT EXISTS reviewers  (
    reviewer_id BIGINT PRIMARY KEY,
    reviewer_name VARCHAR(255),
);
```