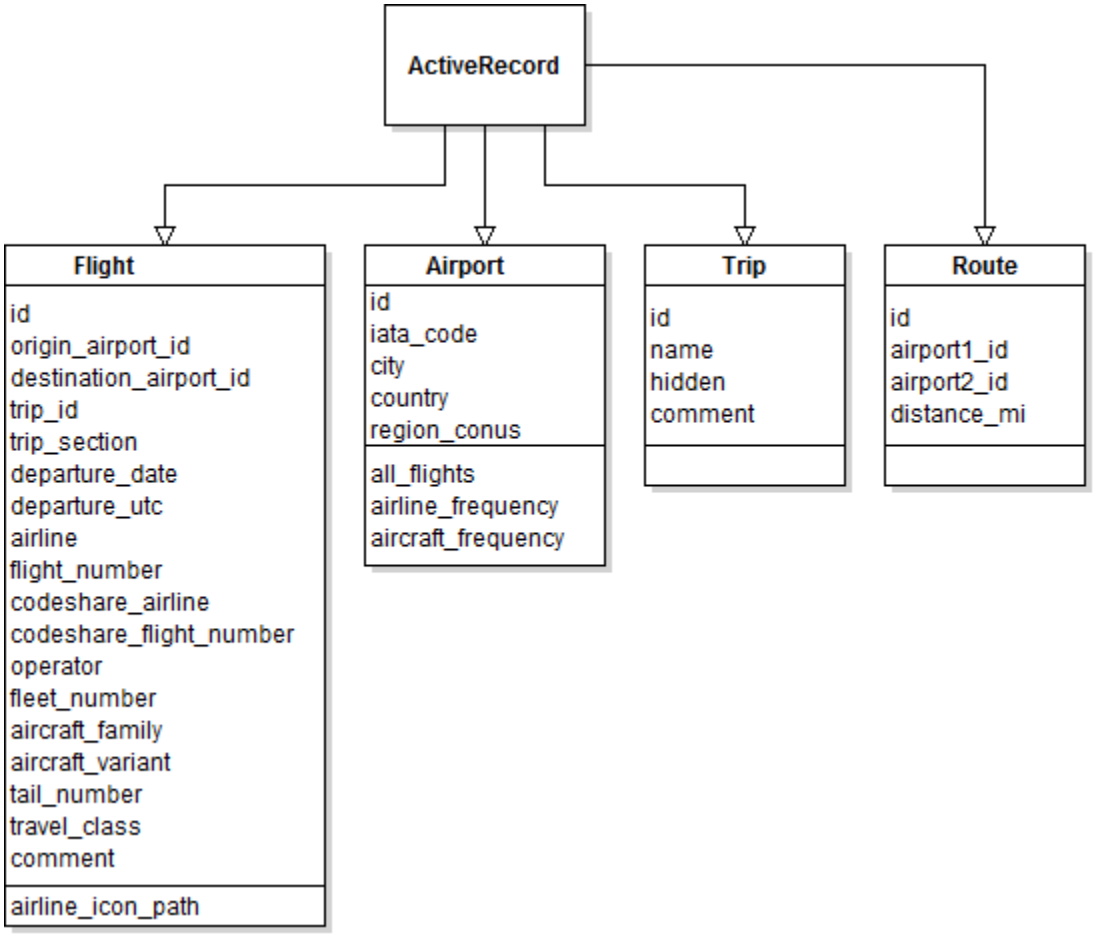
Flight Log Technical Specification

Paul Bogard · February 8, 2015

Contents

Classes 2
Flight 2
Associations 2
Attributes 3
Methods 4
airline_icon_path() 4
Airport 5
Associations 5
Attributes 5
Methods 5
Trip 6
Associations 7
Attributes 7
Methods 8
Database 8
Entity Relationships 8
Size 8

Classes



Flight

Associations

Attributes

Attribute	Type	Description
`id`	integer (required)	Unique flight identifier
`origin_airport_id`	integer (required)	Maps to the id attribute of Airport
`destination_airport_id`	integer (required)	Maps to the id attribute of Airport
`trip_id`	integer (required)	Maps to the id attribute of Trip
`trip_section`	integer (required)	Used to break a trip into subsections
`departure_date`	date (required)	Departure date of the flight (in the local time of the departure airport)
`departure_utc`	datetime (required)	UTC departure date and time, used to sort flights
`airline`	string	Airline branding the flight. For regional subsidiaries, use the parent airline; for codesharing, use the plane's livery.
`flight_number`	integer	The airline's assigned number for this flight
`codeshare_airline`	string	Airline the flight was purchased on and ticketed as
`codeshare_flight_number`	integer	The `codeshare_airline`'s assigned number for this flight
`operator`	string	Airline operating the flight. For mainline flights, this will likely be the same as the `airline` attribute.

`fleet_number`	string	The operator's internal fleet number for the aircraft used for this flight.
`aircraft_family`	string	Manufacturer and family type (e.g. "Boeing 737" and "Airbus A320")
`aircraft_variant`	string	Variant type and model (e.g. "737-800" and "A321")
`tail_number`	string	Tail number for the aircraft used for this flight.
`travel_class`	string	Class of travel (Economy, Business, or First)
`comment`	text	Comment

Methods

Standard Ruby on Rails ActiveRecord methods are available, but not listed in this document.

airline_icon_path()

Returns the path of this Flight's airline's logo icon as a string.

self.aircraft_first_flight(aircraft_family)

Returns the departure_date of the first flight on this aircraft family as a date.

self.airline_first_flight(airline)

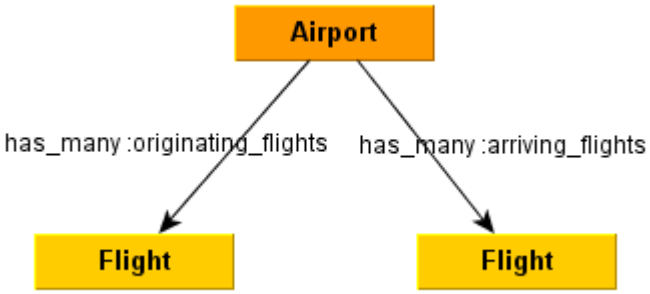
Returns the departure_date of the first flight on this airline as a date.

self.airport_first_visit(airport_id)

Returns the departure_date of the first visit to this airport as a date.

Airport

Associations



Attributes

Attribute	Type	Description
`id`	integer (required)	Unique airport identifier
`iata_code`	string (required)	3-letter IATA code. Must be unique.
`city`	string (required)	Usually the city, with additional information if ambiguous (e.g. “Dayton” and “Chicago-O’Hare” and “Portland (OR)”).
`country`	string (required)	The country that the airport is located.
`region_conus`	bool	True if the airport is in the CONUS region, False otherwise

Methods

Standard Ruby on Rails ActiveRecord methods are available, but not listed in this document.

all_flights(logged_in)

Returns a collection of Flights that have this airport as an origin or destination. If logged_in is false, hidden flights will not be included.

airline_frequency(logged_in)

Returns a hash of the airlines of the flights using this airport, and how many flights involving this airport each airline has. If logged_in is false, hidden flights will not be counted.

aircraft_frequency(logged_in)

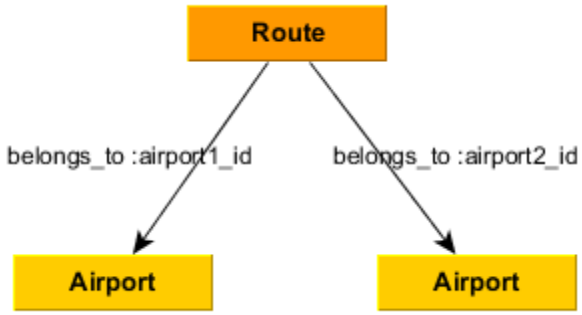
Returns a hash of the aircraft families of the flights using this airport, and how many flights involving this airport each aircraft family has. If logged_in is false, hidden flights will not be counted.

self.frequency_array(flight_array)

Returns a hash of the airports visited in the flights listed in flight_array, and how many times each airport was visited within that list of flights. This method does not filter hidden flights, so care should be taken to pass in an appropriate flight_array.

Route

Associations



Attributes

Attribute	Type	Description
`id`	integer (required)	Unique route identifier
`airport1_id`	integer (required)	Airport 1 id
`airport2_id`	integer (required)	Airport 2 id

`distance_mi`	Integer	Great circle distance between the two airports (in miles)

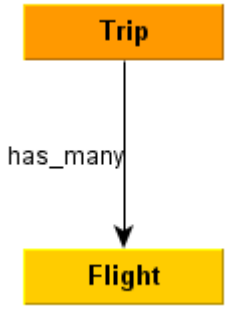
Routes consist of a pair of airports, which are indifferent to the direction flown. Thus, care must be taken to ensure duplicate pairs of airports don't end up in the route table. For example, `[airport1_id,airport2_id] = [5,9]` is the same route as `[airport1_id,airport2_id] = [9,5]`, so only one of these should have a record in the Routes table. In order to ensure this, whenever a new record is submitted, it should check both combinations of airport1, airport2 to ensure a record doesn't already exist; if one does, the existing record shall have its `distance_mi` updated rather than creating a new record. If the record does not exist, then it shall be saved such that `airport1_id` is the lower of the two ids, and `airport2_id` is the higher.

Methods

Standard Ruby on Rails ActiveRecord methods are available, but not listed in this document.

Trip

Associations



Attributes

Attribute	Type	Description
`id`	integer (required)	Unique trip identifier
`name`	string (required)	Trip name
`hidden`	bool	True if the trip is only visible to verified users; False if visible to visitors

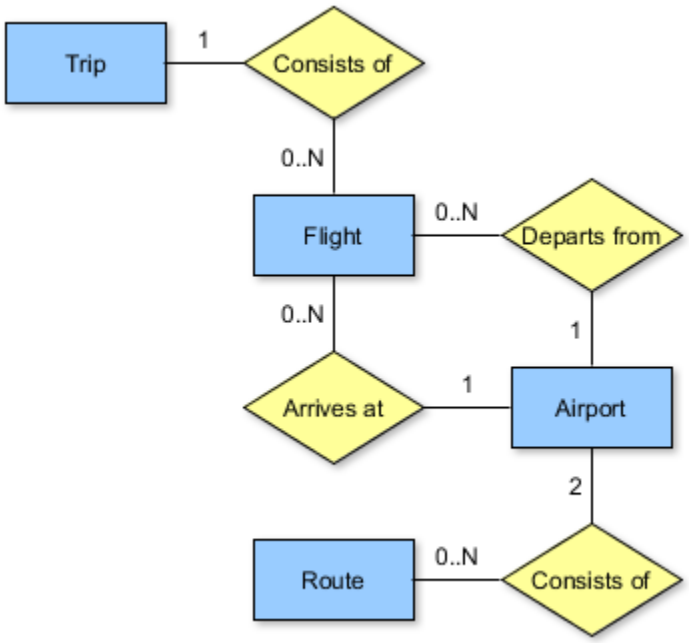
comment	text	Comment

Methods

Standard Ruby on Rails ActiveRecord methods are available, but not listed in this document.

Database

Entity Relationships



Size

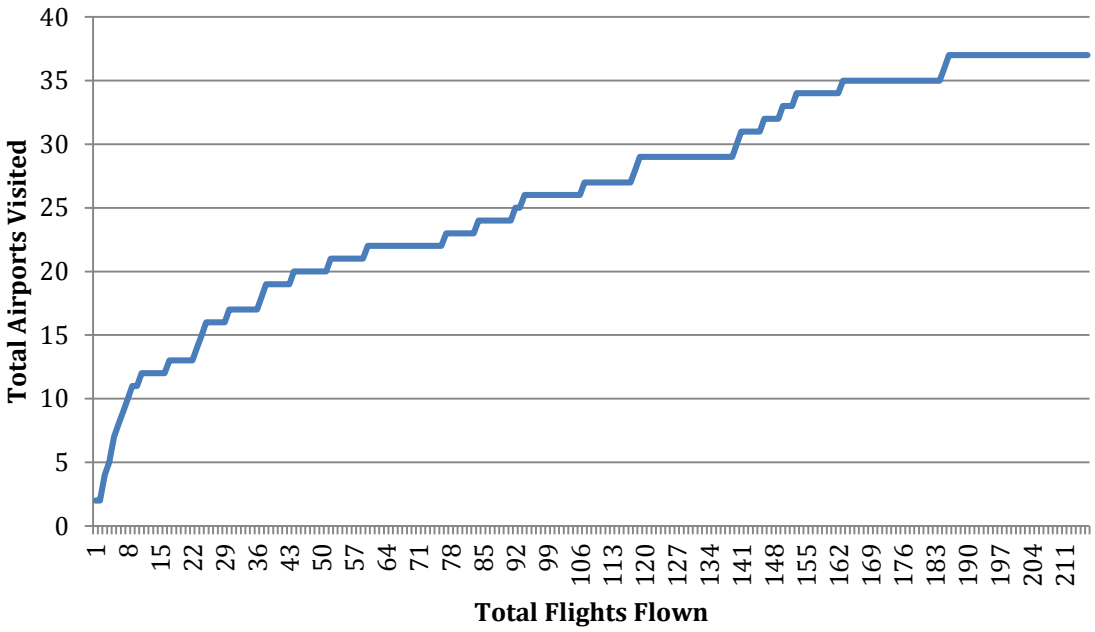
Per the functional specification, this site is intended for a single user (Paul Bogard), which will keep the size small.

For a conservative maximum number of records, assume an average of one trip per day and two flights per day for forty years.

$$\left(\frac{1 \text{ trip}}{1 \text{ day}}\right)\left(\frac{365.25 \text{ days}}{1 \text{ year}}\right)(40 \text{ years}) = 14610 \text{ trips}$$

$$\left(\frac{2 \text{ flights}}{1 \text{ day}}\right)\left(\frac{365.25 \text{ days}}{1 \text{ year}}\right)(40 \text{ years}) = 29220 \text{ flights}$$

At the time of the initial writing of this spec, Paul's flight log contained 219 flights and 37 airports. The number of airports as a function of flights appears to be less than linear. This is logical: the more flights are flown, the more likely it is that the flight will involve airports that have been visited in the past.



To get the worst-case prediction, though, we will assume a linear relationship with a ratio of 37 airports per 219 flights (and a y-intercept of zero).

$$29220 \text{ flights}\left(\frac{37 \text{ airports}}{219 \text{ flights}}\right) = 4937 \text{ airports}$$

Even at these extraordinarily worst-case numbers, these table sizes are easily within the capabilities of MySQL.