Databases — Part 3

The objective of this sequence of works is to understand the concept of databases and to handle it with the SQL language: we will manipulate this language thanks to the `sqlite3` module in Python.

In the last work, we have learn the basics of SQL. In this work, we will see in more detail what is a join, and we will manipulate a database with the `sqlite3` Python module.

1 Complements on join

Last week, we have seen that if we have Tables 1 and 2, then, if we want to print the title, the author and the literary genre of all the books in our library, we can write the following query:

```
SELECT title, author, genre FROM books, genres WHERE books.genre_id=genres.
    identifier;
```

Id.	Genre
1	Poetry
2	Novel
3	Theater

Table 1: Work 13, Exercise 1: The 3 literary genres.

Id.	Author	Title	Genre id.
1	Baudelaire	Les fleurs du mal	1
2	Steinbeck	The Grapes of Wrath	2
3	Molière	L'avare	3
4	Corneille	Le Cid	3
5	Hugo	Les misérables	2
6	Prévert	Paroles	1

Table 2: Work 13, Exercise 1: The 6 books.

I introduced this query as a "join operation". In fact, this is a Cartesian product in which we only keep the rows where `books.genre_id=genres.identifier`. But this is equivalent to the "real" join operation, whose syntax would be:

```
SELECT title, author, genre
FROM books JOIN genres ON books.genre_id=genres.identifier;
```

This join operation probably needs to be explained in more details. There are different types of joins in SQL (natural join, theta join, and equijoin, which is a theta join using the $=$ operator). We don't need to cover them all, we will only cover the equijoin. An equijoin is an operation which takes two tables and outputs one table.

Let's start with two tables R and S with the fields $(A1, A2, \ldots, Am)$ for R and $(B1, B2, \ldots, Bn)$ for S. We can output a new table T where the fields Ai and Aj are equal, for some i and j. This table has the fields $(A1, A2, \ldots, Am, B1, B2, \ldots, Bn)$, and for each row of R, we will search in S if there are some rows where $Ai = Bj$. When this is the case, we add a new row which is the concatenation of this row from R and the corresponding row from S.

On this example, let's do the equijoin `books.Genre_id=genres.Id`. The resulting table is:

books.Id	Author	Title	Genre_id	genres.Id	Genre
1	Baudelaire	Les fleurs du mal	1	1	Poetry
2	Steinbeck	The Grapes of Wrath	2	2	Novel
3	Molière	L'avare	3	3	Theater
4	Corneille	Le Cid	3	3	Theater
5	Hugo	Les misérables	2	2	Novel
6	Prévert	Paroles	1	1	Poetry

Notice that in this table, to distinguish among the different fields with the same name, I have added, for clarity, the name of the table from which this field was extracted.

Let's see that on another example. In Tables 3 and 4, we show two tables. In Table 5 we show the table resulting from the join `B=C` (we could also write `R.B=S.C` but here there is no ambiguity).

```
SELECT * FROM R JOIN S ON B=C;
```

A	B
a	b
d	b
b	b
c	a
c	d

Table 3: Relation R.

C	D	E
b	c	d
b	c	e
a	d	b
a	c	c
c	a	d

Table 4: Relation S.

A	B	C	D	E
a	b	b	c	d
a	b	b	c	e
d	b	b	c	d
d	b	b	c	e
b	b	b	c	d
b	b	b	c	e
c	a	a	d	b
c	a	a	c	c

Table 5: Relation R join S on $B = C$.

2 The hotels example

This week, we will work on a hotel database[1]. You will need the three following files (make sure to put them in the same folder):

http://www.barsamian.am/2023-2024/S6ICTA/TP14_Hotels_database.py
http://www.barsamian.am/2023-2024/S6ICTA/TP14_Hotels_creations.sql
http://www.barsamian.am/2023-2024/S6ICTA/TP14_Hotels_insertions.sql

Once you downloaded them, you can run the python file once. The first run will take a little bit of time, since python needs to perform all the SQL commands. Once it's done, the database will be created (stored inside a file `TP14_Hotels_database.sq3`) and it will be much faster.

Here is a global view of our database :

- hotels: id, name, city, stars
- rooms: id, #hotel_id, floor_number, type, price_per_night_excl_tax
- clients: id, surname, first_name
- occupancies: id, #client_id, #room_id, #hotel_id, date_from, date_to
- bookings: id, #client_id, #room_id, #hotel_id, date_from, date_to

Each hotel has a name, is located into a given city, and has a given number of stars (1 to 4).

Each room is located into a hotel, at a given floor, has a type (simple, double, triple, suite, other), and a price (per night, excluding taxes).

The clients are listed by surname and first name.

Then, we have the occupancies and the bookings.

In the following questions, if nothing specific is asked, it means that we want all the fields from the corresponding table(s).

1. Give the list of the names of hotels, with their city.
2. Give the names of hotels having 3 stars or more.
3. Give the list of hotels for which there is at least one booking.

   Hint: give the list of hotels associated with all the bookings ; then use the `DISTINCT` clause to display each hotel only once.

[1] Source : https://remieyraud.github.io/index.html%3Fp=129.html

4. Give the list of clients that occupied at least one room.

   Hint: the request is very similar to the previous question.

   Use the previous request together with the `EXCEPT` clause to give the list of clients that did not occupy any room.

5. For each hotel, give the total number of rooms.

   Hint: there are at least two ways to do that. First, after a request `cursor.execute(request)`, you can ask, in Python, `cursor.fetchall()`, this will return a list of all the results. Or, in SQL, you can execute the command:

   ```
   SELECT COUNT(field) FROM table;
   ```

6. Who were the clients of the "Hôtel des voyageurs", in Nice?

7. How many triple rooms are in the "Hôtel des ambassadeurs", in Grenoble?

8. For each client, list the name and the city of the hotels in which he occupied a room.

The following questions are really harder, but you can try to at least find partial results.

9. List the rooms from the "Hôtel de la gare", in Bordeaux, that are available for the nights from 2014, April the 12th to 2014, April the 17th.

10. If the taxes represent 19.6% of the price excluding taxes, what will be the amount (taxes included) of each stay of Jean Némarre, that he had to pay before 2015, January the 21st? (A client only pays a stay at the end of it).

11. What are the rooms that have been occupied for more than 40 days?

    Hint: what allows us to uniquely talk about a room? You can find that information in the `.sql` file that create the tables.

3 The hotels example — Bonus questions

1. Remove all the clients that never appear nor in an occupancy nor in a booking.

2. Add a star to the "Hôtel terminus" from Grenoble.

3. Diminish by 12% the price of all the rooms in Paris.

4. (a) Add two columns in the table `hotels`, to store the annual leave of each hotel. These two columns must store the date when the leave starts, and when it stops.

   (b) Modify the data as follows: hotels in Nice are closed from January, the 22nd to February, the 10th; hotels in Grenoble are closed from August, the 1st to August, the 18th; other hotels are closed the 21 first days in November.

5. (a) Create a table `invoices` that has the following fields: id, date_invoice, occupancy_id, price_excl_taxes and has_been_paid. Explain why these data are enough to print an invoice for a stay with the name of the client, the dates of the stay, etc. If you wish to add other fields you may, but you must explain why.

   (b) Let's go back in time in 2014, April's fool's day. For each finished stay (at that date), create a line in this table, with this date as invoice date.