

Let's normalize treesdb

The goal of this exercise is to transform the flat one table treesdb database into a fully normalized database by applying 1NF, 2NF and 3NF forms.

For each entity that you feel would benefit from a stand alone table, the process is

1. create a new table with a primary key
2. inserts values from the trees column
3. add a foreign key to the trees table
4. delete the original column

The entity can be composed of multiple original columns. For instance address and suppl_address

Here's an example with the `name` column

- create a new table called `name`

```
create table tree_name (  
  id serial primary key,  
  name varchar unique not null  
)
```

SQL

Note the unique and not null constraints

- inserts values from the trees column

```
INSERT INTO tree_name (name)  
SELECT DISTINCT name  
FROM trees  
WHERE name IS NOT NULL;
```

SQL

- add the foreign key in the trees table

```
ALTER TABLE trees ADD COLUMN name_id INTEGER;
```

SQL

- update the foreign key column with the correct IDs

SQL

```
UPDATE trees t
SET name_id = tn.id
FROM tree_name tn
WHERE t.name = tn.name;
```

- add the foreign key constraint

SQL

```
ALTER TABLE trees
ADD CONSTRAINT fk_tree_name
FOREIGN KEY (name_id)
REFERENCES tree_name(id);
```

- delete the name column in the main trees table

SQL

```
ALTER TABLE trees
DROP COLUMN name;
```

Identify the logical entities

Which logical entities are you going to extract and regroup in a dedicated table?

- localisation
- naming, species, genre, variety
- dimensions
- what about idbase, remarkable, and anomaly ? can these stay in the trees table ?
- does arrondissement need its own table ?

Let's look at the columns and group them by logical entities

Ids

We can keep the ids in the trees table:

- id
- id_location
- idbase

same with the flags:

- remarkable : boolean

- anomaly : boolean

addresses and locations

It makes sense to regroup all the columns associated with location of the tree.

- address | character varying
- suppl_address | character varying
- arrondissement | character varying
- geolocation | point

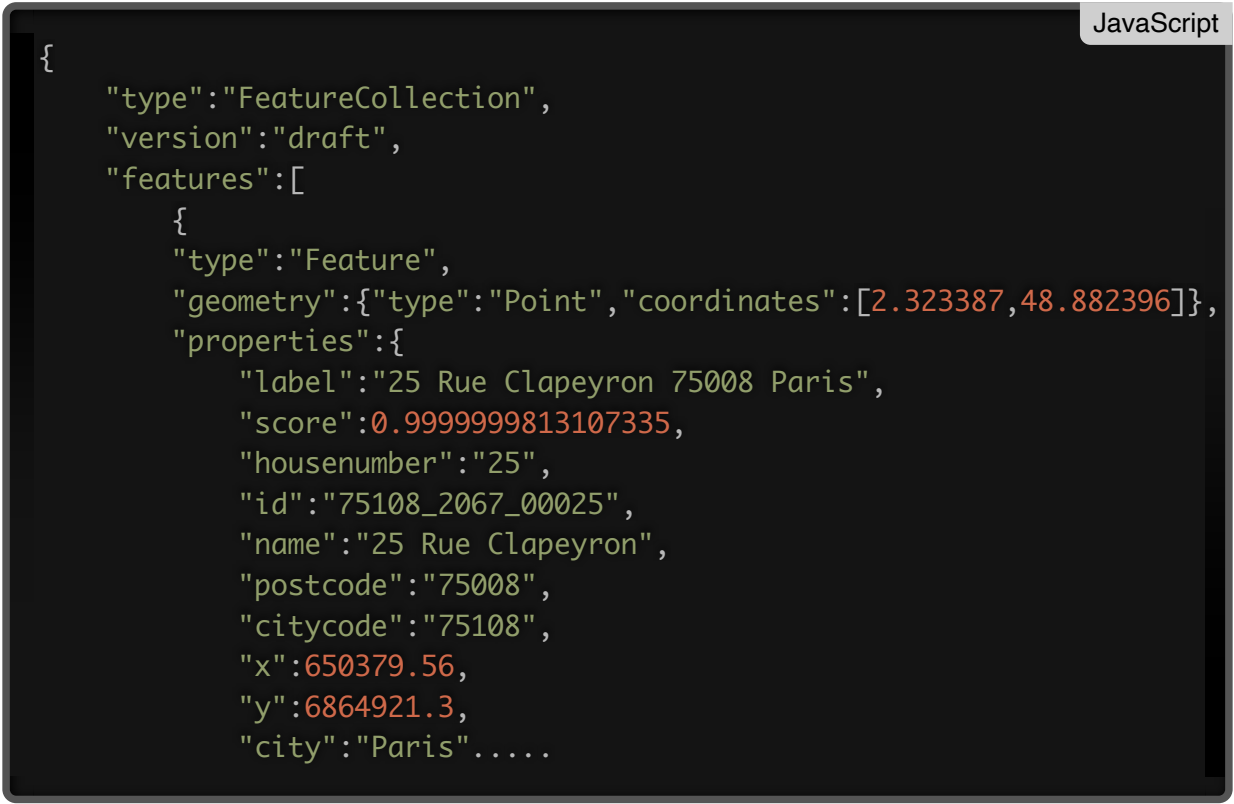
Note the weird format of the arrondissement and the lack of postal code

There is a national database of addresses called BAN (Base Nationale d'Adresse) available at adresse.data.gouv.fr/base-adresse-nationale

It has an api and you can find an address given location coordinates

```
curl "https://api-adresse.data.gouv.fr/reverse/?  
lon=2.323473607304248&lat=48.88237276393427&limit=1"
```

returns



```
{  
  "type": "FeatureCollection",  
  "version": "draft",  
  "features": [  
    {  
      "type": "Feature",  
      "geometry": {"type": "Point", "coordinates": [2.323387, 48.882396]},  
      "properties": {  
        "label": "25 Rue Clapeyron 75008 Paris",  
        "score": 0.9999999813107335,  
        "house number": "25",  
        "id": "75108_2067_00025",  
        "name": "25 Rue Clapeyron",  
        "postcode": "75008",  
        "citycode": "75108",  
        "x": 650379.56,  
        "y": 6864921.3,  
        "city": "Paris".....  
      }  
    }  
  ]  
}
```

so you could sanitize the addresses by feeding the long, lat into the API, and compare it with the trees address and arrondissement and if there's a match you get a clean address properly formatted

domain & stage

domain & stage should be given their dedicated table even if they don't change often:

- domain | character varying
- stage | character varying

taxonomy

How do we deal with **taxonomy** which includes those four columns.

Note that there's is an intrinsic relation between the name, genre, species and variety of a tree. Not sure which one. but it probably should be kept.

name | character varying genre | character varying species | character varying variety | character varying

So, our options are:

- one table per column (2NF) (but we loses the relation between the different categories)

or

- one taxonomy table (keeps the relations between the different categories but will require having NULL values)

measurements

Should the tree measurements be kept in the trees table or should they have their own separate table ?

- circumference | integer
- height | integer
- diameter | double precision

To answer consider these questions

a. **Functional dependency**: Do these dimensions depend solely on the tree's ID, or could they change independently? b. **Update anomalies**: Would keeping them in the main table cause update anomalies? c. **Data redundancy**: Is there any repetition of this data across multiple trees? d. **Query patterns**: How frequently are these dimensions accessed along with other tree data?

One piece of data is painfully missing from the trees table ... when the tree was last measured, the survey_date.

If the survey date was available then we could have multiple measures for a given tree and then it would definitely make sense to put the measurements into a separated dedicated table.

In our context (no date column), there's still a feeling that measurement is a logical / natural object by itself. So I would give them a dedicated table.

However since there is no date of each measurements, each tree has one unique set of measures that are not going to change often (guessing). So we can keep them in the trees table.