Aether: 600.415 Final Project

Matt Ziegelbaum, H. Parker Shelton
mziegelbaum@acm.jhu.edu, parker.shelton@jhu.edu

December 18, 2009

1 Project Description

We have developed a system, aether, that allows users to make queries for information about international air travel. We currently support queries regarding airlines, airports, and routes as well as airplanes used by the various airlines.

Aether is hosted at `aether.acm.jhu.edu`.

2 Database Design

The database has the following structure:

```
CREATE TABLE Airports (
    ID INT(11) NOT NULL primary key,
    Name VARCHAR(255),
    City VARCHAR(255),
    Country VARCHAR(255),
    IATA VARCHAR(3) NOT NULL,
    ICAO VARCHAR(4) NOT NULL,
    Latitude DECIMAL(12,9),
    Longitude DECIMAL(12,9),
    Altitude INT(11),
    Timezone INT(11),
    DST CHAR(1),
    NumRunways INT(2) UNSIGNED
```

```
);

CREATE TABLE Airlines (
    ID INT(11) NOT NULL primary key,
    Name VARCHAR(255),
    Alias VARCHAR(255),
    IATA VARCHAR(2),
    ICAO VARCHAR(3) NOT NULL,
    Callsign VARCHAR(255),
    Country VARCHAR(255),
    Active CHAR(1)
);

CREATE TABLE Equipment (
    ID VARCHAR(3) NOT NULL primary key,
    Name VARCHAR(255)
);

CREATE TABLE Routes (
    Airline VARCHAR(3) NOT NULL,
    AirlineID INT(11),
    Source VARCHAR(4),
    SourceID INT(11),
    Dest VARCHAR(4),
    DestID INT(11),
    Codeshare CHAR(1),
    Stops INT(11),
    Equipment VARCHAR(255),
    TicketPrice FLOAT UNSIGNED,
    International CHAR(1)
);
```

3 Database Population

The core database information on airports, airlines, and routes was loaded with information from CSV files available from `www.openflights.org`, and AirlineRouteMapper (`arm.64hosts.com`). Additional information giving the

number of runways at each airport was extracted from `www.ourairports.com`. The total data inserted into the database can be seen in the included aetherdb.sql file.

As this data set was fairly dirty, a large amount of effort was spent cleaning it appropriately, i.e. ensuring that all airports and airlines had either a valid IATA or ICAO code and each route's origin and destination airports were contained in the database. Thus, 3,854 airlines without a valid IATA or ICAO code, three airlines without a valid callsign, and 70 airports without a valid IATA or ICAO code were deleted from the database. A large number of airports with no inbound or outbound routes exist in the database, but the query for their removal proved to be prohibitively costly to run, and since these airports are all well-formed, their presence does not interfere with query processing.

A heuristic algorithm was generated to populate the route information with approximate ticket prices:

```
CREATE FUNCTION TicketPrice(dist FLOAT, stops INT, startRunways INT,
                endRunways INT) RETURNS FLOAT DETERMINISTIC
BEGIN
    RETURN ((dist*0.9) / LOG((stops+2)*100)) + 100 - 40*(stops)
           - endRunways*8 - startRunways*5;
END
```

This heuristic has the properties that the longer the flight, the more expensive the ticket, flights with more stops are exponentially cheaper, and the larger the size of the departing and arriving airports, the cheaper the ticket. The distance between airports was calculated using a readily-available formula for distance given latitude and longitude. This heuristic was able to produce surprisingly accurate results for its simplicity:

```
American Airlines, Dallas-Fort Worth International Airport to
    Abiline Regional Airport = $67.74
American Airlines, Dallas-Fort Worth International Airport to
    Baltimore-Washington International Airport = $239.31
British Airways, John F. Kennedy International Airport to
    London Heathrow International Airport = $648.66
US Airways, John F. Kennedy International Airport to
    Los Angeles International Airport = $467.45
```

Any flight with the same departure and arrival airports but with with a layover (not shown) is understandably cheaper.

4 Architecture

Aether is implemented as a web interface constructed with HTML/CSS/JavaScript over a Ruby backend and a MySQL database. It implements the Google Maps API, and uses Sinatra to handle database requests. Something about Mongrel and Apache goes here.

5 Stored Procedures

MySQL stored procedures were created to handle requests from the web interface and return result sets to Ruby, which were then filtered, organized, and returned to the client as JSON. The full list of stored procedures implemented can be seen in the included StoredProcedures.txt.

5.1 Dijkstra's Algorithm

The algorithm for both cheapest flight and shortest path is an adaptation of Dijkstra's shortest-path algorithm to the structure of our database and the constraints of MySQL. The implementation of both procedures can be seen in the included StoredProcedures.txt. The query requires approximately 90 seconds to calculate the shortest path through 5,425 nodes and 54,000 edges, quite impressive performance. Credit to Peter Larsson for the original SQL algorithm (`http://www.sqlteam.com/forums/topic.asp?TOPIC_ID=772620`).