University of Toronto
Department of Computer Science
CSC 228 - Summer 1993

Term Project: Where's my bike?

Project Plan Due: in tutorial, July 13
Project Report Due: in tutorial, August 3

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 using disk access as a measure. Finally, your report will describe your implementation and test procedures, and discuss the results obtained in the performance measurement experiments.

The problem

Bike theft has become rampant in Toronto, and despite a large number of successful recoveries, the police are unable to return many bikes to their rightful owners, due largely to inefficiencies in the police filing system.

Your task is to design and implement a system which allows the police to enter information about recovered bikes and then, based on descriptions supplied by the owners, match the bikes to their owners. The specific information you will store about each bike is up to your group, but some suggestions are registration number, make, model, number of speeds, and colour.

To complicate police efforts in tracing stolen bikes, some thieves alter or destroy the registration numbers engraved on the bottom of the frame. Therefore, in adding a recovered bike to the database, your system will have to make up and assign a registration number to any bikes in which the original number is unreadable.

In order to provide quick access to information for most queries, you should use a direct file for storing bike records, based on a hashing function of the registration number. By providing the system with a bike registration number, the police should be able to locate a bike in the data files (if it exists) with a small number of disk accesses. Unfortunately, many bike owners misplace or forget these numbers[1], so your system must also be able to retrieve records based on secondary attributes, that is, using inverted files. For example, a distraught bike owner who failed to register his bike before it was stolen might ask the police to look for a blue Norco 12-speed mountain bike, which disappeared around May 1993. There may be more than one match to such a search, so your system should display them all.

The police currently require storage for approximately 1000 bikes. You should design your program to handle the current requirements, but it should not be overly sensitive to these numbers. This means that your program should be easily expandable, and if expanded, its performance should not suffer noticeably. Furthermore, the main memory capacity is sufficient to hold no more than 100 bike records simultaneously, however you define them. You should also assume that any inverted files you use will be too large to fit in main memory. All other design decisions such as the data file format, the query structure, and the number of records per bucket are completely up to your group.

[1] Also, as described previously, creative thieves may alter or destroy the registration number, thus making the identification task non-trivial. However, we will assume that thieves do not bother painting bikes to change their colour.

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 Project Report

This report will naturally be longer than your reports for the first two assignments, although it should not exceed ten pages[2]. 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.

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.

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 begin thinking about what measurements you want to make. These may include average numbers of disk accesses for insertions, deletions, and retrievals. Since you cannot count true disk accesses, you should measure the number of "read", "write", and "seek" statement executions, keeping in mind that the quantity of data transferred will also affect performance. 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.

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