Spatial Database Design and Analysis

with data from the EPA's Facility Registry System and National Priority List

Sarah Jaraha

Introduction

Proper database design allows data to be accessed without common database errors, and allows for faster querying. By the end of the tutorial, users will have designed, optimized, and queried a spatial relational database using PostgreSQL (with PostGIS extension) and QGIS.

Description of Data

Source and Purpose

The dataset used in this tutorial is sourced from the Facility Registry System (FRS), a database used by the Environmental Protection Agency's (EPA) Office of Information Collection. The FRS stores information about facilities that are "subject to environmental regulations or of environmental interest" (1).

Two main pieces of information stored about facilities in the FRS are the facility's North American Industry Classification System (NAICS) code and the environmental interests of the facility. The NAICS code is a two- to six- digit code indicating the "type of industrial activity occurring at the facility site" (1). Environmental interests include "environmental permits and regulatory programs that apply to the facility site" (1).

A separate EPA dataset provides descriptions for the NAICS codes, at the two-digit level. This dataset will be referred to as the NAICS description dataset.

Instructions on how to import the data can be found in Appendix 1.

Spatial Questions

A spatial question incorporates location and attributes. The following question is non-spatial:

"Which facilities are in the manufacturing economic sector?"

In it's spatial form, the question would read:

"What are the locations of all facilities in the manufacturing economic sector?"

General spatial questions that can be answered using the FRS dataset are listed below:

- What are the locations of facilities with a certain set of environmental and/or industrial attributes?
- What is the distance(s) between facilities with a certain set of environmental and/or industrial attributes?
- Which facilities, with a certain set of environmental and/or industrial attributes, are located within a specific proximity of another facility?

Complimentary Data

The EPA's National Priorities List (NPL) is a list of superfund sites across the nation. Data from the EPA's NPL allows for interesting spatial questions when paired with the FRS data in the spatial queries.

Data Structure and Processing

Structure of FRS Data

Data from the FRS can be downloaded as a zip file, containing nine CSV files. Two of the nine CSVs will be used in this tutorial: NAICS and Facility. Table 1 describes the datasets as well as the fields in each that will be used in the tutorial.

Table 1: Selected Datasets from FRS

Dataset Name	Description (1)	Selected Fields
Facility	basic identification information for a facility site	registry_id, primary_name, state_code, city_name, longitude83, latitude83
NAICS	type of industrial activity occurring at a facility site	registry_id, naics_code, interest_type, code_description

The registry_id field is the unique identifier in the facility table. Longitude83 and latitude83 reference the NAD83 spatial reference system (SRID: 4269).

The NAICS dataset contains NIACS codes as well as environmental interests for facilities.

Normalization Process for FRS Data

Normalization structures a database according to a specific set of rules to avoid common database errors. It also allows for efficient querying. Once the datasets mentioned above have been imported to PostgreSQL, their fields can be strategically organized into relations to create a normalized relational database. The normalization scripts for this tutorial can be found in Appendix 2.

The entity relationship diagram (ERD) in figure 1 shows the normalized database using data from the FRS datasets: Facility and NAICS. It also incorporates data from the NPL description dataset. For the remainder of the report, this database will be referred to as the FRSS database (facility registry system select).

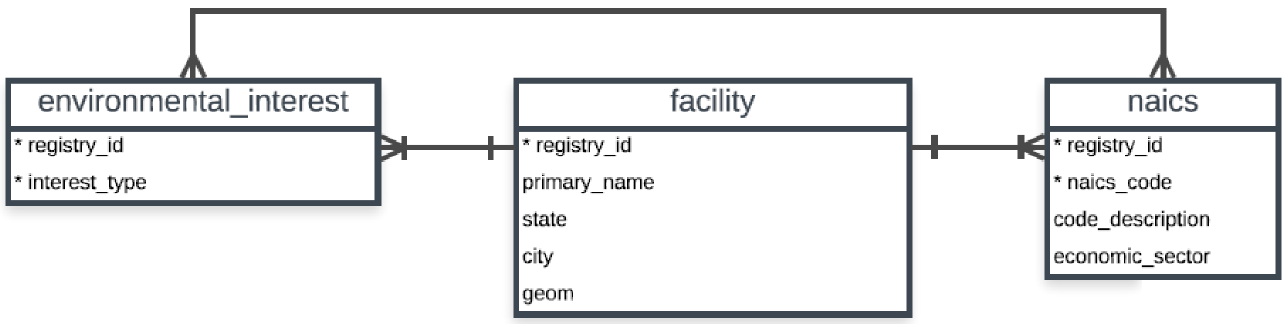

Figure 1: Normalized ERD using FRS data (FRSS database)

Data from the NPL dataset was used to create the table "npl_final_site", shown in figure 2.

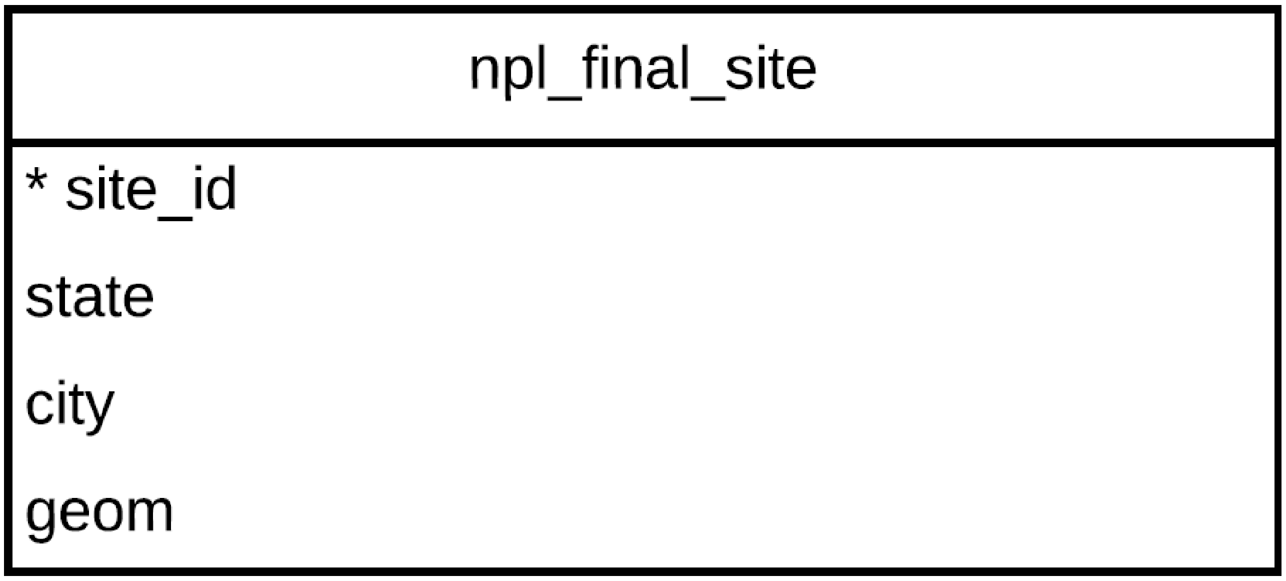

Figure 2: npl_final_site relation from the NPL dataset

A data dictionary for the FRSS database and npl_final_site is provided in Appendix 4. The following section describes each of the relations shown in figure 1 and figure 2.

- **facility**
  - Primary Key: registry_id
  - Data Source: Facility dataset (FRS)
  - Description: Each entry in the facility relation relates to an individual facility that is registered with the FRS. The *geom* field contains the location data for each facility.

- **naics**
    - Primary Key (Composite): registry_id, naics_code
    - Data Source: NAICS dataset (FRS), NAICS definitions dataset
    - Description: A facility can have more than one naics code. Each entry in the naics relation relates to a distinct instance of a facility and its naics code. The *registry_id, naics_code, and code_description* fields were populated using the NAICS dataset. The *economic_sector* field was populated using the NAICS definitions dataset.
- **environmental_interest**
    - Primary Key (Composite): registry_id, interest_type
    - Data Source: NAICS dataset (FRS)
    - Description: The *interest_type* field contains the environmental permit or regulatory program that applies to facility. A facility can have more than one interest_type. Each entry in the environmental_interest relation relates to a distinct instance of a facility and its interest_type.
- **npl_final_site**
    - Primary Key: site_id
    - Data Source: NPL dataset
    - Description: Each entry in the npl_final_site relation relates to a specific site on the NPL list. The *geom* field contains location data for each site.

Database Optimization

Normalization alone does not always produce the most efficient querying. Several methods can be used to optimize a database and make queries run faster. Denormalization and indexing are the two optimization methods that were used to optimize the FRSS database. Optimization scripts can be found in Appendix 3.

- **Indexing**
    - Definition: A data structure that allows a database management system to quickly search the data to which the index is applied, similar to the index of a book. Indexes can reduce query speeds significantly when used properly.
    - Use in FRSS Database: A spatial index was applied to the geom field in the naics_denorm relation. A spatial index was not applied to the geom field in the npl_final_site relation because the npl_final_site relation is relatively small in size (1338 rows).
- **Denormalization**
    - Definition: Removing a database from its normalized form in the interest of efficient querying. Denormalization is appropriate when fields that are frequently used together exist in separate relations. Without denormalization, joins would be used more frequently.
    - Use in FRSS Database: The naics relation was denormalized to include the primary_name, state, and geom fields from the facility relation. The denormalized naics table is called naics_denorm.

The denormalized ERD with naics_denorm is shown in figure 3:

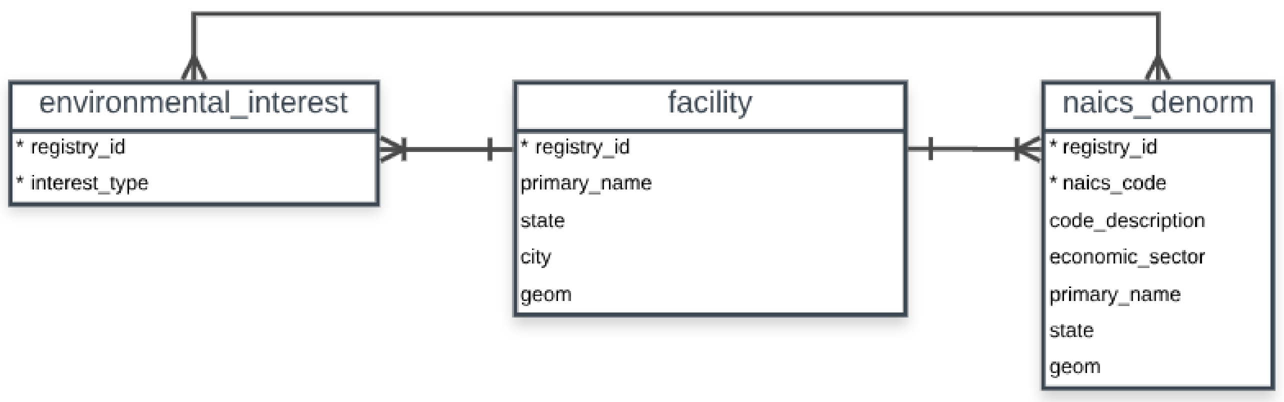

Figure 3: Optimized FRSS database using the denormalized naic_denorm table

Spatial Queries

Now that the FRS database has been normalized and optimized, it can used to answer spatial questions. The spatial questions that will be answered in this tutorial are listed below:

1. What is the distance from each FRS apple orchard in California to its nearest casino?
2. Which economic sectors, and how many FRS facilities of each economic sector, exist within one mile of a specific NPL site?
3. What is the economic sector of the closest FRS facility to each site on the NPL?

The spatial queries used to answer these questions are shown below, follow by a detailed description of each query:

Query 1: *What is the distance from each FRS apple orchard in California to its nearest casino?*

```
SELECT n1.primary_name as apple_orchard,
    (SELECT n2.primary_name
    FROM naics_denorm as n2
    WHERE code_description ilike '%casino%'
    AND state = 'CA'
    ORDER BY n1.geom <-> n2.geom
    LIMIT 1) as casino,
        (SELECT (ST_Transform(n1.geom, 26910)
                 <->
                 ST_Transform(n2.geom, 26910))/1000 as distance_km
        FROM naics_denorm as n2
        WHERE code_description ilike '%casino%'
        AND state = 'CA'
        ORDER BY n1.geom <-> n2.geom
        LIMIT 1) as distance
FROM naics_denorm as n1
WHERE code_description = 'APPLE ORCHARDS.'
AND state = 'CA';
```

Query 1 is composed of one select statement with two subqueries:

1. **apple orchard select statement:** select the primary_name of each FRS apple orchard in California

2. **casino subquery:** select the primary_name of the closest casino to each FRS apple orchard in California
3. **distance subquery:** select the distance from each FRS apple orchard in California to its closest casino

Let's take a closer look at each step:

1. *Select the primary_name of each FRS apple orchard in California*

```
SELECT n1.primary_name as apple_orchard

FROM naics_denorm as n1
WHERE code_description = 'APPLE ORCHARDS.'
AND state = 'CA';
```

The selection is made from the denormalized naics table. A WHERE clause narrows the selection to include only rows where code_description is 'APPLE ORCHARDS.' and state is 'CA'.

2. *Select the primary_name of the closest casino to each FRS apple orchard in California*

```
(SELECT n2.primary_name
FROM naics_denorm as n2
WHERE code_description ilike '%casino%'
AND state = 'CA'
ORDER BY n1.geom <-> n2.geom
LIMIT 1) as casino,
```

There are two main parts to the casino subquery:

- **WHERE clause:** narrows the selection to California casinos
- **KNN operator with ORDER BY:** finds the closest casino to each apple orchard selected in the apple orchard select statement

KNN stands for k-nearest-neighbors. There are two types of KNN operators in PostgreSQL: the KNN bounding-box distance operator (<#>) and the KNN bounding-box centroid distance operator (<->). The latter is used in the fifth line of the casino subquery.

On its own, the KNN operator finds the distance from *each* apple orchard to *each* casino in California. To find the *closest* casino, several procedural steps must be taken:

1. Use the KNN operator to find the distance from each apple orchard to each casino in California
2. Use an ORDER BY clause to order the distances in ascending order
3. LIMIT the results to one output so only the shortest distance will be returned from the ordered list of distances

Apple orchard and casino data come from the same column and in the same table. So, aliases are used to distinguish geometries that represent apple orchards from geometries that represent casinos. In the *apple orchard select statement*, naics_denorm is aliased as n1. So, n1.geom represents the apple orchards that are selected in the apple orchard select statement. In the *casino subquery*, naics_denorm is aliased as n2. So, n2.geom represents casinos that are selected in the casino subquery. Aliasing allows geometries from the same table to be used with a KNN operator.

3. *Select the distance from each FRS apple orchard in California to the closest casino*

```
(SELECT (ST_Transform(n1.geom, 26910)
         <->
         ST_Transform(n2.geom, 26910))/1000 as distance_km
FROM naics_denorm as n2
WHERE code_description ilike '%casino%'
AND state = 'CA'
ORDER BY n1.geom <-> n2.geom
LIMIT 1) as distance
```

There is one difference between the distance subquery and the casino subquery: the returned field. The casino subquery returns the *primary_name* of the casino that is closest to each apple orchard in California. The distance subquery returns the *distance* from the closest casino to apple orchard in kilometers, shown in the excerpt below.

To return the correct distance, attention must be paid to the spatial reference system. The longitude83 and latitude83 fields in the FRS reference the NAD83 spatial refence system, which uses units of decimal degrees. Since the geometries represent longitude latitude pairs, the geometries are also in units of decimal degrees. Decimal degrees are meaningless when measuring distances, so the geometries must be projected to a spatial reference system with meaningful units. The function ST_Transform() makes this possible.

When given a geometry and an SRID, ST_Transform() creates a new geometry with the spatial refence system of the specified SRID. Geometries in this tutorial are transformed to UTM zone 10N (SRID: 26910), which is the UTM zone for California. UTM uses meters as a unit of measurement. Once the distance is calculated in meters, it is divided by 1000 to convert units to kilometers.

The result of Query 1 is shown in table 2:

Table 2: Results of Query 1

FRS Apple Orchard	Closest Casino	Distance (km)
Manzana Products	Sonoma County Airport	9
Kingsburg Apple	Table Mountain Rancheria DBA Table Mountain Casino	57

A mere 9 km separates Manzana Products and its closest casino. Kingsburg Apple is 57 km from its closest casino. Figure 4 shows a map of the FRS apple orchards with their closest casinos.

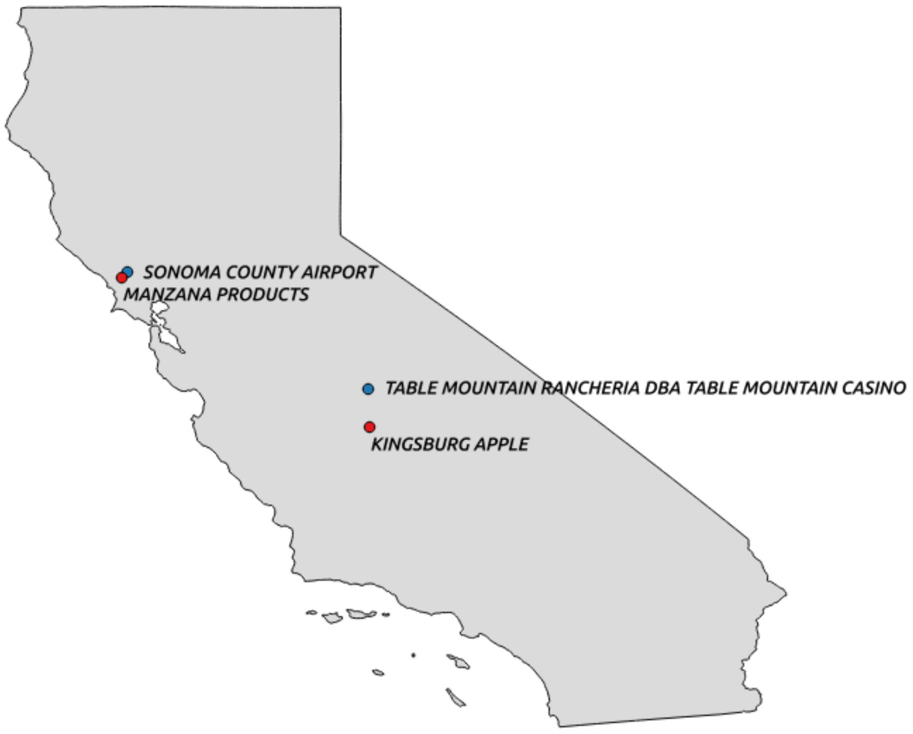

Figure 4: Result for Query 1 Visualized in QGIS

Query 2: *Which economic sectors, and how many FRS facilities of each economic sector, exist within one mile (1610 meters) of a specific NPL site?*

```
SELECT economic_sector, COUNT(economic_sector)
FROM naics_denorm
WHERE state = 'CA'
AND ST_DWithin(
        ST_Transform(geom,26910),
            (SELECT ST_Transform(geom,26910)
            FROM npl_final_site
            WHERE site_id = '0901389'),1610)
GROUP BY economic_sector
ORDER BY 2 DESC;
```

Query 2 operates as follows, beginning from the innermost subquery, and moving outward:

1. Select the geometry of the npl site of interest
2. Return only facilities within a specified distance of the npl site
3. Return the economic sector of the facilities, and the number of facilities in each economic sector that fall within a specified distance of the npl site

Let's take a closer look at each step:

1. *Select the geometry of the NPL site of interest*

```
(SELECT ST_Transform(geom,26910)
FROM npl_final_site
WHERE site_id = '0901389')
```

A WHERE clause is used to select the npl site of interest using the site_id field. The select statement returns the corresponding geometry, transformed to the appropriate SRID.

2. *Return only facilities within a specified distance of the NPL site*

```
ST_DWithin(
        ST_Transform(geom,26910),
            (SELECT ST_Transform(geom,26910)
            FROM npl_final_site
            WHERE site_id = '0901389')
,1610)
```

When given two geometries and a distance, the function ST_DWithin() returns true if the given geometries are within the specified distance, and false if they are not. In Query 2, every facility in the naics_denorm table is compared to the NPL site of interest. If the facility is within one mile (about 1610 meters) of the NPL site of interest, ST_DWithin() returns true.

Like Query 1, geometries must be transformed to the correct spatial reference system with ST_Transform() before they can be used properly with ST_DWithin(). Since the NPL site of interest is in California, geometries are transformed to UTM zone 10N (SRID: 26910).

3. *Return the economic sector of the facilities and the number of facilities in each economic sector that fall within a specified distance of the npl site*

```
SELECT economic_sector, COUNT(economic_sector)
FROM naics_denorm
WHERE state = 'CA'

GROUP BY economic_sector
ORDER BY 2 DESC
```

Once the desired facilities have been returned by the ST_DWithin() function, a GROUP BY clause is used to separate the facilities by economic_sector. The select statement returns the economic sector and the number of facilities in each economic sector that are within one mile of the npl site of interest.

The result of Query 2 shows that, of the 187 FRS facilities within one mile of npl site with site_id '0901389', 125 facilities are in the manufacturing economic sector, followed by 14 in the professional, scientific, and technical services economic sector.

Query 3: *What is the economic_sector of the closest FRS facility to each site on the NPL?*

```
SELECT site_id,
    (SELECT n.economic_sector
    FROM naics_denorm as n
    ORDER BY n.geom <-> npl.geom
    LIMIT 1 ) as economic_sector_closest_facility
FROM npl_final_site as npl;
```

In Query 3, a KNN operator is used between two different tables: naics_denorm and npl_final_site. It operates as follows:

1. Select the economic sector of the closest facility to each NPL site
2. Return the site_id and the corresponding closest economic_sector

Let's take a closer look at each step:

1. *Select the economic sector of the closest facility to each NPL site*

```
(SELECT n.economic_sector
FROM niacs_denorm as n
ORDER BY n.geom <-> npl.geom
LIMIT 1 ) as economic_sector_closest_facility
```

Here, a KNN operator compares the geometry column in npl_final_site with the geometry column in naics_denorm. It is important to make sure that both geometry columns have the same spatial reference system. Since distance is not being measured, the geometries do not need to be transformed with ST_Transform(). The economic sector corresponding to each npl site's closest facility is returned.

2. *Return the site_id and the corresponding closest economic_sector*

```
SELECT site_id,
    (SELECT n.economic_sector
    FROM niacs_denorm as n
    ORDER BY n.geom <-> npl.geom
    LIMIT 1 ) as economic_sector_of_closest_facility
FROM npl_final_site as npl
```

Finlaly, the site_id is returned from npl_final_site along with the economic sector of its closes facility from naics_denorm.

Summary of Main Functions and Operators Used

Function	Description
ST_Transform(geometry, integer)	Returns a new geometry with its coordinates transformed to a different spatial reference system (2).
ST_DWithin(geometry, geometry, integer)	Returns true if the geometries are within the specified distance of one another (3).
KNN Operator (geometry<-> geometry)	Returns the 2D distance between two geometries (4).

Appendix 1: Obtaining and Loading Data

FRS data

1. Navigate to the FRS Data Resources page on the EPA's website:

   https://www.epa.gov/frs/frs-data-resources
2. Find the "Prepackaged Download" called "FRS Combined CSV Download (National)" and select the hyperlink.
3. Download the 723 MB National Zip file.
4. Import the Facility and NAICS datasets to PostgreSQL with the following names, making sure to convert all field names to lowercase:
   - Facility dataset: **frs_facility**
   - NAICS dataset: **frs_naics**

NAICS description data

1. Navigate to the North American Industry Classification System page on the US Census Bureau's website:

   https://www.census.gov/eos/www/naics/downloadables/downloadables.html
2. Download the xls file called "2-6 digit 2012 Code File"
3. Import the xls file to PostgreSQL as **naics_description_2012** and launder the column names in pgAdmin as follows:
   - "2012 NAICS US Code" >>> **naics_code_2012**
   - "2012 NAICS US Title" >>> **naics_title_2012**

NPL data

1. Navigate to the Superfund Data and Reports page on the EPA's website
2. Find the document called "All current Final NPL Sites (FOIA 4)".This is a PDF file that contains an xlsx file. Open the PDF in Adobe Acrobat Reader.
3. Download the attached xlsx file and open it in Excel.
4. Select cells (A5:P1344). Copy the selection and paste it into a new workbook. Save the new workbook as a CSV.
5. Import the CSV to PostgreSQL as **npl_final**, making sure to convert all field names to lowercase.
6. Launder the column names in pgAdmin as follows:
   - "site id" >>> **site_id**

FRS Environmental Interest Types PDF

The FRS Environmental Interest Types PDF provides definitions for the environmental interests.

1. Navigate to the Environmental Interest Types and Program Categories page on the EPA's website: https://www.epa.gov/enviro/environmental-interest-types-and-program-categories
2. Download the PDF file called "Environmental Interest Types and Program Categories"

Appendix 2: Normalization Scripts

```
--create function isnumeric()

CREATE OR REPLACE FUNCTION isnumeric(text) RETURNS BOOLEAN AS $$
DECLARE x NUMERIC;
BEGIN
    x = $1::NUMERIC;
    RETURN TRUE;
EXCEPTION WHEN others THEN
    RETURN FALSE;
END;
$$
STRICT
LANGUAGE plpgsql IMMUTABLE;
-----------------------------------------------------
--CREATE TABLE (FRS_FACILITY_ISNUMERIC)
--FROM FRS_FACILITY
-----------------------------------------------------

--create table: frs_facility_isnumeric
CREATE TABLE frs_facility_isnumeric AS
(SELECT * FROM frs_facility) with no data ;

--insert values: frs_facility_isnumeric
INSERT INTO frs_facility_isnumeric
(SELECT * FROM frs_facility
WHERE isnumeric(longitude83)
AND isnumeric(latitude83));

-----------------------------------------------------
--CREATE TABLE (FACILITY)
--FROM (FRS_FACILITY_ISNUMERIC)
-----------------------------------------------------

--create table: facility
CREATE TABLE facility
(registry_id character varying PRIMARY KEY,
primary_name character varying,
state character varying,
city character varying,
geom geometry(point,4269));

--insert values: facility
INSERT INTO facility
(SELECT registry_id, primary_name, state_code, city_name,
ST_SetSRID(ST_MakePoint(longitude83::double precision, latitude83::double
precision),4269)
FROM frs_facility_isnumeric
ORDER BY registry_id);

-----------------------------------------------------
--CREATE TABLE (NAICS)
```

```
--FROM (FRS_NAICS) + (NAICS_DESCRIPTION_2012)
-----------------------------------------------------

--create table: naics
CREATE TABLE naics AS
(SELECT DISTINCT registry_id, naics_code, code_description
FROM frs_naics
WHERE naics_code IS NOT NULL);

--add primary key
ALTER TABLE naics
ADD PRIMARY KEY(registry_id, naics_code);

--add column: economic_sector
ALTER TABLE naics
ADD COLUMN economic_sector varchar;

--prepare to update column: economic_sector
--account for range entries in naics_description_2012 (31-33, 44-45, 48-49)
INSERT INTO naics_description_2012 (naics_title_2012, naics_code_2012)
VALUES
('Manufacturing', '31'),
('Manufacturing', '32'),
('Manufacturing', '33'),
('Retail Trade', '44'),
('Retail Trade', '45'),
('Transportation and Warehousing', '48'),
('Transportation and Warehousing', '49');

--update economic_sector
UPDATE naics
SET economic_sector = naics_title_2012
FROM
    (SELECT naics_title_2012, naics_code_2012
    FROM naics_description_2012
    WHERE length(naics_code_2012) = 2) as sub
WHERE naics.naics_code ilike sub.naics_code_2012||'%';

-----------------------------------------------------
--CREATE TABLE (ENVIRONMENTAL_INTEREST)
--FROM (FRS_NAICS)
-----------------------------------------------------

--create table: environmental_interest
CREATE TABLE environmental_interest
(registry_id varchar,
interest_type varchar,
PRIMARY KEY(registry_id, interest_type));

--insert values: environmental_interest
INSERT INTO environmental_interest
SELECT DISTINCT registry_id, interest_type
FROM frs_naics
```

```
ORDER BY registry_id;

-----------------------------------------------------
--CREATE TABLE (NPL_FINAL_SITE)
--FROM (NPL_FINAL)
-----------------------------------------------------

--create table: npl_final_site
CREATE TABLE npl_final_site
(site_id varchar PRIMARY KEY,
state varchar,
city varchar,
geom geometry(point,4269));

--insert values: npl_final_site
INSERT INTO npl_final_site(site_id, state, city)
(SELECT DISTINCT  (TRIM(both chr(8236) from (TRIM(both chr(32) from TRIM(both chr(8237)
from site_id)))), st, city
FROM npl_final
WHERE site_id != '');

UPDATE npl_final_site
SET geom = point
FROM
    (SELECT
         (TRIM(both chr(8236) from (TRIM(both chr(32) from TRIM(both chr(8237) from
site_id)))) as sid,
        ST_SetSRID(ST_MakePoint(
        (TRIM(both chr(8236) from (TRIM(both chr(32) from TRIM(both chr(8237) from
longitude)))))::double precision,
        (TRIM(both chr(8236) from (TRIM(both chr(32) from TRIM(both chr(8237) from
latitude)))))::double precision),
        4269) as point
    FROM npl_final
    WHERE longitude != '') as sub
WHERE npl_final_site.site_id=sub.sid;
```

Appendix 3: Optimization Scripts

```
-----------------------------------------------------
--CREATE TABLE (NAICS_DENORM)
--FROM (NAICS and FACILITY)
-----------------------------------------------------

--create table: naics_denorm
CREATE TABLE naics_denorm AS
(SELECT * FROM naics);

--add primary key
ALTER TABLE naics_denorm
```

```
ADD PRIMARY KEY (registry_id, naics_code);

--add columns from facility
ALTER TABLE naics_denorm
ADD COLUMN primary_name varchar,
ADD COLUMN state varchar,
ADD COLUMN geom geometry(point,4269);

--update primary_name, state, and geom fields
UPDATE naics_denorm
SET primary_name = sub.primary_name, state = sub.state, geom = sub.geom
FROM
    (SELECT primary_name, registry_id, state, geom
    FROM facility) as sub
WHERE sub.registry_id = naics_denorm.registry_id;

-----------------------------------------------------
--CREATE INDEX
--ON NAICS_DENORM(geom)
-----------------------------------------------------

CREATE INDEX
ON naics_denorm USING gist(geom);
```

Appendix 4: Data Dictionary

facility

Field	Data Type	Description
registry_id	varchar	unique identifier assigned to each facility in the FRS
primary_name	varchar	commercial name of the facility
state	varchar	state where the facility is located
city	varchar	city where the facility is located
geom	point geometry	location of the facility

naics

Field	Data Type	Description
registry_id	varchar	unique identifier assigned to each facility in the FRS
naics_code	varchar	two- to six- digit code indicating the type of industrial activity occurring at an facility
code_description	varchar	a description that corresponds to a specific NAICS code
economic_sector	varchar	the economic sector of a facility (manufacturing, utilities); represented by the first two digits of the NAICS code

environmental_interest

Field	Data Type	Description
registry_id	varchar	unique identifier assigned to each facility in the FRS
interest_type	varchar	the environmental permit or regulatory program that applies to a facility

npl_final_site

Field	Data Type	Description
site_id	varchar	unique identifier assigned to each site on the NPL
state	varchar	state where an npl site is located
city	varchar	city where an npl site is located
geom	point geometry	location of the npl site

Further Exploration

Indexing: http://revenant.ca/www/postgis/workshop/indexing.html

Denormalization: https://www.vertabelo.com/blog/technical-articles/denormalization-when-why-and-how

KNN: https://postgis.net/docs/geometry_distance_knn.html

ST_Transform(): https://postgis.net/docs/ST_Transform.html

ST_DWithin(): https://postgis.net/docs/ST_DWithin.html

Citations

1. “Facility Registry System State CSV Download File Descriptions.” Environmental Protection Agency, 13 Nov. 2012.
2. “ST_Transform.” - *Spatial and Geographic Objects for PostgreSQL*, postgis.net/docs/ST_Transform.html.
3. “ST_DWithin.” *Spatial and Geographic Objects for PostgreSQL*, postgis.net/docs/ST_DWithin.html.
4. *"<->" Spatial and Geographic Objects for PostgreSQL*, postgis.net/docs/geometry_distance_knn.html.