Projeto Laboratórios de Informática III
Grupo 19 - Fase 2

Projeto desenvolvido por
Alex Sá (A104257), Paulo Ferreira (A96268), e Rafael Fernandes (A104271)

Licenciatura em Engenharia Informática

Universidade do Minho
Escola de Engenharia

Departamento de Informática
Universidade do Minho

Contents

1 Abstract

This report chronicles the advancements and strategic decisions undertaken by Group 19 leading up to the delivery of the initial phase of the project conducted as part of the curriculum for *Laboratórios de Informática III* during the academic year 2023/2024.

The second phase of this project signifies a notable progression from its initial stage. This document meticulously outlines the pivotal developments, optimizations, and novel features implemented by Group 19 during the continuation of the project under the purview of *Laboratórios de Informática III* in the academic year 2023/2024.

2 Introduction

As the project progresses into its second phase, the team has dedicated its efforts to address three pivotal tasks: refining the statistics module, implementing the remaining queries (2, 6, 8, 10), and introducing an interactive mode. Beyond these key objectives, the team has been actively engaged in substantial enhancements, focusing on optimizing memory usage and execution time. This commitment ensures that the project aligns with academic standards and adheres to best practices in the field.

Throughout the evolution of the project, decisions made since the initial phase, both those implemented and those discarded, have been pivotal. This report elucidates the reasoning behind our choices, examining the challenges encountered and detailing improvements that could be applied. The ensuing pages provide a comprehensive exploration of our decision-making process, shedding light on the complexities faced by Group 19 in shaping the project's trajectory.

3 Application Details

3.1 Overview

Offering a comprehensive snapshot of the project's status as of January 22, 2024, it is with satisfaction that we report the successful attainment of the project's intended functionality. The application, at its current stage, stands as a testament to the diligent efforts and strategic decisions made by Group 19, culminating in a robust and functional system.

Commencing with the parsing phase, it maintains a substantial degree of consistency with the approach employed in the initial project phase, albeit with noteworthy adjustments in the quantity of arguments passed for each struct. Notably, calculations such as determining the number of passengers per flight or the count of reservations per user are now computed during parsing.

Additionally, it is pertinent to mention the introduction of two new structs during this phase, akin to small catalogs—namely, the 'genericCatalog' and the 'statsCatalog'. A more in-depth examination of this aspect is provided in Section 4 of this report, shedding light on the significance and role of these new structures within the project's evolving architecture.

Subsequent to the dataset parsing, a dedicated function is invoked to process information sourced from "input.txt," encompassing queries and their corresponding inputs. This phase serves as a testament to the effectiveness of our unchanged data structures, excluding the Passagens Catalog, as detailed in Section 3.3. Furthermore, we delve into the chosen algorithm for each query, providing succinct explanations. It's noteworthy that modifications have been made to the approach proposed in phase one for several queries, and these alterations are elucidated in the subsequent sections.

3.2 Query Solutions

The following section outlines the strategies employed to solve specific queries:

1. **First Query:**
   - **Algorithm:** Hashtable-based approach.
   - **Additional Module:** Statistics for value calculations (e.g., total spent by a user, total price for a reservation).
2. **Second Query:**
   - **Algorithm:** GArray approach, accessing data stored in the usersCatalog.
3. **Third Query:**
   - **Algorithm:** Binary search with range determination (made possible by previous sorting).
   - **Catalog Sorting:** By hotel ID, start date, and reservation ID.
4. **Fourth Query:**
   - **Algorithm:** Similar to the third query.
   - **Catalog Sorting:** By hotel ID, start date, and reservation ID.
5. **Fifth Query:**
   - **Algorithm:** Similar to the third and fourth queries.
   - **Catalog Sorting:** By origin, scheduled departure date, scheduled arrival date, and flight ID.
6. **Sixth Query:**
   - **Algorithm:** Hashtable + GArray approach.
   - **Additional Module:** Statistic module to calculate passengers by year and origin, adding that value to a hashtable in the statsCatalog.
7. **Seventh Query:**
   - **Algorithm:** Utilization of the statistics module to access the pre-calculated median by origin.
   - **Additional Module:** Statistic module for median calculation.
8. **Eighth Query:**
   - **Algorithm:** Binary search with range determination (made possible by previous sorting).
9. **Ninth Query:**
   - **Algorithm:** Similar to the third query.
   - **Catalog Sorting:** By name and user ID.
10. **Tenth Query:**
    - **Algorithm:** Previous calculation of values during parsing and iterating over the usersCatalog, accessing all flights each user is in.
    - **Additional Module:** Statistic module for accessing the catalog with general information.

As of now, this encapsulates a comprehensive summary of how the application works. The subsequent section will delve into a more detailed examination of the catalogs and data management, offering a deeper insight into the intricacies of our project.

3.3 Managing Data

The data management approach in the second phase closely aligns with the principles established in the initial phase, with some notable refinements introduced.

In this iteration, the Passengers Catalog departs from its sorted structure. Instead, all pertinent information is gathered during parsing. Functioning primarily as a lookup mechanism, it employs a UserId to efficiently retrieve the corresponding FlightID or vice-versa.

A significant enhancement is observed in the User structure, now equipped with two arrays. These arrays, populated during the parsing of reservations and passengers, contain pointers to reservations and flights associated with each user. This augmentation streamlines the calculation of certain values in the context of various queries, improving overall query performance.

The decision to retain consistency with the first phase's approach while introducing targeted modifications underscores the effectiveness of our initial design. For a detailed understanding of the data management strategies adopted in the initial phase, refer to Section regarding "Managing Data" of the first-phase report.

As our project evolves, the judicious combination of sorted and unsorted structures continues to address the dynamic demands of our datasets, striking an equilibrium that optimizes both data access and retrieval efficiency.

4 Key Developments in Phase Two

4.1 Implementation of All Queries

All queries have been meticulously implemented in this phase, ensuring the project boasts a comprehensive set of functionalities for robust data retrieval and processing.

4.2 Completion of Query 6

In this phase, the algorithm for Query 6 was successfully implemented and seamlessly integrated into the project. The team's dedication resulted in the full realization of Query 6, enhancing the versatility of the application.

4.3 Total Implementation of Query 8

Query 8 has been fully implemented, contributing to the expansion of the application's query capabilities. The successful execution of this query enhances the overall functionality of our project.

4.4 Implementation of Query 10

A notable achievement in the query field was the successful implementation of Query 10, recognized as one of the most challenging queries. The team's diligent efforts in formulating an efficient algorithm and integration have resulted in Query 10 becoming a fully functional component of the application.

4.5 Optimizations for Existing Queries

Building upon the optimization efforts from phase one, further enhancements were made to improve the efficiency of existing queries. These optimizations ensure the streamlined performance of the application, especially in scenarios with large datasets.

4.6 Adjustment of Priority Validations

Acknowledging the time-intensive nature of hashtable lookups, the group adjusted priority validations. These lookups are now strategically placed in the final step of the validation process, resulting in a notable performance improvement.

4.7 Modification of Date Offset for Extended Date Support

An adjustment to the date offset was implemented to support a broader range of dates. This modification, driven by the need for more bits to store dates, involved changing the offset calculation method. The new approach, involving summation rather than subtraction, addresses the issue without increasing memory usage.

4.8 Enhancements to the Date Module

The Date module underwent a comprehensive refactor in phase two, focusing on improved code readability. New getter and comparison functions were introduced, and the nomenclature and declaration of functions were completely revamped. This overhaul was complemented by thorough documentation to enhance the module's accessibility and understanding.

4.9 Improved Includes and Documentation Readability

Phase two witnessed changes aimed at enhancing the overall readability of includes and documentation. These improvements contribute to a more user-friendly and comprehensible codebase, facilitating collaboration and maintenance.

4.10 Statistics Calculation during Parsing Time

To streamline program efficiency, certain statistics are now calculated during parsing time. This eliminates the need for separate iterations over the array, contributing to overall performance optimization.

4.11 Statistics Catalog/Generic Catalog

The introduction of a Statistics Catalog marks a significant development in the second phase. This catalog is instantiated using a generic struct that encapsulates essential information about the application, incorporating values calculated during the parsing process.

Following parsing, a pointer to the Statistics Catalog is provided to the queries module. This specialized struct comprises small hashtables, arrays, and values, strategically designed to expedite query execution. The inclusion of precomputed data enhances the efficiency of queries, contributing to the overall optimization of the application.

4.12 Output Module Implementation

A dedicated Output module was implemented to manage and present output information generated by the program. This addition enhances the user experience by providing a clear and organized presentation of results.

4.13 Test Executable and Interactive Mode

In the pursuit of a robust testing framework, a test executable was added to the project. This executable prints time and memory usage information about the program, contributing to ongoing efforts to ensure program reliability. Additionally, an interactive mode was implemented, offering a dynamic and user-friendly interface, further enhancing user interaction with the application.

4.13.1 Modular Code Organization: Phase Two Updates

In this phase, the modularity of our project underwent targeted adjustments to accommodate new functionalities and improve existing features. The key modifications include:

- **Interactive Module:** A new module, `interactive`, has been introduced to manage the interactive mode functionality. This module, represented by `interactive.h` and `interactive.c`, encapsulates operations related to user interaction in the interactive mode. The User Interface (UI) is powered by a custom graphics library called Grimoire, further described on it's own chapter. Using the Grimoire as a framework for the creation of the UI, a modular plug-and-play approach was attempted, borrowing some concepts from the Object-Oriented Programming paradigm, to various degrees of success. Due to some constraints and shortcomings, the implementation of the interface did not sit on top of careful planning, and instead was built and altered as need manifested.
- **Output Module:** The introduction of an `output` module enhances the project's capability to manage and present output information. The `output.h` file defines the interface, while `output.c` implements logic for managing and presenting output information generated by the program.
- **Generic Catalog:** A significant addition to our modular structure is the implementation of a `genericCatalog` within the `catalog` directory. The `genericCatalog.h` and `genericCatalog.c` files define and implement operations related to the new generic catalog, serving as a valuable component in optimizing query execution.

These adjustments not only showcase the adaptability of our modular code organization but also emphasize our commitment to refining and expanding the project's functionality while maintaining a clear separation of concerns.

5 Grimoire

The Grimoire is a terminal graphics library created from the ground up for this project. It allows for high-level design and implementation of Graphic Interfaces without requiring the user to manage the underlying commands and memory associated with it.

5.1 Motivation

The motivation to write this library was the fact that the popular and objectively more solid library NCurses contains permanent memory leaks, which would negatively impact our grade. Initially planned to be a drop-in replacement for NCurses, during the early stages of development, a different API was envisioned, moving away from NCurses' API, while retaining a similar syntax.

5.2 Features

The library allows for declaration of colors within the full RGB spectrum, printing ASCII and UTF8 characters, with or without special effects applied and input fetching, including basic UTF8 multibyte support. UTF8 support proved to be challenge to implement, due to it's multibyte structure not allowing for a conventional 2D matrix buffer for the underlying drawing module. To circumvent this limitation, we took advantage of the UNIX terminals ability to automatically convert UTF8 sequences into it's corresponding character, accessible using the printf function. Using a GQueue to store every draw call in sequence, the refresh call passes every draw call to the terminal to be processed, lifting some workload off the program. Another challenging feature was the ability to read user input. In order to be able to read special keys such as arrow keys, function keys, control sequences and others, the terminal is required to be set to raw mode and non-blocking mode. Furthermore, reading multibyte sequences such as escape sequences and UTF8 characters when typed in rapid succession caused the STDIN buffer to contain multiple sequences back-to-back, which broke the initial implementation. In order to partly circumvent this limitation, the reader ceases to read new bytes from the buffer if a new sequence is started, and simply ignores all subsequent bytes present in the buffer.

6 System Architecture and Code Organization

The system architecture and code organization have been continually refined in phase two to accommodate the new features and optimizations. The modular structure, exemplified in the earlier phase, remains a cornerstone for code clarity and maintainability.

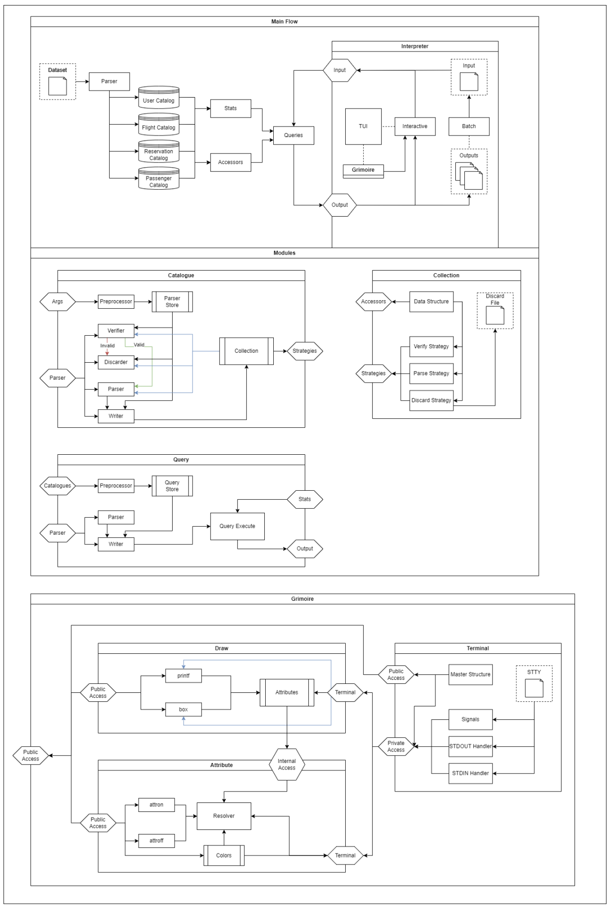

Figure 1: Architecture schematic for the project, as it stands in this phase

7 Performance Evaluation

Phase two's optimizations and enhancements have been accompanied by a comprehensive evaluation of the program's performance. A series of tests were conducted to gauge the impact of the implemented changes on both time and memory usage. Detailed performance metrics are presented in subsequent sections.

	Machine 1	**Machine 2**	**Machine 3**
CPU	Intel Core i7-1165G7	Intel Core i7-1065G7	Ryzen 7 5800H
Cores/Threads	4/8	4/8	8/16
RAM	16GB DDR4 3200	12GB DDR4 2666	16GB DDR4 3200
Disk	1 TB SSD M.2	512 GB SSD M.2	1 TB SSD M.2
OS	Pop!_OS Linux	WSL Ubuntu	Debian 12

Table 1: Hardware of the Machines Used

Metric (s)	**Machine 1**	**Machine 2**	**Machine 3**
Parsing Users	1.4858	1.4990	2.6910
Sorting Users	1.3260	1.4397	2.0808
Parsing Flights	0.2843	0.3142	0.6070
Sorting Flights	0.1250	0.1786	0.2672
Parsing Passengers	16.6096	15.3187	26.3466
Sorting Passengers	0.0000	0.0000	0.0000
Parsing Reservations	11.1418	15.3187	18.3288
Sorting Reservations	1.5466	1.6906	3.2079
Initializing Statistics	6.9727	6.3868	11.5994
Query 1	0.0021	0.0014	0.0031
Query 2	0.0013	0.0007	0.0017
Query 3	0.1144	0.1030	0.2981
Query 4	0.5937	0.3698	1.0153
Query 5	0.2113	0.1857	0.3844
Query 6	2.1124	1.9511	2.7237
Query 7	0.0002	0.0001	0.0002
Query 8	0.1805	0.1791	0.3974
Query 9	0.1936	0.1795	0.1759
Query 10	0.0002	0.0001	0.0002
Solving all queries	3.4285	2.9703	5.0001
Parsing all collections datasets	30.0091	27.6476	47.9733
Sorting all collections datasets	10.0256	9.6159	17.1553
Total program execution time	45.7858	42.5040	73.9019
Memory usage (MB)	2216	2216	2217

Table 2: Median execution times for various metrics on three machines.

8 Experienced Difficulties

While phase two was marked by substantial progress, the team encountered a set of challenges. These challenges, ranging from debugging complexities to algorithmic intricacies, are detailed below.

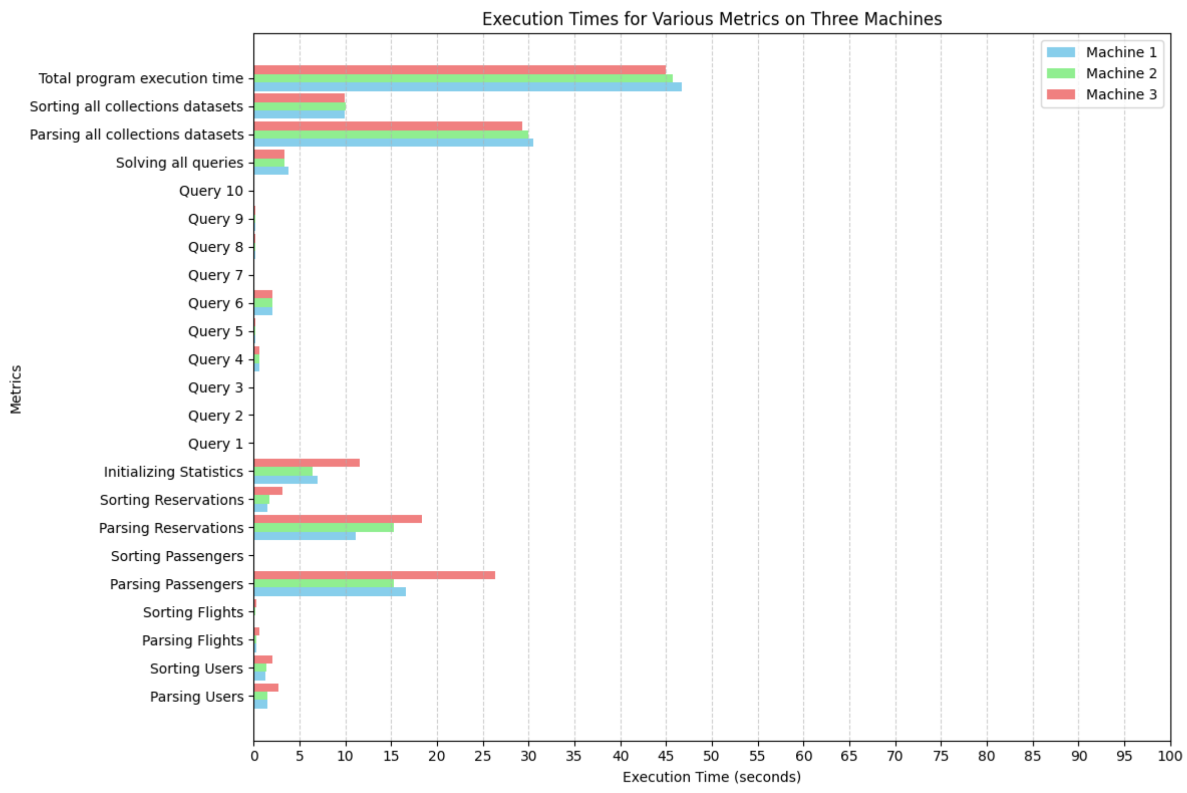

Figure 2: Bar chart showing median execution times for various metrics on three machines.

8.1 Optimization Challenges

The pursuit of optimization, particularly in handling large datasets and refining query execution times, presented intricate challenges. Balancing performance enhancements without compromising code readability and maintainability required meticulous analysis and iterative refinement.

8.2 Algorithmic Complexity in Query 10

The implementation of Query 10 introduced algorithmic challenges, necessitating a thorough examination of computational efficiency. The team engaged in collaborative problem-solving sessions to devise an optimal solution, addressing complexities while adhering to the project's coding standards.

9 Future Considerations

As we finalize the project, certain considerations for potential improvements have been identified. While these areas were acknowledged, it's important to note that, in alignment with the standards of the discipline, they were not pursued in the final phase.

9.1 Threading

The potential introduction of threading to enhance concurrent execution and resource utilization was considered. However, given the discipline standards and the project's current state, this enhancement was deemed unnecessary. Thorough testing and integration would have been required to ensure thread safety and avoid potential race conditions.

9.2 Pagination

Considerations regarding pagination for future development were contemplated. The implementation of an efficient pagination mechanism could enhance user experience, especially when dealing with extensive datasets.

10 Conclusion

The conclusion of phase two signifies a pivotal moment in the journey of our project, undertaken within the framework of *Laboratórios de Informática III* during the academic year 2023/2024. Group 19's steadfast commitment to excellence, meticulous planning, and diligent execution have propelled the project into a mature state, characterized by significant developments, optimizations, and seamless integration of new features.

Our dedication to a modular code organization has not only facilitated code maintenance and navigation but also fostered a clear separation of concerns. The project directory structure stands as a testament to our commitment, with distinct modules encapsulating specific functionalities, enhancing clarity, and reducing complexity.

Challenges encountered, be they optimization intricacies or algorithmic complexities, served as opportunities for collaborative problem-solving. The team's adept navigation through these challenges, involving engaging problem-solving sessions and upholding code quality, highlights our resilience and proficiency.

Looking to the future, identified considerations for improvements, including threading, pagination, and addressing algorithmic complexities in specific queries, remain noted. It's noteworthy that these considerations, while acknowledged, were intentionally not pursued in the final phase. This decision aligns with both the standards of the discipline and the current state of the project.

As the project concludes its current phase, Group 19 reflects with pride on the achieved objectives and the feature-rich nature of the application. The collaborative spirit that has guided us thus far remains a cornerstone as we bring this project to a close. We have not only delivered a robust and efficient solution that meets academic expectations but have also gained valuable insights and skills that will undoubtedly shape our future endeavors.