University of Toronto
Department of Computer Science
CSC 228 - Summer 1994

Term Project: Airport '94

Project Plan Due: in tutorial, July 21
Project Report Due: in tutorial, August 11

This project will involve the use of hashing functions, inverted files, and some simple query processing. You will produce your own test data, and evaluate the performance of your system. Finally, your report will describe your implementation and test procedures.

The problem

Canadian Airlines recently won a court battle to leave the Gemini reservation system which they had shared for many years with Air Canada. This move allowed Canadian to join American Airlines on the Saber reservation system, effectively merging several operations with the larger US-based carrier. Without the partnership of their competitor, Air Canada can no longer afford to maintain Gemini, and have therefore hired you to design and implement a new reservation system.

Your task is to write a system which allows reservation agents to book passengers on flights operated by Air Canada or any of its partners. The program must keep track of seat availability, meal selection, and of course, the passenger roster. When making a reservation, passengers may specify flights either by

- date, origin city, and destination city, or
- date and flight number

Given an origin and destination city for a particular date, your program should be able to provide a reservation agent with a list of flights with available seating.

Once a booking is made in a passenger's name, a unique reservation number is assigned. Passengers may then make changes to their booking by specifying either the reservation number, their full name, or either of the previous two items.

Just before flight time, a passenger list must be printed, including business class and economy seat allocation, meal selection, and any other information you consider relevant. You should provide a special command to generate this list.

In order to provide quick access to information for most requests, you should use a direct file for storing passenger reservation records, based on a hashing function of the reservation number. When provided with this number, your system should be able to locate a reservation in the data files (if it exists) with a small number of disk accesses. Unfortunately, many passengers misplace or forget their reservation numbers so your system must also be able to retrieve reservations based on secondary attributes, including passenger name, or date and flight number. In the event that there is more than one match to such a search, your system should display them all.

Air Canada currently requires storage for approximately 10^5 (one hundred thousand) active reservations. However, for the prototype system you will be developing, you may use a much smaller database, provided you can convince us that with greater file storage capability, your program is easily expandable. This means that you should never rely on main memory being sufficiently large to hold all of the data you need at once.

Although in actual use, your program would be run interactively, you should design the system to run in batch mode. All other design decisions, including (but not limited to) the data file format, the query structure, and the number of records per bucket are completely up to your group.

The Project Plan

The plan should be a formal description of approximately two pages. It should report the membership of your team, the tasks undertaken by each member, and your plans for what is to be accomplished in your program. Your plans should include explicit mention of how you are going to arrange your program code so as to accommodate changes to the capabilities of the final system, which you will likely decide are necessary as you work on the implementation.

The Group Presentation

During the tutorials of August 4 and August 11, each group must make a brief (five minute) presentation of their project, discussing their overall design strategy as well as special features of their system. In most cases, you should select one member of your group to do the actual speaking, although every member should be involved in planning.

The Project Report

This report will likely be longer than your reports for the first two assignments, although it should not exceed ten pages[1]. It will be similar in format, except that you must be careful to explain which members of your group did what tasks. It is possible that you have shared the work equally, but it is also acceptable for one team member to do most of the testing while another does most of the report writing. However, it is not acceptable for all the coding or testing or report writing to be done by one person. Each team member must have at least some involvement in every aspect of the project.

[1] Remember – a carefully written, concise report is of far greater value to the reader than verbose drivel.

Approaching the Project

There are several components to this project, each of which will require attention early in the design. You will have to select a hashing technique for the direct file, and a mechanism for resolving collisions. You will also need to make some decisions regarding the implementation of the inverted files, including the directory structure, and the format of queries and output. Furthermore, you may find it necessary to reduce your ambitions, and you may be told to include requirements which are not part of this handout. For both reasons, it is important to begin by isolating the essential operations of the system and designing the modular structure to accommodate later changes. This is not a difficult task, but you must remember not to rush straight into coding.

If, on the other hand, you find that you have quickly succeeded in building a basic working system, you may wish to add extra features to your program. For instance, you might want to allow a reservation agent to overbook a flight by entering a special password, or alternately, to place passengers on a waiting list for seats, hoping that someone else will cancel their reservation.

It will be natural as you plan your system to think about how to describe it in your report. The report should not be written at the end of the project. Rather, you should write an outline at the beginning, and continue to expand it as you proceed, until by the end of the coding, testing, and experimenting, the report is almost complete.

You should also begin work on your testing strategy very early in the project. Unlike previous assignments, you will need to test this system for both *correctness* (which you are used to) and *performance*. For correctness testing, you will need small, carefully chosen test files, while for performance testing, you will need large, random files, likely generated by a program. Plan this from the beginning. If you leave either type of testing to the last minute, your group will likely find itself in trouble.

Try to build up the system from a simple skeleton to a fully functional package in gradual stages, testing various components as you proceed. With the confidence of having something simple working soon, you can go on to grow a bigger project successfully.