Instituto Superior Técnico

Information Systems and Databases

2022/2023 - First Semester

Database Project, Part 3

Group 6

Shift: SIBD-2PB03

Lab Professor: Francisco Regateiro

Authors and Contribution

89924 Madalena Fernandes 33.3% (27h)
92676 Diogo Delgado 33.3% (27h)
92707 Mariana Lima 33.3% (27h)

Faculty

Pedro Carreira
Francisco Regateiro
Pedro Dias

Lisbon, 3rd January 2023

Content

1 Introduction

In an ever increasingly digital world, it is necessary to process and work on large amounts of data. Information systems and databases come as a solution to the ever increasing problem of big data. Such systems are highly optimised, enabling the efficient use of large and complex databases.

This project was created in the scope of the Information Systems and Databases course. It had the main objective of designing an information system to support a Boating Management System which was developed using a database and a prototype of a web application. An important goal of this project was also to create several SQL queries to analyse the database. The concept of indexes was also covered in order to improve querying times.

2 Application Development

The web application consists of several `.cgi` files linked together in order for it to be well connected. In order to access the web app, use the link `https://web2.tecnico.ulisboa.pt/ist192707/initialpage.cgi`.

The web application consists of an `'Home Page'` from where every other can be accessed. This home page, named `initialpage.cgi`, is directly linked, through buttons, to the following: `sailor.cgi`, `reservation.cgi`, `authorisation.cgi` and `trips.cgi`. Each one of these has a `'Home Page'` button at the top that enables the return to the `'Home Page'`. Figure 1 shows the flowchart that illustrates the organisation of the web application.

The `sailor.cgi` page shows the list of all sailors and the lists of all junior and senior sailors. Next to each sailor there is a `'Remove Sailor'` button that triggers the `removesailor.cgi` file to be executed. This file opens a web page indicating if the sailor was successfully removed and a `'Go Back'` button which triggers the return to the `sailor.cgi` file. Below the junior/senior sailors tables, there are `'Create Junior/Senior Sailor'` buttons. Both these buttons trigger the execution of the `createsailor.cgi`, where the web page presents text input boxes for the user to fill with the data of the new sailor to be created. The `'Cancel'` button present in this page forces the return to the previous web page (`sailor.cgi` file) without any addition to the database. However, the `'Create'` button links to the `updatesailor.cgi` file which attempts to insert into the database the data the user gave. If successful, the page indicates so but if unsuccessful it shows that an error occurred and what failed in the SQL query. The web page always creates a `'Go Back'` button that triggers the `sailor.cgi` file to be executed.

The `reservation.cgi` page presents the list of all reservations. Next to each one there is a `'Remove Reservation'` button that triggers the execution of the `removereservation.cgi` file. This file opens a web page indicating if the reservation was successfully removed and a `'Go Back'` button which forces the return to the `reservation.cgi` file. Below the table, there is a `'Create Reservation'` button. This button triggers the execution of the `createreservation.cgi`, where the web page presents text input boxes for the user to fill with the data of the new reservation to be created. The `'Cancel'` button present in this page forces the return to the previous web page (`reservation.cgi` file) without any addition to the database. However, the `'Create'` button links to the `updatereservation.cgi` file which attempts to insert into the database the data the user gave. If successful the page indicates so but if unsuccessful it shows that an error occurred and what failed in the SQL query. The web page always creates a `'Go Back'` button that triggers the `reservation.cgi` file to be executed.

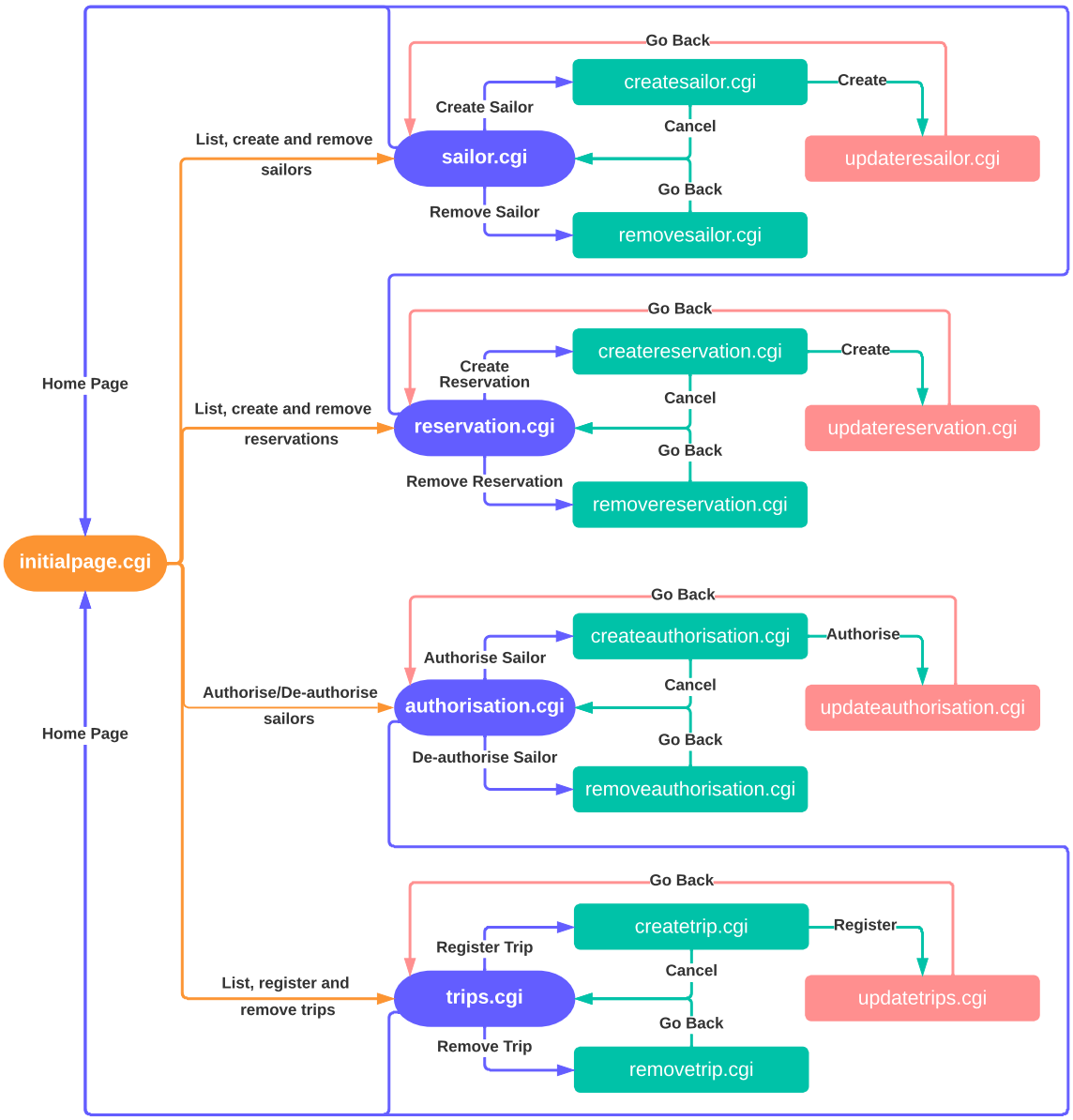

Figure 1: Flowchart of the web application organisation.

The `authorisation.cgi` page shows the list of all authorisations. Next to each one there is a `'de-authorise Sailor'` button that triggers the `removeauthorisation.cgi` file to be executed. This file opens a web page indicating if the sailor was successfully de-authorised and a `'Go Back'` button which forces the return to the `authorisation.cgi` file. Below the table, there is a `'Authorise sailor for reservation'` button. This button triggers the execution of the `createauthorisation.cgi`, where the web page presents text input boxes for the user to fill with the necessary data of the new authorisation to be created. The `'Cancel'` button present in this page forces the return to the previous web page (`authorisation.cgi` file) without any changes to the database. However, the `'Authorise'` button links to the `updateauthorisation.cgi` file which attempts to insert into the database the data the user gave. If successful the page indicates so but if unsuccessful it shows that an error occurred and what failed in the SQL query. The web page always creates a `'Go Back'` button that triggers the `authorisation.cgi` file to be executed.

The `trips.cgi` page shows the list of all trips. Next to each one there is a `'Remove Trip'` button that triggers the `removetrip.cgi` file to be executed. This file opens a web page indicating if the trip was successfully removed and a `'Go Back'` button which triggers the return to the `trips.cgi` file. Below the table, there is a `'Register New Trip'` button. This button triggers the execution of the `createtrip.cgi`, where the web page presents text input boxes for the user to fill with the data of the new trip to be registered. The `'Cancel'` button present in this page forces the return to the previous web page (`trips.cgi` file) without any addition to the database. However, the `'Register'` button links to the `updatetrips.cgi` file which attempts to insert into the database the data the user gave. If successful the page indicates so but if unsuccessful it shows that an error occurred and what failed in the SQL query. The web page always creates a `'Go Back'` button that triggers the `trips.cgi` file to be executed.

2.1 File sailor.cgi

This file is responsible for showing the tables listing the sailors. It sends to the `removesailor.cgi` file the data necessary for the removal of the sailor, *i.e.*, the email (Primary Key). To the `createsailor.cgi` file it sends an integer that works as an indicator for the 'Sailor type', *i.e.* Senior (0) or Junior (1).

2.1.1 File removesailor.cgi

This file deletes a sailor from the database. It receives, from the `sailor.cgi` file, the data necessary for the removal of the sailor, *i.e.*, the email (Primary Key).

2.1.2 File createsailor.cgi

In this file, the data for a new sailor is collected. It receives, from the `sailor.cgi` file, an integer that works as an indicator for the 'Sailor type', *i.e.* Senior (0) or Junior (1) and sends it to the `updatesailor.cgi` file along with the information the user inserted in the text boxes (first name, surname and email).

2.1.3 File updatesailor.cgi

This file inserts a new sailor in the database and receives from the `createsailor.cgi` file the 'Sailor type' along with the information the user inserted in the text boxes (first name, surname and email).

2.2 File reservation.cgi

This file is responsible for displaying the table listing the reservations present in the database. It sends to the `removereservation.cgi` file the data necessary for the removal of the reservation, *i.e.*, start date, end date, country and CNI.

2.2.1 File removereservation.cgi

This file deletes a reservation from the database. It receives, from the `reservation.cgi` file, the data necessary for the removal of the reservation, *i.e.*, start date, end date, country and CNI.

2.2.2 File createreservation.cgi

This file collects data for a new reservation. The information the user inserted in the text boxes (start date, end date, country, CNI and the email of the responsible) is then sent to the `updatereservation.cgi` file.

2.2.3 File updatereservation.cgi

In this file, a new reservation is added to the database. This file receives from the `createreservation.cgi` file the information the user inserted in the text boxes (start date, end date, country, CNI and the email of the responsible).

2.3 File authorisation.cgi

This file displays the table listing the authorisations. It sends to the `removeauthorisation.cgi` file the data necessary for the removal of the authorisation (start and end date of the reservation, country, CNI and sailor email).

2.3.1 File removeauthorisation.cgi

This file is responsible for de-authorising a sailor from a reservation. It receives, from the `authorisation.cgi` file, the data necessary for the removal of an authorisation, *i.e.*, start and end date of the reservation, country, CNI and sailor email.

2.3.2 File createauthorisation.cgi

This file collects the data for a new authorisation. It sends to the `updateauthorisation.cgi` file the information the user inserted in the text boxes (start and end date of the reservation, country, CNI and the email of the sailor).

2.3.3 File updateauthorisation.cgi

This file inserts a new authorisation in the database. It receives from the `createauthorisation.cgi` file the information the user inserted in the text boxes (start and end date of the reservation, country, CNI and the Email of the sailor).

2.4 File trips.cgi

This file presents the table listing the trips present in the database. It sends to the `removetrip.cgi` file the data necessary for the removal of a trip, *i.e.*, takeoff date, arrival date, insurance, latitude (origin), longitude (origin), latitude (destination), longitude (destination), skipper email, reservation start date, reservation end date, boat country and CNI.

2.4.1 File removetrip.cgi

This file removes a trip from the database, receiving from the `trips.cgi` file the data of the corresponding trip to be removed, *i.e.*, takeoff date, arrival date, insurance, latitude (origin), longitude (origin), latitude (destination), longitude (destination), skipper email, reservation start date, reservation end date, boat country and CNI.

2.4.2 File createtrip.cgi

This file collects the necessary data to add a new trip and sends to the `updatetrips.cgi` file the information the user inserted in the text boxes (takeoff date, arrival date, insurance, latitude (origin), longitude (origin), latitude (destination), longitude (destination), skipper email, reservation start date, reservation end date, boat country and CNI).

2.4.3 File updatetrips.cgi

This file is responsible for the insertion of a new trip in the database, receiving from the `createtrip.cgi` file the information the user inserted in the text boxes (takeoff date, arrival date, insurance, latitude (origin), longitude (origin), latitude (destination), longitude (destination), skipper email, reservation start date, reservation end date, boat country and CNI).

3 Indexes

The creation of indexes allows for a more efficient processing of queries, in a certain database, if it's applied properly. For this project, it was asked to create SQL index(es) that would speed up the execution of the following queries:

1. List the names of all boats of a given class and registered after a given year:

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

   This query has an equality test and a range-check condition, which means that a `Hash` or `B+ Tree` index should be used.

In this case, it was chosen to use the `B+ Tree` index, with the column 'year' as the key field, as opposed to choosing a `Hash` function, as it would not be very useful since neither the 'year' or 'boat class' would have a lot of rare data.

As mentioned above, the 'year' would be the best key field, as it filters the most results and we can order the data by ascending values, which increases the efficiency of the query.

2. Count the number of trips of boats by country:

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

In the second proposed query, an output of the usage of a count function was presented, which provokes a sorted aggregation, scanning a table in sequence of the attributes and breaking every time a new value is found.

This subsequently means that an index sorting the table is necessary, being chosen to use a `B+ Tree` with the key value being 'boat_country ', and it being order ascending, as a way to furthermore improve the query's efficiency.

The discussion above can be implemented on SQL language, creating two indexes to answer the questions presented, as presented below.

```
-- 7.1) List the names of all boats of a given class and registered after given
    year
DROP INDEX IF EXISTS boat_idx1;
CREATE INDEX boat_idx1 ON boat (year ASC);

EXPLAIN ANALYZE SELECT boat.name
FROM boat
WHERE year >= '2000' AND boat_class = '3';


-- 7.2) Count the number of trips of boats by country
DROP INDEX IF EXISTS boat_idx2;
CREATE INDEX boat_idx2 ON trip(boat_country ASC);

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

By using the function `EXPLAIN ANALYZE` to the queries, the system executes their statement, and instead of returning the data, provides a query plan detailing what approach the planner took to executing the statement provided. This would help decide the correct index to use, but the values returned are not precise enough, as the number of data used was too small.

It can be concluded that depending on the content of the data included in the tables, the use of indexes can vary and also their respective types.

4 Conclusion

Throughout the completion of this project, it was possible to gain a deeper understanding of the theoretical concepts gained in class, ranging from the conceptualisation of a database and its implementation to the development of a web application.

In this report, there is a detailed explanation of the web application's structure and our reasoning behind the indexes we chose to improve querying times of the queries presented.

Finally, it's possible to say that this project was a resounding success and gave the students great pleasure in learning more about Information Systems and Databases.