Information Systems and Databases 2022/2023

Project Assignment - Part 3

The third assignment consists of the development of SQL queries, integrity constraints and the creation of a web application prototype and OLAP queries.

1. Database Loading

Use the supplied **`schema-part3.sql`** file to create the database. The table schema provided should be consistently filled with the records required to ensure that all SQL queries, to be developed below, have a **non-empty result**. Record creation and the database loading can be carried out through whatever method you find more adequate, including adapting the loading script of the previous project assignment.

2. Integrity Constraints

Write the code to implement the following integrity constraints - which are presented assuming your knowledge of previous phases - with the SQL procedural extensions (Stored Procedures, Triggers, and Deferred Constraint checking) in:

(IC-1) Every Sailor is either Senior or Junior.

(IC-2) The take-off and arrival dates of Trips for the same reservation must not overlap (i.e., one Trip cannot take off before the arrival of another).

The integrity constraints **definable** without resorting to procedural extensions (Stored Procedures and Triggers), should be implemented using other mechanisms if appropriate. However, mechanisms such as ON DELETE CASCADE and ON UPDATE CASCADE **are not allowed**.

3. SQL

Present the most succinct Standard SQL[1] query for each of the following questions. If appropriate, you can use the view created previously.

1. Which country has more boats registered than any other?

[1] You cannot use SQL instructions that are not part of the standard (such as the **LIMIT** instruction or vendor specific extensions).

2. List all the sailors that have at least two certificates.
3. Who are the sailors that have sailed to every location in 'Portugal'?
4. List the sailors with the most skipped trips.
5. List the sailors with the longest duration of trips (sum of trip durations) for the same single reservation; display also the sum of the trips duration.

4. View

Create a view that summarizes the most important information regarding boat trips, combining information from different tables in the relational model. The view should have the following schema:

```
trip_info(
   country_iso_origin, country_name_origin,
   country_iso_dest, country_name_dest,
   loc_name_origin,
   loc_name_dest,
   cni_boat,
   country_iso_boat, country_name_boat,
   trip_start_date)
```

Where:

- `country_name_origin: Foreign Key(Country)`
- `country_name_dest: Foreign Key(Country)`
- `country_name_boat, cni_boat: Foreign Key(Boat)`

5. Application Development

Create a web application prototype to demonstrate to the possible prospect clients. Your application should be developed using Python CGI scripts and HTML pages that allows users to:

a) List, create and remove sailors (including junior and senior)

b) List, create and remove reservations

c) Authorise/De-authorise sailors for reservations

d) List, register, and remove trips (including listing the available locations)

The solution should prize security, preventing attacks by SQL INJECTION. Additionally, the **atomicity of related operations** in the database should be ensured using transactions. The graphical aspect of the application is not fundamental (the use of CSS to improve the look of the screens is not mandatory).

6. Data Analytics Queries

Using the view from Question 4, write two SQL queries that allow you to analyze the total number of trips according to different groups depending on:

1. The start date (i.e., per year, per month independently of year, and per exact date);
2. The location of origin (i.e., per location within countries, per country, and in total).

The submitted solution must use ROLLUP/CUBE/GROUPING SETS instructions, or the UNION of GROUP BY clauses.

7. Indexes

Present the SQL index(es) creating instruction(s) to improve the querying times for each of the cases listed below, explaining what are the operations that would be optimized and how.

Indicate, with proper justification, what type of index(es), over which attribute(s) and over which table(s), it would make sense to create, in order to speed up each query execution. Assume that the size of the tables exceeds the available memory by several orders of magnitude.

Assume that there are no indexes over the tables, aside from those implicit when declaring primary and foreign keys.

7.1 - List the names of all boats of a given class and registered after given year:

```
SELECT boat.name
FROM boat
WHERE year >= <some year>
AND boat_class = <some class>;
```

7.2 - Count the number of trips of boats by country:

```
SELECT boat_country, COUNT(*)
FROM trip
GROUP BY boat_country
```

Report

The project will be graded based on the report submitted and the discussion. The report should contain all answers to the items requested above. In the table below the points awarded to each portion of the work are listed.

Item	**Grading (0-20)**
SQL	5.0
Integrity Constraints	2.0
Application	6.0
Indexes	3.0
Data analytics	4.0

Although not fundamental, the following aspects may be used to appreciate your work:

Polished graphical aspect of the application	+1.0
Pagination logic for lists	+1.0

The report should begin with a cover page with the title "**Database Project, Part 3**", the **name and number of the students, the contribution from each member in relative percentage, along with** the **effort (in hours)** that each member put into the project, the **group number**, the group **shift** and the lab teacher's name. Besides the cover page, the report should have, at most, **8 pages**.

Delivery

The submission in Fénix should be a **zip** file with the following structure:

reportGG.pdf (where GG is the group number)	The report in **pdf** where **GG** is the group number, containing an **explanation of the architecture of the web application with a link to a working version**, the **relations between the various files** and the **indexes**. You should not include the instructions used to populate the database. The report does not need to include the OLAP component.
queries.sql	File with the SQL queries.
populate.sql	File to populate the database with the test data.

`output.txt`	File with the output of each query. Please make sure that the output of each query is not empty.
ICs.sql	File to create the integrity constraints (triggers & stored procedures).
view.sql	File with the instructions to create the view.
analytics.sql	File with the data analytics queries.
web/	Folder with the Python and HTML files. ⚠ Groups must make sure to have the application working online at **web.ulisboa.pt** until the end of the semester.

The project must be delivered in ZIP formatted with the name `delivery-03-GG.zip`[2] (where **GG** is the group number), through Fénix until 23h59 of the delivery date.

Note: Penalties apply to groups that do not follow the delivery instructions. Evaluation elements that are not found in the prescribed as above **will not be taken into account for grading purposes**.

[2] ⚠ The file format must be exclusively ZIP or GZ. Other archive formats (such as RAR) will not be accepted.