

Tutoriel PostGreSQL / PostGIS

Introduction

Ce tutoriel a pour but de comprendre le fonctionnement de l'extension spatiale PostGIS du SGBD PostGreSQL dans un environnement Windows.

Il traite sommairement de l'installation de PostgreSQL puis de PostGIS.

Ensuite il aborde le système de données spatiales, par l'intermédiaire de l'interface d'administration PgAdmin de PostGreSQL.

L'exploitation de ces données géographiques par l'intermédiaire du logiciel Quantum GIS et du Serveur cartographique MapServer est décrite.

Enfin, plusieurs techniques d'importation de données géographiques provenant de différents formats de données, dont MapInfo et ArcGIS, sont proposées.

Ce tutoriel se présente sous la forme de notes de cours, suivi d'exercices d'application.

Les notes de cours ne nécessitent pas d'action particulière sur le PC.

Les exercices se présentent sous la forme d'un encadré au format spécifique :

Travail à faire :

Ceci est un exemple d'énoncé d'exercice d'application.

Ce document a été construit à partir de diverses ressources internet dont :

<http://www.01map.com/david/doc/postgis090/ch02s04.html>

et :

<http://www.postgis.fr/book/print/198>

Il a été élaboré et complété par Christian GERMAIN avec l'aide de l'option MSI 2006-2007 de Bordeaux Sciences Agro, puis régulièrement mis à jour.

Sommaire

1.	Installation	3
2.	Création de structures de données	4
3.	Insertion de données	9
4.	Exécution de requêtes	11
5.	Utilisation de QGIS	13
6.	Importation de données.....	16
7.	Utilisation conjointe de PostGIS et MapServer	17
8.	Liens utiles	18

1. Installation

Les différentes versions de PostGreSQL / PostGIS ne sont pas toujours livrées ensemble. Il est donc conseillé de les installer séparément afin de pouvoir profiter des dernières versions.

a. Téléchargement des packages d'installation :

Télécharger les fichiers d'installation. Pour les besoins du tutoriel, la version 9.3 de PostGreSQL et la version 2.1 de PostGIS ont été utilisées.

Le package d'installation de PostGreSQL peut être téléchargé sur le ftp suivant : <http://www.postgresql.org/download/windows/>

Dans l'arborescence des répertoires, choisissez la version que vous voulez. Ces fichiers contiennent aussi l'extension PostGIS.

Si nécessaire, le package d'installation spécifique à PostGIS peut être téléchargé à l'adresse suivante : http://postgis.net/windows_downloads. Cette page vous proposera un exécutable d'installation de la dernière version de PostGIS.

Travail à faire : Procéder aux téléchargements.
--

b. Installation de PostGreSQL

Décompressez le dossier téléchargé et lancer l'exécutable d'installation. Pour la version 9.3, il s'agit du fichier postgresql-9.3.1-1-windows-x64.exe (pour la version 64 bits)

Configuration de l'installation :

- Par défaut, le programme d'installation installera PostGreSQL en tant que service et il créera un nouveau compte Windows caché ayant des droits limités. Pour des raisons de sécurité, c'est grâce à ce compte caché que le service sera activé. Le programme vous demandera de définir les paramètres de ce nouveau compte (nom d'utilisateur et mot de passe).
- Notez bien le nom d'utilisateur et le mot de passe que vous définirez à cette étape, sans quoi vous ne pourrez utiliser PostGreSQL par la suite.
- Lors du choix des fonctionnalités à installer (module Stack Builder), choisissez d'installer PostGIS. Si d'autres choix vous sont proposés, laissez les options par défaut.

c. Installation de PostGIS

Dans le cas où PostGIS n'aurait pas été installé sur une version existante de PostGreSQL Lancez l'exécutable téléchargé précédemment. (Pour notre exemple postgis-bundle-pg93x64-setup-2.1.1-1.exe). Suivez les instructions d'installation en renseignant lorsque cela vous est demandé les noms d'utilisateur et mot de passe définis comme compte d'utilisation de PostGreSQL.

2. Création de structures de données

Dans cette partie, vous allez apprendre à utiliser le logiciel PgAdmin, qui offre une interface graphique pour gérer vos bases de données PostGreSQL/PostGIS.

a. Le logiciel PgAdmin

Le programme d'installation de PostGreSQL a installé PgAdminIII sur votre système. Vous devriez pouvoir y accéder depuis votre menu Démarrer.

La fenêtre de PgAdmin se décompose en quatre parties :

- En haut, une barre de menus et une barre de raccourcis
- A gauche, l'arborescence de votre serveur PostGreSQL
- En haut à droite, le contenu des objets sélectionnés à gauche
- En bas à droite, la définition SQL de l'objet sélectionné

Vous pouvez alors vous connecter à votre serveur PostGreSQL en double cliquant sur son nom dans la fenêtre de gauche et en introduisant votre mot de passe.

**Travail à faire : Démarrer PgAdmin. Se connecter au serveur.
Vérifier l'installation de PostGreSQL / PostGIS**

Votre serveur se décompose en plusieurs parties (voir figure 1) :

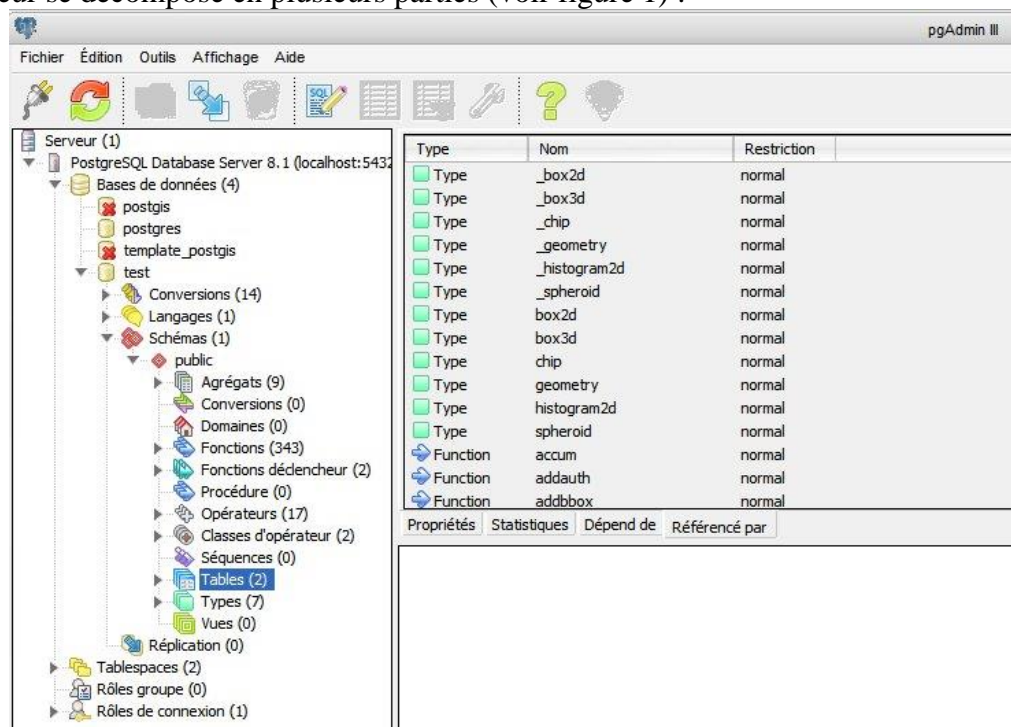


Figure 1 : Aperçu général de PgAdmin

- Bases de données : Contient vos bases de données PostGreSQL/PostGIS. Initialement, ce répertoire contient 2 ou 3 bases de données :
 - postgres : base de données standard PostGreSQL
 - postgis-21_sample : base de données modèle pour PostGIS


En navigant dans l'arborescence, vous trouverez tout le contenu de votre base. Les constituants usuels d'une base de données (tables, vues, procédures) se trouvent dans l'arborescence « schémas ».

- Tablespaces : Espaces de stockage de vos données. Ils définissent des emplacements où seront stockés les fichiers contenant les données.
- Rôles groupe et Rôles de connexion : définissent des comptes d'utilisation de PostgreSQL ou des groupements de comptes sous un profil commun.

b. Créer une base de données

Pour créer une nouvelle base de données, il faut vous positionner dans l'arborescence sur le répertoire Bases de données.

La création peut ensuite se faire de trois façons :

- Clic droit puis Ajouter une base de données
- Menu Edition → Ajouter un objet → Ajouter une base de données
- Bouton 'Créer un objet de même type que celui de l'objet sélectionnée' 

Quelle que soit la méthode retenue, une fenêtre s'affiche et permet de définir les paramètres de la base :

- Onglet Propriétés : définissez le nom de la base, son propriétaire (au choix parmi les comptes utilisateurs créés), son encodage (par défaut ASCII), son modèle (la base créée reprendra le schéma de son modèle), son espace de stockage (tablespace) et ajouter si besoin un commentaire.

Pour créer une table pouvant contenir des objets géographiques, prenez comme modèle la base postgis_21_sample (Cette base ne doit pas être modifiée car elle sert de modèle).

Il se peut qu'en choisissant le modèle, PgAdmin vous signale une erreur de cette forme :



Figure 2 : Erreur de création de base

Il vous faut dans ce cas vous déconnecter de la base postgis_21_sample. Pensez à fermer toutes les fenêtres en rapport avec cette base (requêtes, affichage du contenu...). Si nécessaire placez vous sur la racine de vos bases de données et choisissez le raccourci 'Actualiser'. Attention : La base sera reconnectée dès que vous la sélectionnerez à nouveau.

Dans certains cas rares (notamment lorsque l'on vient de créer manuellement la base modèle « TemplatePostGIS », des processus résiduels peuvent empêcher la création de la base test à partir du modèle. Il suffit de quitter PgAdmin et de le relancer pour régler ce problème.

- Onglet Droits : définissez les droits d'utilisation de la base en fonction des comptes d'utilisateurs.

Remarque sur les clés étrangères.

Si une clé étrangère est nécessaire, la table contenant le champ clé étrangère doit être créée avant la table où cette clé étrangère est appelée. Par exemple, si un champ type_d_objet devait être créé dans une table objet, une table type contenant les champs identifiant et libellé devrait être créée, puis l'identifiant de cette table devrait être appelé dans la table objet. La syntaxe pour créer une clé étrangère, à écrire dans la liste des champs lors de la création de la table (ici table objet), est la suivante :

Type_d_objet integer references type (identifiant) ;

Travail à faire : Créez une base de données nommée 'test' permettant de saisir des données géographiques (codage UTF8 ; Modèle postgis_21_sample).

c. Création de tables

Pour créer une ou des tables dans votre base de données, procédez de la même manière, mais en vous plaçant cette fois dans le schéma de votre base de données, sur le répertoire Tables.

De la même façon, vous pouvez définir les propriétés de la table (nom, tablespace... dans l'onglet Propriétés), ses champs (onglet Colonnes), ses contraintes (onglet Contraintes), les droits d'utilisation (onglet Droits) et voir la requête SQL qui lui sera associée (onglet SQL).

La définition des clés (primaires et étrangères) se fait dans l'onglet Contraintes.

Pour créer un champ qui contiendra une donnée géographique, choisir le type de données 'geometry' (ou 'public.geometry' selon les versions de PostGIS).

Travail à faire : Dans la base 'test', créez les tables suivantes

Indiquez que les colonnes id_pieton, id_route, id_batiment sont les clés primaires des 3 tables.

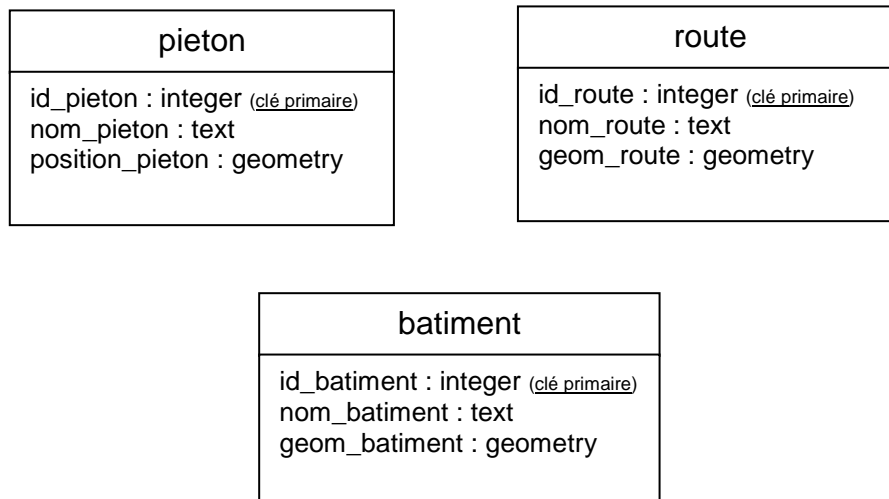


Figure 3 : Diagramme de classes

Remarques sur le type « geometry »

Le type de champ « geometry » permet de stocker n'importe quel type d'objet géographique (point, ligne polygone).

PostGIS permet de stocker des objets géographiques de type différents dans une même colonne de type « geometry ». Toutefois, pour construire une base de données bien structurée, il est strictement recommandé de distinguer dans des tables différentes des objets dont la géométrie est différente.

Dans notre exemple, les routes sont des lignes, les piétons des points et les bâtiments des polygones.

d. Exécuter des requêtes SQL

Dans la fenêtre principale de PgAdmin, vous pouvez exécuter vos propres requêtes SQL en utilisant

le bouton 'Exécuter vos propres requêtes SQL'



. Dans la fenêtre affichée, vous pouvez

paramétrer vos requêtes et les exécuter.

Attention, les requêtes seront exécutées dans la base renseignée en haut de la fenêtre (voir figure 4). Toutes les requêtes standard SQL peuvent être utilisées.

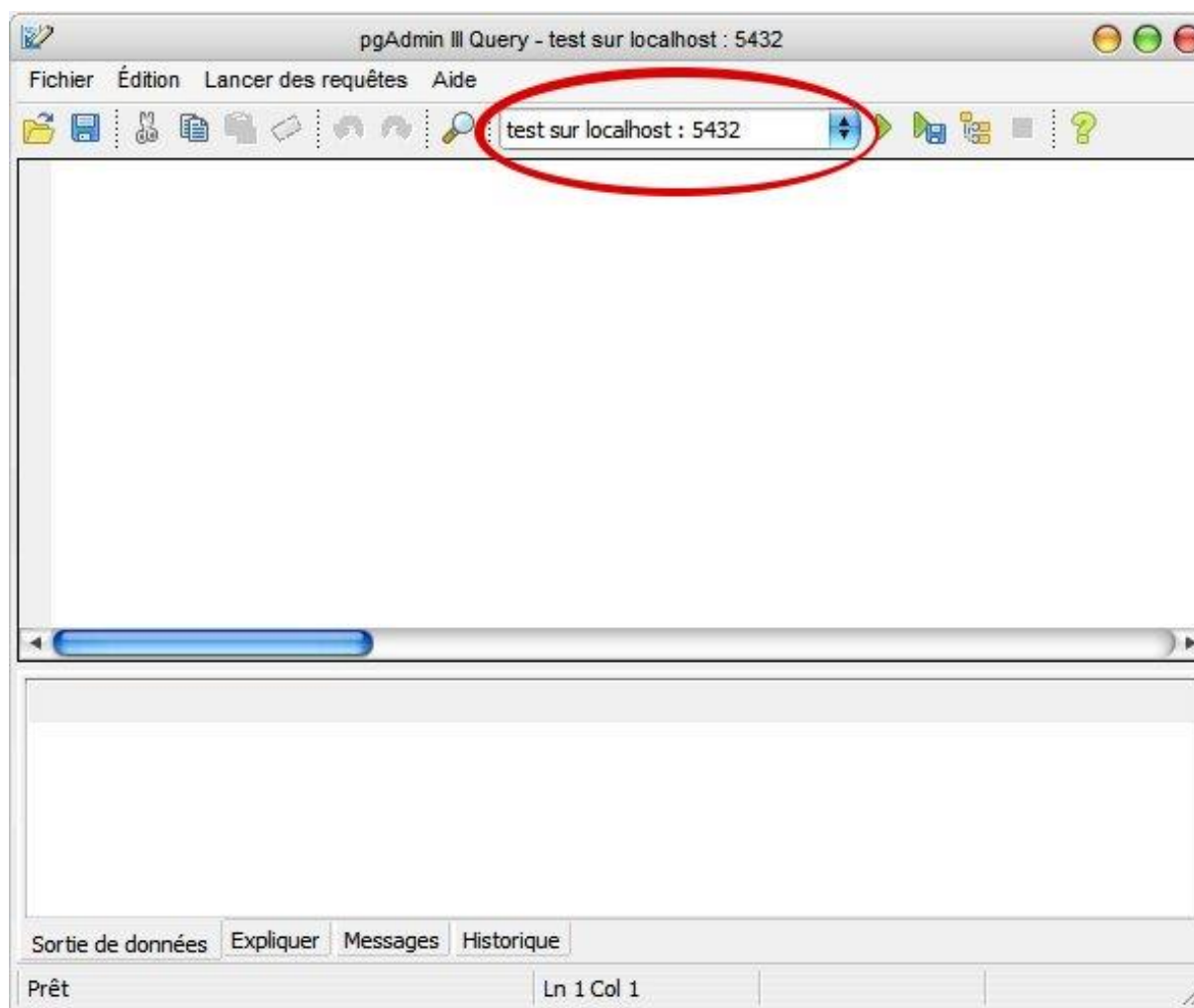


Figure 4 : Fenêtre de requêtes SQL

Nous verrons par la suite comment utiliser cette fenêtre.

e. Transformer une base PostGreSQL en base géographique PostGIS

Si vous avez créé une base en ne précisant pas le modèle, il se peut que la base créée ne soit pas au format géographique. Les formats de données et fonctions géographiques ne sont alors pas connus, donc inutilisables.

Pour transformer une base PostGreSQL en base géographique, il faut utiliser les sources SQL situées dans les répertoires d'installation de PostGreSQL (par défaut C:\Program Files\PostgreSQL\9.3\share\contrib\postgis-2.1).

Dans la base à compléter, ouvrez un par un les fichiers **postgis.sql**, **spatial_ref_sys.sql**, **postgis_comments.sql**, **rtpostgis.sql**, **raster_comments.sql**, **topology.sql**, **legacy.sql** dans une fenêtre SQL et exécutez-les dans cet ordre. Ces scripts vont ajouter des tables à votre base (dont spatial_ref_sys), ainsi que les fonctions spatiales et les vues (dont geometry_columns). Dès lors que ces éléments existent dans votre base, elle est prête à recevoir des données géographiques.

C'est la même méthode qu'il faut utiliser si la base de données PostGisSample n'a pas été créée lors de l'installation de PostGIS !

3. Insertion de données

a. Comment insérer des données

Pour insérer des données dans les tables créées, vous pouvez :

- Soit faire un clic droit sur la table à remplir et choisir l'option 'Afficher les données'. La fenêtre qui s'ouvre vous propose un aperçu de la table, dans laquelle vous pouvez saisir les données que vous voulez insérer.
- Soit procéder directement par des instructions SQL (INSERT).

b. Données de type geometry

Dans le cas de données géographiques (champs de type geometry), il faut utiliser des fonctions PostGIS : les champs « geometry » sont impossibles à paramétrer manuellement.

3 fonctions correspondent aux « points », « lignes » et « polygones » (PostGIS peut utiliser d'autres types d'objets comme multipoints par exemple, nous les rencontrerons plus tard) :

`st_GeometryFromText(<TYPE>(<val 1>, <val 2>, <val 3>,...), <X>)`

<TYPE> représente le type de géométrie (POINT, POLYGON, LINESTRING ...)

<val i>. sont des couples de coordonnées définissant la forme des objets. Un point sera défini par un seul couple, une ligne ou un polygone par un couple pour chaque sommet de la ligne (sommets reliés par des segments de droites).

<X> représente le système de projection utilisé pour la représentation. En renseignant -1, on indique qu'aucun système n'est utilisé.

Pour connaître l'identifiant de la projection que vous voulez utiliser, il vous faut regarder le contenu de la table *spatial_ref_sys*, qui doit faire partie des tables de votre base de données. L'identifiant que vous recherchez est la colonne *srid*, et le nom de la référence se trouve dans la colonne *srtext*. Par exemple, les projections Lambert I, II et III correspondent respectivement aux srid 27571, 27572 et 27573.

Travail à faire : Exécuter les requêtes suivantes : *Un script est disponible !*

```
INSERT INTO pieton (id_pieton, nom_pieton, position_pieton) VALUES (1, 'Albert',
st_GeometryFromText('POINT(300010 2200070)', 27572));

INSERT INTO pieton (id_pieton, nom_pieton, position_pieton) VALUES (2, 'René',
st_GeometryFromText('POINT(300030 2200030)', 27572));

INSERT INTO pieton (id_pieton, nom_pieton, position_pieton) VALUES (3, 'Jean',
st_GeometryFromText('POINT(300035 2200070)', 27572));

INSERT INTO pieton (id_pieton, nom_pieton, position_pieton) VALUES (4, 'Marie',
st_GeometryFromText('POINT(300035 2200060)', 27572));

INSERT INTO batiment (id_batiment, nom_batiment, geom_batiment) VALUES (1, 'Hopital',
st_GeometryFromText('POLYGON((300010 2200010,300040 2200020,300035 2200008,300012
2200004,300010 2200010))', 27572));

INSERT INTO batiment (id_batiment, nom_batiment, geom_batiment) VALUES (2, 'Bar',
st_GeometryFromText('POLYGON((300010 2200040,300020 2200030,300030 2200040,300040
2200035,300050 2200060,300035 2200080,300020 2200060,300010 2200040))', 27572));

INSERT INTO batiment (id_batiment, nom_batiment, geom_batiment) VALUES (3, 'Mairie',
st_GeometryFromText('POLYGON((300010 2200095,300020 2200095,300020 2200135,300010
2200135,300010 2200095))', 27572 ));

INSERT INTO route (id_route, nom_route, geom_route) VALUES (1, 'Route du test',
st_GeometryFromText('LINESTRING(300001 2200092,300050 2200092,300058 2200103)', 27572 ));
```

Il faut maintenant compléter la vue geometry_columns, pour spécifier certaines métadonnées de la couche. Les champs à compléter sont :

- f_table_catalog (le nom de la base. Il est déjà à jour)
- f_table_schema (en général public. Il est déjà à jour)
- f_table_name (le nom de la table. Il est déjà à jour)
- f_geometry_column (le nom de la colonne de type geometry. Il est déjà à jour)
- coord_dimension (2 pour 2D, 3 pour 3D. Il est déjà à jour)
- srid (le numéro du système géodésique ; IL FAUT LE CORRIGER)
- type (en fonction du type de la couche, POINT, POLYGON ; IL FAUT LE CORRIGER).

Hélas, la modification directe n'est pas possible.

On utilisera une commande SQL pour modifier la structure de chaque table (et qui mettra à jour du même coup la vue). La syntaxe est :

```
Alter table <nom de la table>
alter column <nom de la colonne de type geometry>
type geometry (<type d'objet>, <numéro de srid>)
using st_setsrid(<nom de la colonne de type geometry>, <numéro de srid>);
```

Les types (2D) les plus courants sont : POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION. Mais attention, tous ces types ne sont pas acceptés par toutes les fonctions, ni par tous les clients SIG.

Pour en savoir plus les types de données : http://postgis.net/docs/using_postgis_dbmanagement.html

Travail à faire : Exécuter cette instruction pour chaque table. Vérifier ensuite la vue. Vous devez obtenir

	f_table_catalog character varying(255)	f_table_schema character varying(255)	f_table_name character varying(255)	f_geometry_column character varying(255)	coord_dimension integer	srid integer	type character varying(255)
1	test	public	batiment	geom_batiment	2	27572	POLYGON
2	test	public	route	geom_route	2	27572	LINESTRING
3	test	public	pieton	position_pieton	2	27572	POINT

Pour une conception plus rigoureuse, il peut être utile d'ajouter à chaque table géographique des contraintes de vérification pour garantir que le type, le SRID et la dimension des données restent cohérents durant toute la vie de la table.

Pour ajouter une contrainte à une table, faire clic droit sur la rubrique « Contraintes » de la table concernée et « Ajouter un objet » puis « Ajouter une contrainte de vérification ». Donner un nom à la contrainte dans l'onglet « nom ». Décrire les contraintes dans l'onglet « Définition ».

Contrainte qui vérifie la projection des objets ajoutés à la table :

```
st_srid(<col. geometry>) = <n° du SRID>
```

Contrainte qui vérifie que les objets ajoutés sont du bon type :

```
st_geometrytype(<col. geometry>) = '<type>::text' OR <col. geometry> IS NULL
```

(ici <type> sera remplacé par ST_Point ST_Polygon ou ST_LineString)

Contrainte qui vérifie que les coordonnées saisies sont en 2D :

```
st_ndims(<col. geometry>) = 2
```

Travail à faire : Ajouter ces 3 contraintes à chacune des tables « pieton » « route » et « batiment ».

Pour afficher les données contenues dans une table, il suffit de faire un clic droit sur la table et de choisir l'option «Afficher les données ».

Travail à faire : Vérifier votre saisie en affichant le contenu des tables « pieton » « route » et « batiment ».
Explorer également la table des projections (spatial_ref_sys).

4. Exécution de requêtes

a. Requêtes classiques

Toutes les requêtes standard SQL peuvent être utilisées par l'intermédiaire de PgAdmin.

Afin de tester l'existence des données précédemment saisies, essayez les requêtes suivantes.

Travail à faire :

Sélection de toutes les informations

```
SELECT * FROM piéton ;  
SELECT * FROM batiment ;  
SELECT * FROM route;
```

Vérification d'existence d'Albert

```
SELECT * FROM piéton WHERE nom_pieton='Albert';
```

b. Requêtes géographiques

Les requêtes géographiques permettent d'interroger le moteur géographique de PostGIS en lui fournissant des paramètres sur les champs de type 'geometry'. Ces requêtes font appel à des fonctions spécifiques. Nous allons développer quelques exemples de fonctions géographiques simples et utiles. Dans les fonctions suivantes, les termes *geom*, *geom1*, *geom2*... représentent des champs de type 'geometry'. Chaque fonction est illustrée d'un exemple appliqué sur la base que nous avons précédemment créée.

- **st_area2d(geom)** : renvoie l'aire définie par le paramètre de géométrie de l'objet géographique. L'unité dépend du système géodésique (en général m²).
- **st_distance(geom1, geom2)** : renvoie la distance séparant deux objets. L'unité dépend du système géodésique (en général m).
- **st_astext(geom)** : renvoie sous un format compréhensible les paramètres de l'objet. D'autres fonctions existent pour afficher les informations de façon compréhensible, comme **st_asewkt(geom)**. De la même façon, **st_asKML(geom)** convertit *geom* au format KML (utile pour Google Earth). Si on ajoute les en-têtes appropriés au résultat de la requête, on peut construire un fichier KML « à la volée »
- **st_srid(geom)** : renvoie l'identifiant de projection utilisé pour cet objet.
- **st_geometryType()** : renvoie le type d'un objet : ST_Point, ST_Polygon, ST_LineString
- **st_numpoints(geom)** : renvoie le nombre de points composant un objet *linéaire*.
st_npoints(geom) : renvoie le nombre de points composant un objet *polygone*.
- **st_centroid(geom)** : renvoie le centre d'un polygone (le résultat se présente sous la forme d'une suite de caractères alphanumériques définissant le point, le résultat tel quel est donc difficilement exploitable). Pour l'affichage, il vaut donc mieux combiner cette fonction avec **st_astext : st_astext(st_centroid(geom))**.
- **st_pointonsurface(geom)** : renvoie un point situé à l'intérieur d'un polygone. Ce point peut être plus intéressant que le centroïde, car ce dernier n'est pas toujours situé à l'intérieur du polygone (cas des polygones en « U » par exemple).
- **st_startpoint(geom)** et **st_endpoint(geom)** : renvoient respectivement les points de départ et de fin d'une forme géométrique.
- **st_contains(geom1, geom2)** : renvoie une valeur booléenne ('t' ou 'f') indiquant si l'objet *geom1* contient l'objet *geom2*. **st_within(geom1,geom2)** : renvoie une valeur booléenne ('t' ou 'f') indiquant si l'objet *geom1* est contenu dans l'objet *geom2*.
- **st_intersects(geom1, geom2)** : renvoie une valeur booléenne ('t' ou 'f') indiquant si l'objet *geom1* est en intersection avec l'objet *geom2*. A ne pas confondre avec **st_intersection**

(geom1, geom2) qui construit un nouvel objet dont le périmètre est la zone d'intersection entre geom1 et geom2.

- **st_buffer(geom, n)** : renvoie un objet (tampon) recouvrant tous les points situés à une distance $\leq n$ de l'objet **geom**. L'unité de **n** dépend du système géodésique.
- **st_extent(geom_set)** : fonction qui calcule l'emprise géographique¹ totale de l'ensemble des objets d'une table. Opère comme un *sum* : on peut y associer un « *group by* ».
- **box2D(geom), box3D(geom)** : calcule l'emprise géographique individuels des objets d'une table (sans regroupement, donc objet par objet).
- **st_Xmin, st_Xmax, st_Ymin, st_Ymax (box2D)** : calcule les coordonnées d'un rectangle.
- **Opérateur &&** : vérifie si les boîtes englobantes des objets sont en intersection. Exemple : *a.geom && b.geom* est vrai si le rectangle englobant la géométrie de *a* intersecte le rectangle englobant la géométrie de *b*. Cet opérateur exploite l'index spatial (GIST) s'il existe, et accélère considérablement les requêtes (voir section 9).
- **st_transform(geom, srid)** : transforme une géométrie dans un autre système géodésique.

Travail à faire : répondre aux questions suivantes.

- **Afficher l'aire de chaque bâtiment :**
Hopital=228 ; Bar=1050 ; Mairie=400
- **Afficher la distance de chaque piéton par rapport à la route.**
"Albert"=22 ; René= 62 ; Jean=22 ; Marie=32
- **Afficher les paramètres géographiques de tous les bâtiments.**
POLYGON((300010 2200010,... 300010 2200010))
POLYGON((300010 2200040,... 300010 2200040))
POLYGON((300010 2200095,... 300010 2200095))
- **Afficher les paramètres des centres de chaque bâtiment.**
POINT(300025.34502924 2200010.95906433)
POINT(300030.952380952 2200053.0952381)
POINT(300015 2200115)
- **Afficher le nombre de sommets composant la route.**
3
- **Afficher les coordonnées du point de départ de la route.**
POINT(300001 2200092)
- **Pour chaque bâtiment, afficher les coordonnées d'un point situé (avec certitude) à l'intérieur de ce bâtiment.**
POINT(300031.458 2200015), POINT(300030.5 2200050) POINT(300015 2200115)
- **Afficher les piétons se trouvant dans un bâtiment quelconque**
(Jean et Marie dans le bâtiment 2).
- **Afficher les piétons se trouvant dans « bâtiment 2 ».**
(Jean et Marie dans le bâtiment 2).
- **Afficher les piétons situés à moins de 10 m de «bâtiment 2».**
René, Jean et Marie
- **Afficher les coordonnées de l'emprise de la table «bâtiment».**
300010;2200004;300050;2200135
- **Afficher les coordonnées de l'emprise de chaque «bâtiment».**
300010;2200004;300040;2200020
300010;2200030;300050;2200080
300010;2200095;300020;2200135

¹ L'emprise désigne les limites du plus petit rectangle (horizontal ou vertical) englobant le (ou les) objet(s).

5. Utilisation de QGIS

QGIS est un logiciel de cartographie qui permet la saisie et la consultation de données cartographiques. Ce tutoriel a été conçu avec la version 2.0.1 du logiciel. L'installation du logiciel se fait de façon automatique en téléchargeant le package d'installation disponible sur le site <http://www.qgis.org>.

Travail à faire : Procéder au téléchargement et à l'installation de QGIS.

a. Présentation du logiciel

QGIS fonctionne sous forme de projets, contenant des ensembles de couches cartographiques. QGIS ne permet pas le traitement de données ou la formulation de requêtes, mais autorise par contre la consultation, la saisie et la mise à jour des données géographiques de chaque couche ainsi que les informations sémantiques qui leur sont liées. Il permet la gestion de données contenues dans des fichiers (couches « vecteur ») de type MapInfo, Shapefile, Spatial Data Transfert Standard et Geography Markup Language, ainsi que des données serveur de type PostGIS et des données images (couches raster).

Le menu principal de QGIS se présente de cette façon :

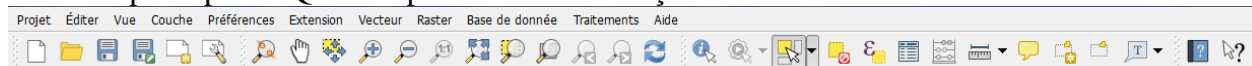


Figure 5 : Menu principal QGIS

Il est composé de 4 parties principales:

Gestion du projet : Créer, ouvrir, sauvegarder ou gérer les impressions.



Gestion des couches : Ouvrir une couche PostGIS, une couche vecteur, une couche raster, visualiser les couches et l'aperçu.



Gestion de l'image affichée : Boutons de manipulation de la couche affichée : zoom, déplacement.



Gestion des données : Afficher les données liées à un objet, sélectionner des objets, mesurer || Saisir ou éditer des données || Afficher les données sémantiques liées à la couche sélectionnée.



Édition : Ajouter, modifier, supprimer des entités géographiques...



b. Manipulation de données PostGIS

Pour insérer une couche de données PostGIS dans QGIS, sélectionnez l'option « Ajouter une couche PostGIS ». Dans la fenêtre de dialogue qui s'ouvre, configurez la connexion à votre serveur PostgreSQL selon les paramètres de votre serveur. En choisissant l'option Nouveau. Le nom qui vous est demandé est le nom sous lequel apparaîtra cette connexion dans la liste. Par défaut, le nom du serveur est *localhost*, les autres paramètres sont le nom de la base (dans notre exemple « test ») et vos noms et mot de passe de connexion à votre serveur PostgreSQL.

Une fois votre connexion paramétrée, revenez à la fenêtre principale, sélectionnez votre connexion et cliquez sur connecter. La liste des couches disponibles s'affiche alors. Chaque table de PostGIS représente ici une couche de données. Les logiciels cartographiques tels que QGIS ne peuvent pas traiter au sein de la même couche des données de formats différents, ce qui conforte le conseil de rigueur que nous avons formulé au 2.c. : Pour construire une base de données structurée, il est préférable de distinguer dans des tables différentes des objets dont la géométrie est différente.

Une fois la base sélectionnée, sélectionner les couches à ajouter au projet et cliquez sur ajouter. Vous devez obtenir l'image de la figure 6 ci-dessous.

La plupart des fonctionnalités de QGIS sont intuitives, et la légende des différents boutons permet de comprendre facilement leur fonctionnement. Nous détaillerons donc simplement comment saisir ou modifier des données.

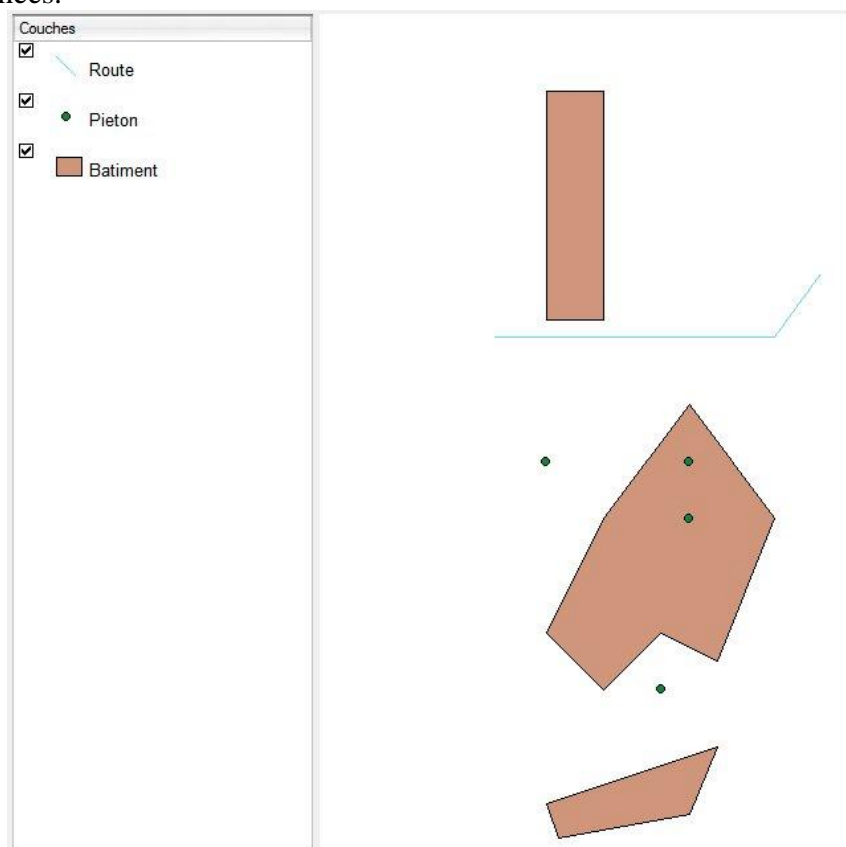


Figure 6 : Vue sous QGIS de la base «test »

Travail à faire :

Se connecter à la base PostGIS « test ».

Afficher les couches de la base « test »

c. Modification de couches

QGIS ne peut saisir ou modifier de données que sur une couche à la fois. Il faut tout d'abord activer l'édition de la couche voulue, procéder aux manipulations, puis désactiver l'édition de la couche afin de sauvegarder les données sur le serveur.

Pour activer le mode édition, il faut faire un clic droit sur la couche à modifier et sélectionner l'option « Basculer en mode édition ».

Une fois l'édition commencée, vous pouvez utiliser les différents boutons d'édition de données pour saisir ou modifier des objets et leurs attributs sémantiques.

Pour terminer l'édition, faire à nouveau un clic droit sur la couche et choisir à nouveau l'option « Basculer en mode édition ». Une fenêtre vous demande alors si vous voulez sauvegarder les données.

Travail à faire :

**Avec QGIS, ajouter un bâtiment et un piéton dans la base PostGIS
« test ».**

Avec PGAdmin, vérifier la mise à jour de la base.

6. Importation de données

a. Importer des données grâce à QGIS

QGIS possède un outil permettant d'importer des données depuis des fichiers de type shapefile. Pour utiliser cet outil, il faut se rendre dans le menu Bases de Données -> Spat -> Importer des shapefiles dans PostgreSQL.

Ce menu ouvre une fenêtre permettant de transformer automatiquement des fichiers de format shapefile (et peut-être Mapinfo) en table de données PostGIS.

Cette fonctionnalité semble encore instable, car elle ne fonctionne pas sur tous les fichiers. Ainsi, certains fichiers risquent de ne pas être convertis par cette méthode.

b. Importer des données grâce aux modules PostGIS

1. Import des fichiers vers PostGIS via une appli de postGIS

PostGIS propose un programme « shp2pgsql-gui.exe » permettant d'importer des fichiers au format « shapefile ». Pour trouver cet outil, il faut se rendre dans le dossier :

\\Program Files\\PostgreSQL\\11\\bin\\postgisgui
(Remplacer le chemin d'accès à PostGIS selon votre numéro de version)

Il faut ensuite :

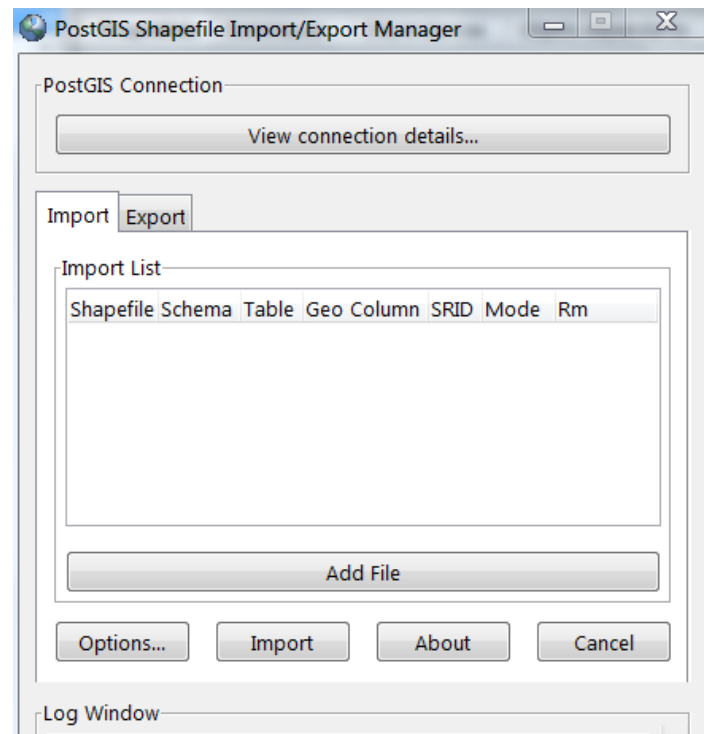
- 1- Sélectionner le ou les fichiers à charger (bouton « Add File »).
- 2- Pour chaque fichier, corriger si nécessaire, le nom de la table, le nom de la colonne *geometry*, et surtout le SRID.
- 3- Vérifier les options (bouton « Options »).
- 4- Appuyer sur le bouton « Import ».

Les options par défaut conviennent généralement. Pour plus de détail sur les options, voir ci-dessous. En cas d'erreur d'importation, une fenêtre décrit les difficultés rencontrées.

ATTENTION :

L'importation échouera lorsque le nom d'un fichier OU le nom de l'un des dossiers du chemin d'accès contiennent des caractères accentués !

Il est prudent de ne pas cocher l'option « Preserve case of column names ». Il est préférable que les noms de colonnes soient en minuscules, pour assurer la compatibilité avec le serveur cartographique MapServer.



Travail à faire :

**Créer une base de données géographique nommée Luchey. Tester l'import de Shapefile dans cette base avec les couches LucheyParc LucheySol avec le srid 27573 (Lambert3)
Vérifier le résultat avec pgAdmin puis avec QGIS.**

7. Utilisation conjointe de PostGIS et MapServer

MapServer est un serveur cartographique Internet qui se conforme aux spécifications "Web Mapping Server" de l'OpenGIS. Ce chapitre suppose qu'un serveur MapServer est disponible et opérationnel. Pour comprendre la syntaxe générale des fichiers MapServer, se reporter à la documentation MapServer.

Ajouter une couche PostGIS dans un MapFile (MapServer)

MapServer accède aux données PostgreSQL/PostGIS comme n'importe quel autre client PostgreSQL. Dans votre Mapfile (fichier de description des cartes MapServer), ajoutez une couche PostGIS.:

```
LAYER                                     # Début de la définition du premier calque
  NAME          luchey                   # Nom du calque

  CONNECTIONTYPE postgis
  CONNECTION "user=postgres dbname=Luchey host=localhost password=stagiaire port=5432"
  DATA "geom from lucheyparc"           # Table postGIS (les données du calque)

  STATUS        DEFAULT                  # Affiche automatiquement la couche
  TYPE          POLYGON                  # Type des données portées par le calque
  CLASS
    NAME        "Carte du Luchey"        # Nom de la classe
    STYLE
      COLOR      232 232 232             # Couleur du fond
      OUTLINECOLOR 32 32 32              # Couleur des traits
    END
  END                                     # Fin de la définition des styles
END                                     # Fin de la définition de la classe
END                                     # Fin de la définition du calque
```

Dans l'exemple ci-dessus, les directives spécifiques à PostGIS sont les suivantes :

CONNECTIONTYPE

Pour les couches PostGIS, cela sera toujours "postgis".

CONNECTION

La connexion à la base de données est gouvernée par "la chaîne de caractères de connexion" ("connection string") qui est constituée d'un ensemble de couples nom/valeur comme ceci (avec la valeur par défaut entre < >) :

```
user=<nom_d_utilisateur> password=<mot_de_passe> dbname=<nom_de_la_base_de_données>
host=<serveur> port=<5432>
```

DATA

La forme de ce paramètre est : "<colonne> from <nom_de_la_table>" où la colonne est une colonne spatiale qui doit être affiché sur la carte.

Travail à faire :

Tester la carte MapServer (MapFile) utilisant les couches de type « shapefile ». Modifier le MapFile afin d'exploiter des couches PostGIS.

8. Liens utiles

Documentation détaillée en français sur PostgreSQL :

<http://www.postgresql.org/docs/manuals/>

Site officiel français de PostgreSQL:

<http://www.postgresqlfr.org>

Tutoriel officiel d'installation de PostgreSQL :

<http://pginstaller.projects.postgresql.org>

Tutoriel de découverte de PostGIS :

<http://old.postgis.fr/documentation/win32/html/ch05.html>

Forum SIG

<http://www.forumsig.org/>.

PHP / PostgreSQL

<http://www.manuelphp.com/php/ref.pgsql.php>

Page d'accueil de MapServer :

<http://mapserver.org/>

Spécifications "Web Map Server" de l'OpenGIS :

<http://www.opengeospatial.org/standards>

Source exploitée dans le chapitre « Utilisation conjointe de PostGIS et MapServer » :

<http://old.postgis.fr/book/print/198>

Gestion de données Spatiales avec PostGIS (en anglais)

http://www.mapbender.org/presentations/Spatial_Data_Management_Arnulf_Christl/Spatial_Data_Management_Arnulf_Christl.pdf

Documentation en anglais : The PostGIS WorkShop

<http://2007.foss4g.org/workshops/W-04/PostGIS%20Workshop.doc>

Syntaxe WKB (Well Known Binary) utilisée pour décrire les points, lignes et polygones

<http://www.gaia-gis.it/spatialite-3.0.0-BETA/spatialite-cookbook-fr/html/wkt-wkb.html>

(Sites visités le 1er Avril 2012)