https://is.gd/xp1wpB

Geocoding and Text Search in PostGIS

paul.ramsey@crunchydata.ca

PostGIS Day STL - November 14, 2019

crunchy data

What is this “geocoding” of which you speak?

Converting a
description of a location
into
geographic **coordinates**.

What is this “geocoding” of which you speak?

“1645 Main Street, Springfield, Illinois”

“Ontario, CA”

“914 -- 582 Massachusetts Blvd NE, Duluth, MN”

“12th and Grand”

“Canadia”

Geocoding is easy!

```
SELECT ST_Centroid(geom)
FROM countries
WHERE name = 'Canada';
```

Wait. What if the countries table uses uppercase names? Or some other casing convention?

Geocoding is easy!!

```
SELECT ST_Centroid(geom)
FROM countries
WHERE upper(name) = upper('Canada');
```

Wait. What if the user fat fingers the name?

Geocoding is easy?

```
SELECT ST_Centroid(geom)
FROM countries
WHERE soundex(name) = soundex('Canada')
AND levenshtein(upper(name),
                upper('Canada')) <= 1;
```

Wait....

Geocoding is **not easy**

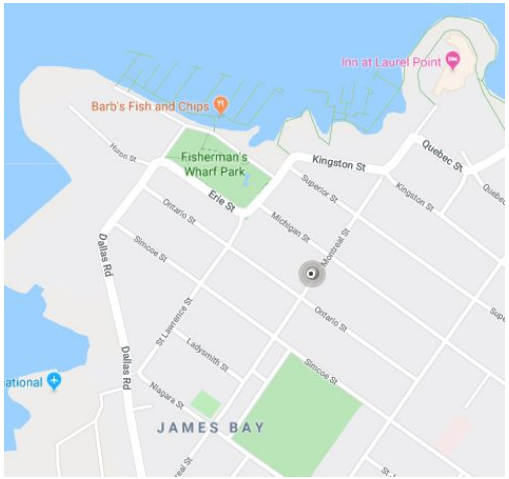
What is this “reverse geocoding” of which you speak?

Converting
geographic **coordinates**.
into a
location **description**

Reverse geocoding is easy!

```
POINT(-123.380491 48.420038)
```

- Canada?
- British Columbia?
- Victoria?
- 232 Montreal Street?
- Unit 298?



Geocoding is **hard**

Geocoding software is hard

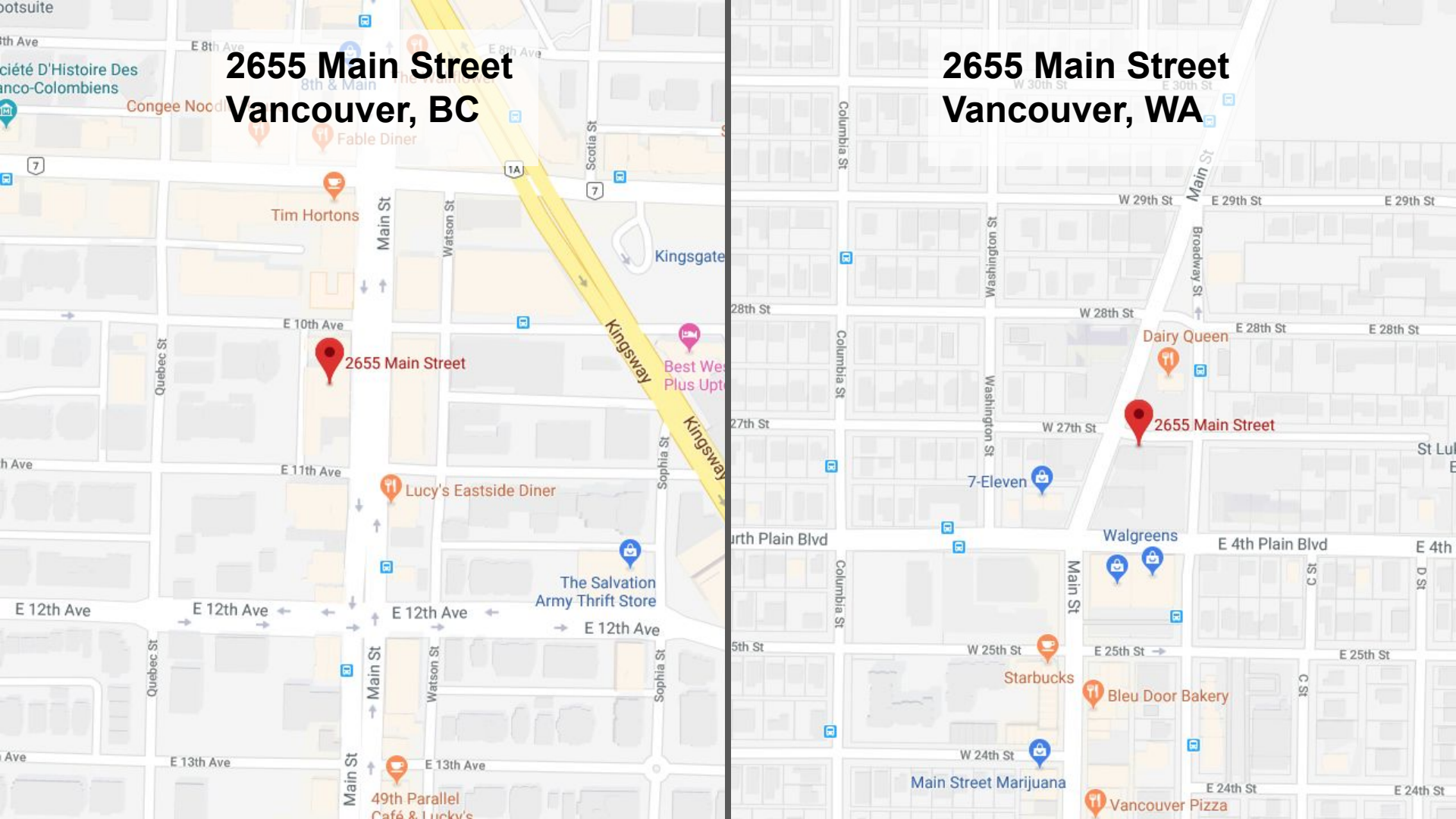
- 23--154 Fort Street S, Victoria, British Columbia
- 23--154 South Fort Street, Victorai, BC
- UNIT 23, 154 S Fort St, Victoria, BC
- 154 S Fort St, UNIT 23, Victoria, BC
- 23 154 Fort St S, Victoria
- 154 Fort St S, Downtown, Victoria
- 154 Fort Street, Victoria, British Columbai
- 154 Fort St, View Royal, BC

Geocoding software is hard

“154 Fort St, Victoria S, British Columbia”

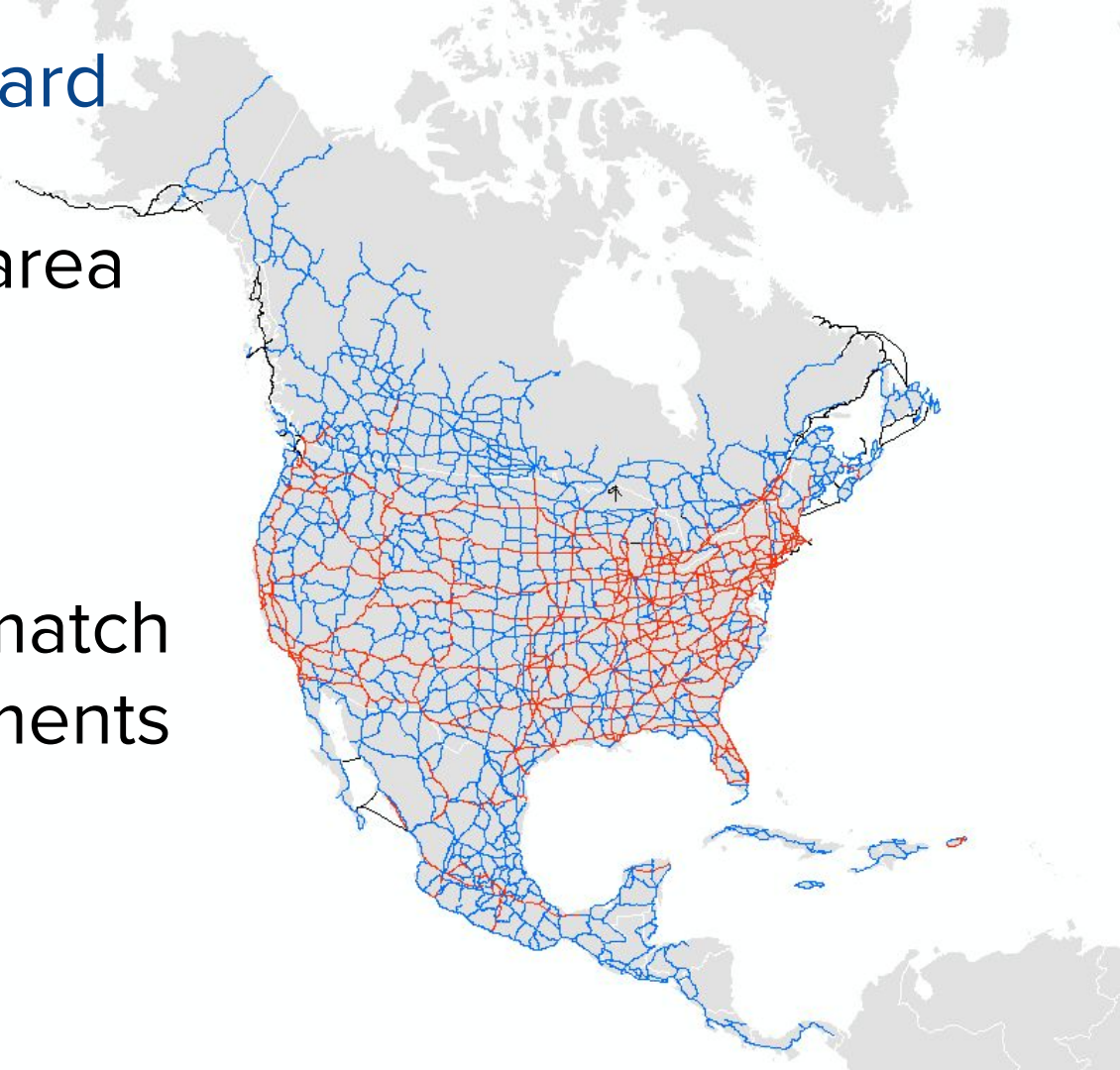
building	
house_num	154
predir	
qual	
pretype	
name	FORT
suftype	STREET
sufdir	SOUTH
ruralroute	
extra	
city	VICTORIA
state	BRITISH COLUMBIA
country	CANADA
postcode	
box	
unit	


Vancouver, BC
Vancouver, WA


2655 Main Street
Vancouver, BC
Congee Noc
Fable Diner
Tim Hortons
Main St
Watson St
Scotia St
1A
7
Kingsgate
E 10th Ave
2655 Main Street
Quebec St
Kingsway
Sophia St
E 11th Ave
Lucy's Eastside Diner
The Salvation
Army Thrift Store
E 12th Ave
E 13th Ave
49th Parallel
2655 Main Street
Vancouver, WA
Columbia St
Washington St
Broadway St
W 29th St
E 29th St
W 28th St
E 28th St
Dairy Queen
W 27th St
7-Eleven
Walgreens
E 4th Plain Blvd
W 25th St
E 25th St
Starbucks
Bleu Door Bakery
W 24th St
Main Street Marijuana
E 24th St
Vancouver Pizza

Geocoding data is hard

- Full coverage of area of interest
- Up to date and complete
- Standardized to match software requirements



Geocoding data is hard

City	Region	Nation	World
• Municipal parcel data • Census data • Open Street Map*	• State roads data • County parcel data • Census data • Open Street Map*	• Census data • Open Street Map*	• Open Street Map*

Geocoding is **hard,** so pay someone

Geocoding services

https://geopy.readthedocs.io/

```
CREATE EXTENSION postgis;
CREATE LANGUAGE plpythonu
```

```
CREATE OR REPLACE FUNCTION
  geocode(address text)
  RETURNS text AS
  $$
    from geopy.geocoders import Nominatim

    geolocator = Nominatim(user_agent='pgsql')
    loc = geolocator.geocode(address)
    wkt = 'SRID=4326;POINT(%g %g)' %
            ((loc.longitude, loc.latitude))
    return wkt
  $$
  LANGUAGE plpythonu VOLATILE
  COST 1000;
```

```
SELECT ST_AsGeoJson(geocode(
  '144 Simcoe Street, Victoria, BC'
  ));
```

```
{"type":"Point",
 "coordinates":[-123.382,48.4195]}
```

But online services...

Geocoding services

ArcGIS Online

- Costly!
  - $17/K for Google geocoding
  - No caching!
- Slow
  - Compared to a database
  - HTTP round trips
- General purpose
  - Don’t necessarily have your data in them

Global geocoding **self-serve** options

Global geocoding options

- Complete?
- Current?
- Consistent?
- Multilingual?

Global geocoding options

Software

Data

Pelias or Nominatum

OpenStreetMap

GeoNames

National geocoding **self-serve** options

crunchydata

National geocoding options

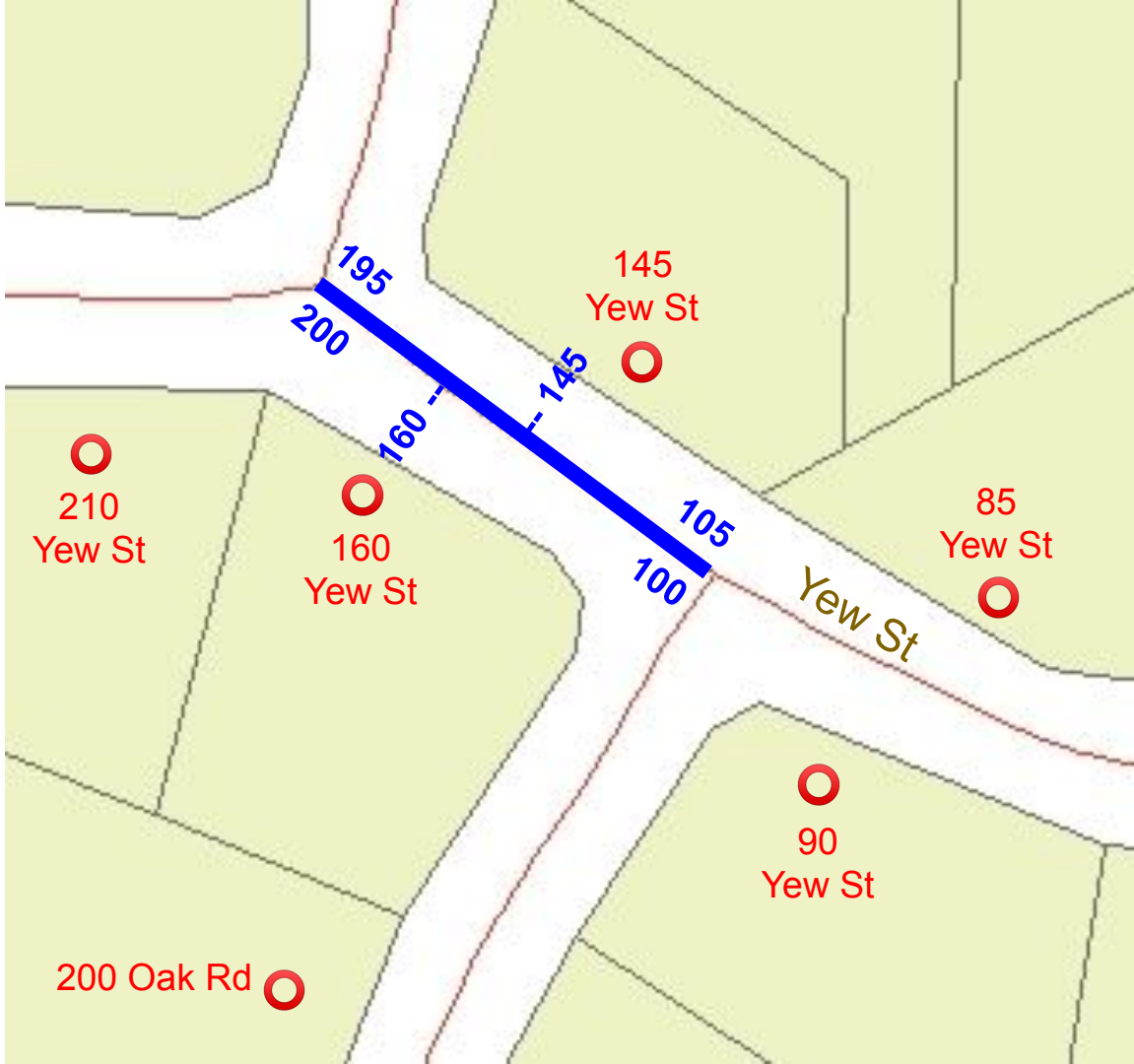
Software

Data

- Complete(ish)
- Current(ish)
- Consistent
- Accurate(ish)
- Interpolated
- Hierarchical

Interpolated centerline geocoding

- Less data required
  - Just roads
- Less accurate
  - Rural routes
- Can match non-existing addresses



PostGIS TIGER Geocoder

```
CREATE EXTENSION postgis;

-- Parse strings to address components
CREATE EXTENSION address_standardizer;
CREATE EXTENSION address_standardizer_data_us;

-- Match address components to TIGER database
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION postgis_tiger_geocoder;
```

address_standardizer

```
SELECT * FROM
parse_address(
     '154 Jackson St,
      San Jose, CA');
```

```
num      | 154
street   | Jackson St
street2  |
address1 | 154 Jackson St
city     | San Jose
state    | CA
zip      |
zipplus  |
country  | US
```

address_standardizer

```
SELECT * FROM
standardize_address(
   'us_lex',
   'us_gaz',
   'us_rules',
   '154 Jackson St,
    San Jose, CA'
);
```

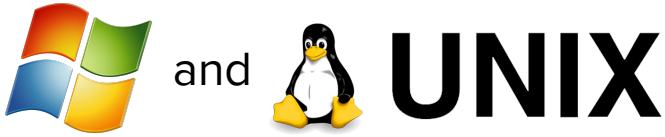
```
building    |
house_num   | 154
predir      |
qual        |
pretype     |
name        | JACKSON
suftype     | STREET
sufdir      |
ruralroute  |
extra       |
city        | SAN JOSE
state       | CALIFORNIA
country     | USA
postcode    |
box         |
unit        |
```

But the data?....

postgis_tiger_geocoder

```
-- Load national data files
SELECT
tiger.Loader_Generate_Nation_Script('crunchy')

-- Using national files as guide,
-- load requested state files
SELECT
tiger.Loader_Generate_Script(
  ARRAY['CA'],
  'crunchy')
```




```
TMPDIR="/data/gisdata/temp/"
UNZIPTOOL=unzip
WGETTOOL=wget
export PGBIN=/usr/bin
export PGPORT=
export PGHOST=
export PGUSER=ec2-user
export PGPASSWORD=ec2-user
export PGDATABASE=tiger2
export PGOPTIONS="-c search_path=public,tiger"
PSQL=${PGBIN}/psql
SHP2PGSQL=shp2pgsql
cd /data/gisdata

cd /data/gisdata
wget https://www2.census.gov/geo/tiger/TIGER2019/STATE/tl_2019_us_state.zip --mirror --reject=html
cd /data/gisdata/www2.census.gov/geo/tiger/TIGER2019/STATE
rm -f ${TMPDIR}/*.*
${PSQL} -c "DROP SCHEMA IF EXISTS tiger_staging CASCADE;"
${PSQL} -c "CREATE SCHEMA tiger_staging;"
for z in tl_*state.zip ; do $UNZIPTOOL -o -d $TMPDIR $z; done
cd $TMPDIR;

${PSQL} -c "CREATE TABLE tiger_data.state_all(CONSTRAINT pk_state_all PRIMARY KEY (statefp),CONSTRAINT
uidx_state_all_stusps  UNIQUE (stusps), CONSTRAINT uidx_state_all_gid UNIQUE (gid) ) INHERITS(tiger.state); "
${SHP2PGSQL} -D -c -s 4269 -g the_geom   -W "latin1" tl_2019_us_state.dbf tiger_staging.state | ${PSQL}
${PSQL} -c "SELECT loader_load_staged_data(lower('state'), lower('state_all')); "
      ${PSQL} -c "CREATE INDEX tiger_data_state_all_the_geom_gist ON tiger_data.state_all USING gist(the_geom);"
      ${PSQL} -c "VACUUM ANALYZE tiger_data.state_all"
```

postgis_tiger_geocoder

```
SELECT * FROM geocode(
  '800 E GREENWICH PL, PALO ALTO, CA', 1);
```

```
addy    | (800,E,Greenwich,Pl,,,"Palo Alto",CA,94303,t,,)
geomout | 0101000020AD100000DB6225ABAE885EC0C395EACB61B84240
rating  | 1
```

postgis_tiger_geocoder

```
SELECT (addy).*,
       ST_X(geomout) as lon,
       ST_Y(geomout) as lat,
       rating
FROM geocode(
  '800 E GREENWICH PL,
   PALO ALTO, CA',
  1);
```

```
address          | 800
predirabbrev     | E
streetname       | Greenwich
streettypeabbrev | Pl
postdirabbrev    |
internal         |
location         | Palo Alto
stateabbrev      | CA
zip              | 94303
parsed           | t
zip4             |
lon              | -122.135660
lat              |   37.440484
rating           | 1
```

postgis_tiger_geocoder

```
SELECT *
FROM geocode('800 E GREENWICH PL, PALO ALTO, CA', 1);
```

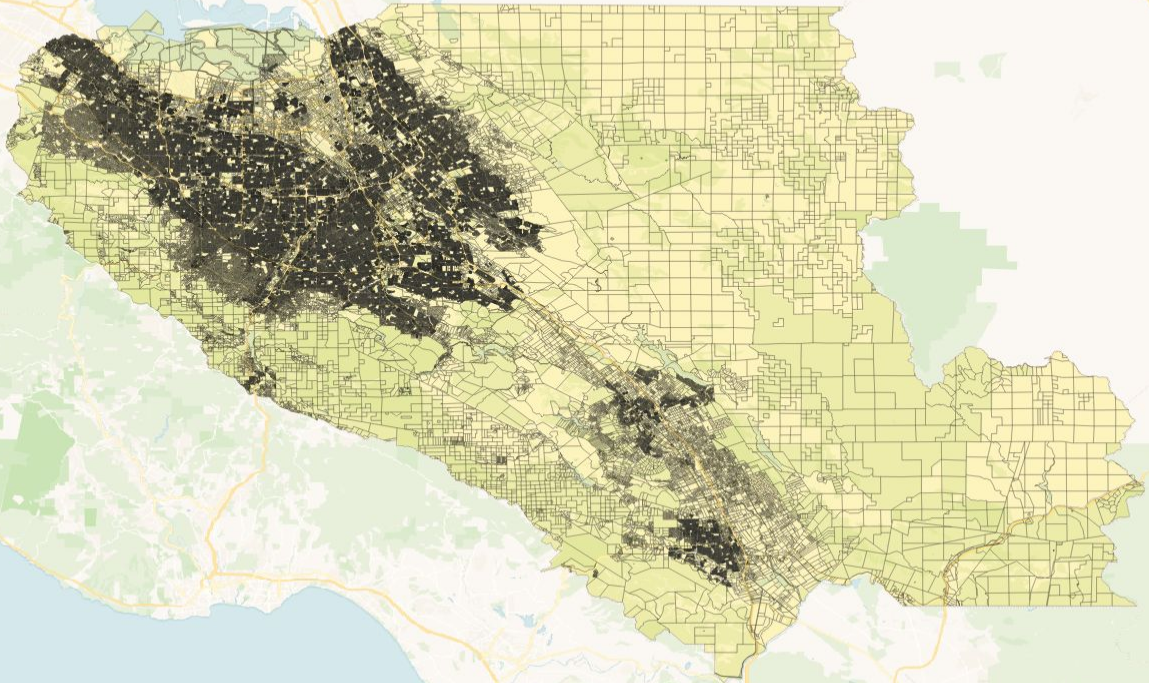
- Standardization: 10ms
- Geocoding: 100ms - 700ms

- St Clara Parcels: avg(160ms)

Regional geocoding **self-serve** options

What if I have my own data?

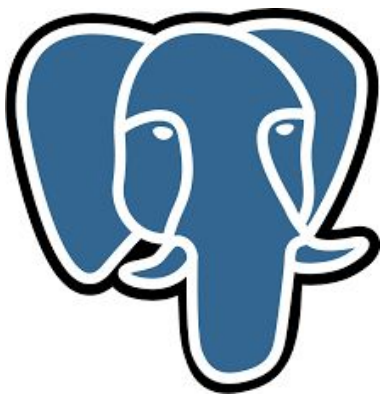
496,103 parcels in Santa Clara county



496,103 parcels in Santa Clara county

house	street_prefix	street_name	street_type	unit	city	state
158		TILLMAN	AV		SAN JOSE	CA
440		JUANITA	WY		LOS ALTOS	CA
1310		SADDLE RACK	ST	136	SAN JOSE	CA
689		GRAND FIR	AV		SUNNYVALE	CA
5668		CALMOR	AV	3	SAN JOSE	CA
23504		OAK VALLEY	RD		CUPERTINO	CA
1705		LEXINGTON	ST		SANTA CLARA	CA
1452		KEONCREST	AV		SAN JOSE	CA
2200		SOUTH	CT		PALO ALTO	CA
6141		YEADON	WY		SAN JOSE	CA
5542		ORA	ST		SAN JOSE	CA
766	W	HOMESTEAD	AV		SUNNYVALE	CA
1569		HEARTHSTONE	DR		SAN JOSE	CA

High speed address autocomplete?



BATTERIES INCLUDED

Add a full-text search column

```
ALTER TABLE parcels
  ADD COLUMN ts tsvector;

UPDATE parcels
  SET ts = to_tsvector('simple',
             concat_ws(' ',
               house,
               street_prefix, street_name, street_type,
               city, state))

CREATE INDEX parcels_ts_x ON parcels USING GIN (ts);
```

Try a full-text search: “800 Greenwich”

```
SELECT house, street_prefix, street_name,
       street_type, city
FROM parcels
WHERE ts @@ to_tsquery('simple',
                       '800 & GREENWICH')
```

```
 house | street_prefix | street_name | street_type |   city
-------+---------------+-------------+-------------+-----------
 800   | E             | GREENWICH   | PL          | PALO ALTO
```

6.773 ms

Try a partial full-text search: “800 G…”

```
SELECT house, street_prefix, street_name,
       street_type, city
FROM parcels
WHERE ts @@ to_tsquery('simple',
                       '800 & G:*')
```

```
 house | street_prefix | street_name | street_type |   city
-------+---------------+-------------+-------------+-----------
 800   |               | GREYSTONE   | CT          | GILROY
 800   | E             | GREENWICH   | PL          | PALO ALTO
```

10.9 ms

Try a partial, ranked search: “800 G…”

```
SELECT house, street_prefix, street_name,
       street_type, city, ts_rank_cd(ts, q) AS rank
FROM parcels,
     to_tsquery('simple', '800 & G:*') q
WHERE ts @@ q
ORDER BY rank DESC
```

```
 house | street_prefix | street_name | street_type |   city    | rank
-------+---------------+-------------+-------------+-----------+------
 800   |               | GLENSIDE    | DR          | SAN JOSE  |  0.1
 800   |               | GREYSTONE   | CT          | GILROY    |  0.1
 ...
```

31.6 ms

String to partial tsearch

Enter Address: 1094 West Ama

```
'1096 & West & Ama:*'
```

- Trim white-space from front and back
- Split tokens on any white-space
- Rejoin using the ‘&’ character
- Stick global match ‘:*’ on the last token, if there is no space before the cursor

String to partial tsearch

```
SELECT to_tsquery_partial('34 Main');
```

```
'34' & 'main':*
```

```
SELECT to_tsquery_partial('34 Main St ');
```

```
'34' & 'main' & 'st'
```

```
SELECT to_tsquery_partial(' 34   Main St ');
```

```
'34' & 'main' & 'st'
```

String to partial tsearch

```
CREATE OR REPLACE FUNCTION to_tsquery_partial(text)
  RETURNS tsquery AS $$
    SELECT to_tsquery('simple',
              array_to_string(
                regexp_split_to_array(
                  trim($1),E'\\s+'),' & ')
              || CASE WHEN $1 ~ ' $' THEN '' ELSE ':*' END
            )
  $$ LANGUAGE 'sql';
```

String to autocomplete query

```
SELECT
  initcap(concat_ws(' ',
    house, street_name, street_type, city)) AS value,
  geom AS geom,
  ts_rank_cd(ts, query) AS rank
FROM parcels,
     to_tsquery_partial('{query}') AS query
WHERE ts @@ query
ORDER BY rank DESC
LIMIT 10;
```

What about abbreviations?

Try a full-text search: “800 E G…”

```
SELECT house, street_prefix, street_name,
       street_type, city
FROM parcels
WHERE ts @@ to_tsquery('simple',
                       '800 & E & G:*')
```

```
 house | street_prefix | street_name | street_type |   city
-------+---------------+-------------+-------------+-----------
 800   | E             | GREENWICH   | PL          | PALO ALTO
```

36.7 ms

Try a full-text search: “800 East G…”

```
SELECT house, street_prefix, street_name,
       street_type, city
FROM parcels
WHERE ts @@ to_tsquery('simple',
                       '800 & EAST & G:*')
```

```
(0 rows)
```

36.7 ms

tsearch dictionaries!

- Allows language-specific “stemming”
  - oaks ➜ oak
  - ran ➜ run
  - rammed ➜ ram
- Allows language-specific synomyms
  - boat ➜ ship
  - nova ➜ star

tsearch synonyms!

- Full-text search .syn file!
  - Map term to synonym, a thesaurus!
- **Abbreviations are synonyms** to fully spelled out address terms

ave avenue
bl boulevard
blvd boulevard
brg bridge
byp bypass
cir circle
circ circle
clf cliff

cmtry cemetery
cnl canal
cp camp
cres crescent
crsn crescent
crk creek
crs course
crst crescent

crt court
ct court
cts courts
cyn canyon
dr drive
espl esplanade
ests estates

expy
expressway
fld field
fls falls
frm farm
fwy freeway
gdns gardens
grn green

hbr harbor
hl hill
hts heights
hwy highway
lk lake
lkout lookout
ln lane
lp loop

TSearch dictionaries for addresses

25 commits | 1 branch | 0 releases | 2 contributors | MIT

Branch: master | New pull request | Find file | Clone or download

pramsey Merge branch 'master' of github.com:pramsey/pgsql-addressing-dictionary — Latest commit 204dcd5 6 days ago

.gitignore	First commit	5 years ago
LICENSE.md	First commit	5 years ago
META.json	bump version to 1.1	5 years ago
Makefile	also remove from Makefile	5 years ago
README.md	Merge branch 'master' of github.com:pramsey/pgsql-addressing-dictionary	6 days ago
addressing_dictionary--1.0--1.1.sql	fix to have an upgrade path from 1.0 to 1.1	5 years ago
addressing_dictionary--1.1.sql	fix to have an upgrade path from 1.0 to 1.1	5 years ago
addressing_dictionary.control	bump version to 1.1	5 years ago

tsearch synonyms!

- https://github.com/pramsey/pgsql-addressing-dictionary

```
CREATE EXTENSION addressing_dictionary;
UPDATE parcels SET ts = to_tsvector('addressing_en',
              concat_ws(' ',
                house,
                street_prefix,
                street_name, street_type,
                city, state))
```

tsearch synonyms!

- https://github.com/pramsey/pgsql-addressing-dictionary

```
SELECT to_tsvector('addressing_en',
                   '1234 n main st');
```

```
'1234':1 'main':4 'n':2 'north':3 'street':5
```

Try a full-text search: “800 East G…”

```
SELECT house, street_prefix, street_name,
       street_type, city
FROM parcels
WHERE ts @@ to_tsquery('addressing_en',
                       '800 & EAST & G:*')
```

```
 house | street_prefix | street_name | street_type |   city
-------+---------------+-------------+-------------+-----------
 800   | E             | GREENWICH   | PL          | PALO ALTO
```

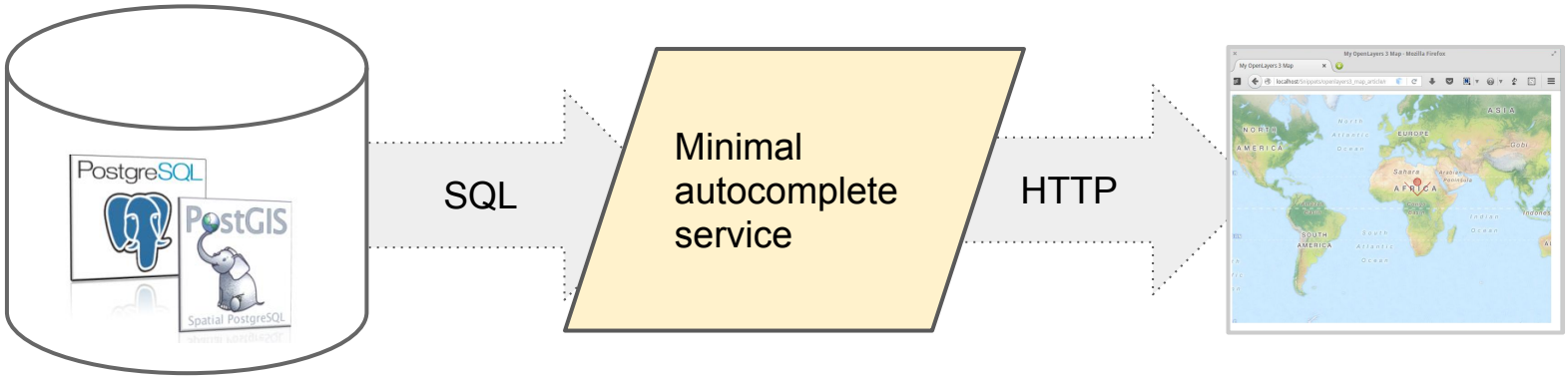
31.5 ms

https://github.com/pramsey/examples-pgsql-full-text

Show me a DEMO!

crunchy data

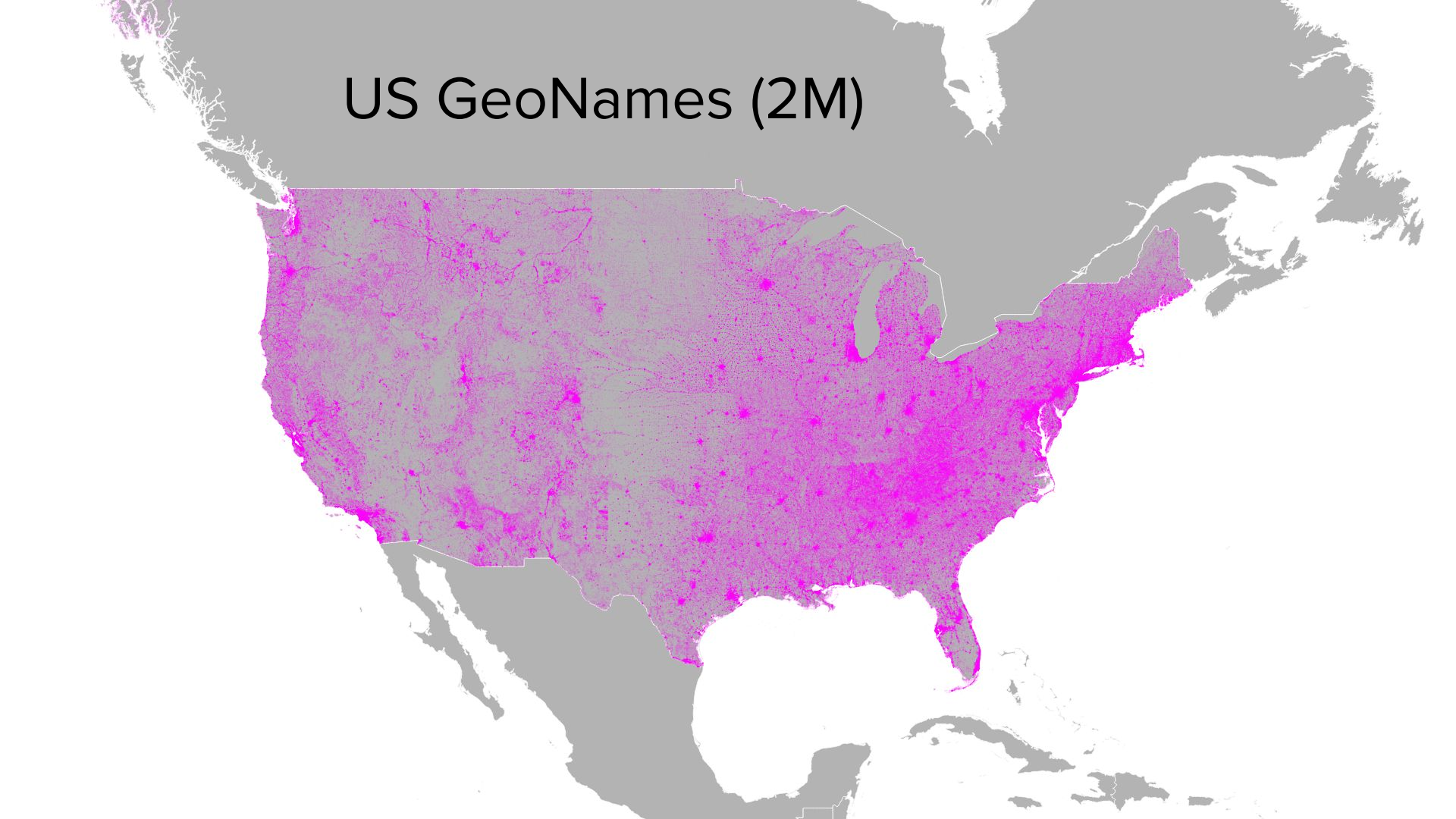
No, just add HTTP!



- (SQL)
  - Execute SQL
  - Return JSON

- Req/Resp (HTTP)
- Read query
- Build SQL

Not just for **addresses**!


US GeoNames (2M)

All the names that include the word {} ...

```
WITH names AS (
     SELECT
        name, kind,
        ST_X(geom) AS x,
        ST_Y(geom) AS y
     FROM geonames
     WHERE ts @@ plainto_tsquery('english', '{}')
)
SELECT json_agg(row_to_json(names.*))::text AS json
FROM names
```

https://github.com/pramsey/examples-pgsql-full-text

Show me a DEMO!

crunchy data

Questions?

paul.ramsey@crunchydata.ca

https://is.gd/xp1wpB