openMairie - synergie relationnel et géographique

Version 1.00

openMairie

23 January 2013

Table des matières

Ce document est proposé comme support à une web conférence organisée par l'ADULLACT intitulée "postgres-postgis : de la synergie du relationnel et du géographique".

La géometrie est un type de champ de la base de données au même titre qu'une date, qu'un champ texte, un champ numérique ... Cette manière de stocker l'information géographique a un impact sur les applications "métier" des collectivités mais aussi sur leurs applications strictement géographiques :

- les applications "métier" ne sont plus organisées en "silo" et peuvent être reliées entre elles avec les données géographiques
- Les relations d'agrégation et de composition du modèle relationnel des applications "métier" peuvent utiliser des liens géographiques : bureau de vote, quartier, périmètres scolaires, POS, secteur de ramassage, de distribution ...
- Le référentiel commun devient géographique et il n'est plus nécessaire d'avoir une codification quartier, ilot ou secteur pour rapprocher des données de diverses applications.
- La géolocalisation peut se faire de manière automatique sur la base de référentiels géographiques existants (adresse postale, parcelle , filaire de voirie)
- Les fonctions de calcul géométrique du SGBD peuvent être implémentées sans avoir besoin d'un interface géographique.
- Le SIG (système d'information géographique) peut lire les données directement dans l'application "métier" sans être obligé de répliquer les données dans le SIG avec des traitements différés.
- Le SIG peut proposer des applications relationnelles avec des fonctionnalités étendues qui va au delà de l'association d'attributs à un objet graphique.
- La mise en œuvre d'application mobile prend tout son sens dans des applications où les données sont géolocalisées.
- Il est possible d'utiliser les outils existants sur le net : géolocalisation via les API existants, les fonds de carte existants, les API de routage, les cartes au format WMS (web map service).

La base de données relationnelle et géographique permet un cadre de développement qui intégre la construction du référentiel, les applications relationnelles et/ou géographique en lien avec ce référentiel et les outils de mobilité (mobile et tablette) en utilisant la géolocalisation de l'utilisateur.

Nous proposons dans ce document d'utiliser quatre outils libres : postgis, Qgis, openLayers et openMairie. L'objectif est de construire une application relationnelle et géographique simple avec ces quatre outils.

Nous allons successivement :

- initialiser la base postgis "test" et faire des calculs géographique avec les fonctions postgis,
- publier une carte avec QGIS à partir de la base "test",
- accéder à la base "test" via l'interface web openlayers.
- mettre en place une interface de saisie de données relationnelles et géographique avec le framework openMairie

CHAPITRE 1

Faire des calculs géographiques avec postgis

Nous allons dans ce chapitre :
– initialiser la base postgres - postgis
– manipuler les fonctions postgis

1.1 Rappel sur l'openGIS

PostGIS respecte les spécifications de l'Open GIS Consortium pour l'intégration de données géographiques dans des bases de données :

"Simple Features Specification for SQL" Version : 1.2.1 du 04/08/2012

http ://www.opengeospatial.org/standards/sfs

PostGIS supporte les objets géométriques définis par l'O.G.C (Open GIS COnsortium).

```
.. image:: ../_static/geometrie.png
```

1.2 installation postgis

Prérequis : Installer les paquets postgis sous ubuntu (cochez la case postgis dans l'installation de postgresql via la logithèque ubuntu)

1.2.1 la base test

créer la base "test"

dans une console (linux ou windows)

```
sudo -s -u postgres
createdb test
```

dans la base "test" :

– executer en schéma public le script usr/share/postgres/XXX/contrib/

postgis.sql : creer 700 fonctions postgis et les 2 tables geometry_columns et spatial_ref_sys

spatial_ref_sys.sql : creer le referentiel spatial : 3749 systèmes de projection (SRID)

Les sytèmes de projection les plus utilisés sont

lambert 93 : 2154 (format des administrations et collectivités françaises)

mercator : 900913 (google, osm, bing)

géographique 4326 latitude et longitude utilisé par les mobiles

Nous travaillons dans cet exemple sur une base postgresql en lamber t93 : 2154

Dans une console pour la version postgresql, 9.1 et postgis 1.5

```
sudo -s -u postgres

psql -d test -f /usr/share/postgresql/9.1/contrib/postgis-1.5/postgis.sql
psql -d test -f /usr/share/postgresql/9.1/contrib/postgis-1.5/spatial_ref_sys.sql
```

Verification des version postgis et des composants

```
select postgis_full_version();

                                    postgis_full_version
-----------------------------------------------------------------------------------------------
POSTGIS="1.5.3" GEOS="3.2.2-CAPI-1.6.2" PROJ="Rel. 4.7.1, 23 September 2009" LIBXML="2.7.8"
USE_STATS
(1 row)
```

Nous allons maintenant créer les tables utiles pour la formation :

une table en lien relationnel “rivoli” qui contient le libellé de la voie avec une clé “rivoli” utilisée par le service des impôts

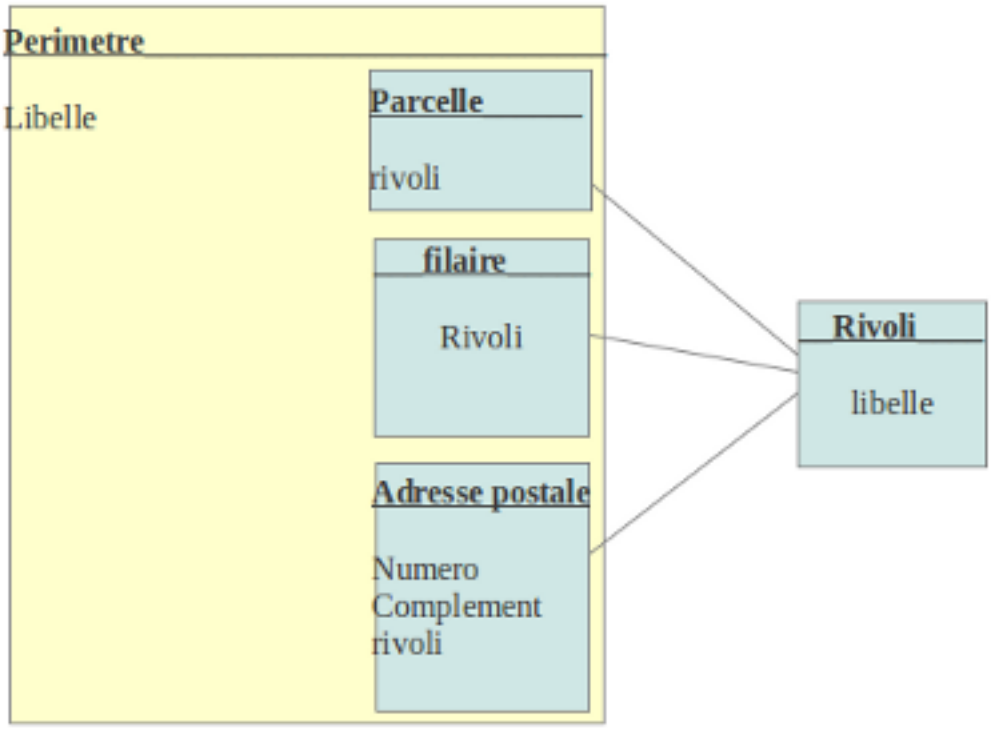
une table d’adresse qui contient les points d adresses

une table du filaire qui contient les troncons des rues de la table rivoli

une table des parcelles qui contient les parcelles associées aux rues sous forme de polygones

une table des perimetres qui seront définis comme polygones

Le modèle relationnel est suivant le schéma ci dessous



– créer des tables, contraintes, sequences de l'application dans la base test

```
psql test
```

```
psql -d test -f /var/www/projet/postgis_conf/trunk/openmairie/data/pgsql/init_metier.sql
```

Ajout de la colone géométrique avec la commande AddGeometryColumn()

```
psql -d test -f /var/www/projet/postgis_conf/trunk/openmairie/data/pgsql/init_metier_sig.sql
```

Dans ce script sont contenues les commandes suivantes

```
SELECT AddGeometryColumn ( 'adresse_postale', 'geom', 2154 , 'POINT', 2 );
SELECT AddGeometryColumn ( 'filaire', 'geom', 2154 , 'LINESTRING', 2 );
SELECT AddGeometryColumn ( 'parcelle', 'geom', 2154 , 'POLYGON', 2 );
SELECT AddGeometryColumn ( 'perimetre', 'geom', 2154 , 'POLYGON', 2 );
```

L'utilisation de la commande AddGeometryColumn() permet de créer les méta-données sur les objets géométriques concernant le type, le srid, la dimension. (AddGeometryColumn(nom_de_Table, nom_de_Colonne_Geometrique, SRID, Type_Geometrie, Dimension))

Ces métadonnées sont dans la table geometry_columns

```
select * from geometry_columns
```

```
f_table_schema|f_table_name    | f_geometry_column | dimension | srid |    type
--------------+----------------+-------------------+-----------+------+------------
 public       | adresse_postale| geom              |         2 | 2154 | POINT
 public       | filaire        | geom              |         2 | 2154 | LINESTRING
 public       | parcelle       | geom              |         2 | 2154 | POLYGON
 public       | perimetre      | geom              |         2 | 2154 | POLYGON
(4 rows)
```

Il est créer avce la commande addGeometryColumn() des contraintes de saisie dans les tables correspondantes :

```
-- table adresse_postale

CONSTRAINT enforce_dims_geom CHECK (st_ndims(geom) = 2),
CONSTRAINT enforce_geotype_geom CHECK (geometrytype(geom) = 'POINT'::text OR geom IS NULL),
CONSTRAINT enforce_srid_geom CHECK (st_srid(geom) = 2154)
```

– entrer les jeux de données avec init_metier_data.sql

dans une console

```
psql -d test -f /var/www/projet/postgis_conf/trunk/openmairie/data/pgsql/init_metier_data.sql
```

la base comprend
– 3 voies (rivoli) : rues de la république, dulau et gambetta
– 91 parcelles du centre ville d'arles,
– 101 adresses postales des 3 rues
– 16 filaires de voirie de ces 3 rues
– 2 périmetres.

1.2.2 Mode de stockage

Le mode de stockage est binaire pour des raison d'économie de place sur le disque de dur et pour avoir un accès plus rapide

Exemple du stockage binaire du champ geom de la table adresse_postale pour la rue dulau

```
 select numero, geom from adresse postale where rivoli = '1255';


 numero |                        geom
--------+---------------------------------------------------
      5 | 01010000206A08000048E17A14EE5C2941D7A3700D5BFC5741
     10 | 01010000206A08000052B81E85CE5C29411F85EB5155FC5741
      7 | 01010000206A0800009A999919EB5C2941CDCCCC7C59FC5741
      6 | 01010000206A080000AE47E17ADC5C29413333337358FC5741
      3 | 01010000206A0800003333333B3F15C294148E17AF45CFC5741
      9 | 01010000206A080000A4703D0AE85C29411F85EBE157FC5741
      8 | 01010000206A08000048E17A14D95C294114AE47E156FC5741
     11 | 01010000206A080000295C8FC2E35C2941EC51B89E55FC5741
      1 | 01010000206A0800001F85EBD1F55C2941E17A141E5FFC5741
      2 | 01010000206A080000EC51B81EEA5C294185EB51785FFC5741
     12 | 01010000206A0800005C8FC275C55C29413D0AD74356FC5741
      4 | 01010000206A080000A4703D0AE55C2941C3F528DC5CFC5741
(12 rows)
```

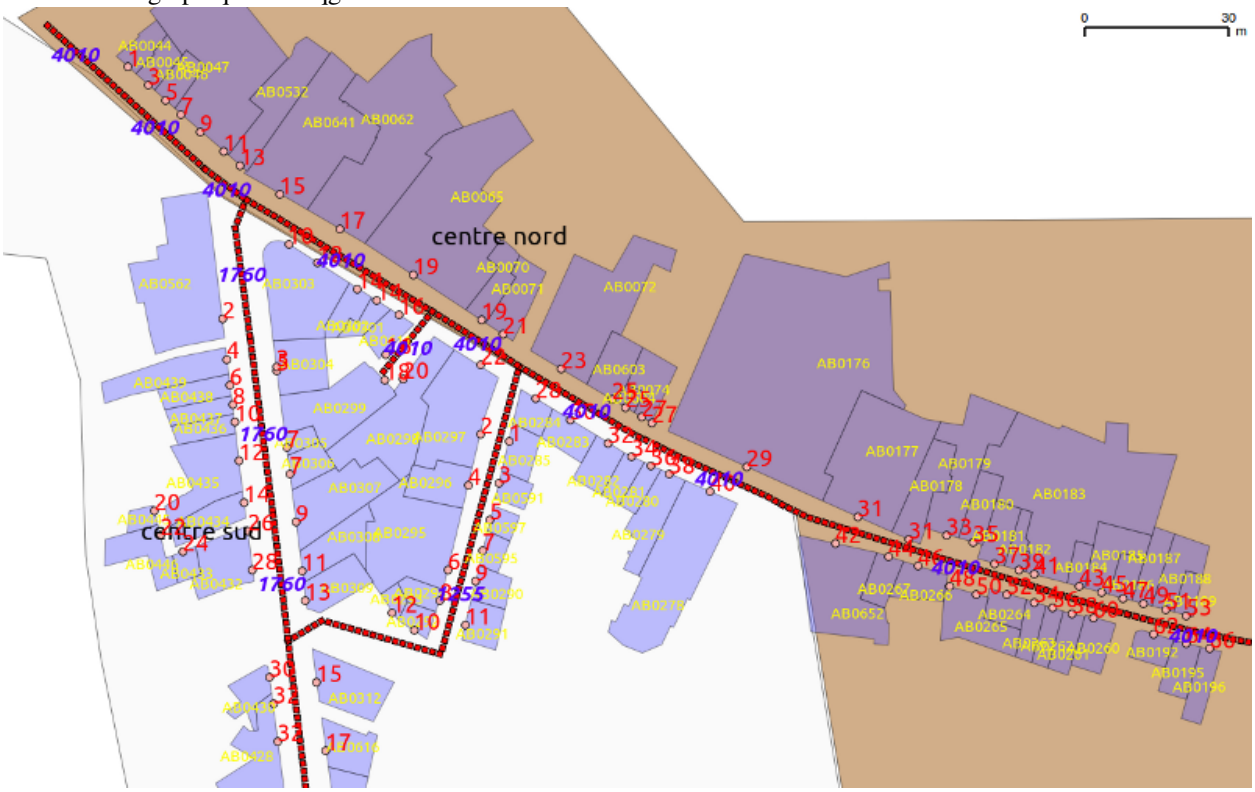
Pour avoir un format lisible, il faut utiliser la fonction astext

```
 select numero, astext(geom) from adresse_postale where rivoli = '1255';

 numero |           astext
--------+----------------------------
      5 | POINT(831095.04 6287724.21)
     10 | POINT(831079.26 6287701.28)
      7 | POINT(831093.55 6287717.95)
      6 | POINT(831086.24 6287713.8)
      3 | POINT(831096.85 6287731.82)
```

```
      9 | POINT(831092.02 6287711.53)
      8 | POINT(831084.54 6287707.52)
     11 | POINT(831089.88 6287702.48)
      1 | POINT(831098.91 6287740.47)
      2 | POINT(831093.06 6287741.88)
     12 | POINT(831074.73 6287705.06)
      4 | POINT(831090.52 6287731.44)
(12 rows)
```

Visualisation graphique sous qgis de la base installée



1.2.3 Implementation avec des schéma de postgres

Il est préférable d'implémenter les applications métiers dans un schéma, ce qui permet d'avoir une vision plus simple de l'organisation des applications : un schéma par application.

Dans notre exemple, nous avons pour des raisons de simplicité laisser l ensemble des tables dans le schéma public mais une organisation cohérente serait :
– un schéma public : 2 tables postgis (geometry_columns et spatial_re_ref-sys) et les 780 fonctions postgis
– un schéma voirie avec les rivoli, adresse_postales et le filaire
– un schéma cadastre avec les parcelles

1.3 Les fonctions postgis

PostGIS permet les requetes spatiales et/ou attributaires complexes, gère les changements de projections, les données 2, 3 ou 4d, les opérations "topologiques" telles que calculs de buffers, d'intersections, d'unions etc.

Nous proposons quelques exemples de fonction postgis avec l'utilisation de jointure avec une table relationnelle de notre exemple (rivoli) et l'utilisation de requête de statistiques (count, max, min, sum).

les fonctions présentées sont les suivantes :
– les mesures de surface : area2d()
– les mesures de longueurs : length2d()
– les mesures de distance : distance()
– les calculs "topologiques" : les intersections, les unions.
Pour plus d'information, il faut voir la documentation complète sur le site (en anglais)

http ://postgis.refractions.net

1.3.1 Calcul de l'aire d'un polygone

La fonction Area2d() renvoie les aires des objets polygones

```
SELECT  libelle ,Area2d(geom) FROM perimetre;

    libelle   |      area2d
 -------------+------------------
  centre sud  | 26482.5316848755
  centre nord | 23795.5244145393
 (2 rows)
```

En la combinant avec la fonction min, on peut rechercher la plus petite parcelle

```
 SELECT parcelle, Area2d(geom) FROM parcelle
 WHERE Area2d(geom) = (SELECT Min(Area2d(geom)) FROM parcelle);

 parcelle |      area2d
----------+------------------
 AB0604   | 18.9489500522614
(1 row)
```

la plus grande, avec la fonction max

```
 SELECT parcelle, Area2d(geom) FROM parcelle
 WHERE Area2d(geom) = (SELECT Max(Area2d(geom)) FROM parcelle);

 parcelle |     area2d
----------+-----------------
 AB0176   | 1529.6137008667
(1 row)
```

1.3.2 Calcul de longueur de voies

La fonction length2d() permet de calculer la longueur

```
 select filaire.rivoli, libelle, length2d(geom)
     from filaire inner join rivoli on rivoli.rivoli=filaire.rivoli
     where filaire.rivoli = '1255';

 rivoli |  libelle  |     length2d
--------+-----------+------------------
 1255   | RUE DULAU | 95.6156276352888
(1 row)
```

la rue dulau est composé d'un segment et fait 95.62 metre de long

Pour la rue de la gambetta

```
 select filaire.rivoli, libelle, length2d(geom)
     from filaire inner join rivoli on rivoli.rivoli=filaire.rivoli
     where filaire.rivoli = '1760';

 rivoli |   libelle    |     length2d
--------+--------------+------------------
 1760   | RUE GAMBETTA | 70.8285526282503
 1760   | RUE GAMBETTA | 66.8940871825706
 1760   | RUE GAMBETTA | 30.1209339519525
 1760   | RUE GAMBETTA | 36.2619704382524
 1760   | RUE GAMBETTA | 26.1567448277959
(5 rows)


 select sum(length2d(geom))
     from filaire  where rivoli = '1760';

         sum
  ------------------
   230.262289028822
  (1 row)
```

La rue Gambetta est composé de 5 segments et fait 230 métres de long

1.3.3 Premier et dernier point du filaire

Nous souhaitons connaitre le premier et le dernier point du filaire (troncon) de la rue de la gambetta

```
 SELECT rivoli,AsText(StartPoint(geom)) as debut ,AsText(EndPoint(geom)) as fin
     FROM filaire where rivoli = '1760' order by geom;

 rivoli |            debut            |             fin
--------+-----------------------------+-----------------------------
 1760   | POINT(831044.46 6287790.48) | POINT(831045.46 6287760.98)
 1760   | POINT(831045.46 6287760.98) | POINT(831050.13 6287725.02)
 1760   | POINT(831050.13 6287725.02) | POINT(831053.25 6287699.05)
 1760   | POINT(831053.25 6287699.05) | POINT(831061.2 6287632.63)
 1760   | POINT(831061.2 6287632.63)  | POINT(831071.92 6287562.62)
(5 rows)
```

Le point de fin de chaque segment correspond au point de début suivant.

1.3.4 Création de la géometrie pour les voies entières

En utilisant la requête ST_UNION pour les tronçons du filaire, nous allons créer la géométrie de la voie entière La fonction ST_LINEMERGE elimine les discontinuité dans le retour de st_union (évite la création d'une multiligne)

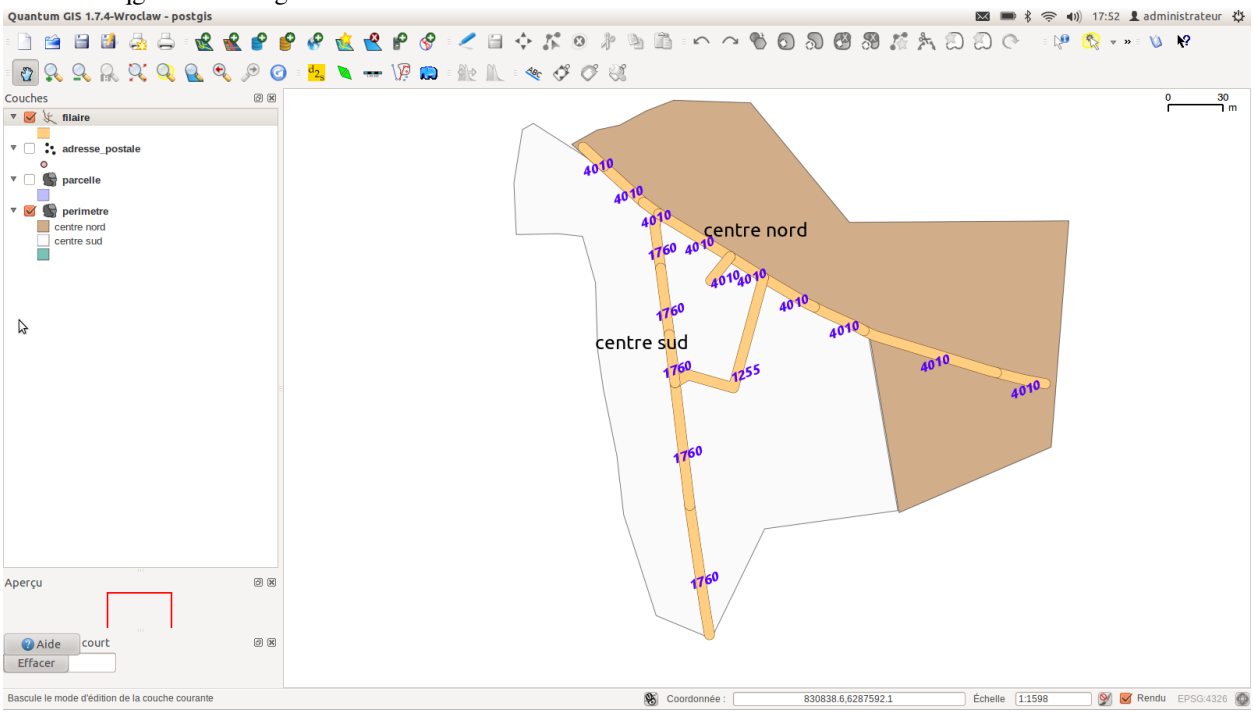
```
SELECT astext(ST_LINEMERGE(ST_UNION(geom)))
    FROM filaire WHERE rivoli ='1760';

LINESTRING(831044.46 6287790.48,831042.25 6287784.64,831045.46 6287760.98,
           831050.13 6287725.02,831053.25 6287699.05, 831061.2 6287632.63,
```

```
            831069.31 6287578.01,831071.92 6287562.62)
(1 row)
```

1.3.5 Calcul d'une intersection entre deux voies (rivoli)

Voici sous qgis un affichage des voies :



Nous allons maintenant calculer le point d'intersection entre la rue Dulau et la rue Gambetta

```
select astext(ST_INTERSECTION(a.geom,b.geom)) from filaire a, filaire b
    where a.rivoli = '1760' and b.rivoli= '1255';

            astext
 ----------------------------
  GEOMETRYCOLLECTION EMPTY
  POINT(831053.25 6287699.05)
  GEOMETRYCOLLECTION EMPTY
  GEOMETRYCOLLECTION EMPTY
  POINT(831053.25 6287699.05)
 (5 rows)
```

Le calcul se fait sur la base de l'intersection des 5 filaires de la rue gambetta avec le filaire de la rue Dulau. 2 points identiques sont en intersection.

En récupérant la géométrie complète de la rue gambetta suite à l'union des filaires de cette rue :

```
select astext(ST_INTERSECTION(geometryFromText('LINESTRING(831044.46 6287790.48,
           831042.25 6287784.64, 831045.46 6287760.98, 831050.13 6287725.02,
           831053.25 6287699.05, 831061.2 6287632.63, 831069.31 6287578.01,831071.92 6287562.62)'
           ,2154), geom))
           from filaire where rivoli= '1255';

       astext
```

```
-----------------------------
 POINT(831053.25 6287699.05)
(1 row)
```

Nous avons le même point.

1.3.6 Calcul de centroid de parcelle

Pour positionner un permis de construire, openFoncier utilise le centroîde de la parcelle concernée. Nous proposons ci dessous d'utiliser la fonction centroid

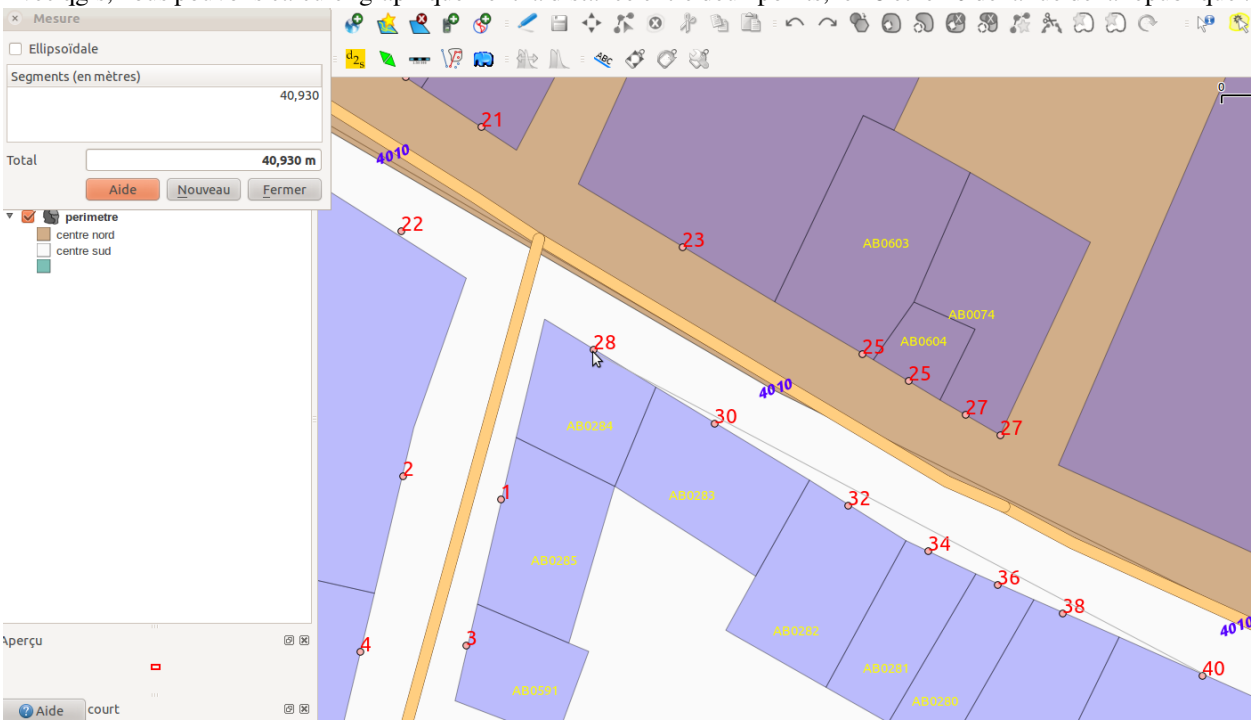
```
select parcelle, astext(centroid(geom)) from parcelle where parcelle = 'AB0298';

 parcelle |                  astext
----------+-------------------------------------------
 AB0298   | POINT(831075.459192478 6287744.75350581)

(1 row)
```

1.3.7 Calcul de distance entre deux points d'adresse

Avec qgis, nous pouvons calculer graphiquement la distance entre deux points, le 28 et le 40 de la rue de la république :



Nous proposons de calculer la distance entre ces deux points d'adresse postale avec une requête postgis

```
select distance(a.geom,b.geom) from adresse_postale a, adresse_postale b
    where a.numero = 28 and b.numero= 40 and a.rivoli = '4010' and b.rivoli = '4010' ;

 distance
------------------
 40.9929469541595
(1 row)
```

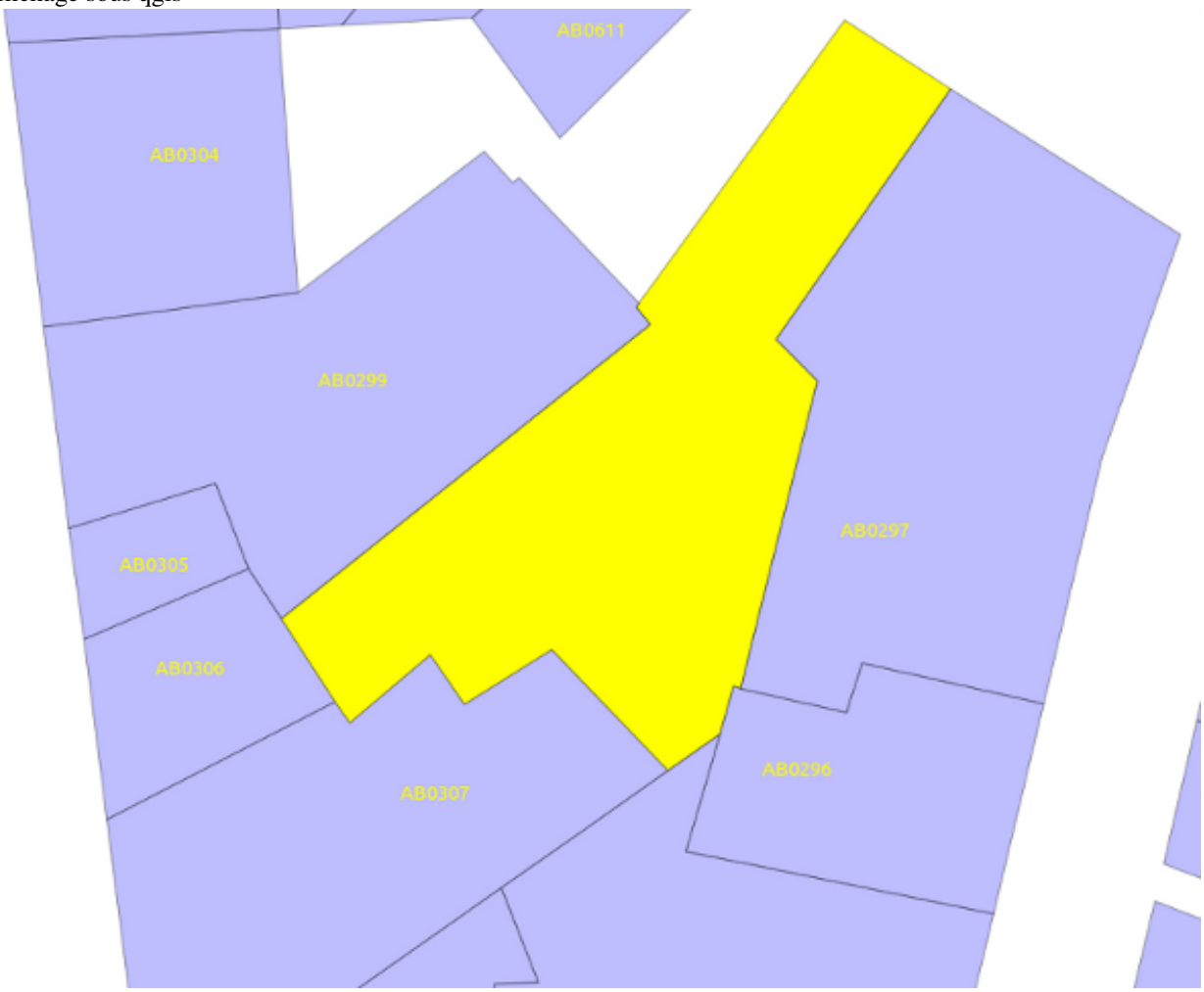
1.3.8 Parcelles avoisinantes d'une parcelle

Nous allons utiliser la fonction st_distance qui si elle est égal à zéro, selectionne les parcelles collées à la parcelle cible 'AB0298' La fonction not st_equals permet d'éliminer la parcelle 'AB0298' dont la distance sest égal à 0 avec la parcelle cible

```
 SELECT b.parcelle FROM parcelle a, parcelle b
     WHERE a.parcelle = 'AB0298' AND Distance(a.geom,b.geom)=0
     AND not st_equals(a.geom,b.geom);

 parcelle
----------
 AB0307
 AB0306
 AB0296
 AB0297
 AB0299
 AB0295
(6 rows)
```

Affichage sous qgis



1.3.9 Périmètre contenant une parcelle

Sous qgis, nous visualisons le périmètre “centre sud” (en orange) comme contenant la parcelle “AB0298”

Nous pouvons connaitre le périmètre dans lequel est contenu la parcelle AB0298 avec une requête postgis :

```
  select b.libelle from parcelle a, perimetre b where contains(b.geom,a.geom)
      and a.parcelle = 'AB0298';

  libelle
------------
 centre sud
(1 row)
```

Cette fonction est très utile pour connaitre le pos correspondant à une parcelle, le bureau d’un électeur, le secteur scolaire d’un élève ...

1.3.10 Nombre de parcelles par périmètre

Nous souhaitons savoir le nombre de parcelles de chaque périmètre

```
   select b.libelle, count(a.parcelle) from parcelle a , perimetre b
       where contains(b.geom, a.geom)  group by b.perimetre;


   libelle    | count
--------------+-------
 centre sud   |    51
 centre nord  |    39
(2 rows)
```

Cette fonction peut permettre des comptage d'électeur par bureau, d'élèves par secteur ...

1.3.11 Associer les codes rivolis au parcelles limitrophes de chaque voie

Nous allons essayer d'associer les codes rivoli aux parcelles limitrophes de voie. Il y a 101 adresse_postale Pour cela nous allons utiliser les adresses postales qui sont associées aux rivolis

```
 -- Premier essai : avec la fonction contains

 select b.numero, b.complement,c.libelle, a.parcelle
     from parcelle a, adresse_postale b, rivoli c where c.rivoli = b.rivoli
     and contains(a.geom,b.geom) order by c.rivoli, b.numero;

 numero | complement |       libelle        | parcelle
--------+------------+----------------------+----------
      2 |            | RUE DULAU            | AB0297
      4 |            | RUE DULAU            | AB0296
      6 |            | RUE DULAU            | AB0295
      8 |            | RUE DULAU            | AB0294
     10 |            | RUE DULAU            | AB0292

 ...
 57 rows

 -- Deuxieme essai avec la fonction intersects

 select b.numero, b.complement,c.libelle, a.parcelle
     from parcelle a, adresse_postale b, rivoli c
     where c.rivoli = b.rivoli and intersects(a.geom,b.geom) order by c.rivoli, b.numero;

 numero | complement |       libelle        | parcelle
--------+------------+----------------------+----------
      2 |            | RUE DULAU            | AB0297
      4 |            | RUE DULAU            | AB0296
      6 |            | RUE DULAU            | AB0295
      8 |            | RUE DULAU            | AB0294
     10 |            | RUE DULAU            | AB0292
 ...
 57 rows

 -- troisième essai avec la fonction distance = 0

 select b.numero, b.complement,c.libelle, a.parcelle
```

```
    from parcelle a, adresse_postale b, rivoli c where c.rivoli = b.rivoli
    and distance(a.geom,b.geom) < 0.30  order by c.rivoli, b.numero;

 numero | complement |       libelle        | parcelle
--------+------------+----------------------+----------
      2 |            | RUE DULAU            | AB0297
      4 |            | RUE DULAU            | AB0296
      6 |            | RUE DULAU            | AB0295
      8 |            | RUE DULAU            | AB0294
     10 |            | RUE DULAU            | AB0292

 ...

 58 rows

 -- quatrième essai avec la fonction distance et un seuil de tolérance de 30 cm

 select b.numero, b.complement,c.libelle, a.parcelle
    from parcelle a, adresse_postale b, rivoli c where c.rivoli = b.rivoli
    and distance(a.geom,b.geom) < 0.30  order by c.rivoli, b.numero;

 numero | complement |       libelle        | parcelle
--------+------------+----------------------+----------
      1 |            | RUE DULAU            | AB0285
      2 |            | RUE DULAU            | AB0297
      3 |            | RUE DULAU            | AB0591
      4 |            | RUE DULAU            | AB0296
      5 |            | RUE DULAU            | AB0597
      6 |            | RUE DULAU            | AB0295
      7 |            | RUE DULAU            | AB0595
      8 |            | RUE DULAU            | AB0294
      9 |            | RUE DULAU            | AB0290
     10 |            | RUE DULAU            | AB0292
     11 |            | RUE DULAU            | AB0291
     12 |            | RUE DULAU            | AB0293

 101 rows
```

Nous retenons cette dernière selection pour mettre à jour le rivoli de parcelle qui n'est pas rempli

```
update only parcelle  set rivoli = adresse_postale.rivoli
    from adresse_postale where distance(parcelle.geom,adresse_postale.geom) < 0.30;

UPDATE 91
```

91 parcelles ont été mises à jour. 9 parcelles ont 2 numéros, 1 parcelle a 3 numéro

```
select a.parcelle, count(a.parcelle) from parcelle a, adresse_postale b
    where distance(a.geom,b.geom) < 0.30
    group by a.parcelle
    having count(a.parcelle) > 1 ;

 parcelle | count
----------+-------
 AB0611   |     2
 AB0047   |     3
 AB0304   |     2
 AB0432   |     2
 AB0303   |     2
```

```
 AB0265   |     2
 AB0297   |     2
 AB0428   |     2
 AB0074   |     2
(9 rows)
```

La parcelle AB0047 a 3 numéros dans la même rue

```
 select a.parcelle, b.rivoli, b.numero  from parcelle a, adresse_postale b
         where parcelle = 'AB0047' and distance(a.geom,b.geom) < 0.30;

 parcelle | rivoli | numero
----------+--------+--------
 AB0047   | 4010   |     11
 AB0047   | 4010   |      9
 AB0047   | 4010   |      7
(3 rows)
```

Par contre la parcelle AB0297 a deux numéros de rues différentes Pour la rue dulau, le 2 correspond à la parcelle AB0297 qui est aussi le 22 rue de la république

```
 select a.parcelle, b.rivoli, b.numero, c.libelle  from parcelle a, adresse_postale b, rivoli c
     where parcelle = 'AB0297' and distance(a.geom,b.geom) < 0.30 and b.rivoli=c.rivoli;


 parcelle | rivoli | numero |       libelle
----------+--------+--------+----------------------
 AB0297   | 4010   |     22 | RUE DE LA REPUBLIQUE
 AB0297   | 1255   |      2 | RUE DULAU
(2 rows)
```

CHAPITRE 2

Publier des cartes avec qgis

Nous vous proposons dans ce chapître :
– de décrire succintement le fonctionnement de qgis serveur
– de décrire la connexion sur une base postgis
– de décrire le fichier xml d'un projet qgis

2.1 Installation du serveur WMS qgis

Le serveur WMS QGIS permet de publier des projets au format WMS et de rendre accessible les projets par des clients web sur internet

La communication entre QGIS Mapserver et notre serveur Web s'appuie sur le protocole CGI/FCGI

```
sudo apt-get install libfcgi-dev
```

L'installation de QGIS Mapserver se fait en utilisant le dépôt ubuntugis

```
apt-get install qgis-mapserver
```

Dans le répertoire /usr/lib/cgi-bin/, le fichier qgis_mapserv.fcgi interprete les requêtes WMS et les retourne ensuite sous forme d'images. QGIS utilise :
– la bibliothèque OGR pour accéder aux données vectorielles (postgis, shapefiles)
– la bibliothèque GDAL pour accéder aux données au format rasters (plus de 100 formats rasters supportés)
– accède au format de l'OGC : WMS, WFS et WFS-T (avec exetension)

2.2 Accès à la base postgis

Paramétrage qgis pour accéder à la base test

Appuyer sur connecter

Selectionner toutes les couches et ajouter

QGIS necessite une clé primaire numérique pour chaque table, c'est pour cela que nous avons mis les oids dans la table parcelle qui a une clé primaire alphabétique.

2.3 Le fichier du projet :

Le fichier du projet est stocké au format xml dans le répertoire qgis : qgis/postgis.qgs

Les informations stockées sont les suivantes :
– les couches ajoutées et les paramètres de connexion
– les propriétés des couches et notament la sémiologie
– la projection de la carte
– l'étendue de la dernière étendue de visualisation

CHAPITRE 3

Accéder aux cartes et aux fonds internet avec openLayers

Nous vous proposons dans ce chapître de décrire quelques fonctionnalités du client web openlayers au travers de notre exemple openlayers/wms.html

openLayers est une bibliothèque javascript qui permet d'accéder à tous les formats standards d'information géométrique (rasters, vecteurs, wms ...)

Nous vous proposons au préalable de décrire le client puis ensuite de regarder notre exemple.

3.1 le client openlayers

Nous proposons dans se chapiître d' utiliser la librairie javascript "openLayers"
– soit après avoir téléchargé la librairie sur http ://openlayers.org
– soit en adressant la librairie dans le fichier html ou php

```
<script src="http://openlayers.org/api/OpenLayers.js"></script>
```

Nous prendrons la deuxième option dans le script "wms" suivant.

3.2 Le web map service de QGIS

Il est décrit ici l'exemple avec le script "openlayers/wms.html" du projet "formation postgis".

Affichage de la carte dans un div : :

Affichage d'un fond openstreetmap (il faut un fond obligatoire)

```
// valorisation des projection
var geographic = new OpenLayers.Projection("EPSG:4326");
var mercator = new OpenLayers.Projection("EPSG:900913");
var world = new OpenLayers.Bounds(-180, -89, 180, 89).transform(
    geographic, mercator
);
var options = {
    projection: mercator,
    displayProjection: geographic,
```

```
    units: "m",
    maxExtent: world,
    maxResolution: 156543.0399
};
var map = new OpenLayers.Map("map-id",options);
var osm = new OpenLayers.Layer.OSM();
    map.addLayer(osm);
map.setBaseLayer(osm);
```

Affichage d'une couche wms "perimetre" du projet "postgis.qgis" depuis le serveur qgis

```
var cs = new OpenLayers.Layer.WMS( Couche perimetre",
        "http://localhost/cgi-bin/qgis_mapserv.fcgi?SERVICE=WMS&VERSION=1.3.0&map=/var/www/projet/pos
        {layers: "perimetre", transparent: true});
map.addLayer(cs);
```

Attention de bien préciser l'emplacement de votre fichier qgis sur votre serveur (ou en localhost)

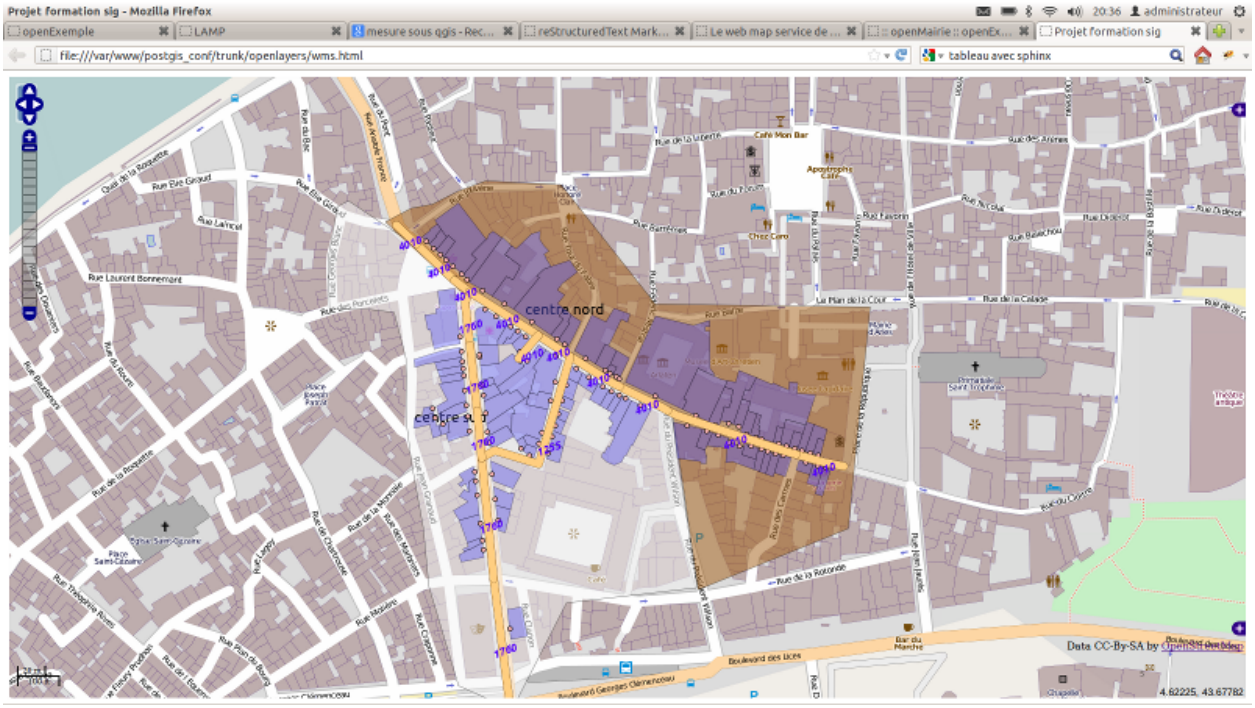
Centrage de la carte sur arles avec transformation coordonnées géographique en mercator

```
var arles_center = new OpenLayers.LonLat(4.62584, 43.67654).transform(geographic, mercator);
map.setCenter(arles_center, 17);
```

mise en place des outils de controles openLayers utilisés pour la carte

```
map.addControl(new OpenLayers.Control.LayerSwitcher());      changement de layers (en haut à droite)
map.addControl(new OpenLayers.Control.OverviewMap());        carte de deplacement (bas droite)
map.addControl(new OpenLayers.Control.ScaleLine());          echelle
map.addControl(new OpenLayers.Control.MousePosition());      position x y de la souris
map.addControl(new OpenLayers.Control.PanZoomBar());         zoom en haut a gauche
```

Affichage du résultat sur le navigateur :



CHAPITRE 4

Créer un interface de saisie relationnelle et géographique avec openmairie

Nous allons dans ce chapître interfacer notre base de onnées test avec le framework openMairie et génerer les formulaires permettant de saisir les données relationnelles et géométriques.

Enfin nous donnerons une liste indicative des applications openMairie en lien géographique existantes et à venir.

4.1 Installation du framework openMairie

Le framework est téléchargeable sur la forge de l'adullact :

http ://adullact.net/frs/ ?group_id=265

Il faut prendre le dernier projet disponnible (à ce jour la version 4.3.0b1)

Il faut le décompresser et mettre le projet dans le répertoire des applications apache (sous linux /var/www) et sous linux donner les droits d'écriture

4.2 Paramètrage du projet

Il faut executer le script d'initialisation du framework (utilisateurs, droits ...)

```
psql -d test -f /var/www/projet/postgis_conf/trunk/openmairie/data/pgsql/init.sql
```

Il faut paramétrer la base postgres dans dyn/database.inc.php

```
// PostGreSQL
$conn[1] = array(
    "formation postgis",
    "pgsql",
    "pgsql",
    "postgres",
    "postgres",
```

```
    "tcp",
    "localhost",
    "5432",
    "",
    "test",
    "AAAA-MM-JJ",
    "public",
    "",
    NULL,
    "mail-default",
);
```

il faut rajouter dans dyn/menu.inc.php dans la partie application l'acces au formulaire rivoli

```
$links[] = array(
    "href" => "../scr/tab.php?obj=rivoli",
    "class" => "rivoli",
    "title" => _("rivoli"),
    "right" => array("rivoli" ),
    "open" => array("tab.php|rivoli", "form.php|rivoli", ),
);
```

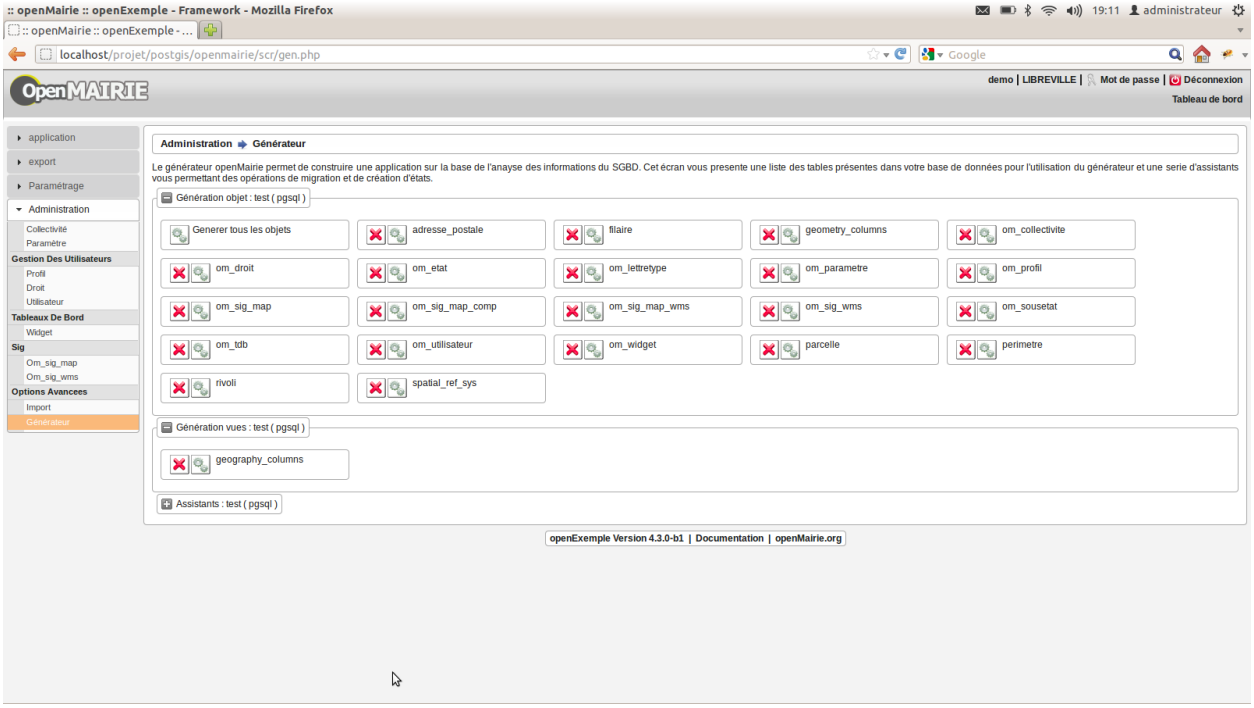
Il faut rajouter les droits d'accès dans l'interface : administration droit ou executer les requetes suivantes

```
INSERT INTO om_droit VALUES
(21, 'rivoli', 3),
(22, 'filaire', 3),
(23, 'parcelle', 3);

SELECT pg_catalog.setval('om_droit_seq', 23, true);
```

Entrer dans l application login : demo , pwd : demo

Il faut ensuite générer les formulaires de saisie dans l'application :
– administration -> generateur -> adresse_postale
– administration -> generateur -> rivoli
– administration -> generateur -> filaire
– administration -> generateur -> parcelle



4.3 Ajout des cartes dans l'interface

Il est proposé d'ajouter les cartes paramétrées pour l'application pour les adresses postales et le filaire. en utilisant le script sql/init_sig_map.sql. Les données auraient pu être saisies dans l'interface openMairie : administration -> om_sig_map

Il faut lancer le script suivant

```
psql -d test -f /var/www/projet/postgis_conf/trunk/openmairie/data/pgsql/init_metier_parametrage.sql
```

Dans adminitrateur-> wms, précisez l'emplacement du fichier qgis pour que le lien wms fonctionne.

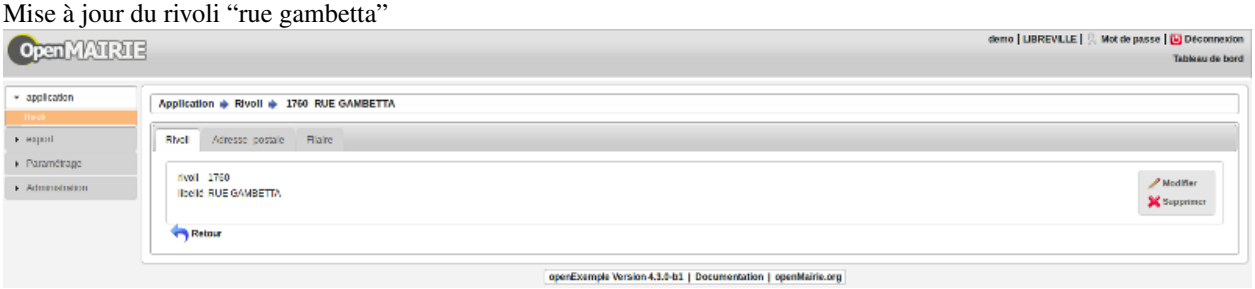
4.4 Les formulaires de saisie relationnel et géographique :

Nous vous proposons dans ce chapitre la visualisation des écrans de saisie.

4.4.1 La saisie de rivoli :

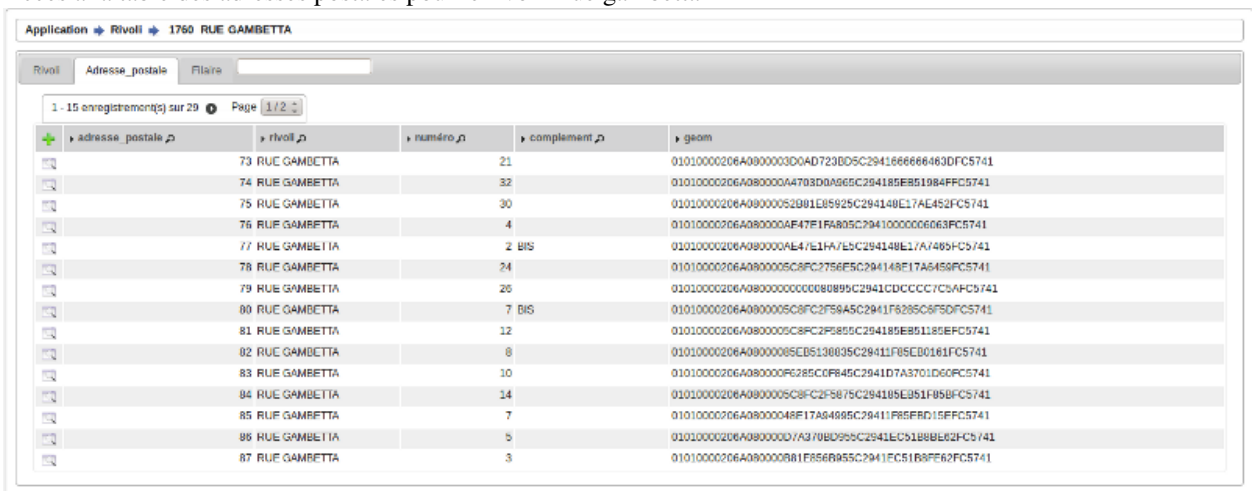
Accès à la table :

Mise à jour du rivoli “rue gambetta”

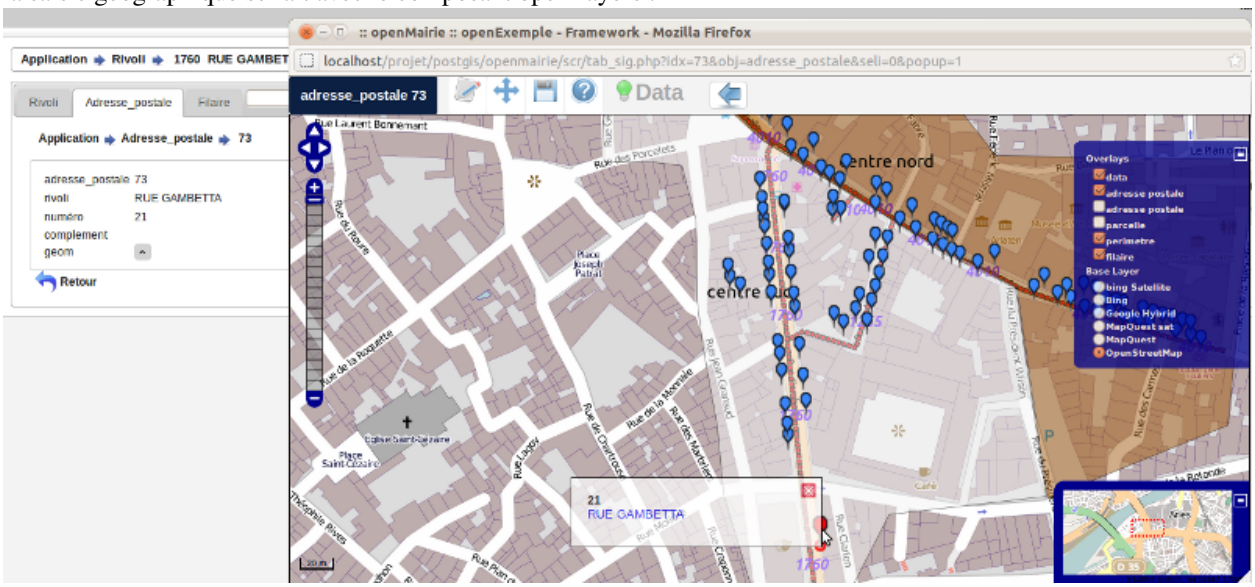


4.4.2 L’onglet adresse_postale :

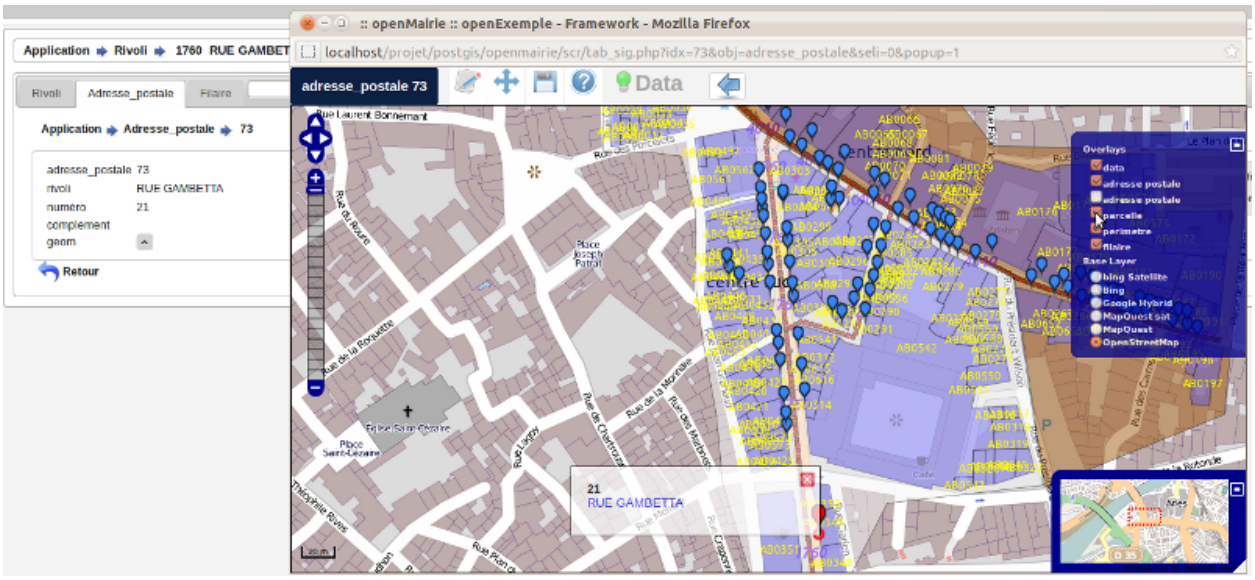
Accès à la table des adresses postales pour le rivoli “rue gambetta”



la saisie geographique se fait avec le composant openLayers :

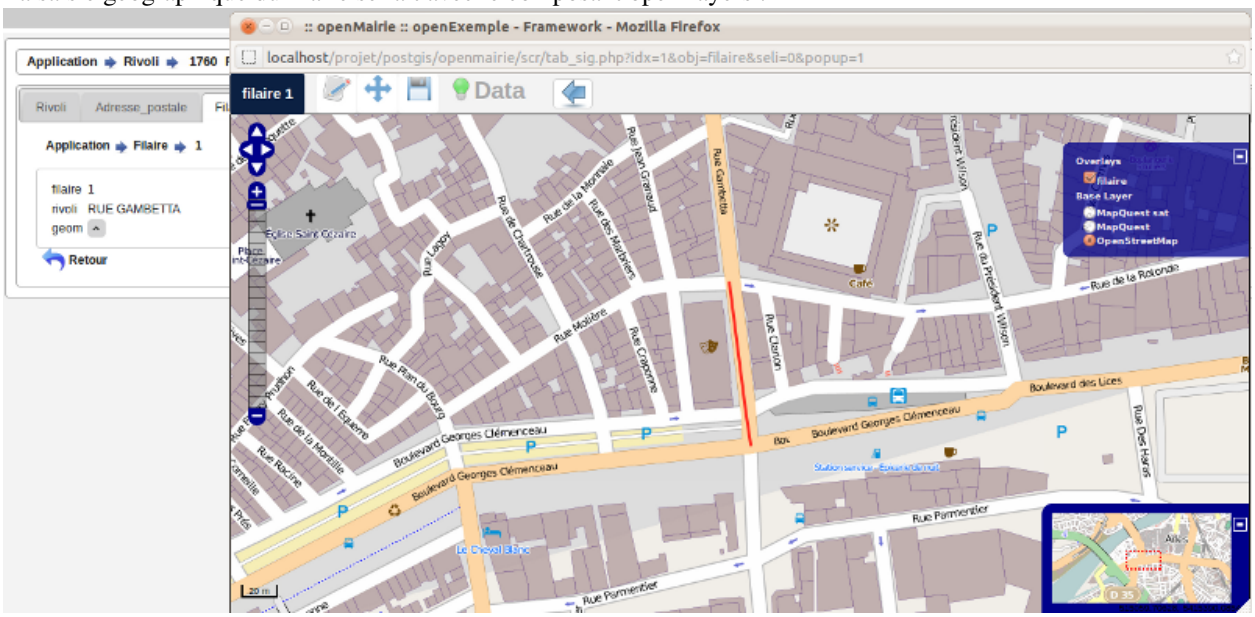


Avec les couches wms :



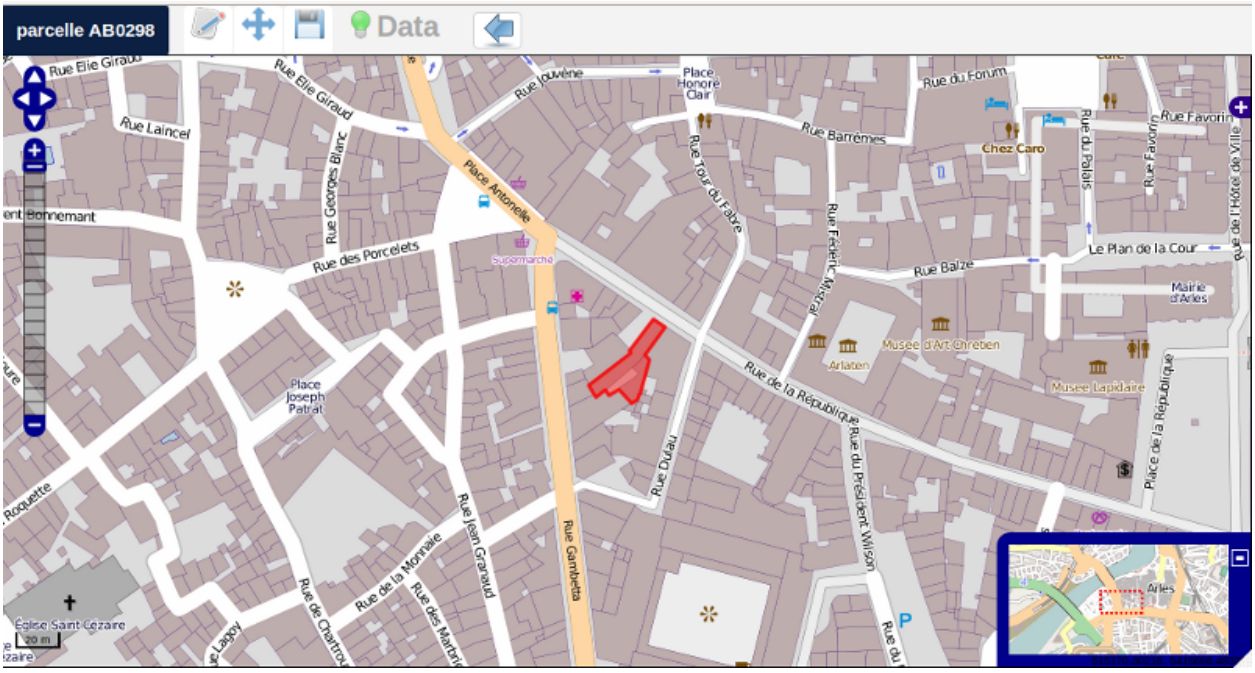
4.4.3 L'onglet filaire :

La saisie géographique du filaire se fait avec le composant openLayers :



4.4.4 L'onglet parcelle :

La saisie géographique du parcelle se fait avec le composant openLayers :



4.5 Les applications openMairie

Il est proposé de décrire les applications openMairie existantes et en cours de développement.

4.5.1 Les applications existantes

openFoncier a été re développé sous postgresql. En effet, openFoncier est utilisé depuis 2007 pour instruire les autorisations du droit des sols. La géolocalisation de l'autorisation se fait de manière automatique soit par le centroïde de la parcelle si elle existe soit sur la base des lots de lotissements en utilisant les couches correspondantes. L'instruction se fait avec une remontée de l'information des couches POS , servitudes surfaciques (zone de préemption par exemple), les servitudes linéaires (voies bruyantes, lignes de chemin de fer, gaz . . .), servitudes ponctuelles (monuments historiques par exemple).

openDomainePublic géolocalise les entreprises qui occupent le domaine public en utilisant les adresses postales et propose une localisation des espaces occupés

openBoisson positionne en utilisant l'adressage postal, les entreprises demandant une licence de débit de boisson. L'utilisation de données géographiques permet de pouvoir calculer automatiquement la distance avec les équipements sensibles. Cette application est utilisée depuis 2011.

OpenCirculation gére les arrêtés de circulation. OpenCirculation utilise le graphe de voirie et les adresses postales pour construire de manière automatique les représentations géographiques des arrêtés : croisement , adresse, entre croisement, entre adresse, secteur ...

OpenTaxePub permet depuis le 1er juillet 2012 de gérer la taxe sur la publicité. Ce logiciel permet de géolocaliser les entreprises et les sites d'implantation des dispositifs publicitaires. Il est utilisé la couche d'adresses postales pour les entreprises. La mise en place du module d'instruction pour l'implantation de nouveaux dispositifs pour le service de l'urbanisme va permettre une utilisation efficace de la couche du règlement de publicité.

OpenTriSelectif permet de gérer la distribution du matériel pour le tri selectif (sacs jaunes et blancs, containers, composteur . . .). Ce logiciel utilise la couche adresses postales pour localiser les usagers et ainsi, l'utilisateur peut vérifier que le demandeur habite bien un logement individuel.

OpenParking utilise l'adressage postal et va permettre d'identifier la géolocalisation des clients et à terme aider à la décision sur le stationnement.

openAdresse a été développé pour gérer le référentiel adresse : rivoli, filaire de voirie, emprise de voirie, adresse postale.

4.5.2 les applications en cours de développement

Nous démarrons une évolution d'openCimetiere actuellement sur une base mysql vers postgresql. L'objectif est d'utiliser les couches cimetières, zones (carré, collines ...) et emplacement sur la base des plans numériques et manuels existants. OpenCimetiere géolocalise les concessions. Le rapprochement des concessions avec la couche des emplacements va permettre de gérer la place libre des cimetières, de voir l'évolution de l'occupation et de fournir des éléments d'aide à la décision.

Il est souhaitable d'intégrer openElec en utilisant la couche adresse postale. En effet, la géolocalisation des électeurs va permettre d'utiliser le SIG pour faire de la simulation de découpage et ainsi équilibrer la répartition des électeurs par bureau.

openGestionCrise a été utilisé pour la première fois en 2012 dans dans le cadre d'une simulation d 'alerte. L'utilisation de ressources géographiques nous pousse à re développer ce logiciel sous postgresql. Il s'agit d'utiliser les couches actuelles du SIG dédiées à la gestion de crise. Le projet n'est pas encore abouti.

Nous avons initialisé le projet openInscription pour les journées de l'emploi de novembre 2012. En 2013, il est proposé de géolocaliser les adresses des participants pour pouvoir faire les études géographiques nécessaires (analyse par quartier, zus ...).

Openvoie devrait gérer la refection des voies et les chantiers de voirie en utilisant le graphe de voirie et les adresses postales.

La ville de Marseille demarre un projet openERP (nom provisoire) pour proposer une gestion des établissement recevant du public et améliore le logiciel openfoncier (marché en cours).

Le service SIG de Vitrolles travaille sur un projet openReference permettant de récupérer les données de l'IGN pour constituer un référentiel municipal.

Liste des applications openMairie opérationnelles et en cours :

Application openMairie	Construction automatique de géométrie	Couches de géolocalisation	Couches SIG
OpenFoncier	Dossier (point ou polygone)	*Adresse postale, Parcelle*	Lotissement, pos et servitudes, PPRI, Hydrants
OpenDomainePublic	Entreprise, Occupation	*Adresse postale*	
OpenBoisson	Licence, périmètre d'exclusion	*Adresse postale*	
OpenCirculation	Arrêtés (point, ligne, polygone), ayant droit (point)	*Adresse postale, filaire de voie*	Quartier, panneaux
OpenTaxePub	Entreprise (point), dispositifs (point)	*Adresse postale*	Règlement de publicité
OpenTriselectif	Usager (point)	*Adresse postale*	
OpenParking	Client (point)	*Adresse postale*	
OpenAdresse	Adresse postale, filaire de voie, emprise de voie		
OpenCimetiere	Concession (point)		Cimetiere, zone, emplacement
OpenElec	Electeur (point)	*Adresse postale*	Bureau, canton
OpenVoie	Travaux	*Adresse postale, filaire de voie*	
OpenGestionCrise	*En cours*		

4.5.3 En guise de conclusion : mobile et openData :

Dans le cadre de la stratégie d'intégration géographique du projet openMairie, le mobile et l'openData constituent les prolongements intéressants de cette expérience.

Le projet openMairie "mobile" commence à prendre forme (future version 4.4.0). Il s'agit de porter les applications existantes en html5 sur mobiles et sur tablettespour permettre une information et une mise à jour des données en mobilité.

projet openMairie avec jquery mobile :

Seules les données géographiques peuvent permettre des rapprochements entre les jeux de données. Ceci souligne l'importance des jeux de données géographiques dans les openData. openMairie souhaite s'insérer dans cette dynamique.

CHAPITRE 5

Indices and tables

- *genindex*
- *modindex*
- *search*
- bibliographie
  http ://www.openmairie.org/telechargement/openMairie-Guidedudveloppeur.pdf/view
  Guide utilisateur QGIS

CHAPITRE 6

Contributeurs

- francois raynaud
- alain baldachino