Database Project

Part 3

Inês Lucas (93383) (33.(3)%) (12h)

Ivan Figueiredo (93386) (33.(3)%) (12h)

Miguel Fernandes (92836) (33.(3)%) (12h)

Group 09

Shift SIBD-2PB05

Professor Francisco Regateiro

1 Integrity Constraints

1.1 Notes about IC-1

In order to ensure this integrity constraint, several triggers/procedures were created.

Upon inserts or updates on tables Sailor, Junior or Senior, our triggers only produce a warning and prevent the situation from happening, in case a sailor is not present into one (and only one) of the tables Junior or Senior.

Upon deletes on tables Senior or Junior, our trigger, instead of raising an exception, it actually deletes the deleted sailor from the table Sailor. This is crucial for our web application design. There is no need to create a trigger upon a delete on table Sailor since, if a sailor was deleted from table Sailor without being removed from the table Senior or Junior, a foreign key constraint would be violated and it would give rise to an error.

1.2 Notes about IC-2

For this integrity constraint, we created a trigger that was also capable of raising an exception in case the takeoff/arrival of a trip was before/after the start/end date of the respective reservation and also if the takeoff was before the arrival date. The remaining conditions of this trigger concerned the avoidance of overlapping trips: if the takeoff and arrival dates of a certain trip are given by, respectively, A and B, we guarantee that no other trip for that reservation can take off between A and B and, in the case where the takeoff date of the new trip is earlier than A, we ensure that its arrival date cannot be later than A.

2 Web Application

Our web application has a main page, `http://web2.tecnico.ulisboa.pt/ist193386/app.cgi`, created with the `app.cgi` file. In this page, there are three buttons: "Sailors", "Reservation and respective Trips" and "Queries Output", as it can be seen in Figure 1.

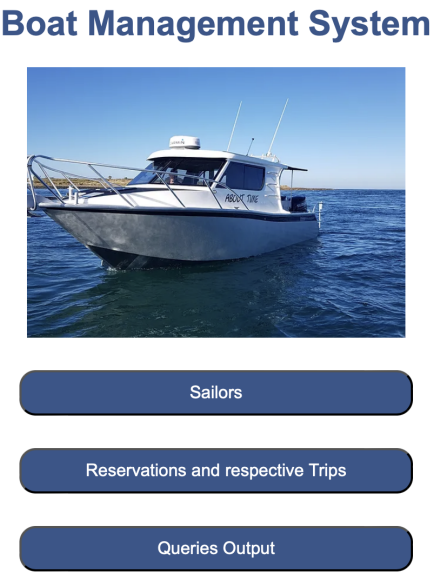


Figure 1: Main Page of Web Application

2.1 List, create and remove sailors

By clicking on "Sailors", a new web page appears, as it can be seen in Figure 2, built based on the instructions in the `sailors.cgi` file.

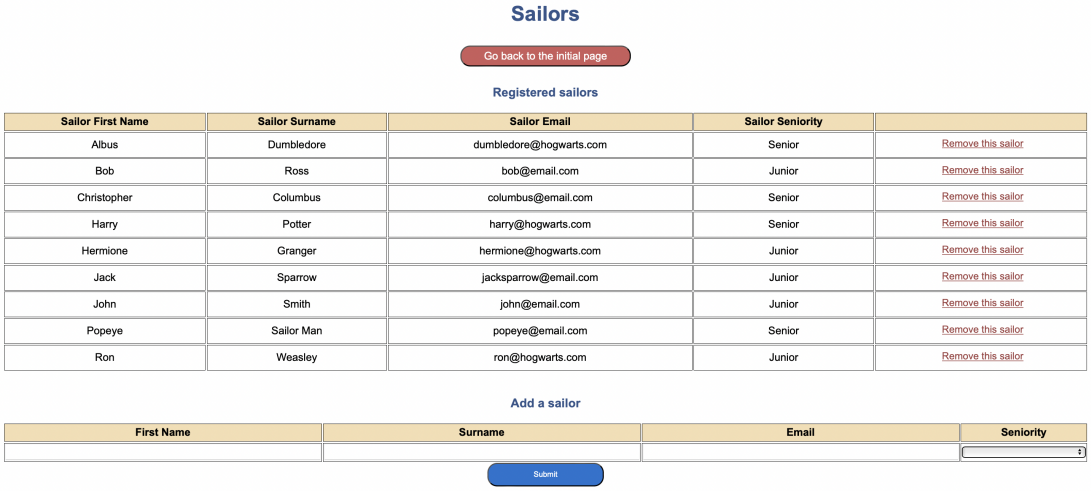


Figure 2: "Sailors" Web Page

In this page, there is a list of all existing sailors in the database, along with their respective seniority. Besides, it is possible to create a new sailor, under the form "Add a sailor", for which all the attributes need to be filled, thanks to `NOT NULL` requirements and to one of the triggers created for IC-1, which assures that whenever a new sailor is added, it must be either Senior or Junior.

In order to remove a sailor, the user simply needs to click on the button "Remove this sailor" corresponding to the sailor to be deleted.

2.2 List, create and remove reservations

From Figure 1, by clicking on "Reservations and respective Trips", the web page presented in Figure 3 appears.

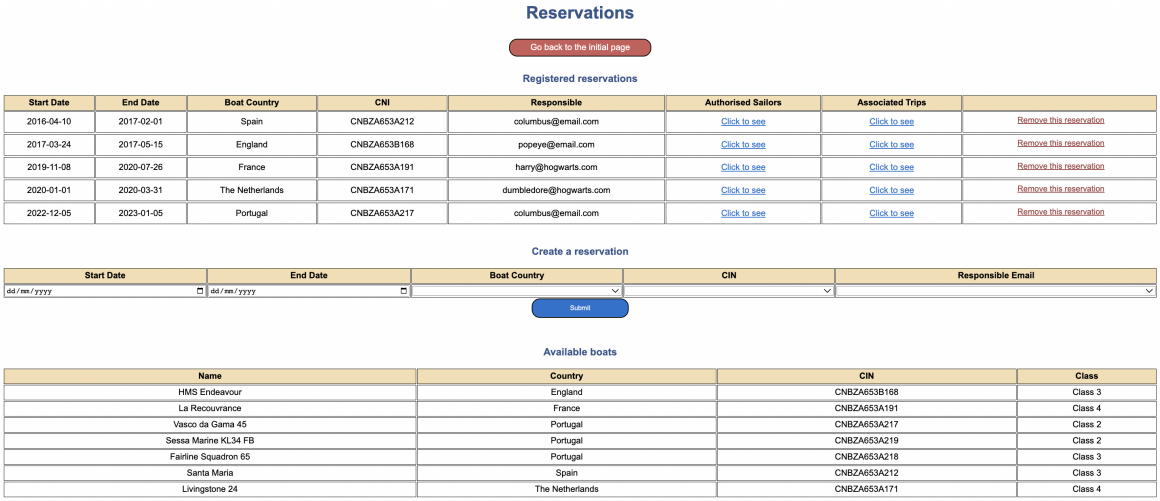


Figure 3: "Reservations" Web Page

This web page is created by the `reservations.cgi` file. Here, the user is shown a list of all existing reservations. In the same way as in the "Sailors" web page, it is possible to create and remove a reservation.

Under "Create a reservation", the "Boat Country", "CIN" and "Responsible Email" attributes appear as a dropdown in order to avoid the user to insert invalid values. Note that the available values for "Responsible Email" are the emails of the senior sailors, since the junior sailors cannot be responsible for any reservation. Once a reservation is created, its responsible sailor becomes automatically an authorised sailor for that reservation, since the responsible of a reservation must be authorised for it.

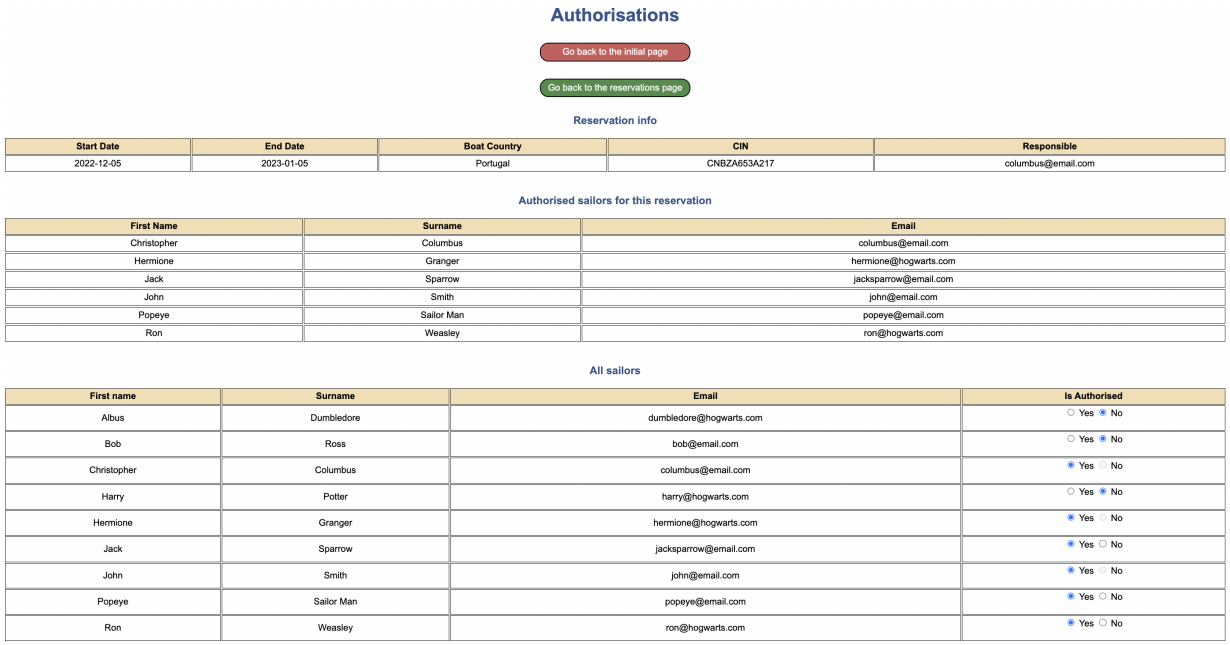
In order to remove a reservation, the user has to click on the button "Remove this reservation" associated to the reservation to be deleted. A reservation cannot be removed if there is any trip associated to it. So the user has to start by removing the trips associated to a reservation before removing the reservation itself. The same happens with the authorised sailors: if any sailor, besides the responsible one, is authorised for a reservation, then the user needs to de-authorise them before

removing the reservation, otherwise an error will be raised. If the only authorised sailor for a reservation is the responsible one and if there is no trip associated to that reservation, then the reservation can be removed by clicking on the respective button.

A list of the available boats is provided at the bottom of this web page in order to ease the access to the "CIN" and "Boat Country" values, necessary to create a reservation.

2.3 Authorise and deauthorise sailors for reservations

By clicking on the button "Click to see" corresponding to a certain reservation, under the column "Authorised Sailors" in Figure 3, the user is redirected to another web page, as shown in Figure 4.



Authorisations

Go back to the initial page

Go back to the reservations page

Reservation info

Start Date	End Date	Boat Country	CIN	Responsible
2022-12-05	2023-01-05	Portugal	CNBZA653A217	columbus@email.com

Authorised sailors for this reservation

First Name	Surname	Email
Christopher	Columbus	columbus@email.com
Hermione	Granger	hermione@hogwarts.com
Jack	Sparrow	jacksparrow@email.com
John	Smith	john@email.com
Popeye	Sailor Man	popeye@email.com
Ron	Weasley	ron@hogwarts.com

All sailors

First name	Surname	Email	Is Authorised
Albus	Dumbledore	dumbledore@hogwarts.com	○ Yes ◉ No
Bob	Ross	bob@email.com	○ Yes ◉ No
Christopher	Columbus	columbus@email.com	◉ Yes ○ No
Harry	Potter	harry@hogwarts.com	○ Yes ◉ No
Hermione	Granger	hermione@hogwarts.com	◉ Yes ○ No
Jack	Sparrow	jacksparrow@email.com	◉ Yes ○ No
John	Smith	john@email.com	◉ Yes ○ No
Popeye	Sailor Man	popeye@email.com	◉ Yes ○ No
Ron	Weasley	ron@hogwarts.com	◉ Yes ○ No

Figure 4: "Authorisations" Web Page

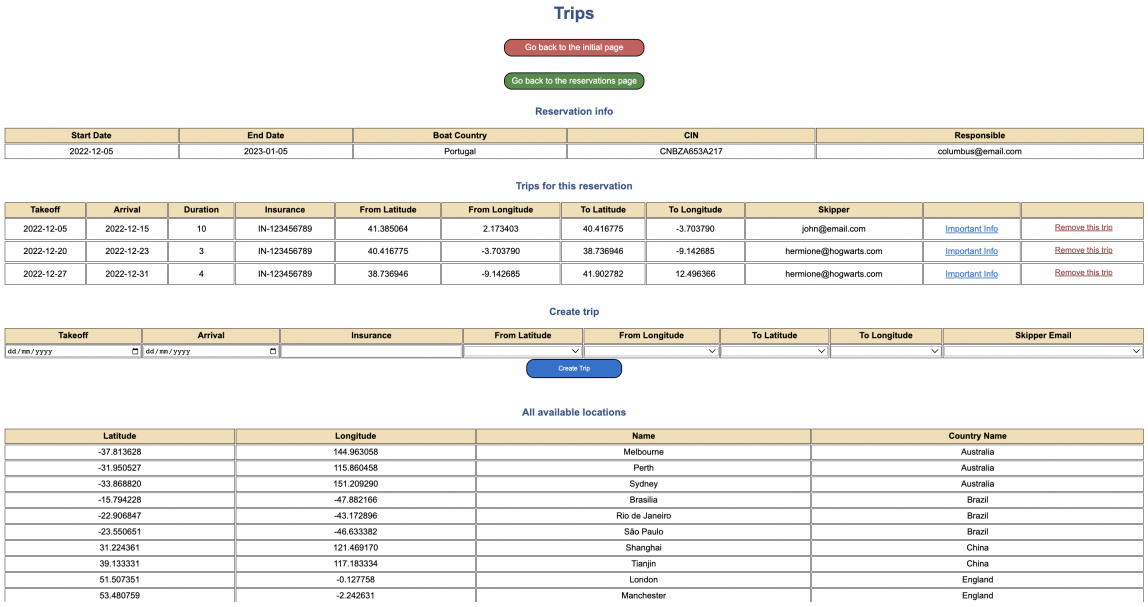
This web page, created by the file `authorisations.cgi`, firstly states the information about the reservation that was clicked on, namely the start and end dates of the reservation, the email of the responsible sailor and the CIN of the boat and the country in which it was registered.

Right below, there is a list of the authorised sailors for that specific reservation, and contains the first name, surname and email of each one.

And finally, the third table lists all sailors, in which it is possible to authorise or deauthorise sailors for this specific reservation, by clicking on "Yes" or "No", respectively, under the column "Is Authorised". Note that the responsible sailor of the reservation can never be deauthorised. In this example, the responsible sailor is the one with email given by `columbus@email.com`. So, if the user tries to deauthorise him by clicking on the button "No", it will not produce effect, since that button is locked. Regarding any other sailor, they can only be deauthorised if they are not skippers of any trip associated to that reservation.

2.4 List, register, and remove trips

By clicking on the button "Click to see" corresponding to a certain reservation, under the column "Associated Trips" in Figure 3, the user is redirected to another web page, present in Figure 5.



Trips

Go back to the initial page

Go back to the reservations page

Reservation info

Start Date	End Date	Boat Country	CIN	Responsible
2022-12-05	2023-01-05	Portugal	CNBZA653A217	columbus@email.com

Trips for this reservation

Takeoff	Arrival	Duration	Insurance	From Latitude	From Longitude	To Latitude	To Longitude	Skipper		
2022-12-05	2022-12-15	10	IN-123456789	41.385064	2.173403	40.416775	-3.703790	john@email.com	Important Info	Remove this trip
2022-12-20	2022-12-23	3	IN-123456789	40.416775	-3.703790	38.736946	-9.142685	hermione@hogwarts.com	Important Info	Remove this trip
2022-12-27	2022-12-31	4	IN-123456789	38.736946	-9.142685	41.902782	12.496366	hermione@hogwarts.com	Important Info	Remove this trip

Create trip

Takeoff	Arrival	Insurance	From Latitude	From Longitude	To Latitude	To Longitude	Skipper Email
dd/mm/yyyy	dd/mm/yyyy						

Create Trip

All available locations

Latitude	Longitude	Name	Country Name
-37.813628	144.963058	Melbourne	Australia
-31.950527	115.860458	Perth	Australia
-33.868820	151.209290	Sydney	Australia
-15.794228	-47.882166	Brasilia	Brazil
-22.906847	-43.172896	Rio de Janeiro	Brazil
-23.550651	-46.633382	São Paulo	Brazil
31.224361	121.469170	Shanghai	China
39.133331	117.183334	Tianjin	China
51.507351	-0.127758	London	England
53.480759	-2.242631	Manchester	England

Figure 5: "Trips" Web Page

The instructions to create this web page are in the `trips.cgi` file. The first table shows the information about the reservation that was clicked on. Below, there is a list of all trips related to the clicked reservation and respective information. Note that, for the provided example, there are three registered trips: the sailor whose email is `john@email.com` is the skipper of one trip and the sailor with email `hermione@hogwarts.com` skips two trips. This means that they cannot be deauthorised and, because of that, the respective "No" buttons are disabled, as can be seen in Figure 4.

It is possible to create a new trip related to this specific reservation by filling the attributes under "Create Trip". Note that the attributes "From Latitude", "From Longitude", "To Latitude", "To Longitude" and "Skipper Email" are populated through dropdowns in order to avoid inserting values that do not exist in the database. Regarding the "Skipper Email" dropdown, maybe it would be expected to find the emails of all authorised sailors for this specific reservation, since the skipper must be an authorised sailor. However, in order to be a skipper, it is mandatory that the sailor has a sailing certificate for the class of the reservation boat. Thus, the sailors displayed in the dropdown "Skipper Email" are the ones obtained by intersecting the authorised sailors with the ones that have a sailing certificate for the boat class. In the presented example, the authorised sailors' emails are `columbus@email.com`, `hermione@hogwarts.com`, `jacksparrow@email.com`, `john@email.com`, `popeye@email.com` and `ron@hogwarts.com`. Consulting the `populate.sql` file, one can see that the sailors who have a certificate to sail boats of "Class 2" (the boat class of this reservation) are `john@email.com`, `hermione@email.com` and `bob@email.com`. Then, only the emails `john@email.com` and `hermione@email.com` will be displayed in the dropdown. If the user goes to the

"Authorisations" web page and authorises the sailor with email `bob@email.com` for this reservation, then it will be added to the respective"Skipper Email" dropdown, since he has a sailing certificate for this class. All the others, even if becoming authorised, will not be displayed in that dropdown.

In order to delete a trip from a reservation, the user has to click on the button "Remove this trip" associated to the trip to be deleted.

A list of the available locations is provided at the bottom of the web page in order to ease the access of the latitude and longitude values needed to create a new trip.

2.5 Consult the information about each trip

If the user clicks on the button "Important Info", in Figure 5, corresponding to a certain trip in the table that lists all trips for a particular reservation, a new web page appears, as shown in Figure 6. The file `view.cgi` contains the code to create it. This web page contains the data stored in the view `trip_info` created in the file `view.sql`.

Figure 6: Web page that displays the most important information about a trip

2.6 Queries output

By clicking on the button "Queries Output" in the main page, in Figure 1, the user is redirected to a web page that displays the output of the requested queries in the project assignment, including the data analytics ones. The output of each query is only displayed when the user clicks on the respective statement. The code associated to this web page is in the file `queries.cgi`

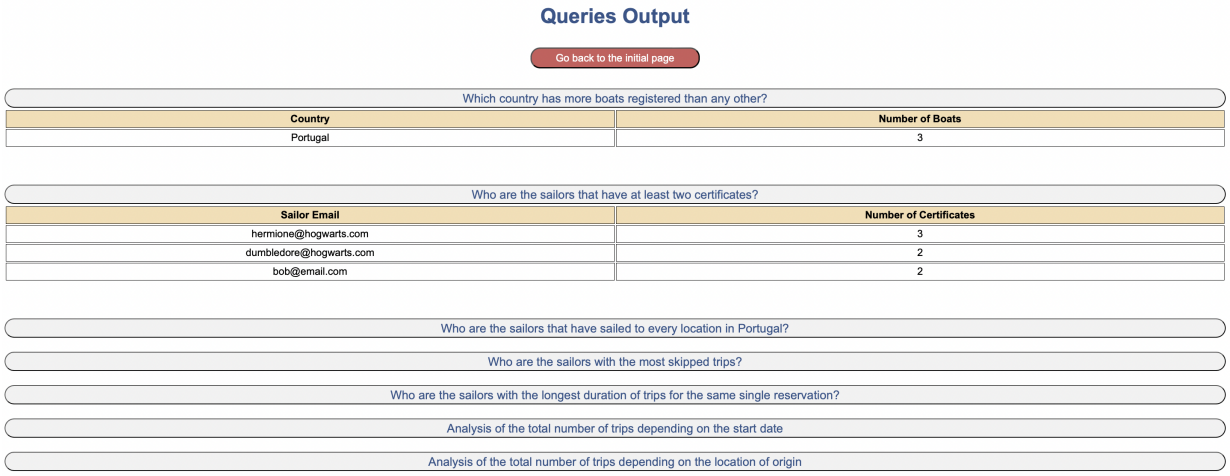


Figure 7: "Queries Output" Web Page

3 Indexes

3.1 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>;
```

In order to improve the scanning of boat table, we start by noticing that this query involves an equality condition (`boat_class = <some class>`) and a range-check condition (`year >= <some year>`). Hash indexes are unusable for range-queries and B+Tree indexes are efficient both with point and range queries. Therefore, the best index type for this case is a B+Tree.

Since two conditions are involved in this query, which means, two attributes need to be tested at the same time, `year` and `boat_class`, we would create a B+Tree composite index on attributes `<boat_class, year>` of the table boat. This index would optimize the scanning of records by enabling to locate records that match the `WHERE` condition faster and without having to search everywhere, since index entries are ordered by the search key value. The order of the attributes in the index was purposely chosen due to the fact that the condition `boat_class = <some class>` is more selective than the range-check condition `year >= <some year>`, reducing significantly the number of entries to be scanned. After having found the `<some class>` records to which the `boat_class` belongs (equality condition on the first attribute of the search key), finding records that also satisfy the range-check condition on the second attribute of the search key is fast, since index entries are ordered and, therefore, it only needs to find the first record with `year >= <some year>` and, then, all the others are instantly found.

The SQL index creating instruction would be the following:

```
CREATE INDEX boat_class_year_idx ON boat(boat_class,year);
```

3.2 Count the number of trips of boats by country:

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

In order to improve the scanning of the trip table, we intend to optimize the grouping by and `COUNT(*)` operations. Therefore, we create a B+Tree index on the attribute `boat_country` of the trip table. In a B+Tree index, the index entries are ordered by the search key value and, therefore, this index will be ordered by `boat_country`. This will allow the query to be executed faster since the records that belong to each sub-group of `boat_country`, to which the aggregating function `COUNT(*)` is applied, are already identified by the index and there is no need to search the whole table in order to

group the records into the sub-groups of `boat_country`. Therefore, besides allowing to more quickly locate the records of the trip table that match the `boat_country` values specified in the `GROUP BY` clause, it also allows to count the number of rows that match each `boat_country` value faster.

The SQL index creating instruction would be the following:

```
CREATE INDEX boat_country_idx ON trip(boat_country);
```