Web-based Smart Bus Schedules

Thomas Nyberg          Jonas Borgström
  c00tng@cs.umu.se       c00jbm@cs.umu.se

13th June 2004

Abstract

Travelling within a city using any form of public transportation will require the use of time tables. Unfortunately the use of these time tables may prove difficult or at least very time consuming, especially if the passengers have to switch buses to reach their destination. The system described here will address this and similar problems, in an attempt to create a solution that is both easy to use as well as powerful. Using simple tools and databases a web based system for querying time tables and helping in both choosing and locating source and destination is developed.

1 Introduction

Travelling through a city using some form of vehicles is a common task for most people. For those who use busses or trains, access to good timetables is essential. However, flipping through huge tables and searching for a suitable departure is a time consuming chore. For the unfortunate people needing to travel across several different routes (changing trains or busses) it can become a quite difficult problem.

This document describes one way to soften this problem for travellers, using a web based system to access and view the different time tables associated with using the local bus to travel within a city. The system provides two distinct methods to access the schedules and time tables, thus (hopefully) easing the burden on the passengers.

By providing the system with a starting point (i.e. the closest bus stop to the passengers current location) and a destination, suitable routes are calculated and displayed to the user. Sometimes travelling from one point to another requires the passenger to switch busses. If this is the case, the system clearly shows when and where the passenger should get off and which bus to resume travel with.

The other method to access the time tables provides the user with an “airport”-like view of the imminent departures and their respective destinations from any given bus stop. This can be used at bus stops or other terminals to provide a quick overview for potential passengers, which busses are departing and when they are scheduled to leave.

\[^{1}\text{This is commonly referred to as the travelling salesman problem.}\]
2 Approach

The two major parts of this system is the web based front end and the database acting as a back end. These two parts was to some extent created separately, and this allowed for reasonable fast development. The limiting factor on the types of queries the system could answer, was mostly the data contained within the database. This is natural, since we can only provide the information we either have or are able to deduce based upon information we do have.

The data for the time tables were acquired directly from Ultra (see section 2.1), and this data contained enough information to answer queries such as “How can I travel from point A to point B?”.

Since the front end for this system is web based, the software used in the production was chosen to facilitate this. The database chosen was the (popular) PostgreSQL database. The complete set of software used in the project, is as follows:

- PostgreSQL 7.4
- Python 2.3
- ClearSilver 0.9.8

2.1 Acquiring Time Tables

To populate the database some form of raw data is required, preferably real data. To acquire this data, contact was made with Ultra by means of email. About two weeks later the kind people at Ultra responded with a zip-archive containing a dump of their database. Unfortunately the dump consisted of 10 MB worth of csv formatted data that did not contain any schema or other information explaining what the data represented.

After studying the raw data for some time we managed to extract enough usable information to answer queries based on time, name and/or geographical location.

2.2 Schema and ER-diagram

As was mentioned earlier, the original schema for the data acquired from Ultra is largely unknown. This fact affects the resulting choice of schema and representation within the database. Determining the normal form of the resulting schema requires certain knowledge about the data itself. The goal of using a normal form such as Boyce Codd is to reduce redundancy within the database, but what if you are uncertain of whether data is redundant or not?

This was the situation faced when the schema for this system was created. Instead of either analysing the data (very timeconsuming) or assuming that the data was or was not redundant (which might have introduced more difficulties and

---

²More information can be found at http://www.postgresql.org
³The company running local busses in the city of Umeå
⁴comma separated values
problems), a more flexible design was chosen. The resulting ER-diagram can be seen in figure 1.

Figure 1: The schema of the underlying database is illustrated in the ER-diagram seen above, (the precomputed tables for use with the optimisations are not shown). The tables are quite simple and there are a few relations between them.

2.3 Queries

The query that generates possible travel routes from point A to point B using two different busses, suffered by severe performance problems. The initial version had an execution time of about 30 seconds and consumed hundreds of megabytes of memory. The performance was even worse before all necessary indices were created.

Fortunately we managed to push the execution time down to about 0.5 seconds using explicit \texttt{INNER JOIN} syntax to perform all of the joins within the query in the optimal order to limit the memory usage, combined with the performance boost supplied by the improved query optimiser in PostgreSQL 7.4.

The user is most of the time only interested in the three to five best departures for a given query. This fact can be used to do further optimisations by pre-calculating which parts of the database that need to be included in the search. This allows queries to ignore irrelevant parts of the database by an inexpensive join against a pre-calculated table instead of doing this expensive calculation in real time.
2.4 Web Interface

Users interacts with the system using a lightweight and easy to use web interface. The interface lets the user select search criterias like departure and arriving bus stops and earliest departure time. As figure 2 shows, the search results are presented as a list of available bus routes and if possible, the bus stops will be hyperlinked to a map pinpointing their exact location.

This interface is implemented as a cgi\(^5\)-program written in Python\(^6\). The web interfaces uses a MVC design pattern where the model fetches relevant schedule information by querying the PostgreSQL database. The resulting html-page is generated by applying the result in the model on a ClearSilver\(^7\). This approach has one major advantage. The program can be modified to return the result in many different formats like xml, csv, plain text, etc. by modifying the template and without changing a single line of the actual program.

![Image of the interface showing search results for a journey from Carlstid to Ostkroken with departure times and bus stops]

Figure 2: The resulting interface is quite simplistic, but allows for efficient input and querying of the underlying database. One menu allows for easy navigation between the two main features. The search result above shows the case when a passenger has to switch between busses to reach their destination. The bus stop to switch at is clearly indicated, as well as hyperlinked to a map pinpointing its location.

3 Result

As was displayed in figure 2, the completed system has a minimalistic look and feel. The main advantage of this is the avoidance of bloat, meaning it should be quite possible to access and use the site from various systems such as textbased

---

\(^5\)Common Gateway Interface, http://www.w3.org/CGI/

\(^6\)Python programming language, http://www.python.org

\(^7\)Language-neutral template system, http://clearsilver.net
browsers, across modems or even through the use of mobile phones (even though this has never actually been tested).

The system is capable of answering the different queries relatively fast, as well as display the result in a meaningful and accessible way to the user. As always there are improvements and small things that can be improved upon (e.g. all references to a bus stop should also have a link pointing to a map of that bus stop), but as a proof of concept this system performs well.

Further performance improvements would be possible by reducing the redundant information within the database. The main cause of this was that the original schema was unknown, and there was simply too much data to analyse given the time allocated for this project. Assumptions made on the data might have worked, but might as easily provide inaccurate answers to the queries.

These problems would have been avoided had the original schema for the data been provided, as well as some additional information about exactly what constraints existed on the data.

### 4 Conclusions

The success of public transportation (such as busses or trains) is dependent upon the number of passengers, and how often they travel by these means. In turn the passengers require easy access to the various time tables and schedules that determine when, where and how busses move.

This system allows passengers to both swiftly and easily find which busses to take, where the bus stops are and when the departures are. It also allows for instances where travellers are required to change busses in order to reach the destination. This is an important feature, since it is very timeconsuming to perform such planning by hand.

The implementation of a system like this is fairly straightforward. The web interfaces can rapidly be both developed and changed by the use of applications such as ClearSilver. To separate appearance and functionality allows for the system to be easily managed and improved upon. The same back end and database can therefore be used, while providing different front ends to various platforms such as mobile phones, over the web or by simply displaying the information on public terminals or monitors.