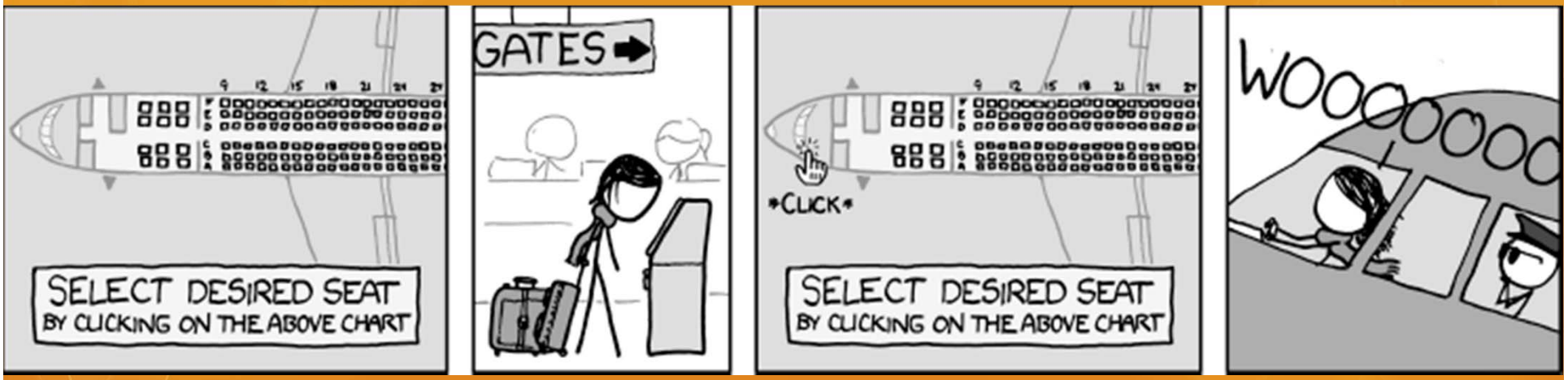


Using PostgreSQL for Flight Planning

Blake Crosby

PGCon 2011

Agenda

- Introduction
- Tools
- Math Class
- Everything you need to know about the Earth
- Ground School
- Challenges
- Displaying the data
- Q&A Session

Some Introductions...

- Senior System Administrator
- Keep cbc.ca up and running
- Manage Apache+Tomcat & Postgres servers.

- PHP Programmer
- Pilot (~250 hours)
- Co-Founder of World Flight Planner

facebook.com/blakecrosby

@blakecrosby

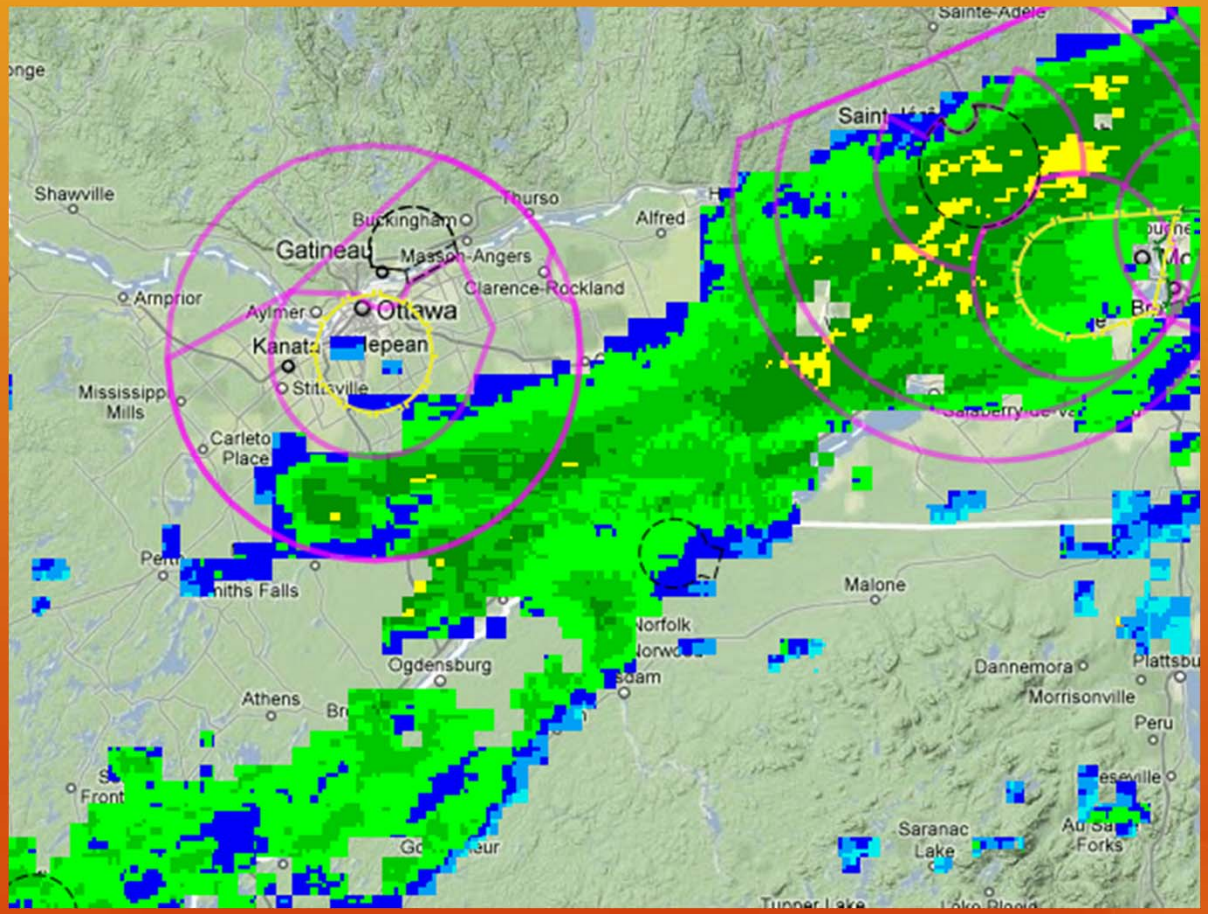
fly.blakecrosby.com

me@blakecrosby

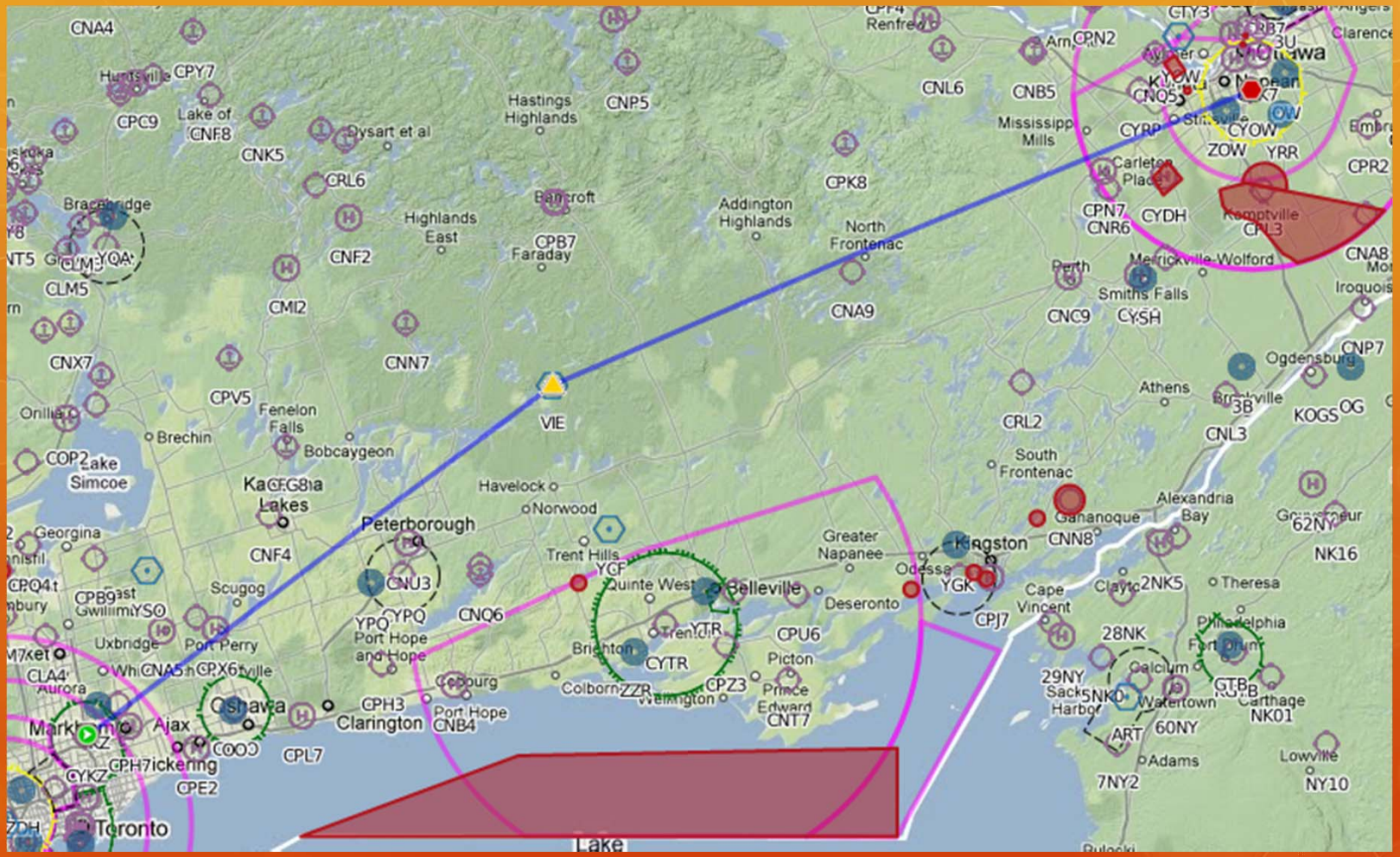
World Flight Planner

- www.worldflightplanner.com
- Launched in 2009
- Gives pilots the tools to plan safe and efficient flights:
  - Weather (windspeed, temperature, etc…)
  - Notices to Airmen (NOTAMs)
  - Airspace information
- Front end is written in Ruby on Rails
- Back end uses PHP, Perl, Postgres+PostGIS.

World Flight Planner



World Flight Planner



World Flight Planner

Route: Flight To Ottawa

Aircraft: Cessna Skyhawk 172S POH

Departure: CYKZ [More Info]
Destination: CYOW [More Info]

Departure Time: Mon May 16 12:00:00 UTC 2011
Generated At: Mon May 16 03:35:26 UTC 2011

Print page

Calculate W&B

Identifier	Alt	Distance			Wind	MC	MH	TAS	EGS	ETE	Fuel		Notes
		Leg	Down	To Go							Used	Remain	
CYKZ 650 ft												51.9	
VIE 115.10 MHz	5500	79.7	0.0	181.9	053@21	062	063	122	101	0:47	8.0	43.9	↑ 5min / 9nm
CYOW 374 ft	5500	102.2	79.7	102.2	071@33	077	079	122	89	1:09	11.7	32.2	↓ 12min / 18nm
		181.9								1:56	19.7		

Route: VIE [view on map]
MOCA: 2132 ft [view elevation profile]

Forms: ICAO Flight Plan

Fuel Log

Taxi	0.3	Flight Fuel	21.1	Available	53.0
Run-up	0.8	Reserve	5.1	Required	26.2
En route	19.7	**Total Required**	**26.2**	Extra	26.8
Approach	0.0				
Taxi	0.3				
Total for Flight	**21.1**				

The Data (About 1GB)

- Over 150,000 unique points, which include:
  - 23,800 Airports
  - 7,300 Heliports
  - 10,500 Traditional radio navigation aids
  - 104,700 Waypoints for GPS/Computer navigation
- Over 2,900 unique polygons, which include:
  - Over 560 restricted airspace definitions
  - Over 2,400 terminal, control, and airport airspace definitions
- Over 800 unique line strings.
- Over 1,600,000 points of wind data covering 72 hours and 15 different altitudes.

Tools We Used:

uDig User-friendly Desktop Internet GIS

mapnik

OpenLayers

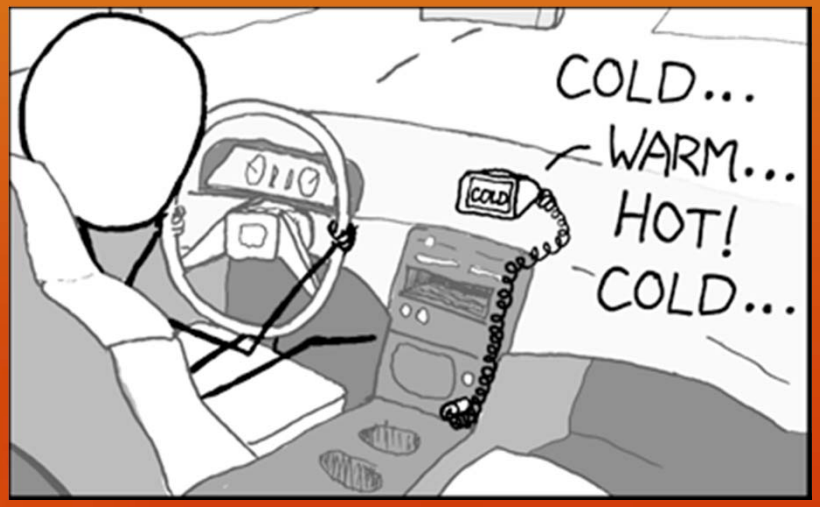
MapServer
open source web mapping

PostgreSQL + PostGIS:

- Allows us to use SQL to answer questions like:
  - How long is my route?
  - Will it pass through any restricted airspace?
  - Will I have a head or tail wind during the flight?
  - What are the nearest airports along my route?
  - Is there any severe weather along the way?
- Postgres' indexes are fast.
  - MySQL not so with spatial data

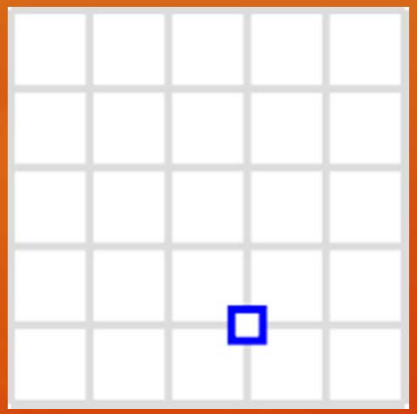
PostgreSQL + PostGIS:

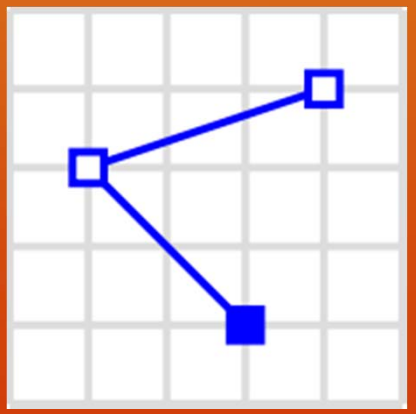
- Gives pilots information to make them safer by:
  - Only showing them data relevant to their flight
  - Visualizing complex data on a map
- The key to a successful flight is detailed and accurate planning

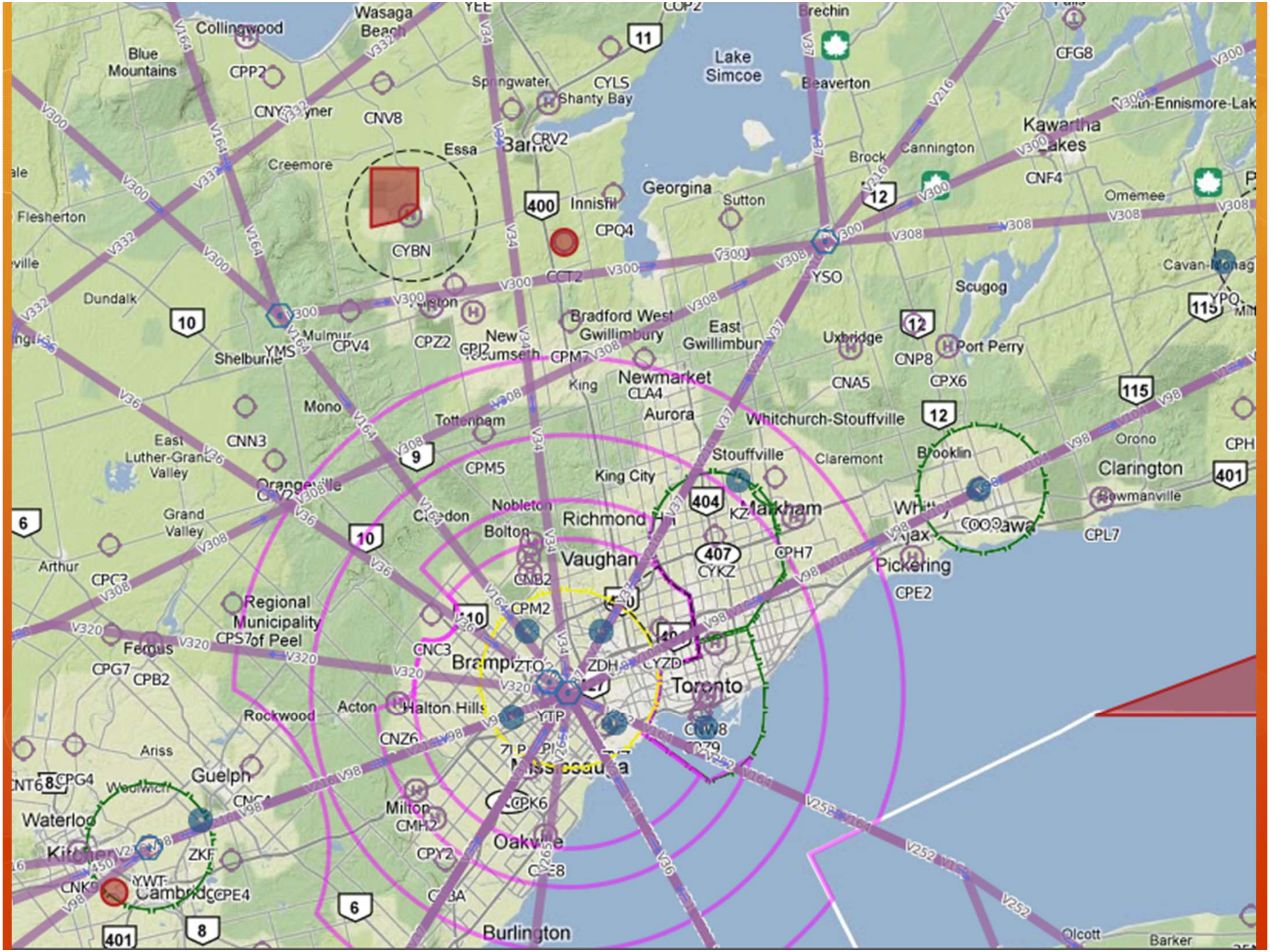


Geographic Data (Vector)

- A Point
  - (-75.6692 45.3225)
- A Line String
  - (-75.99 36.89,-74.16 38.78,-73.36 39.97,-72.86 40.10)
- A Polygon
  - **(-75.99 36.89**,-74.16 38.78,-73.36 39.97,-72.86 40.10,***-75.99 36.89***)

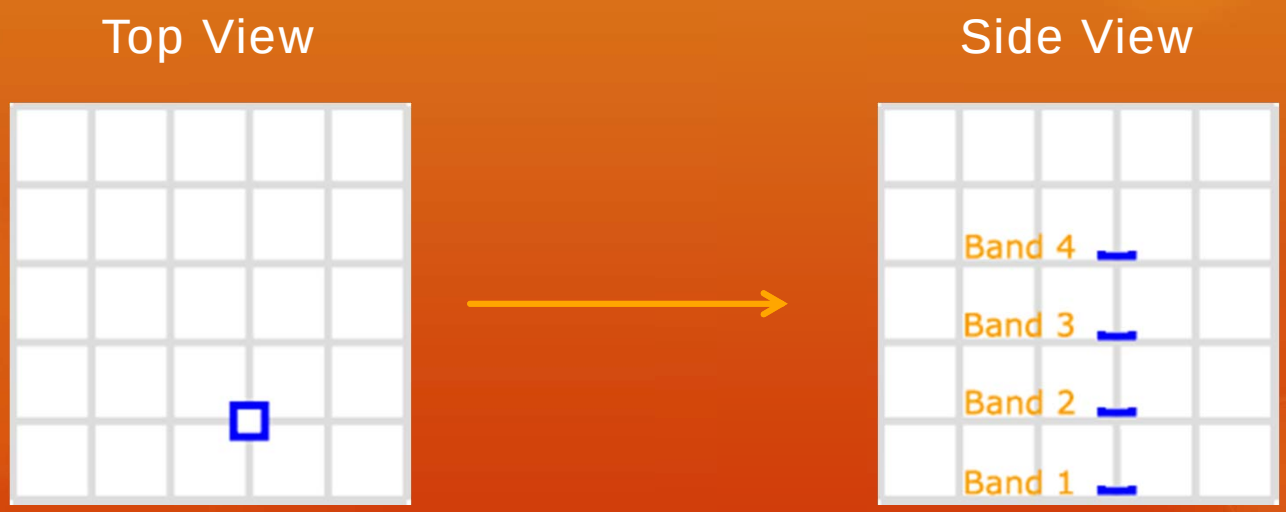





Wasaga Beach
Collingwood
Blue Mountains
Lake Simcoe
Brechin
Beaverton
CFG8
CPP2
CNV8
CYLS
Shanty Bay
Springwater
Barrie
CRV2
Essa
Creemore
Innisfil
Georgina
Sutton
Brock
Cannington
Kawartha Lakes
CNF4
Omemee
Flesherton
CYBN
CPQ4
CCT2
YSO
Scugog
Cavan-Monag
Dundalk
Shelburne
YMS
CPV4
CPZ2
Mulmur
New Tecumseth
CPM7
Bradford West Gwillimbury
East Gwillimbury
Uxbridge
Port Perry
CNP8
CNA5
CPX6
Newmarket
CLA4
King
Aurora
Whitchurch-Stouffville
Mono
Tottenham
East Luther-Grand Valley
CNN3
Orangeville
CPM5
Stouffville
Claremont
Brooklin
Orono
Clarington
King City
Nobleton
Bolton
Richmond Hill
Markham
Vaughan
CYKZ
CPH7
Pickering
CPE2
CPL7
Rowmanville
Grand Valley
Arthur
Regional Municipality of Peel
CPS7
Ferrous
CPG7
CPB2
CNC3
Brampton
ZTO
ZDH
CYZD
Toronto
Acton
Halton Hills
YTP
Rockwood
CNZ6
CNW8
Ariss
Guelph
CPG4
Waterloo
Kitchener
ZKF
YWT
Cambridge
CPE4
Milton
CMH2
CPK6
Oakville
CPY2
Mississauga
Burlington
Olcott
Barker
10
11
12
400
404
407
401
115
9
6
8

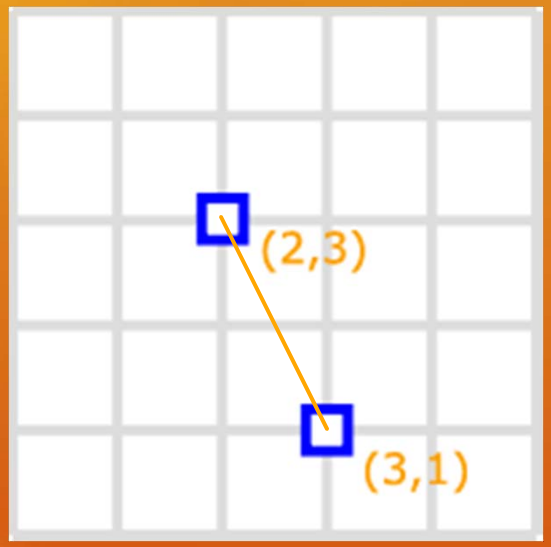
Geographic Data (Raster)

- Multi dimensional data per point in space.
  - Known as “bands”
- Multiple formats for storing data
  - GeoTIFF
  - Gridded Binary (GRIB)



Cartesian Math

- What is the shortest distance between two points?



A Straight Line

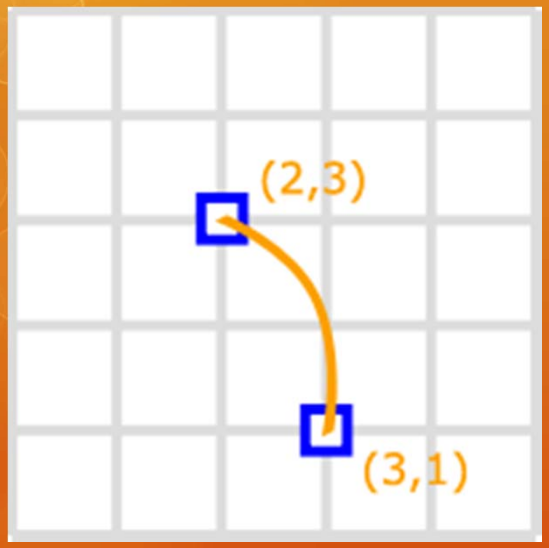
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d = \sqrt{(3 - 2)^2 + (1 - 3)^2}$$

$$d = 2.23606$$

Spherical Math

- What is the shortest distance between two points on a sphere?



$$\Delta\hat{\sigma} = \arctan\left(\frac{\sqrt{(cos\emptyset_f sin\Delta\lambda)^2 + (cos\emptyset_s sin\emptyset_f - sin\emptyset_s cos\emptyset_f cos\Delta\lambda)^2}}{sin\emptyset_s cos\emptyset_f + cos\emptyset_s sin\emptyset_f cos\Delta\lambda}\right)$$

$$d = r\Delta\hat{\sigma}$$

Thankfully, PostGIS can handle this!

A Geodesic Path
(Great Circle)

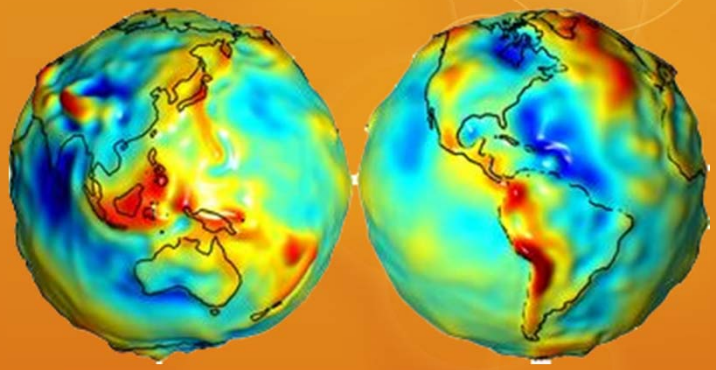
The Earth

- Is not flat
- Is not a sphere
- Is actually an ellipsoid

- It's important to know this distinction.

A flight from Ottawa to Amsterdam

- If the Earth was a perfect sphere
  - 5,631.441 KM
- Using the WGS84 Geoid
  - 5,648.054 KM



- A Geoid is a definition for the shape of the Earth.
- The longer the distance, the greater the error.

- There is still a problem, however.

A flight from Ottawa to Amsterdam

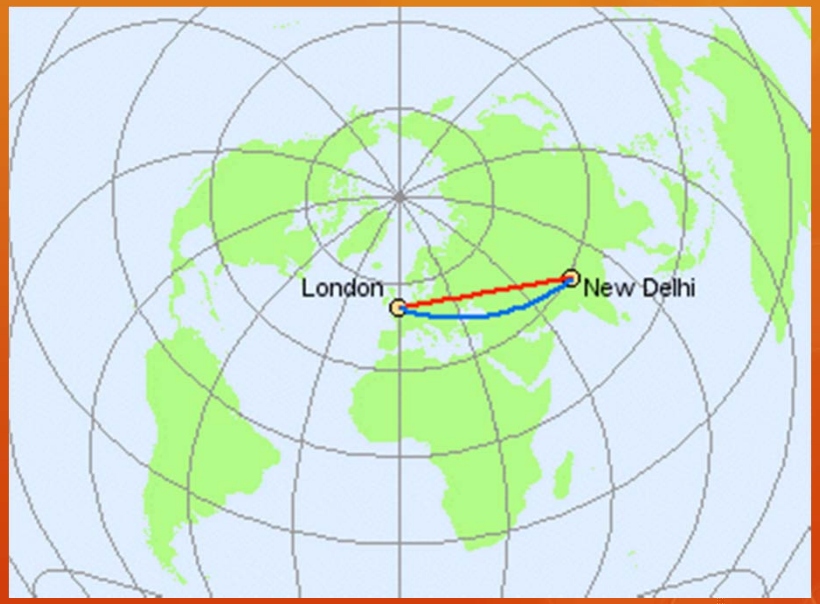
- If the Earth is round? How do we display it on a flat surface (ie: your computer monitor)?
- Answer: Projections

Mercator Gnomonic

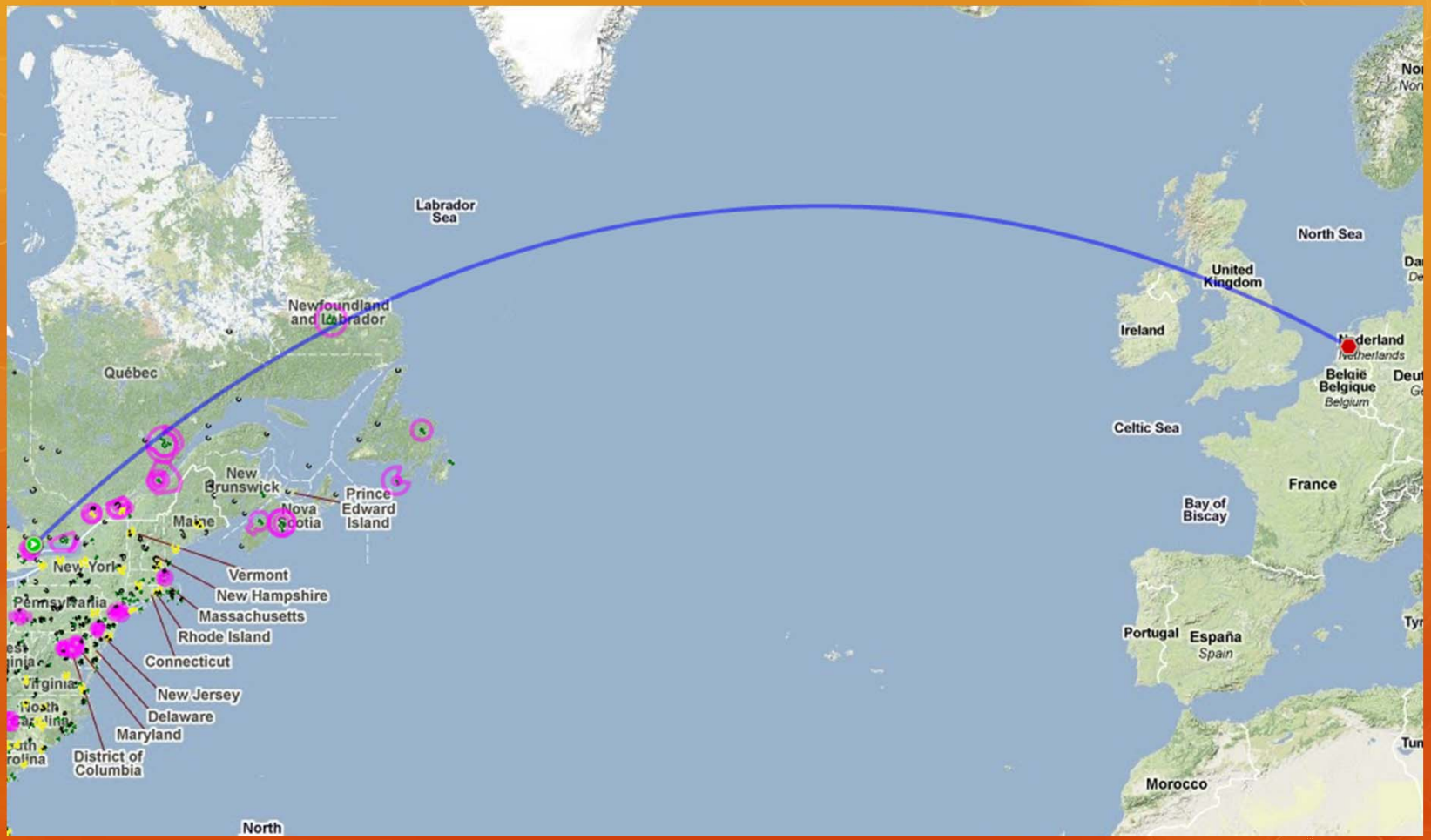
Projections ... Another Example

Mercator

Gnomonic



A flight from Ottawa to Amsterdam



Ground School

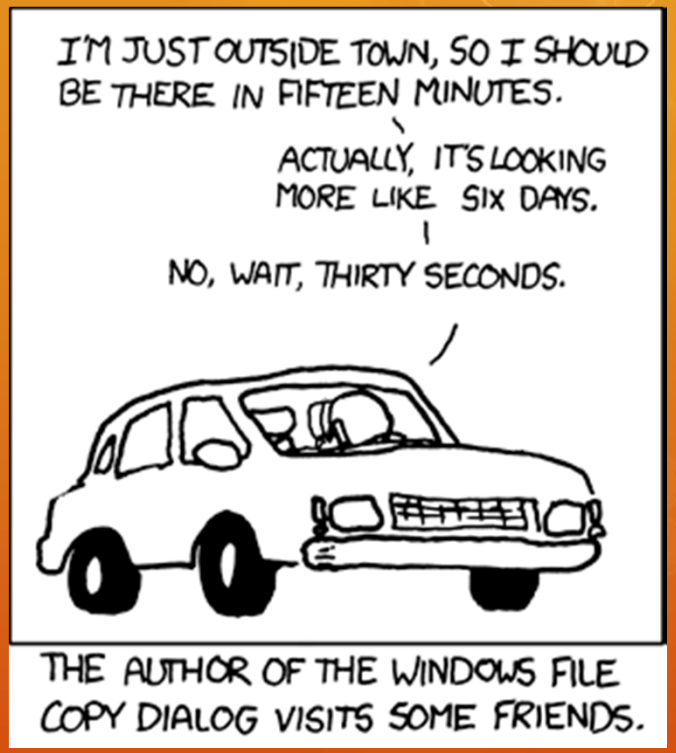
- Aircraft want to get from A → B as quickly as possible.

- Items that help you:
  - Direct routes (no roads to follow)
  - Mother nature (tail winds)
- Items that hinder you:
  - Mother Nature (head winds, thunderstorms)
  - Airspace (going around restricted airspace)

- Tools we use to get where we're going:
  - GPS
  - Compass & Charts
  - Air Traffic Control
  - Radio Navigation Aids (NDBs and VORs)

Ground School

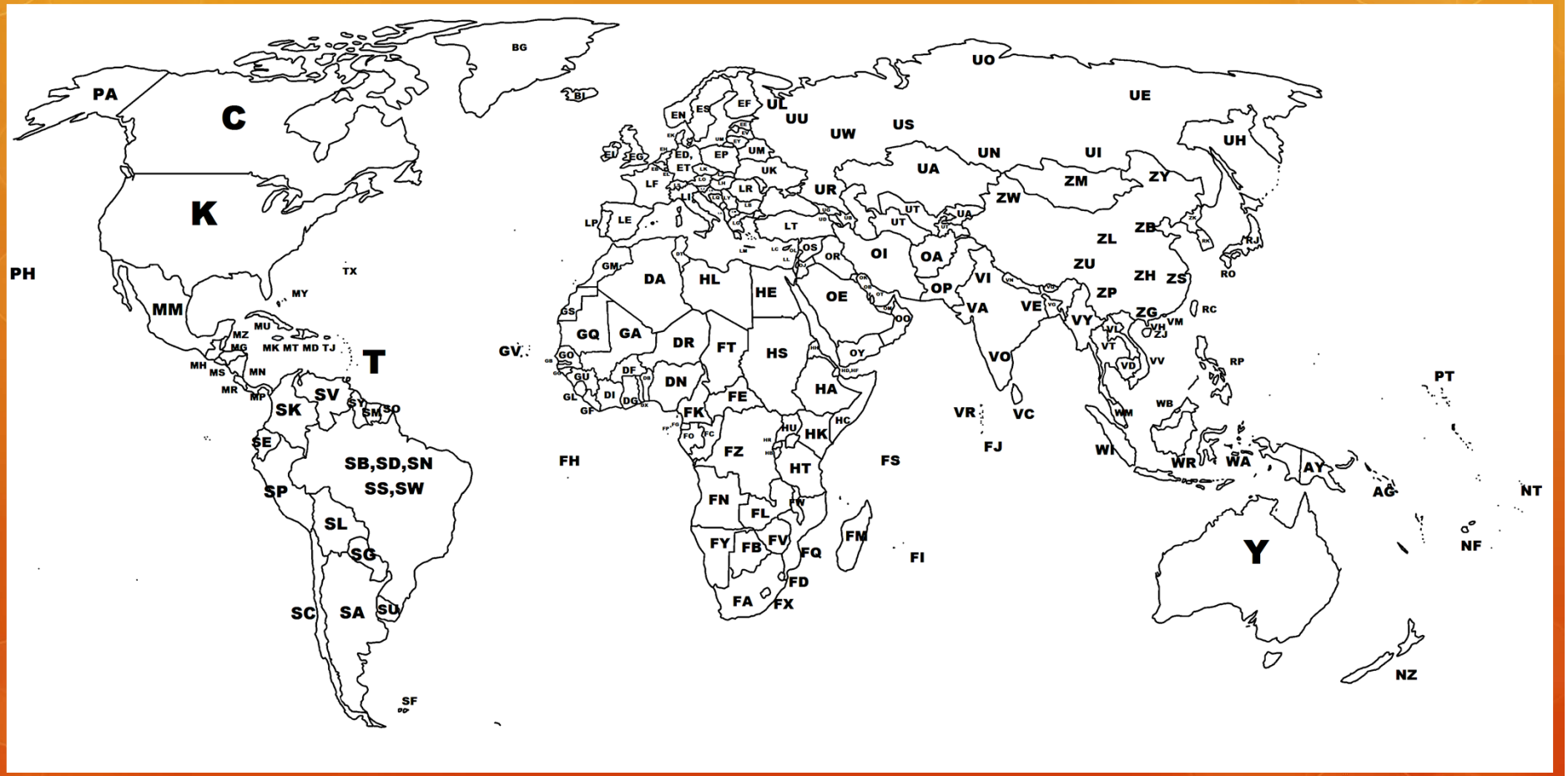
- In order to calculate the best route we need to have:
  - Locations of all the airports (departure and destination points)
  - Locations of all the navigation aids (waypoints)
  - Airspace definitions
  - Wind data for any point on the planet (and altitude)



Challenges: Navaids

- Every airport has an *unique* identifier:
  - Ottawa, CA: CYOW
  - London, UK: EGLL
  - Sydney, AU: YSSY
- Every radio navaid has an identifier:
  - Ottawa, CA NDB: OW
  - Owensboro, US NDB: OW
  - Norwood, US, NDB: OW
- Idents are not unique, not even in the same country!

Challenges: Navaids: ICAO Prefix



Challenges: Navaids

Navaid Schema

```
flightplan=> \d navaid
                          Table "public.navaid"
  Column   |   Type   |                       Modifiers
-----------+----------+-------------------------------------------------
 id        | integer  | not null default nextval('navaid_id_seq'::regclass)
 ident     | text     | not null
 name      | text     | not null
 country   | text     | not null
 city      | text     |
 state     | text     |
 location  | geometry | not null
 elevation | integer  |
 type      | integer  | not null
Indexes:
    "navaid_pkey" PRIMARY KEY, btree (country, type, ident)
Check constraints:
    "enforce_dims_location" CHECK (ndims(location) = 2)
    "enforce_geotype_location" CHECK (geometrytype(location) = 'POINT'::text)
    "enforce_srid_location" CHECK (srid(location) = 4326)
```

Navaid Schema

```
SELECT navaid.country,country.name,navaidtype.type,navaid.ident,
       navaid.city,st_astext(navaid.location)
FROM navaid,navaidtype,country
WHERE navaid.country = country.id
AND navaid.type = navaidtype.id
AND navaid.ident = 'OW';
```

country	name	type	ident	city	st_astext
UL	Russia	NDB	OW	CHEREPOVETS	POINT(37.9633333333333 59.2433166666667)
WI	Indonesia	NDB	OW	PALEMBANG	POINT(104.67645 -2.91065)
CY	Canada	NDB	OW	OTTAWA	POINT(-75.5611166666667 45.36)
K3	Contiguous United States	NDB	OW	OWATONNA	POINT(-93.1552666666667 44.0738333333333)
K5	Contiguous United States	NDB	OW	OWENSBORO	POINT(-87.1622166666667 37.6356)
RJ	Japan	NDB	OW	OSAKA	POINT(135.40365 34.8054833333333)
K6	Contiguous United States	NDB	OW	NORWOOD	POINT(-71.1284166666667 42.11975)
DN	Nigeria	NDB	OW	OWERRI	POINT(7.21075 5.40641666666667)

Navaid Schema

```
SELECT ident,location,elevation FROM navaid WHERE ident = 'CYOW' or ident = 'EHAM';

 ident |                        location                        | elevation
-------+--------------------------------------------------------+-----------
 CYOW  | 0101000020E6100000093A6DA0D3EA52C0E17A14AE47A94640 |       374
 EHAM  | 0101000020E6100000521BE8B4810E134033C4B12E6E274A40 |       -11
(2 rows)
```

- Using PostGIS' ***st_astext()*** function:

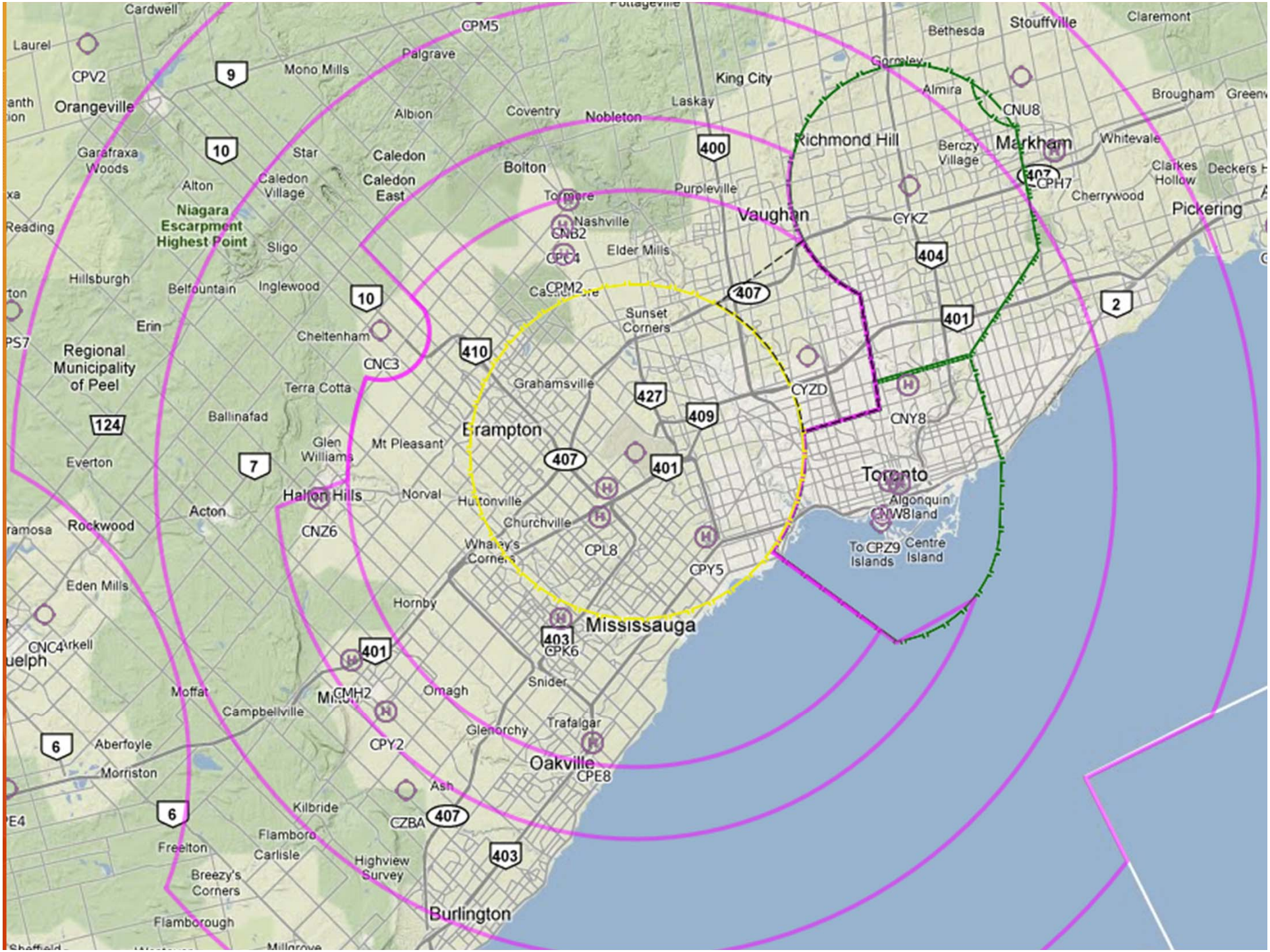
```
SELECT ident,st_astext(location),elevation FROM navaid WHERE ident = 'CYOW' …
 ident |            st_astext             | elevation
-------+----------------------------------+-----------
 CYOW  | POINT(-75.6691666666667 45.3225) |       374
 EHAM  | POINT(4.76416666666667 52.30805) |       -11
(2 rows)
```

Challenges: Airspace & Arcs

- Creating simple polygons is easy.
  - Squares, triangles, octagons, etc…
- However airspace is a complex polygon:

```
The airspace to 2000´ (1400´ AAE) within the area bounded by a line
beginning at:

N43°43'30.37" W079°24'07.83" to
N43°47'10.29" W079°25'01.74" thence clockwise along the arc of a circle
                             of 5 miles radius centred on
N43°51'44.00" W079°22'12.00" (Buttonville Muni, ON - AD) \ to
N43°55'57.12" W079°18'29.25" thence counter-clockwise along the arc of a
                             circle of 2 miles radius centred on
N43°56'09.00" W079°15'44.00" (Markham, ON - AD) \ to
N43°54'09.53" W079°15'59.81" to
N43°48'57.53" W079°14'39.57" to
N43°44'38.30" W079°18'37.97" to
N43°43'30.37" W079°24'07.83" point of beginning
```


Cardwell
Laurel
CPV2
Orangeville
9
Mono Mills
Palgrave
CPM5
Albion
Coventry
Nobleton
King City
Laskay
Bethesda
Stouffville
Claremont
Almira
CNU8
Brougham
Garafraxa Woods
10
Star
Caledon
Caledon Village
Caledon East
Bolton
400
Richmond Hill
Berczy Village
Markham
Whitevale
Alton
Niagara Escarpment Highest Point
Sligo
Tormore
Nashville
CNB2
CPC4
Purpleville
Vaughan
CYKZ
CPH7
Cherrywood
Clarkes Hollow
Pickering
Reading
Hillsburgh
Belfountain
Inglewood
Elder Mills
CPM2
407
404
401
2
Erin
Cheltenham
CNC3
Sunset Corners
Regional Municipality of Peel
410
Terra Cotta
Grahamsville
427
409
CYZD
CNY8
124
Ballinafad
Brampton
Glen Williams
Mt Pleasant
7
Everton
Halton Hills
Norval
Huttonville
Churchville
Toronto
Algonquin
CNW8land
Acton
Rockwood
CNZ6
CPL8
CPY5
To CPZ9 Islands
Centre Island
Eden Mills
Hornby
Mississauga
403
CPK6
CNC4
Arkell
Guelph
401
Moffat
CMH2
Omagh
Snider
Campbellville
Glenorchy
Trafalgar
6
Aberfoyle
CPY2
Oakville
CPE8
Morriston
Ash
Kilbride
CZBA
Flamboro
Freelton
Carlisle
Highview Survey
Breezy's Corners
Burlington
Flamborough

Challenges: Airspace & Arcs

- There are two ways to define arcs:
  - Using a lot of individual lines (***st_curvetoline()***)
  - Using the circular line string geometry type.

- We created a custom function called createarc():
  - Starting Point
  - Ending Point
  - Centre
  - Direction (clockwise or anti-clockwise) OR
  - Size (major or minor arc)
- Returns a line string with 128 points per circle.
- http://trac.osgeo.org/postgis/wiki/UsersWikiplpgsqlfunctions

Challenges: Airspace & Arcs

```
CREATE FUNCTION st_createarc(startpoint geometry, endpoint geometry, arcenter geometry, direction text) RETURNS geometry
    AS $$
  DECLARE
    cwpointonarc geometry;
    ccpointonarc geometry;
    ccarc text;
    cwarc text;
    cwdirection float;
    ccdirection float;
    midpointrads float;
    arcenterutm geometry;
    startpointutm geometry;
    endpointutm geometry;
    thearc geometry;
    majorarc text;
    minorarc text;
  BEGIN
        arcenterutm := st_transform(arcenter,utmzone(arcenter));
        startpointutm := st_transform(startpoint,utmzone(arcenter));
        endpointutm := st_transform(endpoint,utmzone(arcenter));

        midpointrads := abs(st_azimuth(arcenterutm,startpointutm) - st_azimuth(arcenterutm,endpointutm));
        IF midpointrads > pi() THEN midpointrads:= (midpointrads - pi())/2; ELSE midpointrads:= midpointrads/2; END IF;
        IF midpointrads > st_azimuth(arcenterutm,startpointutm) THEN midpointrads:= st_azimuth(arcenterutm,startpointutm)/2; END IF;
        IF midpointrads > st_azimuth(arcenterutm,endpointutm) THEN midpointrads:= st_azimuth(arcenterutm,endpointutm)/2; END IF;
        cwdirection := -1*midpointrads;
ccdirection := midpointrads;
cwpointonarc := ST_Translate( ST_Rotate( ST_Translate( startpointutm, -1*ST_X(arcenterutm), -1*ST_Y(arcenterutm)), cwdirection), ST_X(arcenterutm),
ST_Y(arcenterutm));
ccpointonarc := ST_Translate( ST_Rotate( ST_Translate( startpointutm, -1*ST_X(arcenterutm), -1*ST_Y(arcenterutm)), ccdirection), ST_X(arcenterutm),
ST_Y(arcenterutm));
        cwarc := 'CIRCULARSTRING('||ST_X(startpointutm)||' '||ST_Y(startpointutm)||','||ST_X(cwpointonarc)||'
'||ST_Y(cwpointonarc)||','||ST_X(endpointutm)||' '||ST_Y(endpointutm)||')';
ccarc := 'CIRCULARSTRING('||ST_X(startpointutm)||' '||ST_Y(startpointutm)||','||ST_X(ccpointonarc)||' '||ST_Y(ccpointonarc)||','||ST_X(endpointutm)||'
'||ST_Y(endpointutm)||')';
IF st_length(st_curvetoline(cwarc)) > st_length(st_curvetoline(ccarc)) THEN majorarc := cwarc; minorarc := ccarc; ELSE majorarc := ccarc; minorarc := cwarc;
END IF;
IF direction = 'major' THEN RETURN st_transform(st_setsrid(st_curvetoline(majorarc),utmzone(arcenter)),st_srid(arcenter)); ELSE IF direction = 'minor' THEN
RETURN st_transform(st_setsrid(st_curvetoline(minorarc),utmzone(arcenter)),st_srid(arcenter)); END IF; END IF;
        IF direction = 'cw' THEN RETURN st_transform(st_setsrid(st_curvetoline(cwarc),utmzone(arcenter)),st_srid(arcenter)); ELSE IF direction = 'cc' THEN
RETURN st_transform(st_setsrid(st_curvetoline(ccarc),utmzone(arcenter)),st_srid(arcenter)); END IF; END IF;
  END;
```

Challenges: Airspace & Arcs

- POLYGON((-79.402175 43.7251027777778,-79.4171499999998 43.7861916666658,-79.4222437262452
43.7879570742534,-79.4272120658825 43.7899012082911,-79.4320430798239 43.7920194026045,-79.4367251549832
43.7943065726288,-79.4412470320076 43.7967572274822,-79.4455978322078 43.7993654830112,-79.4497670836202
43.8021250757779,-79.453744746142 43.805029377957,-79.4575212356774 43.8080714131086,-79.4610874472389
43.8112438727908,-79.4644347769474 43.8145391339738,-79.4675551428767 43.8179492772153,-79.4704410046928
43.8214661055553,-79.4730853820377 43.8250811640865,-79.4754818716122 43.8287857601555,-79.4776246629135
43.832570984146,-79.4795085525871 43.8364277307976,-79.4811289573557 43.8403467210089,-79.4824819254884
43.8443185240738,-79.483564146781 43.8483335802993,-79.4843729610167 43.8523822239503,-79.4849063648838
43.8564547064677,-79.4851630173278 43.8605412199043,-79.4851422433212 43.8646319205226,-79.4848440360353
43.8687169524979,-79.4842690574044 43.8727864716697,-79.4834186370756 43.8768306692841,-79.482294769742
43.8808397956703,-79.4809001108588 43.8848041837929,-79.4792379707503 43.8887142726232,-79.4773123071141
43.8925606302725,-79.4751277159383 43.8963339768302,-79.4726894208477 43.9000252068512,-79.4700032609015
43.9036254114369,-79.4670756768673 43.9071258998551,-79.4639136960017 43.9105182206454,-79.460524915371
43.9137941821575,-79.4569174837496 43.9169458724713,-79.4531000821383 43.9199656786492,-79.4490819029465
43.9228463052717,-79.4448726278895 43.9255807922104,-79.4404824046516 43.9281625315941,-79.4359218223728
43.9305852839229,-79.4312018860181 43.9328431932935,-79.426333989692 43.9349308016924,-79.4213298889651
43.9368430623254,-79.4162016722806 43.9385753519459,-79.4109617315123 43.9401234821526,-79.4056227317493
43.9414837096269,-79.400197580382 43.9426527452849,-79.3946993955692 43.9436277623193,-79.3891414741654
43.9444064031117,-79.3835372591913 43.9449867849971,-79.3779003069297 43.9453675048665,-79.3722442537316
43.9455476425945,-79.3665827826175 43.9455267632851,-79.3609295897605 43.9453049183284,-79.3552983509361
43.9448826452662,-79.3497026880254 43.9442609664671,-79.3441561356589 43.9434413866138,-79.3386721080845
43.9424258890107,-79.3332638663468 43.9412169307193,-79.3279444858594 43.9398174365354,-79.3227268244559
43.9382307918239,-79.3176234909975 43.9364608342277,-79.3126468146204 43.9345118442746,-79.3078088146976
43.9323885349042,-79.3031211715935 43.9300960399423,-79.2985951982817 43.9276399015533,-79.2942418128998
43.9250260567001,-79.2900715123064 43.9222608226491,-79.2860943467069 43.9193508815543,-79.2823198954097
43.9163032641615,-79.2787572437714 43.9131253326725,-79.2754149613857 43.9098247628133,-79.2723010815687
43.9064095251516,-79.2694388888888 43.9029083333333,-79.2666138888889 43.9026472222222,-79.244325
43.8159805555556,-79.3105472222222 43.7439722222222,-79.402175 43.7251027777778),(-79.4171499999998
43.7861916666658,-79.4171499999987 43.7861916666654,-79.41715 43.7861916666667,-79.4171499999998
43.7861916666658),(-79.2694388888888 43.9029083333333,-79.2694388888889 43.9029083333333,-79.2694388888869
43.902908333331,-79.2694388888888 43.9029083333333))

Challenges: Airspace & Arcs

- If we could do it all over again…
- Stick with the circular curve geometry types:
  - CIRCULARSTRING
  - COMPOUNDCURVE
  - CURVEPOLYGON

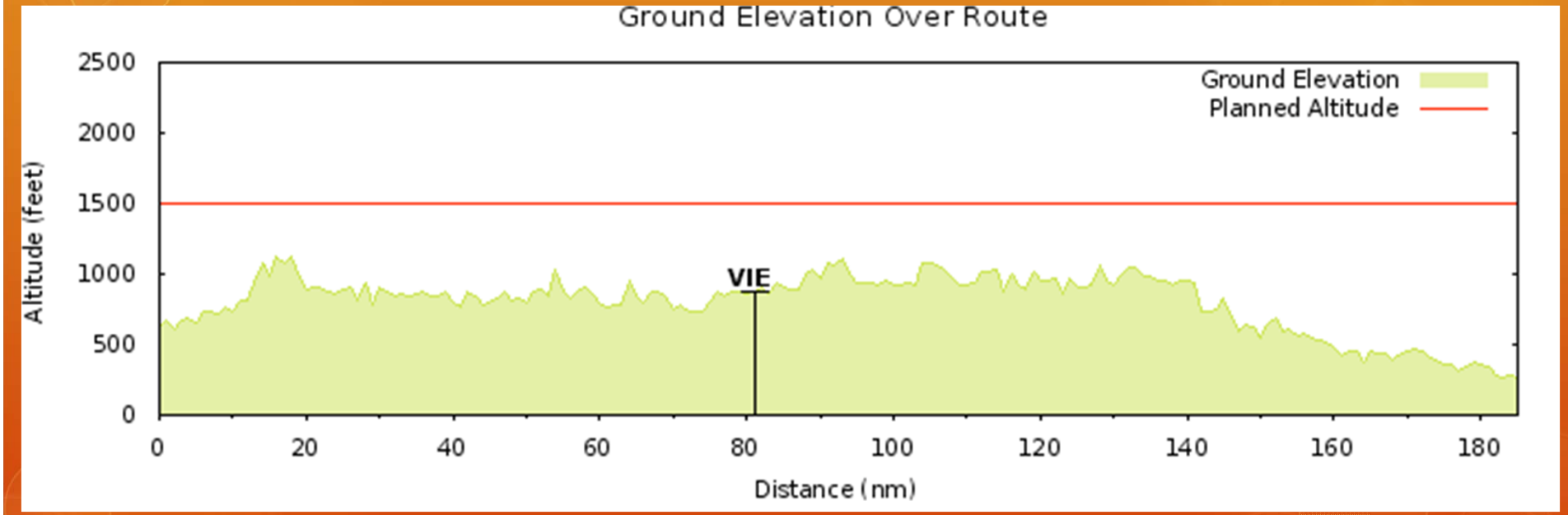
Challenges: Storing Raster Data

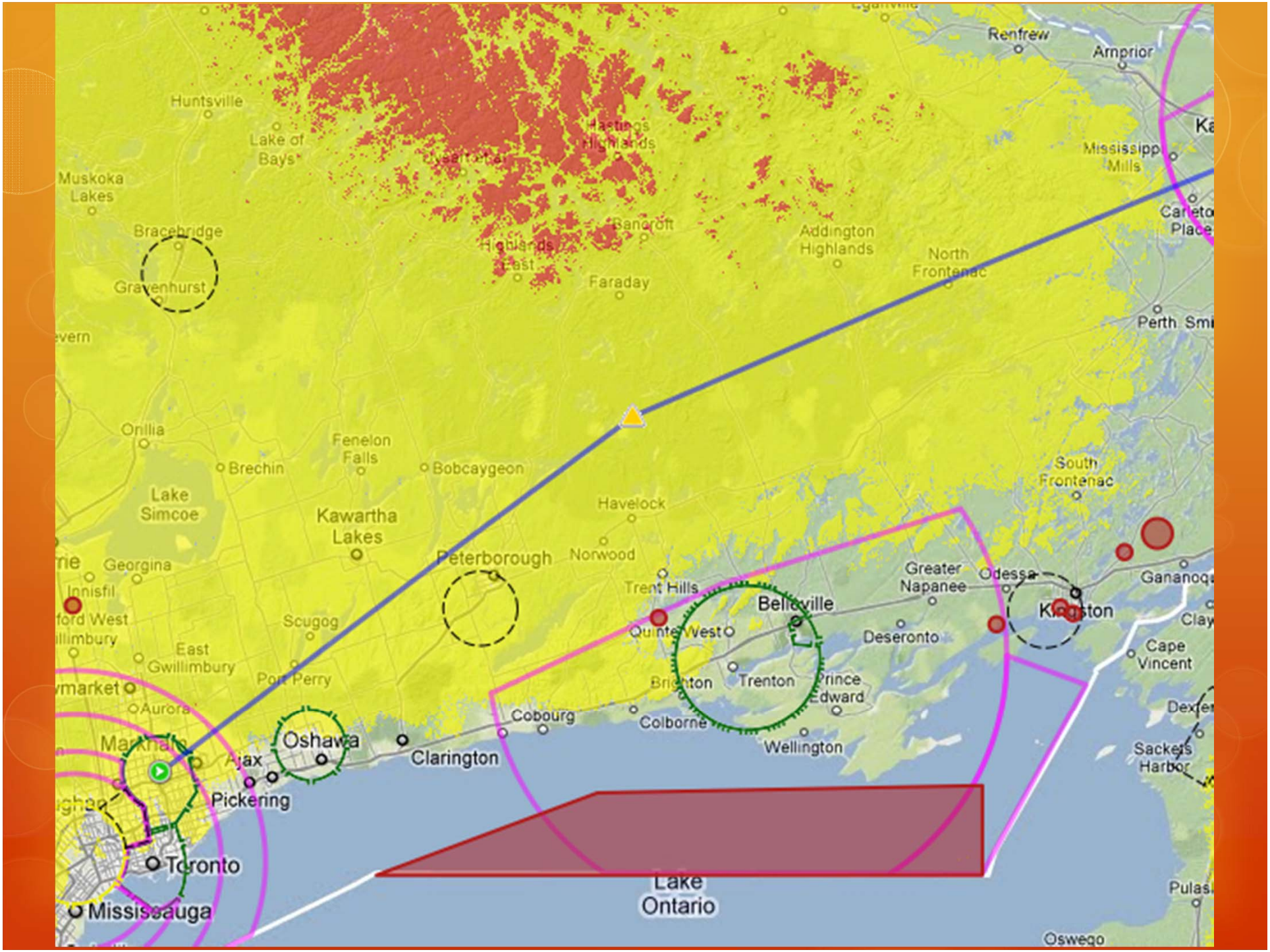
- PostGIS doesn't support raster data (in version 1.5)
- Weather data from NOAA is stored in GRIB format

- Problem: How to store raster data in PostgreSQL?

Challenges: Storing Raster Data

- Solution 1: We Don't




Renfrew
Arnprior
Huntsville
Lake of Bays
Muskoka Lakes
Bracebridge
Gravenhurst
Bancroft
Addington Highlands
North Frontenac
Faraday
Perth
Orillia
Fenelon Falls
Brechin
Bobcaygeon
Lake Simcoe
Kawartha Lakes
Havelock
Peterborough
Norwood
South Frontenac
Georgina
Innisfil
Scugog
East Gwillimbury
Port Perry
Trent Hills
Belleville
Quinte West
Greater Napanee
Deseronto
Odessa
Cape Vincent
Aurora
Cobourg
Colborne
Brighton
Trenton
Prince Edward
Wellington
Oshawa
Ajax
Pickering
Clarington
Toronto
Mississauga
Lake Ontario
Sackets Harbor
Oswego

Challenges: Storing Raster Data

- Solution 2: We use the array datatype.

```
flightplan=> \d grib
                                Table "public.grib"
  Column  |           Type           |                        Modifiers
----------+--------------------------+-----------------------------------------
 id       | integer                  | not null default nextval('gribtemp_id_seq1':
 data     | double precision[]       | not null
 pressure | double precision         | not null
 valid    | timestamp with time zone | not null
 location | geometry                 |
```

Challenges: Storing Raster Data

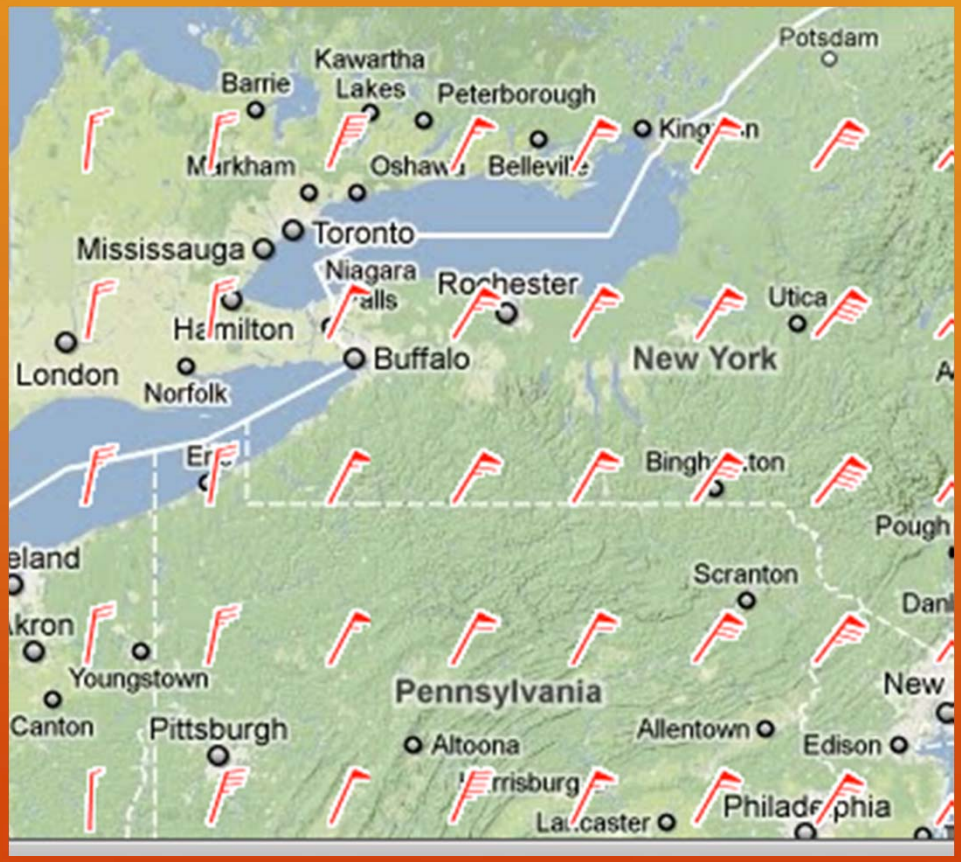
- Convert from GRIB to CSV to Array (using Perl)
- Each element in the array is a different altitude

```
flightplan=> SELECT data,st_astext(location),valid FROM grib LIMIT 1;

                    data                    |    st_astext    |        valid
--------------------------------------------+-----------------+----------------------
 {{-42.4,30,11.4},{-33.6,13,1.4},…} | POINT(-179 -90) | 2011-05-16 12:00:00+00
(1 row)
```

- Fast querying, but long load time (~40 min)
- Version 2.0 of PostGIS will have full support for rasters
  - Hurry Up! ☺

Challenges: Storing Raster Data



Flight Planning: How far is it?

- ***st_distance()*** function:

```
SELECT   st_distance(
                    st_geomfromtext('POINT(4.76416666666667 52.30805)',4326),
                    st_geomfromtext('POINT(-75.6691666666667 45.3225)',4326)
         );

   st_distance
------------------
 80.7361072873446
(1 row)
```

- ***st_distance()*** returns the Euclidian distance. Not Good!
- WGS84 units are in “degrees”.
- So what are our options?

Flight Planning: How far is it?

- ***st_distance_spheroid()*** function:

```
SELECT   st_distance_spheroid(
                  st_geomfromtext('POINT(4.76416666666667 52.30805)',4326),
                  st_geomfromtext('POINT(-75.6691666666667 45.3225)',4326),
                  'SPHEROID["WGS 84",6378137,298.257223563]'
         );

    st_distance_spheroid
------------------------
    5648054.61931799
(1 row)
```

- ***st_distance_spheroid()*** returns the geodesic distance. Good!
- Also returns distance in meters.
- ***Remember: If you want accurate results on the Earth, you must use spheroid functions!***

Flight Planning: Closest Airports

- st_distance_sphere()
- st_dwithin() + st_buffer()
- Nearest neighbour function
  - Boston GIS website

Qantas Air Safety Card
(ca. 2007)

Flight Planning: Closest Airports

```
SELECT ident,name
FROM navaid
WHERE type = '1'
        AND st_distance_sphere(geomfromtext('LINESTRING
        (-79.3687166666667 43.86085,-75.6691666666667
        45.3225)',4326),location) <= '20000';

 ident |             name
-------+------------------------------
 CNP8  | GREENBANK
 CNA9  | PLEVNA/TOMVALE
 CYKZ  | TORONTO/BUTTONVILLE MUNICIPAL
 CYOO  | OSHAWA
 CYRP  | CARP
 CYPQ  | PETERBOROUGH
 CYOW  | OTTAWA/MACDONALD-CARTIER INTL
 CNU8  | TORONTO/MARKHAM
 CYRO  | ROCKCLIFFE
 CYZD  | TORONTO/DOWNSVIEW
 CPS2  | KEENE/ELMHIRST'S RESORT
 CNR6  | CARLETON PLACE
(12 rows)
```

Flight Planning: Closest Airports

```
SELECT DISTINCT navaid.ident,name
FROM navaid,runway
WHERE type = '1'
        AND st_distance_sphere(geomfromtext('LINESTRING     (-
79.3687166666667 43.86085,-75.6691666666667 45.3225)',
        4326),location) <= '20000'
        AND runway.length >='4000'
        AND runway.ident = navaid.ident;
```

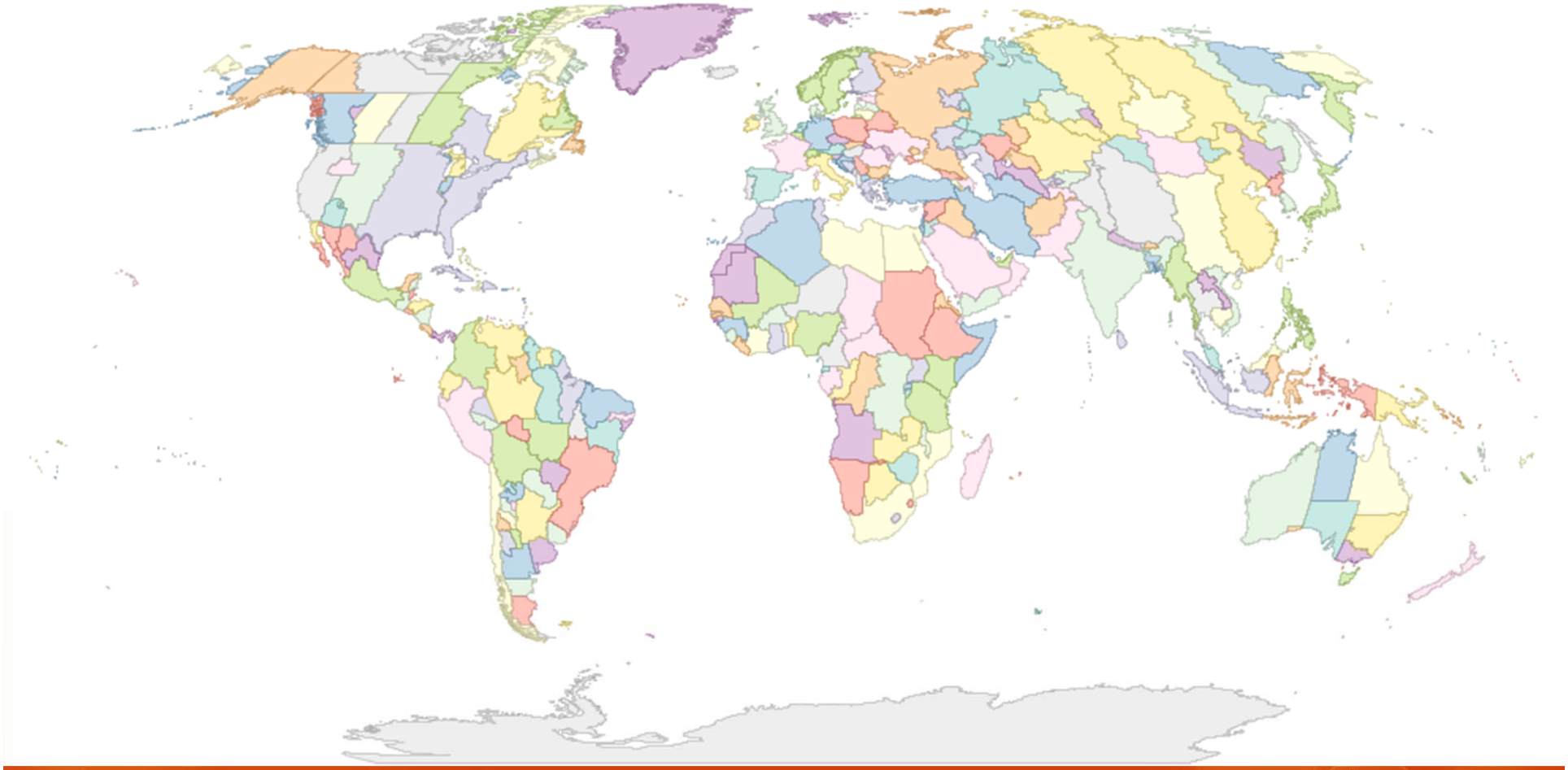
```
 ident |             name
-------+------------------------------
 CYOO  | OSHAWA
 CYPQ  | PETERBOROUGH
 CYOW  | OTTAWA/MACDONALD-CARTIER INTL
 CYZD  | TORONTO/DOWNSVIEW
(4 rows)
```

Flight Planning: Airspace?

- st_intersects()

```
SELECT name,class,notes
FROM airspace
LEFT OUTER JOIN airspaceinfo ON airspace.id = airspaceinfo.airspaceid
        WHERE st_intersects(geomfromtext('LINESTRING(-79.3687166666667
                                43.86085,-75.6691666666667 45.3225)',
                                4326),geometry);

              name                 | class | notes
-----------------------------------+-------+-------
 Toronto TCA 3                     | TCA   |
 Peterborough Control Zone         | E     |
 Ottawa International Control Zone | C     |
 Toronto TCA 4                     | TCA   |
 Buttonville Control Zone          | D     |
 Buttonville Control Zone 2        | D     |
 Ottawa TCA                        | TCA   |
 Ottawa TCA                        | TCA   |
 Ottawa TCA                        | TCA   |
(9 rows)
```



Flight Planning: Time Zones

- Time in the aviation world: Always UTC
- However, it's nice to know your ETA in local time.

```
SELECT timezones.tzid
FROM timezones
WHERE st_dwithin(st_geomfromtext('POINT(4.76416666666667
52.30805)',4326),the_geom,0.01);

       tzid
-----------------
 Europe/Amsterdam
(1 row)
```

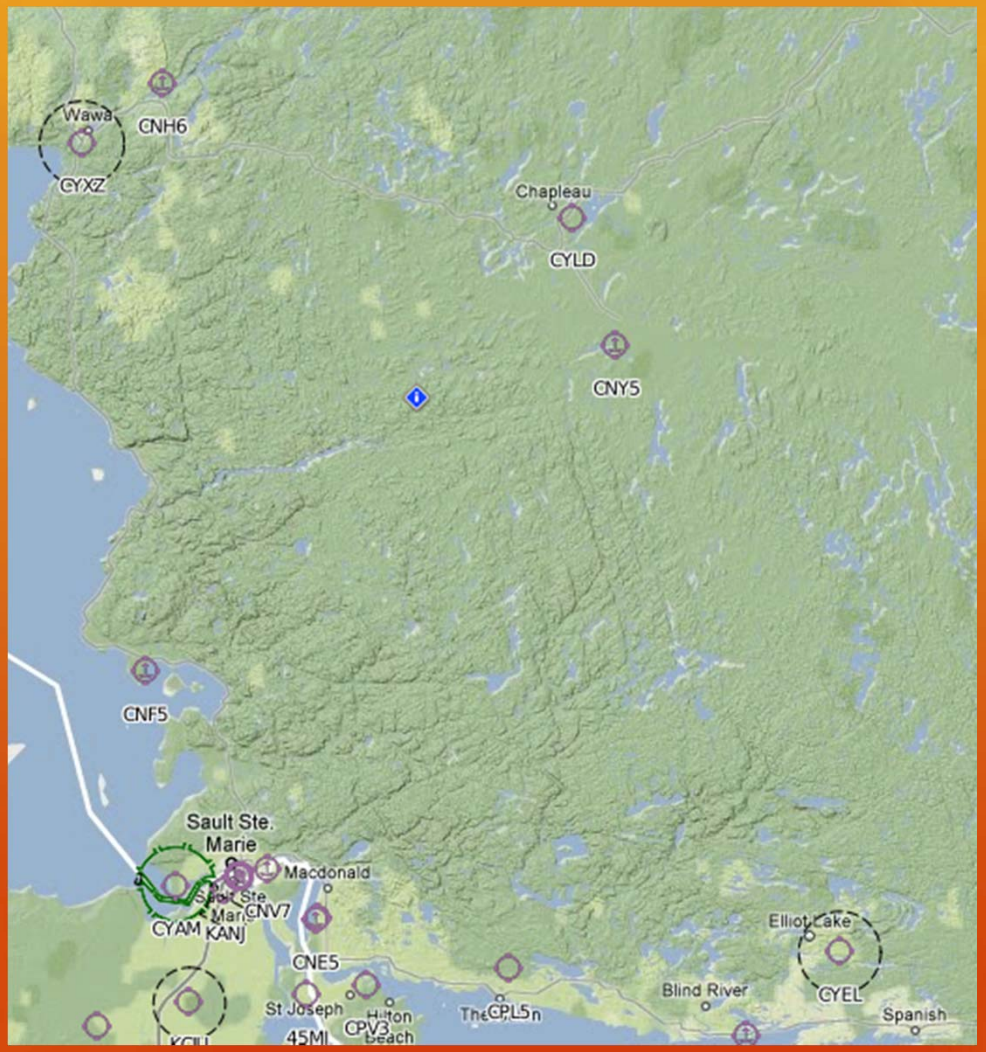
Pilot Reports

- Reports from pilots about weather conditions

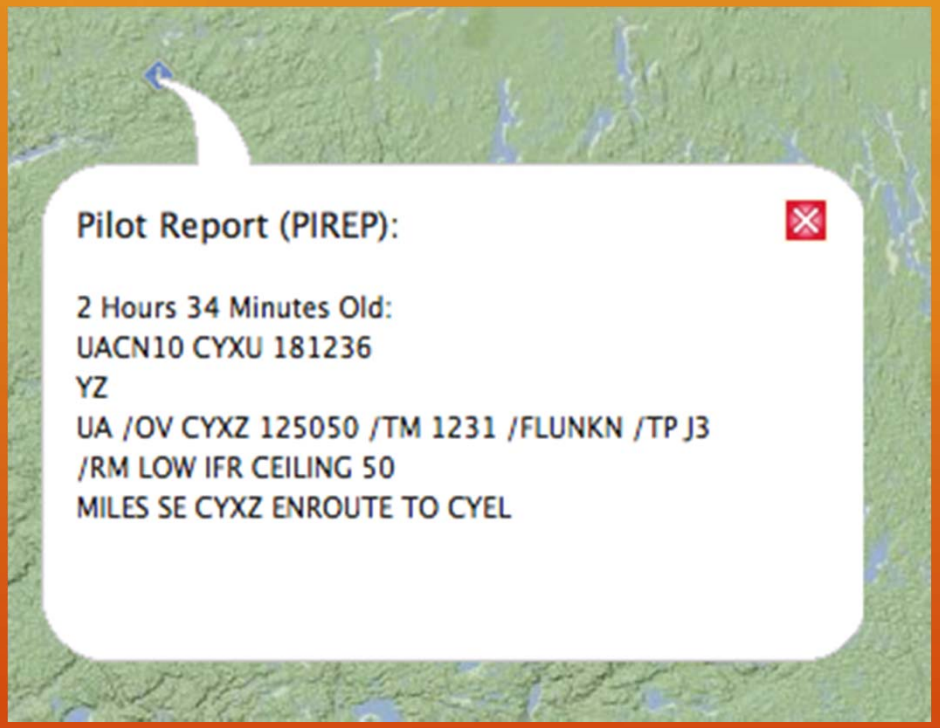
```
UACN10 CYXU 181236
YZ
UA /OV CYXZ 125050 /TM 1231 /FLUNKN /TP J3 /RM LOW IFR
CEILING 50 MILES SE CYXZ ENROUTE TO CYEL
```

- st_translate()
- RotateAtPoint()
  - http://trac.osgeo.org/postgis/wiki/UsersWikiplpgsqlfunctions

Pilot Reports



Pilot Reports



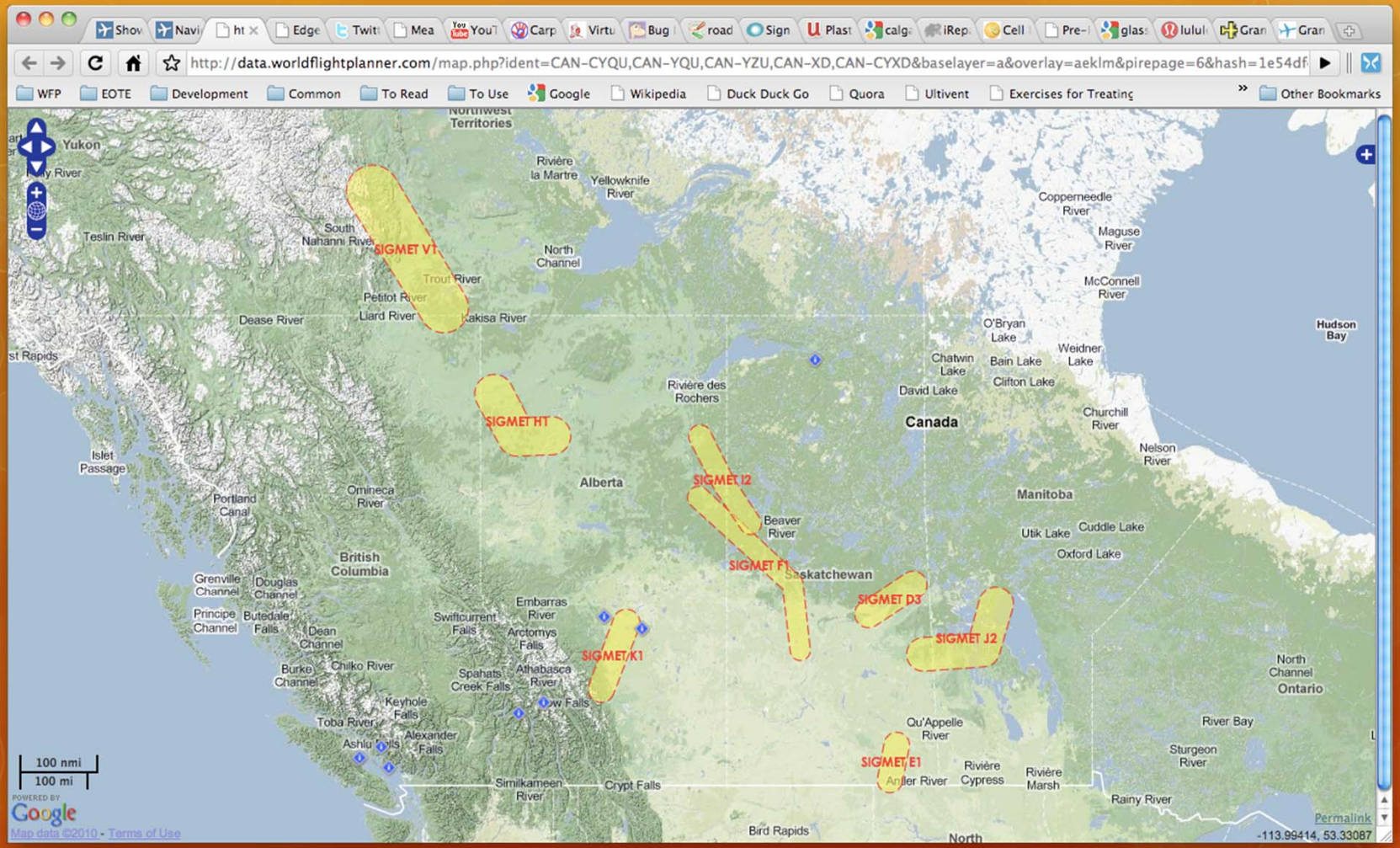
Severe Weather

- SIGMET (Signifigant Meterological Report)
- Can be quite complex and hard to visualize.

```
WSCN33 CWTO171805
SIGMET A5 VALID
171805/172205 CWTO
WTN 30 NM OF LN /4622N 07925W/NORTH BAY–
/4458N07918W/MUSKOKA– /4302N08109W/ LONDON.
TS MAX TOPS 300 OBSD ON RADAR. LN MOVG EWD AT 20
KT.LT LCHG IN INTSTY.
```

- Create a line string
- st_buffer() to create 30nm wide polygon

Severe Weather



Displaying Data To Users

- How do you display your route and data to end users?

Rendering

Navigating

The Future: Geography Type

- PostGIS 1.5 introduced the geography datatype
- Great because it assumes you're on an ellipsoid and not a plane
- Bad because it doesn't support all of the functions that we need to use
- Slowly moving our queries & data over to the geography type as function support increases.

In Conclusion...

- Allows us to export the data in multiple formats:
  - KML
  - Well Known Text (WKT)
  - GML
  - Plain old ints (using st_x() and st_y() functions)
- Allows us to JOIN spatial and non spatial data easily.
- Allows us to perform complex math on geometries:
  - Joining and splitting polygons
  - Clipping shapes to fit inside a specific country
- Allows us to use any programming language without having to have GIS support. (just SQL support)

Thank You

- Questions?