

Introduction to Databases



Project

Designing the database of a travel reservation system

1. Introduction

The goal of this project is to assess your knowledge of the main notions presented in classroom. The project must be implemented by **groups of max 2 students**. The project is divided in two parts:

- The first part consists in designing a relational database for the given application context, importing the data of a given dataset into a SQLite database and querying the data in SQL.
- The second part (OPTIONAL) consists in designing a labelled property graph importing the data into Neo4j and querying the data in Cypher.

The project is meant to be done during class hours on November 19th and 26th (the last two classes of the database course).

The deadline to submit the work is **Friday November 29th**.

Here is what **you have to submit**:

- A report in PDF containing the answers to all the questions and exercises, except the (SQL/Cypher) queries (max 10 pages).
- The file `.db` containing the SQLite database with all the data.
- A file `.sql` with all the SQL queries.

If you develop the optional part, you'll also submit:


- The directory containing the Neo4j database with all the data.
- A `.txt` file with all the Cypher queries.

Please compress all the files in a **zip archive** and send it to gianluca.quercini@centralesupelec.fr. In the email, **specify the name and family name of the group**. I'll send you a confirmation when I receive the project, so if you don't receive any confirmation, do not hesitate to contact me.

2. Application Context

We intend to manage the data of a travel reservation system with clients all over the world. Upon

registration, customers are automatically given a numeric identifier and they are asked to indicate their first and family names, their gender, date of birth, a phone number, an email address and their country of residence. Any customer can book a trip that includes the reservation of **one or more flights** and, possibly, **one or more hotels**.

 Alice wants to fly from Paris, France to New York City (NYC), USA and she intends to stay in NYC for 10 days. Her trip includes two flights: an outbound flight from Paris to NYC and an inbound flight from NYC to Paris; and an hotel in NYC.

A flight is operated by an airline company, of which the system keeps its name (e.g., *British Airways*), the country where the airline is incorporated and, when available, its IATA code (e.g., *BA*, a two-letter code identifying the airline), its ICAO code (e.g., *BAW*, a three-letter code identifying the airline) and alternate name or alias (e.g., *British*).

A flight connects two airports, each having a name (e.g., *London Heathrow Airport*), and, possibly, a IATA (e.g., *LHR*) and ICAO code (e.g., *EGLL*); an airport serves a specific location (e.g., *London, UK*) and its precise position is given by its geographic coordinates (latitude and longitude).

A flight connecting two airports at specific departure and arrival times is identified by a flight number. Two flights operated by two different airline companies cannot have the same flight number, but the same flight number can denote two flights operated by the

same airline company on different days.


✋ Emirates flight EK074 leaves Paris, France at 10 AM and arrives at Dubai, UAE at 7:40 PM (regardless of the departure day).

For each flight booked by a customer, the system keeps the seat number, the travel class (e.g., economy or business), the price and the date of the flight. Usually, airlines include details on the type of aircraft they plan to use on their flight schedules; these details include the name of the aircraft (e.g., *Boeing 787-8*) and, when available, the IATA code (e.g., 788, a unique three-letter identifier for the aircraft) and the ICAO code (e.g., B788, a unique four-letter identifier for the aircraft).

The system maintains a list of hotels, with their names, addresses and an average review score, which is a real number denoting the average grade assigned to the hotel by its customers. Customers can write a review for an hotel; in which case the system stores the text of the review, the date and its author. When a customer books an hotel, the system keeps the price paid, the check-in and check-out dates and whether the breakfast is included.

3. The Dataset

You can download the dataset by clicking [here](#). The dataset consists of 7 CSV files: *aircrafts.csv*, *airlines.csv*, *airports.csv*, *hotels.csv*, *customers.csv*, *hotel_bookings.csv*, *flight_bookings.csv*. The separator character in each file is the tab character ('\t').

 Take some time to look at the content of the files to understand the structure that your tables will have.


4. Creation of a Relational Database

You'll now proceed to the definition of a relational database for our travel reservation system. First, you need to define the **logical schema** and then you'll define the tables that compose the database.

4.1 The Logical Schema

Before defining the logical schema of the database, answer the following questions:

- a. Can you use the name of the hotel as a primary key? Justify your answer.
- b. Can you use the flight number as a primary key to identify a flight? Justify your answer and, in case of a negative answer, propose a solution.
- c. Knowing that it is unlikely that two reviews have the same textual content, would you use it as a primary key? Justify your answer.
- d. Knowing that the IATA code uniquely identifies an aircraft, would you choose it as a primary key for the entity `Aircraft`? Justify your answer.


 Read carefully the context of the application.


- e. Consider the following statement:


It is possible to express in the ER diagram the constraint that a customer cannot review an hotel if she never stayed at that hotel

Is it true or false? Justify your answer.

Exercise 1. Given the application context and the dataset, propose an Entity-Relationship diagram describing the logical model of a relational database.

 Specify all the attributes for each entity and relation.


 For each entity, underline the attributes composing the primary key.

 For each relation, clearly indicate the minimum and maximum cardinality.

4.2 Normalization

Exercise 2. Translate the logical schema into a collection of tables. For each table:

- Indicate its name and the names of the columns (but not their types).
- Underline the columns that are part of the primary key.
- Indicate the entity in the ER diagram to which the table corresponds.

 To make sure you choose the right data types for the columns, you can also check the values in the dataset.

Which **normal form** are your tables in?

Exercise 3. For each table:

- Indicate all the **functional dependencies** that you can find.
- Indicate the normal form that the table satisfies. Justify your answer.

Normalize your tables.

Exercise 4. Normalize each table up to the **Boyce-Codd Normal Form (BCNF)**.

4.3 The Physical Schema

You can now define the **physical schema** of your database.

Exercise 5. Write the SQL code to create the tables. For each table:

- Indicate the **primary key**.
- Indicate the **foreign keys**.
- Indicate NOT NULL and UNIQUE constraints,

if needed.

4.4 Database Creation and Data Import



Given all the installation problems with MySQL, it is highly recommended that you use SQLite as a DBMS!

SQLite is a software library written in C that implements a Relational DataBase Management System (RDBMS). SQLite does not need you to install a server. A database is simply created and managed by calling the functions of this library. Typically, the database is just a file stored in your computer.

Create a Database in SQLite

In order to create a database in SQLite, you just need to download and install [DB Browser for SQLite](#), a graphical interface to any SQLite database.

After the installation:

- Open the program and click on **New Database**;
- The program will ask you to specify a name for the database file and the folder where you want to store it.



Suggested **extensions** for the database file are .db, .sqlite, .sqlite3, .db3.

- The program will display a pop-up window where you can create the tables. Alternatively, you can

directly execute the SQL code that you wrote before to create your tables.

Exercise 6. Create a SQLite database with the tables that you defined in Exercise 5.

Data Import

In order to import the data into the database, you can write a Python code that reads the input CSV files and populates the tables of the database.



When importing the data into the database, you might get some errors concerning the violation of some constraints (primary key or foreign key). In this case you might want to:

- check the validity of the schema of the database and its constraints.
- if the schema and the constraints seem to be valid, think of how you can fit your data into the schema

Here is a sample Python code that:

- connects to a SQLite database contained in a file called *my_db.db*.
- reads a file *test.csv*
- imports its content into a table named `test`

```
1 | import sqlite3
2 | import csv
```

```
3  import datetime
4
5  # Connect to the database cc
6  conn = sqlite3.connect('my_c
7
8  # Create an object cursor, u
9  # database
10 cursor = conn.cursor()
11 try:
12     # Open the input CSV fil
13     with open('test.csv', 'r
14         # Delete all data fr
15         cursor.execute("DELE
16         # Commit the modific
17         conn.commit()
18         csv_reader = csv.rec
19         # Skip the header of
20         next(csv_reader)
21         for line in csv_reac
22             # The values of
23             string_column =
24             # The values of
25             integer_column =
26             # The values of
27             float_column = f
28             # The values of
29             date_column = dc
```

```
30         # Insert the val
31         cursor.execute("
32             (string_cc
33 # If anything goes wrong, rc
34 except sqlite3.Error as errc
35     # Print the reason of th
36     print("Error while loadi
37 # Whether an exception is rc
38 # instructions are executed.
39 finally:
40     # Commit the modificatic
41     conn.commit()
42     # Closes the connection
43     cursor.close()
44     conn.close()
45     print("The connection to
```

Queries

Exercise 7.

Write the following queries in SQL:

1. Count the number of rows in each table.
2. Get the average ticket price on Air France flights.
3. Count the number of customers by country. Sort by decreasing order.

4. Select the names of the airports in Paris, France. Sort by name in alphabetical order.
5. Select the name of the cities (city and country name) with the most airports. Sort by city and country name in alphabetical order.
6. Select the name of the airline companies that use an Airbus A300. Sort by name in alphabetical order.
7. Select the identifier, first and family names of all customers who flew to Sydney, Australia. Sort by identifier.
8. Select the identifier, name, city and country of the busiest airport (the one with more outbound and inbound flights). Sort by identifier.
9. Select the average price in the economy class.
10. Select the average price in the business class.
11. Select the name, city and country of the destination airport of french customers. Sort by name in alphabetical order.
12. Select the destination cities (specify city and country name) preferred by women. Sort by city and country.
13. Select the destination cities (specify city and country name) preferred by men. Sort by city and country.

14. Count the number of customers by country flying to Paris. Sort the result in decreasing order.
15. Count the number of hotels by city. Sort the result in decreasing order.
16. Determine the amount of money spent by Tatiana REZE in flights.

Exercise 8.

1. Write a query to get all the information of a customer with a given family name. Run the query multiple times and note the average running time of the query.
2. Create an index on the column containing the family name of a customer.
3. Rerun the same query multiple times and note the average running times of the query.
4. Do you observe any difference? Can you explain what is going on here?

5. Optional Part: Neo4j

- Define a labelled property graph model for the database of the travelling agency.
- Write in Cypher all the queries of the Exercise 7.

- Import the data into Neo4j and test the queries.