

Guide de l'utilisateur de PostgreSQL/PostGIS

INDEXATION DU DOCUMENT

	<i>TITRE :</i>	<i>REFERENCE :</i>	
Guide de l'utilisateur de PostgreSQL/PostGIS			
<i>ACTION</i>	<i>NOM</i>	<i>DATE</i>	<i>SIGNATURE</i>
RÉDIGÉ PAR	TECHER David	3 juin 2007	

SUIVI DU DOCUMENT

INDICE	DATE	MODIFICATIONS	NOM
11.5.	2007-06-01	annexe "PgRouting pour le calcul d'itinéraire" - ajout d'une section pour la fonctionnalité shortest_path(). Suppression de la section PgDijkstra dans le chapitre Tutoriaux, reprise et adaptée dans cet annexe.	TECHER David
11.4.	2007-05-06	annexe "PgRouting pour le calcul d'itinéraire" - ajout des commandes d'installation de PgRouting sur Mac OS X	RIBOT Nicolas
11.3.	2007-05-06	annexe "PgRouting pour le calcul d'itinéraire" - ajout d'un section pour les fonctions shortest_path_astar_as_geometry_internal_id_directed() et tsp_astar_as_geometry_internal_id_directed()	TECHER David
11.2.	2007-05-02	annexe "PgRouting pour le calcul d'itinéraire" - ajout pour la fonction tsp()	TECHER David
11.1.	2007-04-23	Ajout de l'annexe "Dblink: interroger plusieurs serveurs PostgreSQL distants"	TECHER David
11.	2007-04-02	Mise à jour: 1. Avant-propos: Ajust d'un exemple concernant PgRouting 2. Annexes: Ajust de l'annexe "PgRouting pour le calcul d'itinéraire"	TECHER David

SUIVI DU DOCUMENT

INDICE	DATE	MODIFICATIONS	NOM
10.1	2007-01-27	Mise à jour du document pour PostGIS 1.2.0: <ol style="list-style-type: none">1. Chapitre 1: Corrections des versions concernant MinGW et Msys pour PostgreSQL 8.2.1:2. Chapitre 2: Section "Erreurs reportées par les utilisateurs" Ajout de l'erreur reportée par Hervé QUINQUENEL3. Chapitre 5 Ajout de la sous-section "Exemples d'objets géométriques en WKT"	TECHER David

SUIVI DU DOCUMENT

INDICE	DATE	MODIFICATIONS	NOM
10	2007-01-21	<p>Réorganisation du document pour GNU/Linux et pour</p> <ol style="list-style-type: none">1. PostgreSLQ 8.2.12. PostGIS 1.2.03. Geos 2.2.34. Proj 4.5.0 <p>Modifications majeurs</p> <ol style="list-style-type: none">1. Ajout du chapitre concernant la compilation sous Ubuntu Edgy Eft2. Réorganisation du document en parties distinctes <p>Partie I : Installation sous Windows</p> <p>Partie II: Installation sous GNU/Linux - Ubuntu Edgy Eft</p> <p>Partie III: PostgreSQL</p> <p>Partie IV: PostGIS</p> <p>Annexes</p> <ol style="list-style-type: none">3. Amélioration de la préface "Avant-propos" et du chapitre 24. Chapitre "Paramétriser PostgreSQL": mise à jour pour GNU/Linux, renommage du chapitre en "Paramétriser PostgreSQL sous WIndows et GNU/Linux"5. Correction de la conclusion pour le document6. AJout d'un annexe sur "Créer un modèle de base de données PostgreSQL contenant les fonctionnalités de PostGIS pour un utilisateur aux droits restreints sur une base"7. Ajout de l'annexe "PostgreSQL et Stunnel"8. Ajouts de questions dans la FAQ: Comment connaître les objets d'une table qui intersectionnent une fenêtre? Comment importer un shapefile dans une base encodée en UTF-8? Comment convertir une BOX3d en POLYGON?	TECHER David

SUIVI DU DOCUMENT

INDICE	DATE	MODIFICATIONS	NOM
8.1	2006-10-23	<p>Mise à jour du document pour PostGIS 1.1.3:</p> <p>1. Chapitre 4: Section: Exemple de requêtes spatiales Il: Question "Comment créer une fonction en PL/PGSQL qui puisse ...": L'exemple fourni fonctionne maintenant</p> <p>2. Annexe F.A.Q: QuestionComment passer du format MapInfo à PostGIS? Mise à jour de la section en y incluant la méthode directe entre ogr2ogr vers postgis par exemple pour un fichier de MapInfo et en permettant de modifier le nom de la colonne géométrique</p>	TECHER David

SUIVI DU DOCUMENT

INDICE	DATE	MODIFICATIONS	NOM
8	2006-09-14	<p>Mise à jour du document pour PostGIS 1.1.3:</p> <ol style="list-style-type: none">1. Avant-propos: Ajout d'un exemple de motivation concernant la DDE du GardAjout de l'image concernant l'architecture PostgreSQL/PostGIS/Geos/ProjQuelques petites améliorations... <ol style="list-style-type: none">2. Chapitre 2: Suppression de la sous-section "Rendre sa copie portable qui était en doublon"3. Chapitre 3: Correction pour la section "Les variables d'environnement de PostgreSQL"4. Chapitre 4: Refontes de la section "Cas pratique avec MapServer" en plusieurs sections du même type en ajoutant l'exemple concernant la réunification pour le réseau hydrologique d'un département, exemple pour NumGeometries() "Exemples de requêtes spatiales II": exemple pour SnapToGrid(), possibilité d'utiliser aussi postgis_proj_version(), postgis_geos_version() et \db de psql "Démo en ligne" corrigé uniquement avec SVG et non plus avec MapServer5. Annexe F.A.Q: Ajout de la question: "Est-il possible de connaître l'extent sur des objets de nature géométrique différente?"	TECHER David
7	2006-09-04	Mise à jour du document pour PostGIS 1.1.3: <ol style="list-style-type: none">1. Chapitre 3: Ajout de la section "Gestion des fichiers de logs"	TECHER David
7	2006-09-01	Mise à jour du document pour PostGIS 1.1.3: <ol style="list-style-type: none">1. F.A.Q: Ajouts des sections "PostgreSQL: Connaitre les champs d'un table, ainsi que leurs type?" "PostgreSQL: Connaitre l'OID d'une table?"	TECHER David

SUIVI DU DOCUMENT

INDICE	DATE	MODIFICATIONS	NOM
7	2006-08-23	Mise à jour du document pour PostGIS 1.1.3: 1. Chapitre 4: "Exemples de requêtes spatiales II": amélioration de l'exemple pour PgDijkstra	TECHER David
7	2006-08-22	Mise à jour du document pour PostGIS 1.1.3: 1. Annexe F.A.Q: Ajout de la question: "Exporter des données en CSV avec PostgreSQL"	TECHER David
7	2006-08-19	Mise à jour du document pour PostGIS 1.1.3: 1. Chapitre 4: "Exemples de requêtes spatiales II": Ajout d'un exemple concernant PgDijkstra 2. Avant-propos: Quelques améliorations pour les premiers paragraphes.	TECHER David
7	2006-08-18	Mise à jour du document pour PostGIS 1.1.3: 1. Chapitre 4: "Cas pratique avec MapServer": Ajout deux la question concernant l'effacement de tables géométriques "Exemples de requêtes spatiales II": Comment se procurer de l'aide sur la synthaxe SQL de PostgreSQL? Utilisation des fonctions Centroid() et PointOnSurface() 2. Annexe F.A.Q: Ajout de questions pertinentes. 3. Partie annexes du document: ajout des mémentos concernant les commandes SQL de PostgreSQL et de psql	TECHER David

SUIVI DU DOCUMENT

INDICE	DATE	MODIFICATIONS	NOM
7	2006-08-13	Mise à jour du document pour PostGIS 1.1.3: 1. Chapitre 4: "Cas pratique avec MapServer": usage des index sur les données attributaires; "Cas pratique avec MapServer": ajout de question concernant les projections. 2. Annexe F.A.Q: Ajout de questions pertinentes. 3. Partie annexes du document: ajout des mémentos concernant les commandes SQL de PostgreSQL et de psql	TECHER David
6	2006-04-26	Mise à jour du document pour PostGIS 1.1.2: 1. Chapitre 2: Amélioration du chapitre, notes sur Geos et PostGIS 2. Chapitre 4: Exemple pour les fonctions line_locate_point et line_substring(); Utilisation de MapServer pour l'exemple sur les triggers. 3. Annexe F.A.Q: Notes sur les tests de régression de PostGIS sous Windows	TECHER David
5	2006-01-12	Chapitre 4: Ajout d'exemple pour une vue, et un déclencheur avec PL/PGSQL. Ajout de la bibliographie du document.	TECHER David
4	2006-01-06	Ajout du chapitre "PostGIS et les langages" et transformation de la section "Conclusion et discussion" du chapitre 4 en un chapitre séparé.	TECHER David
3	2006-01-04	Ajout de l'historique pour la documentation	TECHER David
2	2006-01-03	Annexe B: Précision sur l'impossibilité d'utiliser dbsize pour les versions de PostgreSQL < 8.1.0	TECHER David

SUIVI DU DOCUMENT

INDICE	DATE	MODIFICATIONS	NOM
2	2006-01-03	Chapitre 2: Ajout de la section "Erreurs de compilations reportées par les utilisateurs" Contributeurs et testeurs: Alain FERCHAL Jamal BERRICH	TECHER David
1	2005-12-31	Chapitre 4: Ajout de deux exemples de requêtes pour la section "Exemples de requêtes spatiales II"	TECHER David

Table des matières

I Installation sous Windows	10
1 Pré-requis	12
1.1 MinGW	12
1.2 Msys	12
1.3 Suppléments : Zlib, GetText, LibIconv, Bison, Flex, wget et MsysTDK	12
2 Compilation et Installation	14
2.1 Création des répertoires des sources et du répertoire de destination (facultatif)	14
2.2 PostgreSQL	15
2.2.1 Téléchargement	15
2.2.2 Compilation et Installation	16
2.2.3 Ajout des outils/supports contributifs pour PostgreSQL	16
2.3 Geos et Proj	16
2.3.1 Téléchargement	16
2.3.2 Compilations et Installations	17
2.3.3 Création de la DLL pour PROJ	17
2.4 PostGIS	17
2.4.1 Téléchargement	17
2.4.2 Compilation et Installation	18
2.5 Finalisation de la distribution, rendre sa distribution portable	19
2.6 Erreurs de compilations reportées par les utilisateurs	19
2.6.1 Erreur de compilation pour PostgreSQL 8.2.1 et PostGIS 1.2.0	20
2.6.2 Erreur de libiconv	20
2.6.3 Erreur de compilation avec PostGIS pour pgsql2shp.exe	21
2.6.4 Erreur de PostGIS avec make	21

II Installation sous GNU/Linux, Ubuntu Edgy Eft	22
3 Installation sous Ubuntu Edgy Eft	23
3.1 Pré-requis	23
3.2 PostgreSQL	23
3.2.1 Téléchargement et configuration	24
3.2.2 Test de régression (optionnel)	25
3.2.3 Installation	25
3.3 Geos et Proj	26
3.4 PostGIS	26
III PostgreSQL - configuration et administration minimale	28
4 Paramétriser PostgreSQL sous Windows et GNU/Linux	29
4.1 Se procurer une distribution déjà compilée - Optionnel pour Windows -	29
4.2 Les variables d'environnement de PostgreSQL	29
4.3 Le super-utilisateur de PostgreSQL	31
4.3.1 Le super-utilisateur ?	32
4.3.2 Création du super-utilisateur	32
4.3.3 Pouvoir accéder à la session du super-utilisateur sous DOS sans changer la session en cours	33
4.4 Initialisation de PostgreSQL : groupe de bases de données par défaut (Windows)	34
4.5 Initialisation de PostgreSQL : groupe de bases de données par défaut (GNU/Linux)	35
4.6 Autoriser les connexions TCP/IP et Démarrer/Arrêter le serveur	36
4.6.1 Autoriser les connexions TCP/IP	36
4.6.2 Méthode 1 : Démarrage/Arrêt manuel	37
4.6.3 Méthode 2 : Démarrage/Arrêt automatique pour Windows	37
4.6.3.1 Installation du service	38
4.6.3.2 Lancer le service	38
4.6.3.3 Désinstaller ou arrêter le service - Optionnel -	38
4.6.4 Méthode 2 : Démarrage/Arrêt automatique pour GNU/Linux	39
4.6.4.1 Modification du script source	39
4.6.4.2 Installation	41
4.7 Devenir soi-même super-utilisateur de PostgreSQL	42
4.8 Autoriser les machines du réseau intranet/extranet à se connecter au serveur	42
4.8.1 Intranet : Authentification par méthode de confiance (trusting) et connexion à toutes les bases	42
4.8.2 Extranet : Authentification par MD5, connexion à une seule et unique base de données pour un utilisateur unique de PostgreSQL - Optionnel -	43
4.8.2.1 Création de l'utilisateur (createuser)	43
4.8.2.2 Configuration de PostgreSQL	44
4.8.2.3 Accorder au nouvel utilisateur les droits d'accès aux données déjà existantes dans la base	44
4.9 Gestion des fichiers de logs	46
4.9.1 Création du répertoire pour les logs	46
4.9.2 Activation des paramètres	46

IV PostGIS	48
5 Tutoriaux	49
5.1 Créer une base avec PostGIS	49
5.2 Effectuer des requêtes : le moniteur interactif psql de PostgreSQL	50
5.3 Effectuer des requêtes : PgAdmin III	50
5.4 Exemples de requêtes spatiales I	50
5.4.1 Création de la base et d'une table	50
5.4.2 Ajout de la colonne géométrique à la table - AddGeometryColumn()	51
5.4.3 Objets géométriques spécifiés par l'O.G.C dans PostGIS	52
5.4.4 Exemples d'objets géométriques	53
5.4.5 Insertion d'objets géométriques - GeometryFromText()	53
5.4.6 Insertion des données dans la table	53
5.4.7 Question : Quelles sont les aires des objets ? - Area2d() -	56
5.4.8 Question : Quel sont les types géométriques des objets ? - GeometryType() -	57
5.4.9 Question : Qui est dans le bâtiment 2 ? - Distance() -	57
5.4.10 Question : Qui est dans le bâtiment 2 ? - WithIn() -	57
5.4.11 Question : Quel est l'objet géométrique le plus proche du piéton 2 ? - Min(), Distance() -	58
5.5 Exemples de requêtes spatiales II	58
5.5.1 Démo de quelques requêtes en ligne avec SVG	61
5.5.2 Chargement des données par SQL	61
5.5.3 Chargement de données par ESRI Shapefiles (shp2pgsql)	67
5.5.4 Question-Pratique : Quelle est la version de PostgreSQL ?	69
5.5.5 Question-Pratique : Où se trouve notre répertoire de bases de données ? - PGDATA -	70
5.5.6 Question-Pratique : Qui sont les utilisateurs de PostgreSQL ?	70
5.5.7 Question-Pratique : Quelles sont les infos sur les outils compilés pour PostGIS ?	71
5.5.8 Question-Pratique : Quel est le listing des bases de PostgreSQL ?	71
5.5.9 Question-Pratique : Quelles sont les tables contenues dans la base ?	71
5.5.10 Question-Pratique : Utiliser une vue pour simplifier la recherche du listing des tables.	72
5.5.11 Question-Pratique : Avec psql, comment obtenir rapidement un bref rappel de la syntaxe des commandes SQL de PostgreSQL ?	72
5.5.12 Question : Où sont stockées les informations relatives aux données spatiales (métadonnées) des tables avec PostGIS ? - table geometry_columns -	73
5.5.13 Question-Pratique : Comment créer une fonction en PL/PGSQL qui puisse faire le différentiel entre les tables référencées par geometry_columns et toutes les tables contenus dans le schéma public de la base (tables non géospatiales) ?	74
5.5.14 Question : Comment sont stockées les données géométriques avec PostGIS ?	75
5.5.15 Question : Quelles sont les aires et les périmètres des bâtiments ?	77
5.5.16 Question : Qui est dans le bâtiment Résidence des Mousquetaires ?	77
5.5.17 Question : Quelles distances séparent les bâtiments ?	78

5.5.18 Question : Combien de points composent chaque objet de la table great_roads ? - NumPoints() -	79
5.5.19 Question : Dans la table great_roads, quels sont les premier et dernier point de la Rue Paul Valéry ? - StartPoint(), EndPoint() -	80
5.5.20 Question : Quels sont les coordonnées des centres des bâtiments (buildings) ? - Centroid() -	80
5.5.21 Question : Comment garantir de toujours avoir un point sur un POLYGON autre que son centre en dépit de sa convexité/concavité ? - PointOnSurface() -	80
5.5.22 Question : Quels sont les points d'intersection entre les petites routes (small_roads) et les grandes routes (great_roads) ?	81
5.5.23 Quelle distance (relative au tracé de la rue Paul Valéry) dois-je couvrir si je pars de l'entrée la Rue Paul Valéry (table great_roads) jusqu'à son point de rencontre (intersection) avec la Rue Voltaire (table small_roads) ? - line_locate_point(),line_interpolate_point() -	82
5.5.24 Extraire de la Rue Paul Valéry la portion de route joignant le premier cinquième 1/5 au quatrième cinquième (4/5) de cette rue - line_substring() -	86
5.5.25 Question : Quel bâtiment est le plus proche de la personne 2 ?	88
5.5.26 Question : Quel bâtiment ne contient aucune personne ?	88
5.5.27 Question : Quels sont les personnes présentes dans les bâtiments ?	89
5.5.28 Question : Combien y-a-t-il de personnes par bâtiments ?	90
5.5.29 Question : Quel est l'aire d'intersection entre la rivière et les parcs ?	90
5.5.30 Question :Quel bâtiment est contenu dans le parc Mangueret I ? - Contains() -	92
5.5.31 Question :Quelles sont les personnes proches de la rivière dans un rayon de 5 mètres ? - Buffer () -	93
5.5.32 Question : Quel parc contient un "trou" ? - Nrings() -	94
5.5.33 Question : Quels sont les bâtiments que rencontrent la ligne qui relie la personne 5 à la personne 13 ? - MakeLine() -	95
5.5.34 Question : Comment arrondir la position des personnes (table personnes) avec un chiffre après la virgule ? -SnapTogrid() -	97
5.5.35 Application : Utiliser les déclencheurs (triggers) en PL/PGSQL de PostgreSQL pour suivre à la trace la personne 7 quand elle se déplace. Selon sa position, savoir quel est le bâtiment qui lui est le plus proche ou le bâtiment dans lequel elle se trouve ?	98
5.5.35.1 Fonction, table et trigger nécessaires	98
5.5.35.2 Modifications de la position par la commande UPDATE et GeometryFromText	99
5.5.35.3 Suivi des déplacements	99
5.5.35.4 Affichage avec MapServer	100

6 Etudes de cas	103
6.1 Cas pratique I : Etude détaillée, manipulation de données, cas pratique avec MapServer	103
6.1.1 Importation des communes du Languedoc-Roussillon	103
6.1.2 Afficher les informations relatives au Lambert II Etendu depuis la table spatial_ref_sys - get_proj4_from_srid() -	103
6.1.3 Question : Comment faire pour effacer la colonne géométrique sans effacer les données attributaires ? - DropGeometryColumn() -	104
6.1.4 Question : Comment faire pour effacer une table géométrique, avant une réimportation si les métadonnées à utiliser vont changer ?	104
6.1.5 Question : Comment faire si on a oublié de préciser l'identifiant de système de projection ? - UpdateGeometrySrid() -	105

6.1.6	Question : Si on s'est trompé dans le système de projection, comment faire pour reprojeter dans un autre système de manière déﬁnitive ? - Transform() -	105
6.1.7	Création d'index spatiaux Gist, Vacuum de la base	106
6.1.8	Question : Qu'elle est l'étendue géographique/emprise de la table communes_lr ? - Extent() -	106
6.1.9	Visualisation des données avec MapServer	107
6.1.9.1	Pour une image de longueur 500 pixels, quelle doit être la hauteur de l'image en fonction de l'étendue géographique ?	107
6.1.9.2	Mapfile et Script PHP	108
6.1.10	Question : Quelles sont les communes avoisinantes de Montpellier ?, Utilité des index spatiaux - Distance(), && -	111
6.1.11	Utilité des index spatiaux - temps demandé pour exécuter la requête	112
6.1.11.1	Avec la condition pour &&	112
6.1.11.2	Sans la condition pour &&	113
6.1.12	Créer une table communes_avoisinantes correspondant aux communes avoisinantes de MONTPELLIER, extraite et conforme à la structure de la table communes_lr, exploitable par MapServer.	114
6.1.13	Requête 1 : Qu'elle est l'intersection entre MONTPELLIER et les communes de LATTES et de JUVIGNAC ?- Intersection()- Que vaut cette géométrie en SVG ? - AsSVG(),	115
6.1.14	Requête 2 : Qu'elle est la commune ayant la plus petite aire ?	117
6.1.15	Mapfile générale pour la table communes_avoisinantes et les deux requêtes précédentes	118
6.1.16	Exercice : Obtenir une table departements_lr qui contient les contours départementaux du Languedoc-Roussillon à partir de la table communes_lr	120
6.1.17	Exercice : Trouver les communes du Gard et de l'Aude qui sont limitrophes à l'Hérault et les afficher grâce à MapServer.	125
6.1.18	QGIS : Affichage des tables précédentes	126
6.1.19	Exemple de projet sous GNU/Linux avec MapServer/PostgreSQL/PostGIS : savoir si à une échelle donnée, quel sera le meilleur format d'impression de A4 à A0 pour savoir si un polygone ne débordera du cadre de la carte	128
6.2	Cas pratique II : Réunifier des tronçons d'un réseau hydrologique d'un département	132
6.2.1	Objectifs	133
6.2.2	Mise en oeuvre	133
6.2.3	Questions	134
6.2.3.1	De combien d'objets est constituée la rivière de la Drôme ? - NumGeometries() -	134
7	PostGIS et les langages	136
7.1	avec du C -interface client de PostgreSQL -	136
7.1.1	Modifications des fichiers	136
7.1.2	Sortie et exécution	140
7.2	avec du PHP - un peu de SVG -	142
8	Conclusion	149

V Annexes	150
A PgRouting pour le calcul d'itinéraire	152
A.1 Installation sous Windows	152
A.2 Installation sous Ubuntu Dapper/Edgy	153
A.2.1 GAUL	153
A.2.2 BGL	154
A.2.3 CGAL	154
A.2.4 PgRouting	154
A.3 Installation sous Mac OS X	154
A.3.1 FINK	154
A.3.2 GAUL	154
A.3.3 BGL	155
A.3.4 CGAL	155
A.3.5 PgRouting	155
A.4 Chargement des fonctionnalités de PgRouting	156
A.5 Fonctionnalité Shortest_Path() - Dijkstra	156
A.5.1 Dijkstra : module de routing pour PostgreSQL pour la recherche du plus cours chemin	156
A.5.1.1 Description	156
A.5.1.2 Création d'un réseau routier	156
A.5.1.3 Calcul des coûts sur les noeuds du graphe pour rendre la Rue Voltaire à sens unique	159
A.5.1.4 Exemples d'utilisation	160
A.6 Importation d'un shapefile concernant des tronçons	164
A.7 Obtention des noeuds du réseau	165
A.8 Fonctionnalité shortest_path_astar()	168
A.8.1 Exemple pour les noeuds 38 et 48.	168
A.8.1.1 Pour l'aller (source=38 et target=48)	168
A.8.1.2 Pour le retour (source=48 et target=38)	169
A.8.2 Démo en ligne avec MapServer	170
A.8.3 Tester soi-même la démo avec MapServer.	170
A.8.4 Tester sur un jeu de données réelles : jeu de tests GEOROUTE IGN	170
A.8.4.1 Conversion du fichier MapInfo vers PostGIS	171
A.8.4.2 Instructions SQL	171
A.8.4.3 Exemple	172
A.9 Fonctionnalité TSP()	175
A.9.1 Exemple	175
A.9.2 Programme en C pour les appels successifs à shortest_path_astar()	176
A.9.2.1 Le programme	177
A.9.3 Limites du programme	179

A.10 Fonctionnalités shortest_path_astar_as_geometry_internal_id_directed() et tsp_astar_as_geometry_internal_id_directed()	179
A.10.1 Importation d'un jeu de données NavTeq	179
A.10.2 Noeuds du réseau et direction pour le routage	181
A.10.3 Modifications nécessaires sur la table streets_edges	182
A.10.4 Exemple avec shortest_path_astar_as_geometry_internal_id_directed()	182
A.10.5 Exemple avec tsp_astar_as_geometry_internal_id_directed()	186
B Créer un modèle de base de données PostgreSQL contenant les fonctionnalités de PostGIS pour un utilisateur aux droits restreints sur une base	189
B.1 Création du modèle contenant les fonctionnalités de PostGIS	189
B.2 Création de l'utilisateur et de la base de données	190
B.2.1 Création de l'utilisateur	190
B.2.2 Création de la base de données	190
B.2.3 Restriction des droits sur les tables geometry_columns et spatial_ref_sys	190
B.2.4 Vérification	190
C Dblink : interroger plusieurs serveurs PostgreSQL distants	193
C.1 Matériel requis pour la simulation	193
C.2 Compilation et installation	194
C.3 Mise en oeuvre	194
C.3.1 Premiers tests	195
C.3.2 Test attendu	195
D Pouvoir faire des sauvegardes des bases locales ou distantes d'un serveur PostgreSQL du réseau intranet vers une machine-cliente	196
D.1 Proposition 1 : Sauvegarde et restauration rapide par une base	196
D.1.1 Sauvegarde au format tar du schéma public de la base de données - pg_dump	196
D.1.2 Restauration - pg_restore	199
D.2 Proposition 2 : Sauvegarde pour toutes les bases et toutes les tables sans les définitions des fonctions de PostGIS ou autre- Script général (Côté-client)	199
D.3 Proposition 2 : Restauration	202
D.3.1 Format sql	202
D.3.2 Format tar.gz	202
E PostgreSQL et les index	204
E.1 Importation d'un jeu de données	204
E.2 Importer des données au format CSV dans PostgreSQL	204
E.3 Index B-tree, opérateur =	205
E.4 Index B-tree, fonctions	206
E.5 Index B-tree, recherche sur motif, like "chaine%"	206
E.6 Index B-tree, recherche sur motif, like "%chaine"	207
E.6.1 Fonction inversant une chaîne en PostgreSQL	207
E.6.2 Création de l'index sur le champs "Commune" de la table insee	208
E.6.3 Requêtes	208

F Connaître l'espace disque occupé par les données (dbsize)	210
F.1 Dbsize - directement dans le backend de PostgreSQL -	210
F.2 Script Shell	211
F.3 Script PHP	212
G PostgreSQL et Stunnel	214
G.1 Introduction	214
G.2 Pré-requis	214
G.3 Motivations : sniffer une connexion non sécurisée avec NAST, limites d'une connexion par mot de passe en md5 !	215
G.3.1 Test sans mot de passe	215
G.3.2 Test avec MD5	216
G.4 Stunnel : sécurisation de la connexion	218
G.4.1 Pré-requis : OpenSSL	218
G.4.2 Installation de Stunnel	218
G.4.3 Mise en oeuvre	219
G.5 Installation en service de Stunnel	221
G.6 Pour aller plus loin	222
H MapServer : faire une image avec zones réactives	223
H.1 Démo en ligne et ressource en ligne	223
H.2 Création de la base de données	225
H.3 Importation des données dans une base PostGIS	226
H.4 La mapfile	226
H.5 Script php	227
I Foire Aux Questions	232
I.1 Existe-il des installateurs pour PostgreSQL et PostGIS qui évitent d'avoir à les compiler soi-même ?	232
I.2 Quel logiciel utilisé pour gérer/administrer un/des serveurs PostgreSQL sous Windows ?	232
I.3 Quel logiciel utilisé pour visualiser ses données de PostGIS ?	232
I.4 Comment migrer des données de PostGIS à travers un réseau intranet/extranet ?	233
I.5 Comment passer du format MapInfo à PostGIS ?	233
I.6 PostGIS : Comment passer de PostGIS à ESRI Shapefile ?	234
I.7 PostGIS : Est-il possible de calculer l'extent sur des objets de nature géométrique différente ?	234
I.8 PostgreSQL : Comment exporter des données au format CSV ?	235
I.8.1 A partir de psql	235
I.8.2 Avec ogr2ogr	235
I.9 Comment connaître les objets d'une table qui intersectionnent une fenêtre ?	236
I.10 Comment convertir une BOX3d en Polygon ?	236
I.11 PostGIS : Comment mettre à jour et passer de PostGIS 0.X. à 1.0.X pour PostgreSQL 8.0.X ?	236
I.12 Notes pour PostGIS 1.1.2 : Tests de régression	237

I.13 Sous GNU/Linux : depuis un terminal comment se connecter à une machine distante sans avoir à taper à chaque fois le mot de passe ?	239
I.14 Sous GNU/Linux : comment connaître l'ensemble des connexions client-serveur actives ?	239
I.15 PostgreSQL : Connaitre l'OID d'une table ?	240
I.16 PostgreSQL : Connaitre les champs d'un table, ainsi que leurs type ?	241
I.17 Sous GNU/Linux : Comment migrer une base encodé en LATIN9 ou un shapefile encodé en LATIN9 vers une base encodée en UTF-8 ?	241
I.18 Sous GNU/Linux : Comment créer un espace logique avec PostgreSQL ?	241
I.19 PostgreSQL : Peut-on copier un schéma dans un autre schéma et renommer l'ancien schéma ?	242
I.20 Comment ajouter le support PL/Perl pour PostgreSQL sous Windows ?	243
I.21 Comment charger l'extension de PostgreSQL sous PHP ?	243
J Mémento sur les commandes SQL de PostgreSQL	244
K Mémento sur les commandes internes de psql	261
L Suivi de PostGIS	263
M PhpMapScript	265
M.1 Table mapserver_desc	265
M.2 Script PHP	265
M.3 Sortie : Mapfile	271
N Auteurs et contributeurs de PostGIS	276
O Bibliographie	277
O.1 [1]	277
O.2 [2]	277

Table des figures

1	Accidents corporels ayant eu lieu le long de la Route Nationale 113 dans le Gard en 2006 avant le lundi 2006-09-11 (Source : Accidentologie dans le Gard, http://www.01mapserver.com/applis/dde30)	1
2	Communes du Languedoc-Roussillon : exportation au format KML exploitable par Google Earth.	3
3	Réseau routier et ses noeuds obtenu grâce à PgRouting sur la base GEOROUTE IGN	4
4	PgRouting sur la base GEOROUTE IGN - sens aller.	5
5	PgRouting sur la base GEOROUTE IGN - sens retour.	6
6	Architecture PostgreSQL/PostGIS/Geos/Proj	7
7	PostgreSQL/PostGIS font un carton dans le monde du logiciel libre !	8
2.1	Commandes de création des répertoires	15
2.2	Les répertoires des sources (créés par MinGW)	15
2.3	Installation terminée.....C'est dans la boite !	19
4.1	Exemple d'emploi de variables d'environnement de PostgreSQL 8.0.0 pour une connexion cliente. À adapter ici pour la version 8.2.1	31
4.2	Utilisation de la commande runas	33
4.3	Nouvelle fenêtre DOS obtenue	34
4.4	Initialisation du groupe de bases de données pour PostgreSQL 8.0.3 (ancienne image)	35
4.5	Démarrage manuel de PostgreSQL	37
4.6	Arrêt manuel de PostgreSQL	37
4.7	Installation et démarrage du service (postgresqlwin32) pour PostgreSQL	38
4.8	Exemple de connexion dans PgAdmin III	45
5.1	Le moniteur psql - Connexion à la base testgis.	51
5.2	psql : Fonction AddGeometryColumn() de PostGIS	52
5.3	psql : Insertion des données dans la table test	55
5.4	Visualisation de la table test dans la base testgis	56
5.5	Visualisation des bâtiments (tables buildings et parcs et rivers).	59
5.6	Visualisation des personnes (table personnes).	60
5.7	Visualisation des petites et grandes routes (tables small_roads et great_roads).	61
5.8	Personnes présentes dans le bâtiment Résidence des Mousquetaires	78
5.9	Points d'intersection entre les tables small_roads et great_roads	82

5.10 Parcours effectué sur 71% de la rue Paul Valéry	84
5.11 Portion de route allant du 1/5 au 4/5 de la Rue Paul Valéry	87
5.12 Bâtiment ne contenant aucune personne : EDF	89
5.13 Intersection entre la rivière et les parcs	91
5.14 Bâtiment contenu dans le parc Mangueret I : Office du Tourisme	92
5.15 Buffer de 5 mètres sur la rivière : les personnes 8 et 10 y sont présentes	94
5.16 Ligne reliant les points désignant les personnes 5 et 13	96
5.17 Bâtiments (table buildings) que rencontre la ligne reliant les points désignant les personnes 5 et 13	97
5.18 Déplacement successifs de la personne 7	102
6.1 Extent avec PostGIS. Ce qu'on appelle aussi emprise en S.IG.	107
6.2 Affichage des communes du Languedoc-Roussillon (MapServer+PhpMapScript)	110
6.3 Les communes avoisinantes de Montpellier	112
6.4 Intersection entre MONTPELLIER et les communes de LATTES et de JUVIGNAC	117
6.5 LATTES : la commune ayant la plus petite aire.	118
6.6 Affichage des départements du Languedoc-Roussillon	124
6.7 MapServer : Communes du Gard (30) et de l'Aude (11) limitrophes à l'Hérault (34)	126
6.8 QGIS : Affichage des tables communes_lr, departements_lr et communes_avoisinantes	127
6.9 QGIS : Zoom sur le département de l'Hérault	128
6.10 Reconstitution du réseau hydrologique de la DROME (26). La drôme est en jaune sur l'image.	134
7.1 Sortie en SVG sur quelques requêtes de la section "Exemples de requêtes spatiales II" du chapitre 4	148
A.1 Installeur 1.0.0.a	153
A.2 Graphe orienté, la rue Voltaire est à sens unique (en jaune sur le graphe, du noeud 4 au noeud 1)	159
A.3 Shortest_Path() : chemin le plus court pour aller de 6 à 8	161
A.4 Shortest_Path() : chemin le plus court pour aller de 2 à 5 (puisque la Rue Voltaire est à sens unique)	162
A.5 Le réseau avec 5 rond-points	165
A.6 Les noeuds du réseau	166
A.7 Parcours à l'aller	169
A.8 Parcours au retour	170
A.9 Une portion des noeuds de mon réseau (sources+targets)	173
A.10 Parcours à l'aller	174
A.11 Parcours au retour	175
A.12 fonctionnalité TSP() pour les noeuds 7,41,28,42 et 25	176
A.13 GoogleEarth : réseau routier à Washington DC	180
A.14 GoogleEarth : réseau routier à Washington DC	181
A.15 Fonction shortest_path_astar_as_geometry_internal_id_directed() sur source=5274 et target=5488 (sens aller) . .	183
A.16 Fonction shortest_path_astar_as_geometry_internal_id_directed() sur source=5274 et target=5488 (sens aller) . .	184
A.17 Fonction shortest_path_astar_as_geometry_internal_id_directed() sur source=5488 et target=5274 (sens retour) . .	185

A.18 Fonction shortest_path_astar_as_geometry_internal_id_directed() sur source=5488 et target=5274 (sens retour)	186
A.19 Fonction tsp_astar_as_geometry_internal_id_directed()	187
A.20 Fonction tsp_astar_as_geometry_internal_id_directed() vue depuis GoogleEarth	188
C.1 Mes deux bases distantes : testgis et testgis2	194
D.1 Sauvegarde du schéma public de la base madatabase - pg_dump -	198
D.2 Restauration de la table buildings dans la base madatabase - pg_restore -	199
H.1 ImageMap à l'initialisation	224
H.2 ImageMap lors du passage du pointeur de la souris sur l'Hérault	225
I.1 Une fenêtre intersectionnant des objets (ici les communes en région bourgogne)	236

Liste des tableaux

1.2 Versions des outils utilisés	13
5.2 Exemple d'objets géométriques en WKT avec PostGIS	54
L.2 Suivi de PostGIS	264

Résumé

PostGIS est l'extension qui confère au SGBD PostgreSQL la spécificité de bases de données spatiales. Ce document est un recueil de notes pour une compilation et une installation personnelle de PostGIS/PostgreSQL sous Windows et GNU/Linux. Elle est le fruit du travail des contributeurs du site postgis.fr, pour la communauté française en SIG. Ceci est la version 11.6 de ce document mise à jour le 2007-06-01. L'historique est disponible à <http://www.davidgis.fr/documentation/win32/historique.html>. La version imprimable au format PDF est disponible à <http://www.davidgis.fr/documentation/win32/pdf/doc.pdf>

Avant-propos

Cette préface débute par divers cas de figure concernant PostGIS. Je ne permet de m'égarer dans les quelques lignes qui suivent rapidement avant de revenir à PostGIS au moment util. Vous comprendrez ma démarche en prenant trois exemples je l'espère assez parlant et convaincant pour introduire PostGIS.

MOTIVATION I - L'EXEMPLE CLASSIQUE DE LA CARTE

Quand on démarre une documentation, on dit souvent "Il vaut mieux fournir un exemple parlant que tout un blabla sans intérêt !!".



FIG. 1 – Accidents corporels ayant eu lieu le long de la Route Nationale 113 dans le Gard en 2006 avant le lundi 2006-09-11
(Source : Accidentologie dans le Gard, <http://www.01mapserver.com/applications/dde30>)

Sur l'image précédente, j'ai interrogé un serveur cartographique pour connaître l'ensemble des accidents qui ont eu lieu le long de la Nationale 113 dans le Gard(30). Mais on peut aussi demander au serveur PostgreSQL/PostGIS par exemple

"Donne-moi l'ensemble des villes que traverse la Nationale 113 dans le Gard pour lesquels on dénombre des accidents ainsi que le nombre d'accidents pour les dites-villes"

En SQL avec PostgreSQL, je traduirais la demande formulée ci-dessus par la requête suivante

Exemple 0.1 Exemple de requête avec PostGIS

```
SELECT DISTINCT l.com_top AS communes, count(foo.date) AS "nombre d'accidents"
  FROM limites_communales l,
       (SELECT a.date,a.the_geom FROM rezo_route b,base_accident1 a
        WHERE b.route='N113' AND a.the_geom && expand(b.the_geom,100)
          AND distance(a.the_geom,b.the_geom)<100) AS foo
       WHERE l.the_geom && foo.the_geom AND intersects(l.the_geom,foo.the_geom) ←
             GROUP BY l.com_top;
```

qui me renverra comme réponse

Exemple 0.2 Résultat de la requête

communes		nombre d'accidents
AIGUES-VIVES		1
BELLEGARDE		1
BOUILLARGUES		2
NIMES		26

Mais ceci n'est qu'un exemple parmi tant d'autres de ce qu'est capable de faire PostGIS !!! Notons au passage que cette requête a été ici exécutée en moins de 100 millisecondes ! C'est au travers de ce genre de requête que cette documentation essaiera de vous fournir les pistes nécessaires pour vous former à l'utilisation de PostGIS. Mais pour celà commençons par une remarque tout à fait sommaire beaucoup plus générale.

MOTIVATION II - EXPORTER UNE REQUETE EN UN FICHIER KML ET LE VISUALISER AVEC GOOGLE EARTH

Par rapport à l'exemple précédent, on peut aussi essayer par exemple d'aller plus loin. Au lieu d'utiliser MapServer, pourquoi ne pas utiliser GoogleEarth pour tenter de visualiser nos données ? Alors pourquoi GoogleEarth ? Comme tout le monde le sait, 2006 a été l'année de la grande banalisation de la carte sur le Web notamment grâce à un outil comme GoogleEarth. L'engouement du grand public pour cet outil gratuit d'utilisation a bien été ressentie de par le Web et nul ne peut le démentir. Qui dit GoogleEarth dit aussi le fameux fichier kml. Voyons rapidement que passer de PostGIS à kml - au sens de convertir des données de PostGIS vers le format kml - se fait sans souci.

Si on a un outil comme Gdal (version >= 1.4.0) d'installé sur sa machine, il est aussi possible d'utiliser ogr2ogr pour exporter nos données dans un fichier kml

NOTE

Le site de Gdal est <http://www.gdal.org/> où vous trouverez aussi bien les sources pour GNU/Linux que pour Windows.

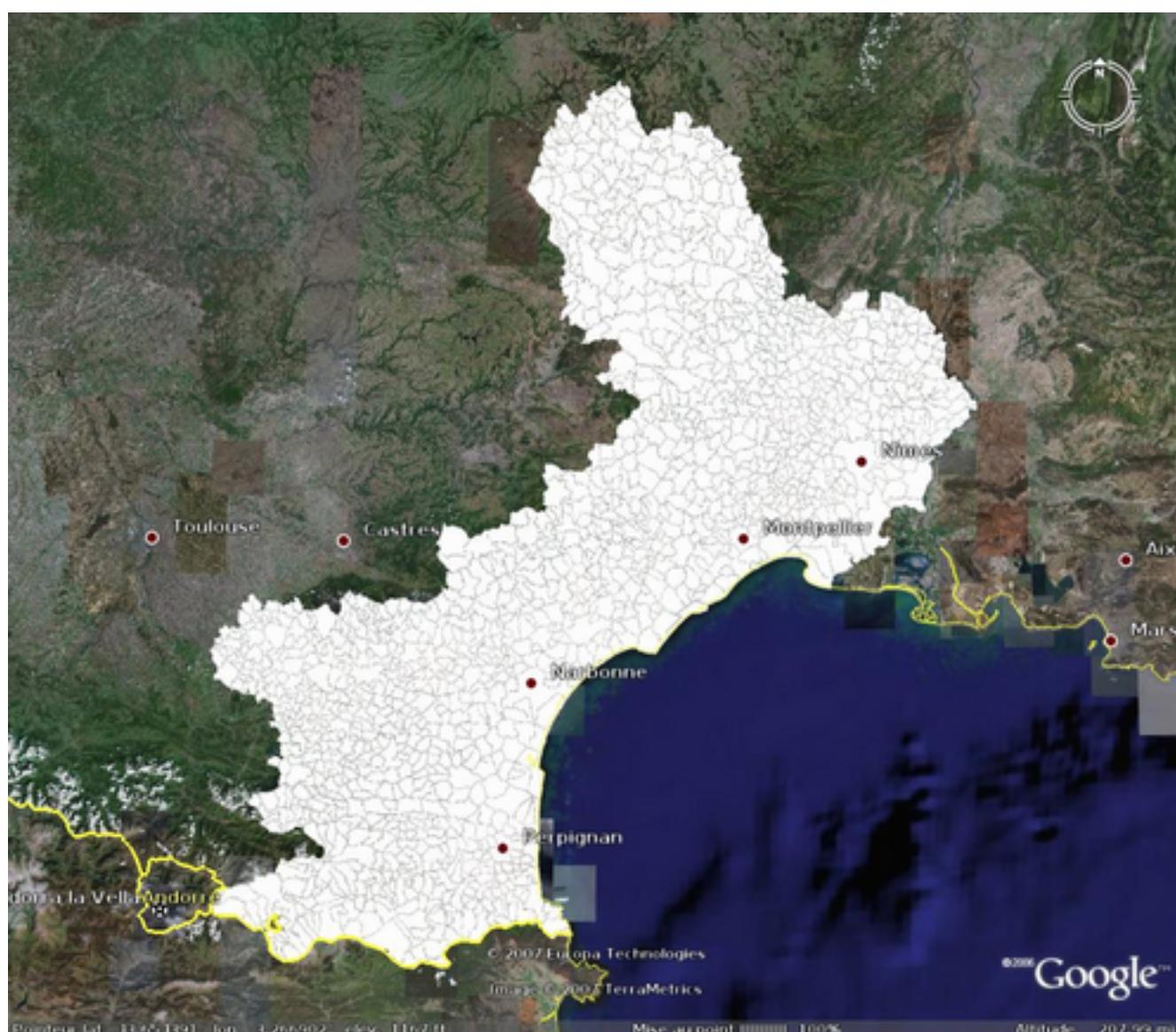


FIG. 2 – Communes du Languedoc-Roussillon : exportation au format KML exploitable par Google Earth.

L'exportation ici de PostGIS vers KML aura été rendu possible par la ligne de commande suivante

Exemple 0.3 Conversion de PostGIS en KML : utilisation de ogr2ogr

```
/opt/gdal-1.4.0/bin/ogr2ogr -f KML /root/communes_lr.kml PG:'host=localhost port=5432 ←  
    dbname=diren user=postgres' \  
-sql "SELECT (dump).geom FROM (SELECT dump(transform(the_geom, 4326)) FROM communes_lr) AS ←  
    foo"
```

Ici nos données sont automatiquement reprojetées à la volée en utilisant la fonction `Transform()` dans le système WGS84¹. La fonction `dump()` permet quant à elle de fractionner un objet multiple en renvoyant les enregistrements contenant l'objet en question. Par exemple, `dump()` fractionnera un `MULTIPOLYGON` en un lot d'enregistrements ordonnés (au sens ligne de tableau) de `POLYGON` contenus dans ce `MUTIPOLYGON`...

¹WGS84 : système utilisé pour le GPS , avec l'identifiant du système de projection valant 4326)

NOTE

Ici j'ai utilisé un outil comme ogr2ogr de GDAL mais il est également possible d'utiliser GeoServer (<http://docs.codehaus.org/display/GEOS/Home>) qui permet de faire des sorties en KML^a

^aGeoServer est un projet écrit en Java, supportant aussi bien le WFS Transactionnel que la WMS. GeoServer est certifié OGC pour le support WMS 1.1.1 et WFS-T 1.0. Pour simplifier, on dira que c'est un serveur de données pouvant se connecter à des bases de données comme PostgreSQL/PostGIS, des fichiers (Shapefile,...). Il supporte beaucoup de formats de sortie comme PNG, SVG, KML, PDF

MOTIVATION III : PGROUTING POUR LE CALCUL D'ITINERAIRE SUR UN RESEAU

PgRouting - couplé à PostGIS - est un module pour le calcul d'itinéraire. Développé en étroite collaboration par Camptocamp (Suisse/France) et Orkney (Japon), ce module s'est enrichi en début 2007. Si nous considérons par exemple la portion du réseau

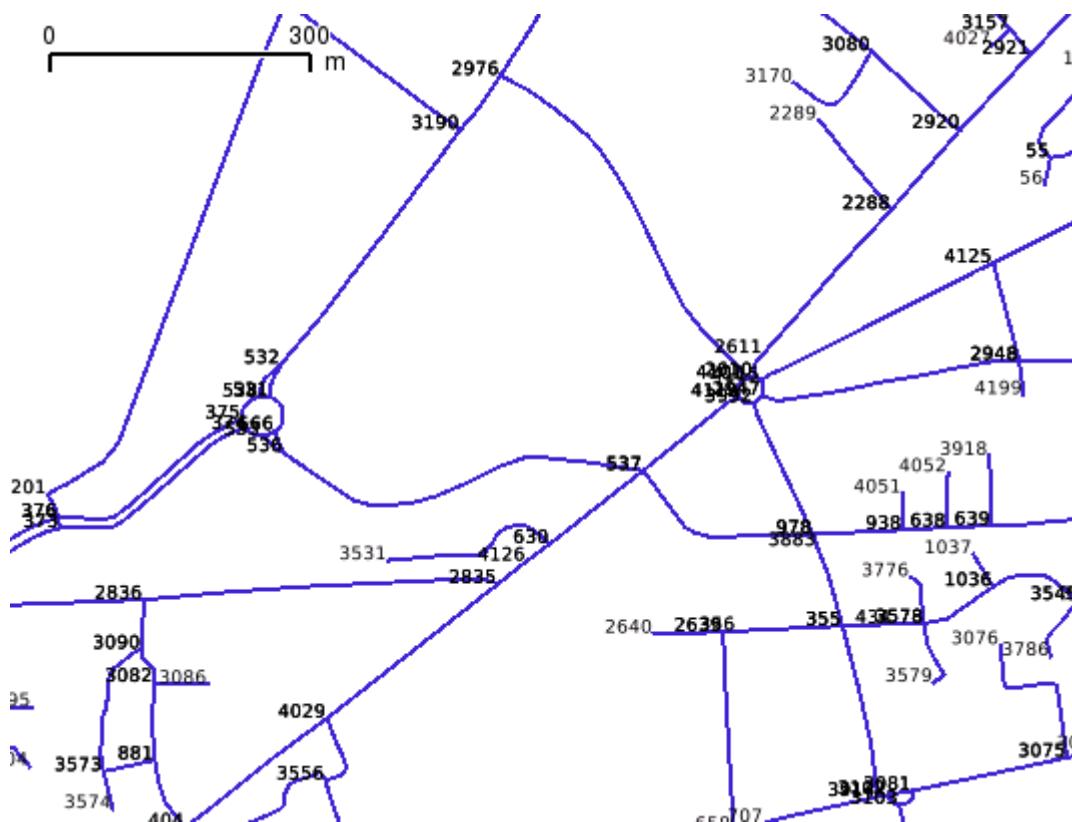


FIG. 3 – Réseau routier et ses noeuds obtenu grâce à PgRouting sur la base GEOROUTE IGN

avec des données réelles, on aurait par exemple



FIG. 4 – PgRouting sur la base GEOROUTE IGN - sens aller.

et pour l'autre itinéraire



FIG. 5 – PgRouting sur la base GEOROUTE IGN - sens retour.

Nous aurons l'occasion de voir ces fonctions plus en détail plus tard. Pour l'instant, trois exemples intéressants nous l'avons vu pour le moment du potentiel de PostGIS. Ne nous égarons pas pour le moment. Comme il s'agit ici d'introduire notre travail sur PostGIS, revenons donc à PostGIS de manière conventionnel en commençant par un bref historique.

UN BREF RAPPEL SUR POSTGIS

PostGIS est la cartouche spatial qui confère au SGDBRO² PostgreSQL le statut de bases de données spatiales. Développé et maintenu par l'équipe de [Refractons Research](#), son cadre d'utilisation est celui fourni selon les spécificités GPL (<http://www.gnu.org/copyleft/gpl.html>). Son développement a commencé à l'été 2001. Les débuts de PostGIS étaient assez limités en terme d'offre

PostGIS n'est pas le seul projet open source développé par Refractons. Je profite ici d'en rappeler un autre notamment [UDIG](#) - viewer PostGIS mais aussi viewer pour du WMS/WFS.

C'est au cours de l'automne 2003 que la version 1.0.0 Geos (projet aussi développé par Refractons) fut couplé à PostGIS. Geos offrit des fonctionnalités supplémentaires - étendre les opérateurs relationnelles spatiaux - à PostGIS telle que `IsValid()`, `Intersection()`, (...) Il était aussi possible à l'époque d'utiliser le projet Proj. Couplé à ces deux outils, PostGIS offre ainsi plus de 400 fonctionnalités spatiales.

Mais quand est-il actuellement sur PostGIS ?

Beaucoup de projets en SIG utilisent PostgreSQL/PostGIS, comme base de données spatiales. Certains sont même venu à abandonner Oracle Spatial pour se tourner vers ce couple. Histoire de licence, coût exorbitant,...chacun trouvera là ses propres arguments. Même certaines solutions SIG propriétaire proposent dans leur éventail un module ou une extension pour pouvoir

²SGBDRO : Système de Gestion de Bases de Données Relationnelles Orienté Objet. Dans un SGBDR classique, les données sont stockées dans des structures de données à deux dimensions : les tables. L'aspect relationnel-objet permet d'étendre ce modèle relationnel : support des tableaux, héritage, fonctions. ...Je ne m'attarderai pas plus sur les grands termes.

se connecter à PostgreSQL/PostGIS : FME...Je citerais aussi à titre d'exemple l'Institut Géographique National - I.G.N - qui a aussi misé sur PostGIS par rapport à d'autre solutions. Dans les premières années de son utilisation PostGIS était utilisé sur GNU/Linux, FreeBSD. Actuellement, il est possible de les utiliser aussi su MAC OS X et Windows.

Pour ce dernier système d'exploitation, son utilisation de 2001 jusqu'au début de l'année 2004 requierait d'avoir cygwin d'installé ! C'était la croix et la bannière déjà pour installer Cygwin. Depuis début 2004, des efforts des contributeurs ont été fait pour maintenir à jour une meilleure installation sous Windows. Je me rappelle encore mes premières heures passées à tenter de les compiler tous les deux sous Windows :D.

Comme j'ai fait parti des premiers à avoir testé PostGIS pour sa compilation sous Windows, un jour je me suis dis. "Tiens ça serait bien de mettre tout ça au propre sur Internet pour les divers utilisateurs !..."

Ce qui est le but du document. Ici j'essaies au fur et à mesure de mes expériences d'enrichir ce document dont je fais la mise à jour fréquemment selon les sorties de nouvelles versions. C'est un travail qui me prend du temps et de la réflexion. Il n'est pas parfait mais j'essaie au mieux de le rendre meilleur au fur et à mesure de mes avancées..

Bien qu'il existe aujourd'hui les installateurs sous Windows, je maintiens toujours les premiers chapitres de ce document pour des tests de compilation fréquents, mais aussi pour garder une trace de HOWTO comme on dit pour une nouvelle version. Mais comme on dit aussi "Rien ne vaut un travail bien fait que si on s'est donné la peine de le faire soi-même !"

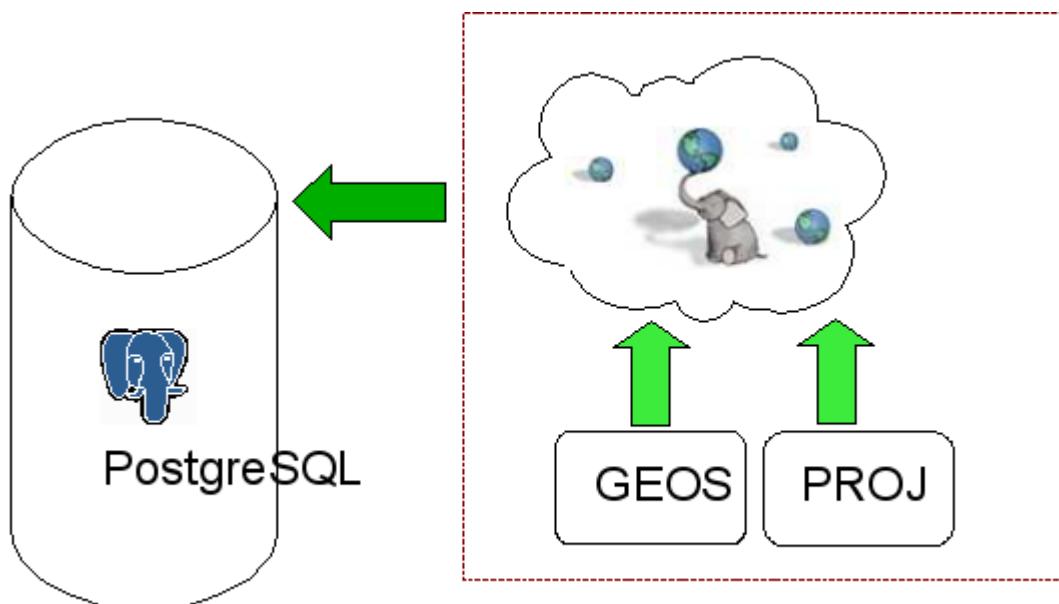


FIG. 6 – Architecture PostgreSQL/PostGIS/Geos/Proj

Comme je commence maintenant à parler de compilation, notons déjà les points suivants

LES OUTILS QUE NOUS COMPILEONS/UTILISERONS

Les divers installations possibles concernent :

PostgreSQL 8.2.1 (<http://www.postgresql.org>) : version pour laquelle - depuis la 8.0.0 maintenant - le serveur tourne en natif sous Windows . Les versions de Windows supportées sont Windows 2000/2003 ou XP. Ne pas oublier son incompatibilité avec NT 4. Le système de fichier du disque - sur lequel PostgreSQL sera installé - le plus "apprécié" sera du NTFS. Le site français de PostgreSQL est <http://www.postgresqlfr.org>

PostGIS 1.2.0 (<http://postgis.refractions.net>) : tout le monde connaît...Le site de contribution en français est <http://www.postgis.fr>

Geos 2.2.3 (<http://geos.refractions.net>) : librairie qui enrichit/complète la panoplie des fonctions spatiales (topologiques, relationnelles au sens géométrique, etc...) de PostGIS ;

Proj 4.5.0 (<http://www.remotesensing.org/proj/>) : librairie permettant la reprojection dans les divers systèmes de projection connus (Lambert II Etendu, Lambert III, , Lambert Zone II etc...).



FIG. 7 – PostgreSQL/PostGIS font un carton dans le monde du logiciel libre !

Ce document n'a pas la prétention d'être un document de substitution par rapport aux documentations officielles de PostGIS, PostgreSQL, Geos et Proj. Bien au contraire, il s'agit d'un support de travail de deux contributeurs - principalement PostGIS - pour la commune francophone en SIG. Ce document est autant que possible mis à jour régulièrement.

Le document est divisé en plusieurs parties. Voici un bref descriptif des divers thèmes abordés.

PLAN DU DOCUMENT

PARTIE I - Installation sous Windows

Chapitre 1 : Pré-requis. On y dresse l'environnement nécessaire pour la compilation des divers outils sur Windows, notamment MinGW et Msys

Chapitre 2 : Compilation et Installations. On y traite des diverses commandes à effectuer pour mener à bien les compilations. L'intégralité de la distribution (ensemble des binaires obtenus) proposée est une distribution minimale par rapport au panel proposée par l'équipe de PostgreSQL sous Windows. Mais elle est amplement suffisante pour apprendre à utiliser PostGIS/PostgreSQL sous Windows

PARTIE II - Installation sous GNU/Linux, Ubuntu Edgy Eft

Chapitre 3 : Installation sous Ubuntu Edgy Eft. Nous verrons comment installer nos outil sous Ubuntu.

PARTIE III - PostgreSQL - configuration et administration minimale

Chapitre 4 : Paramétrier PostgreSQL sous Windows et GNU/Linux. Ici, nous verrons comment administrer de manière minimale notre serveur PostgreSQL. Les points comme

PARTIE IV - PostGIS

Chapitre 5 - Tutoriaux. Au travers d'exemples pédagogiques, nous apprendrons à prendre en main PostGIS. Nous dresserons un premier bilan des premières fonctionnalités spatiales utiles et à connaître.

Chapitre 6 - Etudes de cas. Les connaissances acquises au cours du chapitre précédents serons mis à profit pour des études de cas sur des données réelles. Nous verrons notamment comment importer des données issus de Shapefiles, créer des index spatiaux, optimiser des requêtes etc...

PARTIE V - Annexes

Divers cas d'administration concernant PostgreSQL/PostGIS sont mentionnés dans les annexes de cette documentation. Cette partie étant souvent mise à jour régulièrement, je vous invite donc à la feuilleter en long et en large au lieu de dresser ici son contenu. Merci

IMPORTANT

 REMARQUE IMPORTANTE POUR WINDOWS : Dans le cas où vous ne souhaiteriez pas compiler les outils par vous-même (= passer les deux premiers chapitres), deux solutions s'offrent à vous. SOLUTION I : disposer d'une distribution déjà compilée et installer PostgreSQL manuellement. Dans ce cas, merci de consulter au chapitre 3 la section "Se procurer une distribution déjà compilée" et de suivre la documentation à partir du chapitre 3. SOLUTION II : ne rien installer. Dans ce cas, rendez-vous à l'annexe "Foire Aux Questions" pour obtenir des installateurs déjà existants et suivez la documentation à partir du chapitre 4 ! CE DOCUMENT NE FOURNIT QU'UNE INSTALLATION MINIMALE SUR L'UTILISATION DE POSTGIS ET DE POSTGRESQL !

AVERTISSEMENT

Droits sur ce document

 Ce document est fourni à titre indicatif. En le lisant ou en essayant les commandes par vous-même, vous vous engagez à ne me porter aucune responsabilité éventuelle aussi bien d'ordre matériel que software. L'auteur ne saurait être tenu responsable pour un désagrément que ce soit.

Il est libre de droit donc vous pouvez aussi bien l'utiliser :

pour un cadre purement pédagogique : un cours dispensé en Université ou pour une formation faite par un prestataire
pour une conférence, rédaction de mémoire ou de rapport etc..MERCI EN TOUT CAS DE CITER LES SOURCES LE CAS EVENTUEL !

Les images des données affichées dans ce document ont reçu l'approbation des instances compétentes pour leur affichage que je tiens à remercier ici : DIREN Languedoc-Roussillon, DRASS Languedoc-Roussillon, DIREN et DRASS de Bourgogne.

Notes de rédaction de ce document Ce document a été rédigé en DocBook Version 4.2. La version imprimable au format pdf de ce document a été rédigée à partir du projet DBLaTEX <http://dblatex.sourceforge.net> maintenu par Benoit GUILLOON. C'est la version 0.1.8 de DBLaTEX que j'ai utilisé.

BONNE LECTURE A TOUS !

Première partie

Installation sous Windows

Notes à l'intention des utilisateurs

POUR CEUX QUI VEULENT COMPILE PAR EUX-MEME DE A à Z

Dans les deux chapitres qui suivent, nous verrons comment compiler manuellement les outils sur Windows.

POUR CEUX QUI NE VEULENT RIEN COMPILE

Rendez-vous alors dans la partie Annexes/F.A.Q de ce document et suivez les instructions des deux premières questions fournies. Merci ensuite de vous rendre au chapitre 3 pour avoir un état des lieux de ce que les installeurs auront fait comme travail en arrière-plan, ceci à titre informatif.

Chapitre 1

Pré-requis

Avant de procéder à une compilation des outils, il est nécessaire de disposer d'un environnement permettant les diverses compilations sous Windows : MinGW/Msys et Perl.

MinGW ainsi que Msys sont nécessaires pour pouvoir compiler PostgreSQL et les autres. MinGW "propose" l'utilisation de make ainsi que du compilateur gcc sous Windows. Msys servira surtout pour pouvoir utiliser la commande configure.

NOTE

Les tests de compilation de ce document ont été effectués pour Windows XP Home Edition.

1.1 MinGW

Télécharger MinGW en vous rendant à l'URL suivante :

MinGW <http://prdownloads.sf.net/mingw/MinGW-4.1.0.exe?download>

Il s'agit d'une installation standard. Précisez juste c :\MinGW comme répertoire d'installation. Ajoutez au début de la variable d'environnement PATH de votre machine C :\MinGW\bin ; C :\MinGW\lib

1.2 Msys

Télécharger Msys en vous rendant à l'URL suivante :

Msys <http://prdownloads.sourceforge.net/mingw/MSYS-1.0.10.exe?download>

Lors de l'installation de Msys, choisissez comme proposé par l'installateur : c :\msys\1.0 comme chemin d'installation. Une fenêtre DOS s'ouvrira. Trois questions vous seront poser. Répondez par

question 1 : y

question 2 : y

question 3 : c :/MinGW

1.3 Suppléments : Zlib, GetText, Liblconv,Bison, Flex,wget et MsysTDK

Ces utilitaires/librairies sont aussi nécessaires pour les compilations de PostgreSQL/PostGIS. Ils complèteront notre environnement de travail de MinGW/Msys. Ces outils se trouvent respectivement aux URL suivantes

bison et flex <http://gnuwin32.sourceforge.net/packages.html> (Lien SetUp) qui seront nécessaires pour la compilation du sous-répertoire "contrib" de PostgreSQL. Flex sera nécessaire pour PostGIS ;

msysDTK <http://www.mingw.org/download.shtml> (Fichier .exe) ;

zlib <http://gnuwin32.sourceforge.net/packages.html> (Lien SetUp) qui permettra de pouvoir faire des sauvegardes des données grâce aux utilitaires de PostgreSQL (pg_dump, psql etc...) ;

GetText <http://gnuwin32.sourceforge.net/packages.html> (Lien :SetUp) qui permettra de traduire en français les aides concernant les utilitaires de PostgreSQL ;

LibIconv <http://gnuwin32.sourceforge.net/packages.html> (Lien :SetUp)

wget : <http://gnuwin32.sourceforge.net/packages/wget.htm> (SetUp) qui nous permettra de télécharger les sources en ligne de commande

Lors de leur installation, précisez comme répertoire d'installation c :\MinGW - sauf pour msysDTK qui devra être installé vers le même répertoire d'installation que msys à savoir c :\msys\1.0 - . A titre d'information, les versions des outils sont récapitulées dans le tableau suivant :

MinGW	MinGW-4.10exe
bison	bison-2.1.exe
Flex	flex-2.5.4.a-1.exe
Msys	MSYS-4.0.10.exe
msysDTK	msysDTK-1.0.1.exe
gettext	gettext-0.14.4.exe
libiconv	libiconv-1.8-1.bin.exe
zlib	zlib-1.2.3.exe
wget	wget- ??? .exe

TAB. 1.2 – Versions des outils utilisés

Chapitre 2

Compilation et Installation

Nous allons ici procéder aux diverses compilations et installations. Les diverses étapes doivent avoir lieu dans l'ordre chronologique des sections successives de ce chapitre. Nous commencerons par définir une hiérarchie des répertoires des sources pour pouvoir mieux nous retrouver pour la suite et accueillir les sources que nous aurons téléchargés.

L'installation de nos binaires auront lieu vers le répertoire C :\PostgreSQL\8.2.1.

Nous allons lancer un terminal MinGW. Pour cela, double-cliquez sur C :\msys\1.0\mys.bat

Pour la suite, dans les lignes de commandes à saisir dans MinGW ou dans les répertoires de téléchargement des sources (voir les sous-sections "Téléchargement"), XXX ou \$USERNAME désigne le répertoire C :\msys\1.0\home\XXX que Msys vous aura créé. Regardez pour cela le nom du répertoire contenu dans c :\msys\1.0\home . Notons aussi que, dans les lignes de commandes de ce chapitre, la caractère \ correspondra à ce répertoire. De plus, dans les lignes de commandes, le caractère \ indique que la ligne de commande en cours continue sur la ligne suivante (pour des raisons d'affichage...) Pour la compilation de PostgreSQL et de Geos, veuillez à préparer le café ou de la lecture de magazines :-)

Nous clôturerons ce chapitre par les divers erreurs de compilation reportées par les utilisateurs.

2.1 Crédit des répertoires des sources et du répertoire de destination (facultatif)

Une fois la fenêtre de MinGW lancée, tapez les commandes suivantes

```
cd ~  
  
mkdir sources  
  
mkdir sources/PostgreSQL  
  
mkdir sources/PostGIS  
  
mkdir sources/Geos  
  
mkdir sources/Proj
```

Avec ces commandes, nous mettrons donc nos sources dans c :\msys\1.0\home\XXX\sources. Ces répertoires correspondent au répertoires racines des sources. Sinon pour résumer la création des répertoires précédents, on peut aussi utiliser une boucle FOR :

```
for i in PostgreSQL PostGIS Geos Proj; do mkdir -p ~/sources/$i; done
```



```
MINGW32:~  
david@OLIVIA ~  
$ cd ~  
  
david@OLIVIA ~  
$ mkdir sources  
  
david@OLIVIA ~  
$ mkdir sources/PostgreSQL  
  
david@OLIVIA ~  
$ mkdir sources/PostGIS  
  
david@OLIVIA ~  
$ mkdir sources/Geos  
  
david@OLIVIA ~  
$ mkdir sources/Proj  
  
david@OLIVIA ~  
$ mkdir /c/PostgreSQL  
david@OLIVIA ~  
$
```

FIG. 2.1 – Commandes de création des répertoires

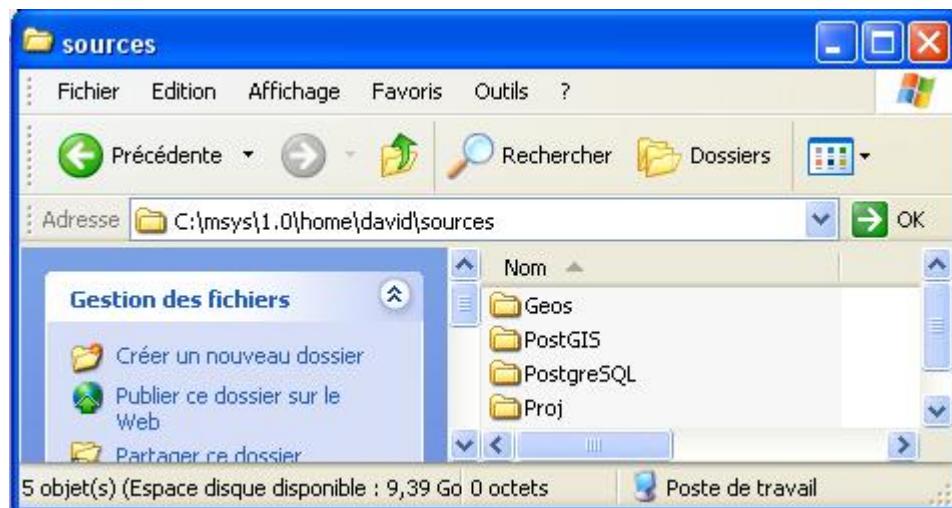


FIG. 2.2 – Les répertoires des sources (crées par MinGW)

2.2 PostgreSQL

2.2.1 Téléchargement

Il faut télécharger les sources de PostgreSQL version 8.2.1 dont l'URL est <ftp://ftp.fr.postgresql.org/source/v8.2.1/postgresql-8.2.1.tar.bz2>. Pour se faire, nous utiliserons les commandes suivantes

```
cd ~/sources/PostgreSQL  
wget ftp://ftp.fr.postgresql.org/source/v8.2.1/postgresql-8.2.1.tar.bz2
```

qui téléchargera nos sources vers **c : \msys\1.0\home\XXX\sources\PostgreSQL**

2.2.2 Compilation et Installation

Depuis MinGW, tapez les commandes suivantes. Arrivé à la commande make, allez-vous préparer un café !!! Y'en a un pour au moins 20 à 40 minutes de compilation selon votre machine.

```
cd ~/sources/PostgreSQL
tar xvjf postgresql-8.2.1.tar.bz2
cd postgresql-8.2.1
configure --enable-nls
make
make install
```

La commande **cd** - tout le monde sait - sert à se rendre dans un répertoire. La commande **tar** sert à décompresser des archives compressés se terminant par .tar, .tar.bz2 ou d'autres formats. La commande **configure** permet de vérifier les dépendances envers/avec les autres outils/librairies présentes dans l'environnement. et de les valider. L'option **--prefix** permet de savoir où sera installée la distribution du matériel. Ici ce sera **/usr/local/pgsql** qui en chemin DOS revient à C :|msys\1.0\local\pgsql. La commande **make** sert à compiler les sources présentes dans les divers sous-répertoires du matériel à installer. La commande **make install**, une fois la compilation réussie va copier les divers binaires résultants (fichiers .dll et .exe etc...) dans les divers répertoires prévues à cet effet.. Dans le cas de PostgreSQL, vous devriez donc vous retrouverez avec les répertoires bin, lib, include, doc, share dans /usr/local/pgsql correspondants à ces fameux répertoires prévus à cet effet..

NOTE

Il existe bien d'autres options possibles que peut prendre en compte PostgreSQL lors de la compilation hors mis celle utilisée ici (**--enable-nls**). Pour obtenir un descriptif des options possibles, vous n'avez qu'à saisir **configure --help**. Je ne propose ici qu'une option parmi d'autres qui ne pose pas de problème sous Windows.

On précisera pour les options passé au script **configure**

--enable-nls : permet d'avoir le support de langue adéquate nls (Native Language Support), notamment le français.

2.2.3 Ajout des outils/supports contributifs pour PostgreSQL

C'est dans le répertoire **contrib** des sources de PostgreSQL que se trouvent les supports (une bonne partie) offrant plus de fonctionnalités type. Pour pouvoir les ajouter à notre distribution, il nous suffira de faire :

```
cd ~/sources/PostgreSQL/postgresql-8.2.1/contrib
make
make install
```

2.3 Geos et Proj

Nous allons maintenant nous intéresser à l'installation de Geos et de Proj

2.3.1 Téléchargement

Il faut télécharger les sources respectives de Geos et de Proj dont les URL respectives sont

<http://geos.refractions.net/geos-2.2.3.tar.bz2>

<ftp://ftp.remotesensing.org/proj/proj-4.5.0.tar.gz>

```
cd ~/sources/Geos
wget http://geos.refractions.net/geos-2.2.3.tar.bz2
cd ~/sources/Proj
wget ftp://ftp.remotesensing.org/proj/proj-4.5.0.tar.gz
```

Les commandes ci-dessus copieront respectivement les sources vers **c : \msys\1.0\home\XXX\sources\Geos** et **c : -\msys\1.0\home\XXX\sources\Proj**

2.3.2 Compilations et Installations

Depuis MinGW, tapez les commandes suivantes. Pour la compilation de Geos, un deuxième café ne serait pas de trop !!!

```
cd ~/sources/Geos
tar xvjf geos-2.2.3.tar.bz2
cd geos-2.2.3
configure --prefix=/usr/local/pgsql && make && make install
cd ~/sources/Proj
tar xvzf proj-4.5.0.tar.gz
cd proj-4.5.0
configure --prefix=/usr/local/pgsql && make && make install
```

2.3.3 Crédit de la DLL pour PROJ

Il faut pour notre distribution créer la librairie **libproj.dll** dans le répertoire **/usr/local/pgsql/lib**

```
cd /usr/local/pgsql/lib
gcc -shared -o libproj.dll -Wl,--out-implib=libproj.dll.a -Wl,--export-all-symbols -Wl, \
--enable-auto-import -Wl,--whole-archive libproj.a -Wl,--no-whole-archive /c/mingw/lib/ \
libmingw32.a
```

NOTE

Depuis la version 2.2.1 de Geos, il n'est plus nécessaire maintenant d'avoir à créer la DLL pour Geos. En effet, celle-ci est automatiquement créée lors de la compilation de Geos.

2.4 PostGIS

Nous allons maintenant nous intéresser à l'installation de PostGIS

2.4.1 Téléchargement

Téléchargez les sources de PostGIS version 1.2.0 à cette URL

<http://postgis.refractions.net/download/postgis-1.2.0.tar.gz>

et copiez ce fichier vers **c :\msys\1.0\home\XXX\sources\PostGIS** en faisant

```
cd ~/sources/PostGIS
wget http://postgis.refractions.net/download/postgis-1.2.0.tar.gz
```

2.4.2 Compilation et Installation

Depuis MinGW, commençons par décompresser les sources :

```
cd ~/sources/PostGIS
tar xvzf postgis-1.2.0.tar.gz
```

Il nous faut temporairement modifier la variable d'environnement PATH pour que les binaires de geos, proj et de PostgreSQL soient accessibles.

```
export PATH=/usr/local/pgsql/bin:/usr/local/pgsql/lib:$PATH
```

Puis depuis MinGW, faites

```
cd ~/sources/PostGIS/postgis-1.2.0
configure
```

Si tout se passe bien, la commande configure devrait se terminer par

```
SUMMARY
-----
HOST_OS: mingw32

PGSQL: /usr/local/pgsql/bin/pg_config
GEOS: /usr/local/pgsql/bin/geos-config (with C-API)
      (ldflags: -L/usr/local/pgsql/lib)
PROJ: prefix=/usr/local/pgsql libdir=/usr/local/pgsql/lib
ICONV: 1 -liconv

PORTNAME: win32
PREFIX: /usr/local/pgsql
EPREFIX: ${prefix}
DOC: /usr/local/pgsql/doc/contrib
DATA: ${datarootdir}
MAN: ${datarootdir}/man
BIN: C:/msys/1.0/local/pgsql/bin
EXT: C:/msys/1.0/local/pgsql/lib (\${libdir})
-----
```

Il se peut que cet affichage diffère du moins par un message concernant datadir mais cela est sans incidence.

Il ne reste plus qu'à compiler et installer PostGIS :

```
cd ~/sources/PostGIS/postgis-1.2.0
make && make install
```

2.5 Finalisation de la distribution, rendre sa distribution portable

Nous allons créer la structure de répertoire C :\PostgreSQL\8.2.1. On pourra aussi le faire depuis MinGW, en faisant

```
mkdir -p /c/PostgreSQL/8.2.1
```

Le reste du document se base sur le fait que nous allons récupérer le contenu du répertoire c :\msys\1.0\local\pgsql et le coller dans le répertoire C :\PostgreSQL\8.2.1 :

```
cd /usr/local/pgsql
cp -r . /c/PostgreSQL/8.2.1/
```

Pour pouvoir rendre votre distribution - tout ce qui est contenu dans le répertoire C :\PostgreSQL\8.2.1 - utilisable sur une nouvelle machine, récupérez (au moins) les DLL suivantes depuis le répertoire C :\MinGW\bin :

libintl3.dll ;
intl-2.dll que vous devrez renommer en intl.dll ;
libiconv2.dll et libiconv-2.dll
zlib1.dll

et copiez les dans le répertoire C :\PostgreSQL\8.2.1\bin. Sinon il suffit de référencer C :\MinGW\bin dans la variable PATH de votre machine pour la suite.

Vous pouvez aussi faire un peu de ménage dans votre nouvelle distribution en supprimant les fichiers libproj*a et libgeos*a contenu dans C :\PostgreSQL\8.2.1\lib

Ca y est, les outils sont installés !!! C'est dans la boite !



FIG. 2.3 – Installation terminée.....C'est dans la boite !

Il ne reste plus qu'à définir l'environnement de travail de PostgreSQL. Ce dernier point sera l'objet du prochain chapitre dans la partie 3 du document.

2.6 Erreurs de compilations reportées par les utilisateurs

Dans cette rubrique, sont mentionnées les erreurs/bugs trouvées par les utilisateurs, que je tiens à remercier pour leurs travaux et leurs contributions. Certaines sont signalées depuis la compilation de PostGIS 1.1.0. Je les signale à titre indicatif.

2.6.1 Erreur de compilation pour PostgreSQL 8.2.1 et PostGIS 1.2.0

Erreur reportée par Hervé QUINQUENEL - le 2007-01-27

```
make -C doc all
make[1]: Entering directory '/home/david/postgresql-8.2.1/doc'
gzip -d -c man.tar.gz | /bin/tar xf -
for file in man1/*.1; do \
    mv $file $file.bak && \
    sed -e 's/\fR(l)/\fR(7)/*' $file.bak >$file && \
    rm -f $file.bak || exit; \
done
/bin/sh.exe ../../config/mkinstalldirs man7
mkdir -p -- man7
for file in man1/*.1; do \
    sed -e '/^\.TH/s/"1"/"7"/' \
        -e 's/\fR(l)/\fR(7)/*' \
        $file >man7/'basename $file | sed 's/.1$/./7/*' || exit; \
done
make[1]: Leaving directory '/home/david/postgresql-8.2.1/doc'
make -C src all
make[1]: Entering directory '/home/david/postgresql-8.2.1/src'
make -C port all
make[2]: Entering directory '/home/david/postgresql-8.2.1/src/port'
gcc -O2 -Wall -Wmissing-prototypes -Wpointer-arith -Winline -Wdeclaration-after-statement - \
    -Wendif-labels -fno-strict-aliasing
-I../../src/port -DFRONTEND -I../../src/include -I./src/include/port/win32 -DEXEC_BACKEND \
    "-I../../src/include/port/win32" -c -o crypt.o crypt.c
gcc -O2 -Wall -Wmissing-prototypes -Wpointer-arith -Winline -Wdeclaration-after-statement - \
    -Wendif-labels -fno-strict-aliasing
-I../../src/port -DFRONTEND -I../../src/include -I./src/include/port/win32 -DEXEC_BACKEND \
    "-I../../src/include/port/win32" -c -o fseeko.o fseeko.c
gcc -O2 -Wall -Wmissing-prototypes -Wpointer-arith -Winline -Wdeclaration-after-statement - \
    -Wendif-labels -fno-strict-aliasing
-I../../src/port -DFRONTEND -I../../src/include -I./src/include/port/win32 -DEXEC_BACKEND \
    "-I../../src/include/port/win32" -c -o getusage.o getusage.c
In file included from ../../src/include/rusagetestub.h:17,
                 from getusage.c:18:
c:/MinGW/bin/../lib/gcc/mingw32/3.4.2/../../../../include/sys/time.h:27: error: \
    redefinition of 'struct timezone'
c:/MinGW/bin/../lib/gcc/mingw32/3.4.2/../../../../include/sys/time.h:40: error: conflicting \
    types for 'gettimeofday'
../../../../src/include/port.h:292: error: previous declaration of 'gettimeofday' was here
c:/MinGW/bin/../lib/gcc/mingw32/3.4.2/../../../../include/sys/time.h:40: error: conflicting \
    types for 'gettimeofday'
../../../../src/include/port.h:292: error: previous declaration of 'gettimeofday' was here
make[2]: *** [getusage.o] Error 1
make[2]: Leaving directory '/home/david/postgresql-8.2.1/src/port'
make[1]: *** [all] Error 2
make[1]: Leaving directory '/home/david/postgresql-8.2.1/src'
make: *** [all] Error 2
```

Il faut en fait utiliser MinGW 4.1.0 et Msys 4.0.10 et pas de version de MinGW 5.0.X.

2.6.2 Erreur de libiconv

Erreur reportée par Alain FERCHAL - le 2006-01-03 - pour PostGIS 1.1.0

Lors d'un certain make install, l'erreur suivante se produit

["Le point d'entrée de procedure libiconv_close est introuvable dans la bibliothèque de liaisons dynamique libiconv-2.dll"]

Après une recherche sur internet, le mieux semble de télécharger ce fichier diffutils-2.8.7-1-dep.zip disponible à cette adresse <http://ovh.dl.sourceforge.net/sourceforge/gnuwin32/diffutils-2.8.7-1-dep.zip>

Il contient un fichier libiconv2.dll qu'il faut ensuite placer dans C :\MinGW\bin.

Page d'info trouvée sur internet : <http://projects.edgewall.com/trac/ticket/2233>

2.6.3 Erreur de compilation avec PostGIS pour pgsql2shp.exe

Erreur reportée par Jamal BERRICH - le 2006-01-03 - pour PostGIS 1.1.0

Lors de la compilation de PostGIS, on obtenait à un moment l'erreur suivante

[gcc.exe : C :\msys/1.0/libpq.dll : No such file or directory]

dûe à une erreur de frappe de ma part dans le fichier postgis-1.1.0/configure.in où j'avais écrit

PGFELIBS="\$echo \${\${PGCONFIG} --libdir}/libpq.dll"

au lieu de

PGFELIBS="\$echo \${\${PG_CONFIG} --libdir}/libpq.dll"

Erreur corrigée !

2.6.4 Erreur de PostGIS avec make

Erreur reportée par Alain FERCHAL - le 2006-01-03 - pour PostGIS 1.1.0

Si vous obtenez l'erreur suivante

```
$ make
./autogen.sh
Can't locate Autom4te/General.pm in @INC (@INC contains : /usr/share/autoconf /usr/lib/perl5/5.6.1/msys /usr/
lib/perl5/5.6.1 /usr/lib/perl5/site_perl/5.6.1/msys /usr/lib/perl5/site_perl/5.6.1 /usr/lib/perl5/site_perl .) at /usr/bin/au-
tom4te line 40.
BEGIN failed--compilation aborted at /usr/bin/autom4te line 40
make : *** [configure] Error 2
```

Le mieux c'est de télécharger la bonne version de msysDTK à savoir la 1.0.1 et non la 1.0.0 ou autre

Deuxième partie

Installation sous GNU/Linux, Ubuntu Edgy Eft

Chapitre 3

Installation sous Ubuntu Edgy Eft

Etant un petit utilisateur sans prétention de GNU/Linux, j'expose ici la façon d'installer nos outils pour Ubuntu Edgy Eft

3.1 Pré-requis

En tant que root

```
apt-get update
apt-get install gcc-4.1 g++-4.1 flex bison libreadline5-dev libssl-dev zlib1g-dev gettext ←
gettext-base
```

avec

1. gcc-4.1,g++-4.1,make : pour pouvoir bénéficier d'un environnement de compilation complet. On s'assurera ici d'avoir au moins la version 3.81 de make (commande : make --version) ;
2. flex,bison : si nous devions travailler avec les dépôts CVS de postgresql ;
3. gettext,gettext-base : pour avoir le support de langue en français (NLS : Native Support Language)
4. zlib1g-dev : la librairie zlib est nécessaire si vous voulez pouvoir faire des sauvegarde de vos données par la suite (pour l'utilitaire pg_dump de PostgreSQL)
5. libreadline5-dev : la librairie readline est nécessaire pour pouvoir bénéficier de l'historique des commandes, de l'édition de la ligne en cours) avec l'utilitaire psql de PostgreSQL.

NOTES AUX UTILISATEURS UBUNTU

Les utilisateurs Ubuntu en particulier m'excuseront ici de ne pas faire usage ici de la commande sudo. Etant avant tout un utilisateur Debian avant tout je partais dans ma documentation du principe que vous savez comment activer le su pour le root. Contrairement à la philosophie Ubuntu, on privilégie le apt-get install, comme j'ai toujours eu l'habitude de compiler mes outils préférés avec les sources, c'est donc de la compilation de source dont il s'agira ici.

3.2 PostgreSQL

Nous commencerons par télécharger les sources de ce dernier

Pour obtenir un descriptif d'une compilation/installation standard de PostgreSQL, vous pouvez aussi consulter <http://docs.postgresqlfr.org/8.2/INSTALL.html>

```
wget ftp://ftp.fr.postgresql.org/v8.2.0/postgresql-8.2.0.tar.bz2
```

que l'on décomprime ensuite

```
tar xvjf postgresql-8.2.0.tar.bz2
```

Commençons donc par configurer PostgreSQL avec les

```
cd postgresql-8.2.0
CC=gcc-4.1 CXX=g++-4.1 ./configure --enable-nls=fr --with-openssl
```

Il ne nous reste plus qu'à le compiler et à l'installer

```
make
make install
```

NOTE

1. Par défaut je rappelle que PostgreSQL a /usr/localpgsql comme répertoire d'installation si l'on ne précise pas le répertoire d'installation dans la commande ./configure avec l'option --prefix.
 2. Pour obtenir un descriptif possible des diverses options, je rappelle que celui-ci peut-être obtenu en faisant ./configure --help
-

N'oublions pas comme à notre habitude de compiler/installer les modules de contribution

```
cd contrib
make
make install
```

3.2.1 Téléchargement et configuration

Nous commencerons par télécharger les sources de ce dernier

```
wget ftp://ftp.fr.postgresql.org/v8.2.0/postgresql-8.2.0.tar.bz2
```

que l'on décomprime ensuite

```
tar xvjf postgresql-8.2.0.tar.bz2
```

Commençons donc par configurer PostgreSQL avec les

```
cd postgresql-8.2.0
CC=gcc-4.1 CXX=g++-4.1 ./configure --enable-nls=fr --with-openssl
```

Il ne nous reste plus qu'à le compiler en faisant

```
make
```

3.2.2 Test de régression (optionnel)

Pour pouvoir réaliser les tests de régression il est nécessaire de créer un utilisateur postgres

```
root@jenna-edgy:/mnt/sources/postgresql-8.2.0# adduser postgres
Ajout de l'utilisateur « postgres »...
Ajout du nouveau groupe « postgres » (1001).
Ajout du nouvel utilisateur « postgres » (1001) avec le groupe « postgres ».
Création du répertoire personnel « /home/postgres ».
Copie des fichiers depuis « /etc/skel »

Enter new UNIX password:
Retype new UNIX password:
passwd : le mot de passe a été mis à jour avec succès
Modification des informations relatives à l'utilisateur postgres
Entrez la nouvelle valeur ou « Entrée » pour conserver la valeur proposée
    Nom complet []:
    No de bureau []:
    Téléphone professionnel []:
    Téléphone personnel []:
    Autre []

Ces informations sont-elles correctes [o/N] ? o
```

Rendons cet utilisateur propriétaire du répertoire des sources

```
root@jenna-edgy:/mnt/sources/postgresql-8.2.0# chown postgres:postgres/mnt/sources/ ←
    postgresql-8.2.0 -R
```

Connectons-nous maintenant en tant que utilisateur postgres et lançons les test de régression

```
su postgres
make check
```

qui renverra en affichage

```
===== removing existing temp installation =====
===== creating temporary installation =====
===== initializing database system =====
===== starting postmaster =====
running on port 55432 with pid 26379
===== creating database "regression" =====
CREATE DATABASE
ALTER DATABASE
===== installing plpgsql =====
CREATE LANGUAGE
.
.
.
.
test stats ... ok
test tablespace ... ok
===== shutting down postmaster =====
server stopped

=====
All 103 tests passed.
=====
```

3.2.3 Installation

Procérons maintenant à l'installation

```
make install
```

NOTE

1. Par défaut je rappelle que PostgreSQL a `/usr/local/pgsql` comme répertoire d'installation si l'on ne précise pas le répertoire d'installation dans la commande `./configure` avec l'option `--prefix`.
2. Pour obtenir un descriptif possible des diverses options, je rappelle que celui-ci peut-être obtenu en faisant `./configure --help`

N'oublions pas comme à notre habitude de compiler/installer les modules de contribution

```
cd contrib  
make  
make install
```

Pour finir, nous ferons

1. pour que nos librairies soient accessibles

```
echo /usr/local/pgsql/lib >> /etc/ld.so.conf
```

2. pour que les outils de PostgreSQL soient accessibles, il suffit d'ouvrir `/etc/bash.bashrc` et d'ajouter les lignes suivantes à la fin de ce fichier et afin de renseigner le répertoire qui contiendra nos données (`cd PGDATA`)

```
export PATH="/usr/local/pgsql/bin:$PATH"  
export PGDATA="/usr/local/pgsql/data"
```

3.3 Geos et Proj

C'est une installation des plus classiques que je détaille par ici

```
wget http://geos.refractions.net/geos-3.0.0rc3.tar.bz2  
tar xjf geos-3.0.0rc3.tar.bz2  
cd geos-3.0.0rc3  
CC=gcc-4.1 CXX=g++-4.1 ./configure && make && make install
```

de même pour Proj

```
wget ftp://ftp.remotesensing.org/proj/proj-4.5.0.tar.gz  
tar xzf proj-4.5.0.tar.gz  
cd proj-4.5.0  
CC=gcc-4.1 CXX=g++-4.1 ./configure && make && make install
```

puis

```
echo /usr/local/lib >> /etc/ld.so.conf  
ldconfig
```

3.4 PostGIS

Si vous n'avez pas encore effectué la mise à jour de votre variable PATH dans `/etc/bash.bashrc` [section 2.3], tapez la commande suivante pour rendre temporairement accessibles les binaires de PostgreSQL

```
export PATH="/usr/local/pgsql/bin:$PATH"

wget http://postgis.refractions.net/download/postgis-1.2.0.tar.gz
tar xzf postgis-1.2.0.tar.gz
cd postgis-1.2.0
CC=gcc-4.1 CXX=g++-4.1 ./configure && make && make install
```

Troisième partie

PostgreSQL - configuration et administration minimale

Chapitre 4

Paramétrer PostgreSQL sous Windows et GNU/Linux

Dans les chapitre précédents des parties I et II, nous avons effectué les diverses installations nécessaires de nos outils. Afin de tirer profit de notre installation, il est maintenant temps de passer à la finalisation qui va concerner le serveur PostgreSQL. Afin que ce dernier soit fonctionnel, il est nécessaire de définir diverses variables d'environnement de PostgreSQL, de définir le super-utilisateur de PostgreSQL qui fera dans un premier temps son initialisation, d'installer le service etc... Nous verrons ces divers points en détail dans ce chapitre.

Dans le cas de l'utilisation de PostgreSQL, surtout les aspects techniques, - et surtout l'aspect traitant des rôles dans PostgreSQL englobant la notion d'utilisateur que j'avoue avoir négligé dans cette documentation faute de temps, je ne saurais trop vous conseiller la traduction de la documentation réalisée par Guillaume LELARGE, qui effectue un travail de traduction incroyable, disponible à l'adresse suivante <http://traduc.postgresqlfr.org/pgsql-8.2.1-fr/>

4.1 Se procurer une distribution déjà compilée - Optionnel pour Windows -

Si vous n'avez pas suivi les deux premiers chapitres pour les diverses compilations nécessaires, il vous est toujours possible de disposer d'une distribution conforme aux deux premiers chapitres. Cette distribution est disponible à l'adresse suivante

http://www.davidgis.fr/download/postgresql-8.2.1_postgis-1.2.0_geos-2.2.3_proj-4.5.0.zip

Pour être en conformité avec le début de la documentation, créez-vous le chemin PostgreSQL\8.2.1 sur votre partition C : ou D : de votre ordinateur en vérifiant que le système de fichier de la partition est du NTFS. Vous obtiendrez donc C :\PostgreSQL\8.2.1 ou D :\PostgreSQL\8.2.1. Décompressez ensuite le fichier téléchargé dans le répertoire en question.

NOTE

Il est possible de choisir un autre répertoire pour l'installation de la distribution. Si tel est votre choix, il faudra faire attention et adapter les commandes/instructions de la documentation au choix de répertoire que vous aurez décidé.

Pour être en conformité avec la suite de la documentation, il est aussi demandé de disposer au moins de MinGW et de Msys (voir chapitre 1)

4.2 Les variables d'environnement de PostgreSQL

Que vous soyez sous Windows XP ou Windows 2000, créez les variables d'environnement suivantes - propres à PostgreSQL - :

Exemple 4.1 Exemples de variable d'environnement sous Windows

PGPORT= 5432

PGDATA = C :\PostgreSQL\8.2.1\data

Exemple 4.2 Exemple de variables d'environnement sous GNU/Linux

PGPORT= 5432

PGDATA = /usr/local/pgsql/data

PGDATA permettra à PostgreSQL de connaître le répertoire racine des données des diverses bases [...]. PGPORT lui dire sur quel port écouter. L'intérêt de ces variables sera vu plus tard.

Ajoutez aussi au début de votre variable **PATH** les chemins d'accès

Exemple 4.3 Modification de la variable d'environnement PATH pour Windows,ajout de PGDATA

PATH=C :\PostgreSQL\8.2.1\bin ;C :\PostgreSQL\8.2.1\lib ;(le reste de votre PATH usuel)

PGDATA=C :\PostgreSQL\8.2.1\data

Les utilisateurs Windows veilleront à avoir effectuer les modifications ci-dessus. De même, les utilisateurs GNU/linux veilleront à avoir effectué les modifications ci-dessous pour la suite du chapitre.

Exemple 4.4 Modification de la variable d'environnement PATH, ajout de PGDATA pour GNU/linux dans /etc/bash.bashrc

```
...
... (en fin de fichier)
export PATH="/usr/local/pgsql/bin:$PATH"
export PGDATA="/usr/local/pgsql/data"
```

Cet ajout en début de votre variable PATH permettra à PostgreSQL d'accéder aux divers utilitaires accompagnant la distribution (cf. répertoire bin) mais aussi aux diverses bibliothèques - fichiers *.dll - (cf. répertoires de lib). Il sera aussi possible par la suite d'accéder aux bibliothèques de Geos et de Proj.

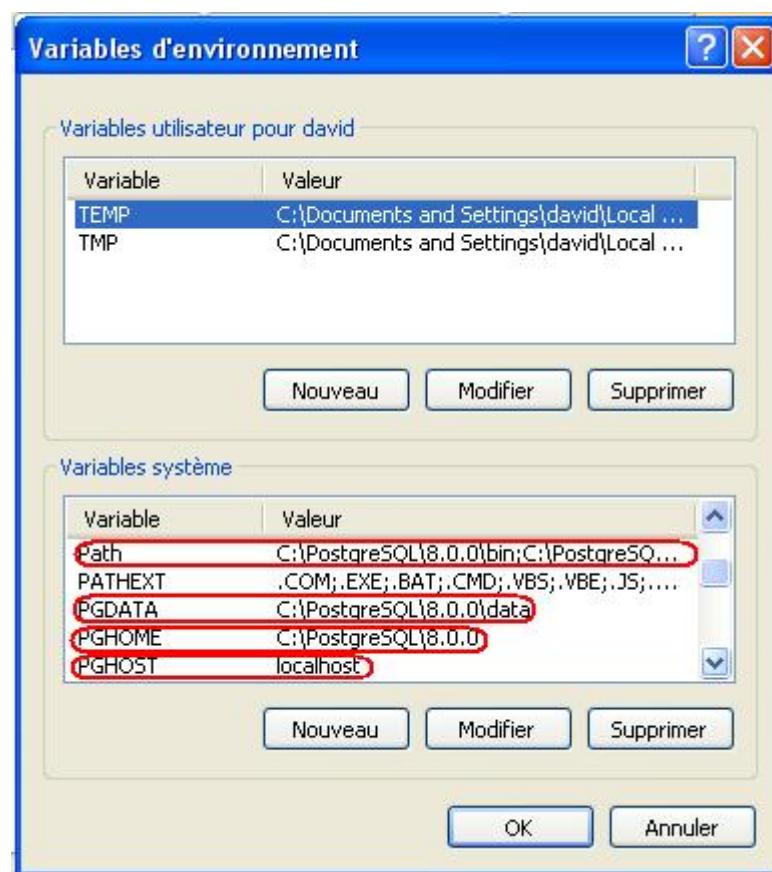


FIG. 4.1 – Exemple d'emploi de variables d'environnement de PostgreSQL 8.0.0 pour une connexion cliente. À adapter ici pour la version 8.2.1

IMPORTANT

Variables d'environnement lors d'une session client à un serveur distant
PostgreSQL utilise d'autres variables qui lui sont propres. Si lors d'un travail répété, vous seriez par exemple amener à vous connecter souvent au même serveur distant et à la même base sur ce serveur - à condition toutefois que votre connexion au serveur soit acceptée par ce dernier -, il vous est aussi possible de définir les variables PGUSER, PGDATABASE et PGHOST à nouveau.

Si vous accédez au serveur temporairement au travers d'une fenêtre DOS, il vous est par exemple possible de définir

```
set PGDATABASE=votre_base_distante
set PGHOST=IP_du_serveur
set PGPORT=port_sur_lequel_écouter
set PGUSER=utilisateur_PostgreSQL_sur_le_serveur
```

et d'utiliser psql le moniteur interactif de PostgreSQL pour faire vos requêtes. Nous verrons cet utilitaire au chapitre suivant.

Pour les utilisateurs GNU/Linux, il suffira tout simplement de remplacer le mot-clé **set** par **export**

4.3 Le super-utilisateur de PostgreSQL

Nous allons ici voir quelques règles régissant les droits avec le serveur PostgreSQL puis nous créerons - si besoin est - ce super-utilisateur.

4.3.1 Le super-utilisateur ?

Comme nous l'énoncions, au début de ce chapitre, la prochaine étape concerne l'initialisation du répertoire qui accueillera nos futures bases de données : PGDATA. Pour l'instant, ce répertoire n'existe pas encore. Or avec PostgreSQL, seul le super-utilisateur a le droit de le créer.

Mais alors qui est ce super-utilisateur ?

Ce super-utilisateur - comme première propriété - doit être un utilisateur physique de votre machine. Oui mais depuis la version 8.2.1 de PostgreSQL, les programmeurs de PostgreSQL pour des raisons de sécurité (impliquant des failles/fuites de sécurités selon les droits accordés à cet utilisateur sous Linux qui peuvent être dûs à des attaques de requêtes de type "SQL injection" par le réseau ou par une mauvaise utilisation des modules objets chargés dans les bases par de simples utilisateurs etc...) ont imposés que cet utilisateur devait avoir des droits limités sur la machine sur lequel tourne PostgreSQL.

Or, sous Windows, on distingue deux types d'utilisateurs :

ceux qui ont des droits administratifs - les administrateurs - ils peuvent tout faire sur la machine, installer/désinstaller des logiciels, lancer des services, supprimer des utilisateurs, modifier les variables d'environnement physiques etc...

ceux qui ont des droits limités : ils peuvent accéder aux logiciels mis à leur disposition par les administrateurs, ne peuvent accéder à certains répertoires, etc...

Le super-utilisateur - comme seconde propriété - sera donc un utilisateur ayant des droits limités

Oui mais - comme troisième propriété - comme son titre l'indique, il sera le grand manitou, le gourou au sens de PostgreSQL de la secte des utilisateurs de PostgreSQL. Il a le droit de créer des utilisateurs, d'étendre/restreindre leur droit de lecture/modification aux bases de données du serveur. Il aura aussi le droit de leur accorder le droit de créer ou non des bases par exemple, de leur attribuer des mots de passe etc...En un mot, il aura droit de vie/mort sur ses sujets :-)

Mais ici, nous resterons dans la politique dite du "trusting". Je m'explique. Pour la suite du document je vais supposer que vous êtes administrateur sur la machine sur laquelle vous avez installé PostgreSQL et les autres outils. Nous allons demander au super-utilisateur de nous confier ses supers pouvoirs (à vous) et de faire confiance à toute machine du réseau qui se connecterait au serveur PostgreSQL sans demander de mot de passe à sa majesté.

Il ne reste donc plus en première instance qu'à créer ce fameux super-utilisateur.

4.3.2 Crédation du super-utilisateur

Sur votre machine, voyez si s'il n'existe pas déjà un utilisateur ayant des droits limités et possédant un mot de passe. Si ce dernier n'existe pas, créez-le. Attention, il est impératif que cet utilisateur ait des droits limités sur la machine et qu'il ait un mot de passe.

Pour faciliter la lecture des futures commandes de ce document, j'appelerais postgres le super-utilisateur de PostgreSQL et david un administrateur sous Windows - qui vous correspondra sous Windows. Sous Windows, postgres et david auront comme mot de passe empr888. Donc, pour récapituler - même si je me répète -, postgres est un utilisateur limité sous Windows et david est administrateur de la machine (vous correspondant) à qui on souhaite conférer les pouvoirs de super-utilisateur de postgres le moment venu. Vos futures commandes seront donc à modifier en fonction des spécifications ici mentionnées. Les lignes de commandes permettant de distinguer les deux fenêtres des utilisateurs commenceront par '# sous david :' (correspondant à votre session) ou bien '# sous postges :' (correspondant à la fenêtre du super-utilisateur que vous aurez plus tard). TOUTES LES COMMANDES SUIVANTES SONT A SAISIR DANS DES FENÊTRES DOS.

Sous GNU/Linux, la création d'un tel utilisateur a déjà été mentionné dans Partie II, chapitre 3, section 2-2. Si tel n'était pas le cas (vous auriez sauté cette étape), pour créer l'utilisateur postgres, il suffit de faire

```
root@jenna-edgy:/mnt/sources/postgresql-8.2.0# adduser postgres
Ajout de l'utilisateur « postgres »...
Ajout du nouveau groupe « postgres » (1001).
Ajout du nouvel utilisateur « postgres » (1001) avec le groupe « postgres ».
Création du répertoire personnel « /home/postgres ».
Copie des fichiers depuis « /etc/skel »
Enter new UNIX password:
Retype new UNIX password:
passwd : le mot de passe a été mis à jour avec succès
Modification des informations relatives à l'utilisateur postgres
Entrez la nouvelle valeur ou « Entrée » pour conserver la valeur proposée
    Nom complet []:
    No de bureau []:
    Téléphone professionnel []:
    Téléphone personnel []:
    Autre []:
Ces informations sont-elles correctes [o/N] ? o
```

Les utilisateurs GNU/Linux peuvent ignorer la prochaine sous-section en tapant tout simplement su postgres

4.3.3 Pouvoir accéder à la session du super-utilisateur sous DOS sans changer la session en cours

Imaginons que nous soyons connectés sous Windows sous la session de david. La prochaine étape va constituer à initialiser le répertoire PGDATA. Ceci se fait en ouvrant normalement une fenêtre DOS sous la session de postgres. Pour éviter de changer de session et accéder plus facilement à cette fenêtre, ceci est possible depuis la session de david.

Pour cela, il faut utiliser la commande runas de Windows. Je sais qu'elle existe sous Windows XP mais je ne garantis pas ici son existence sous Windows 2000. L'obtention de la fenêtre en question se fait en saisissant dans une fenêtre DOS :

```
# sous david:
runas /user:postgres cmd
```

Il suffit alors comme demandé de saisir le mot de passe de postgres.

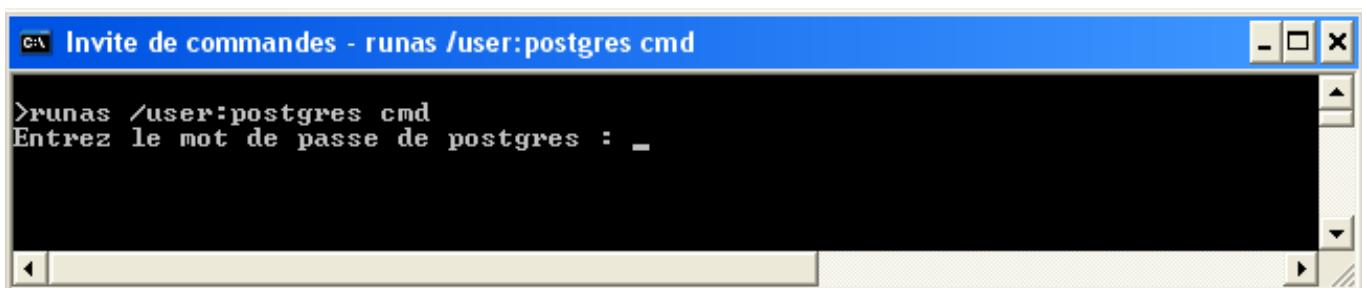


FIG. 4.2 – Utilisation de la commande runas

Une nouvelle fenêtre DOS appartenant à postgres devrait s'ouvrir. C'est cette fenêtre qui désignera comme nous l'avons dit '# sous postgres' maintenant

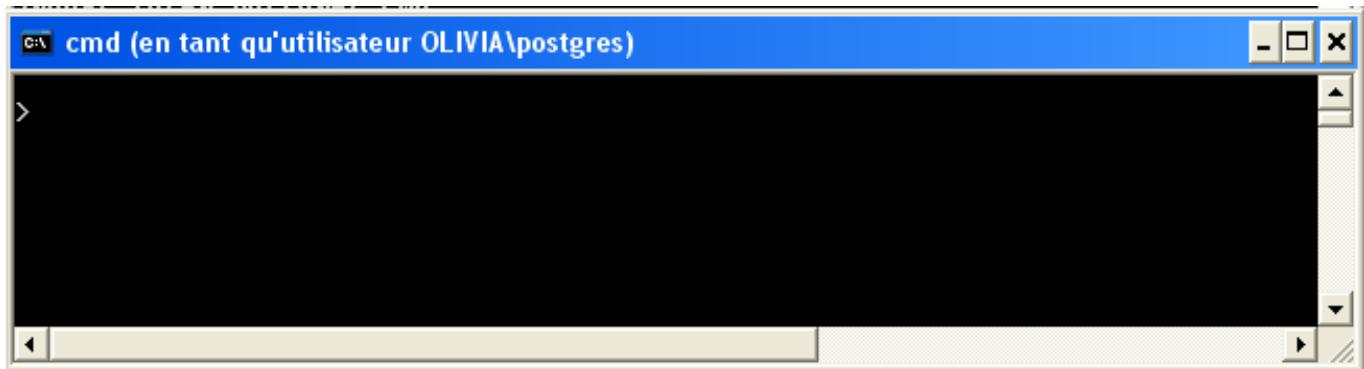


FIG. 4.3 – Nouvelle fenêtre DOS obtenue

AVERTISSEMENT

Code 850 en DOS et 1252 et Police Lucida Console – facultatif –

Certains utilitaires de PostgreSQL (psql, initdb...) que vous verrez plus tard ne sont pas adaptés aux fenêtres DOS pour afficher correctement les caractères sur huit bits. Si nous laissons l'affichage par défaut actif de la fenêtre DOS dont le code par défaut est 850, certains caractères risquent de ne pas s'afficher correctement. On risquerait par exemple d'avoir 'é' affiché sous forme de 'ù'.

Le mieux est de mettre la police en **Lucida Console** et de mettre le code de page de DOS en 1252. Le changement de Police peut se faire en faisant un clic-droit sur la barre bleue de la fenêtre et de choisir «Propriétés». Puis dans l'onglet «Police», cochez «Lucida Console». Dans DOS, tapez ensuite la commande suivante : **cmd.exe /k chcp 1252**



4.4 Initialisation de PostgreSQL : groupe de bases de données par défaut (Windows)

L'initialisation - au sens du système de fichiers - consiste à la création du (premier) groupe de vos futures bases de données par défaut. Cela va consister "à remplir" le répertoire correspondant à la variable d'environnement PGDATA de PostgreSQL qui n'existe pas encore. Elle répond aux besoins suivants. Quel encodage choisir par défaut ? Quel doit être le mode de connexion par défaut ? etc... Mais le reste des configurations des paramétrages du serveur peut par la suite être reprise dans les fichiers .conf de PostgreSQL.

Pour l'initialisation, il faut utiliser l'outil initdb.

Dans la fenêtre de postgres,

Exemple 4.5 Mise en place du cluster pour Windows

```
# sous postgres:  
initdb -A trust -E SQL_ASCII
```

Par défaut ici, nous utiliserons le jeu d'encodage SQL_ASCII (cf. **-E SQL_ASCII**) et autoriserons toute personne voulant se connecter au serveur par le réseau sans demande de mot de passe (cf. **-A trust**).

The screenshot shows a Windows Command Prompt window titled "cmd (en tant qu'utilisateur OLIVIA\postgres) - cmd /k chcp 1252". The window displays the output of the "initdb" command, which initializes a new PostgreSQL cluster. The output includes various configuration steps such as creating directories, setting locale, and initializing system tables. It concludes with instructions to start the server using "postmaster" or "pg_ctl".

```
>initdb -A trust -E SQL_ASCII
Les fichiers appartenant à ce système de bases de données doivent appartenir à l'utilisateur.
Cet utilisateur doit aussi posséder le processus serveur.

Le groupe de bases de données sera initialisé avec la locale French_France.1252.

création du répertoire C:/PostgreSQL/8.0.3/data ... ok
création du répertoire C:/PostgreSQL/8.0.3/data/global... ok
création du répertoire C:/PostgreSQL/8.0.3/data/pg_xlog... ok
création du répertoire C:/PostgreSQL/8.0.3/data/pg_xlog/archive_status... ok
création du répertoire C:/PostgreSQL/8.0.3/data/pg_clog... ok
création du répertoire C:/PostgreSQL/8.0.3/data/pg_subtrans... ok
création du répertoire C:/PostgreSQL/8.0.3/data/base... ok
création du répertoire C:/PostgreSQL/8.0.3/data/base/1... ok
création du répertoire C:/PostgreSQL/8.0.3/data/pg_tblspc... ok
sélection de la valeur par défaut de max_connections... 100
sélection de la valeur par défaut de shared_buffers... 1000
création des fichiers de configuration... ok
création de la base de données template1 dans C:/PostgreSQL/8.0.3/data/base/1... ok
initialisation de pg_shadow... ok
activation de la taille illimitée des lignes pour les tables systèmes... ok
initialisation de pg_depend... ok
création des vues système... ok
chargement de pg_description... ok
création des conversions... ok
initialisation des priviléges sur les objets intégrés... ok
création du schéma d'informations... ok
lancement du vacuum sur la base de données template1... ok
copie de template1 vers template0... ok

Succès. Vous pouvez maintenant lancer le serveur de bases de données en utilisant:

    ""postmaster -D "C:/PostgreSQL/8.0.3/data"
ou
    ""pg_ctl -D "C:/PostgreSQL/8.0.3/data" -l journaltrace start

>
```

FIG. 4.4 – Initialisation du groupe de bases de données pour PostgreSQL 8.0.3 (ancienne image)

4.5 Initialisation de PostgreSQL : groupe de bases de données par défaut (GNU/Linux)

La mise en place du cluster diffère légèrement par rapport à celle de Windows ici. Il faut avant tout commencer par créer le répertoire nécessaire au cluster. En tant que root faites :

Exemple 4.6 Création du répertoire pour les bases de données

```
mkdir /usr/local/pgsql/data
chown postgres:postgres /usr/local/pgsql/data
```

NOTE

Si vos variables PATH et PGDATA sont à jour, vous devriez à la place pouvoir saisir

```
mkdir $PGDATA
chown postgres:postgres $PGDATA
```

Maintenant, il faut se logguer en tant que postgres pour pouvoir lancer la commande de initdb

```
su postgres
initdb -A trust
```

L'encodage par défaut sous Ubuntu est UTF-8. Vos futures bases seront donc encodées par défaut en UTF-8. Un moyen de vérifier cela est de faire

```
# locale
LANG=fr_FR.UTF-8
LANGUAGE=fr_FR:fr:en_GB:en
LC_CTYPE="fr_FR.UTF-8"
LC_NUMERIC="fr_FR.UTF-8"
LC_TIME="fr_FR.UTF-8"
LC_COLLATE="fr_FR.UTF-8"
LC_MONETARY="fr_FR.UTF-8"
LC_MESSAGES="fr_FR.UTF-8"
LC_PAPER="fr_FR.UTF-8"
LC_NAME="fr_FR.UTF-8"
LC_ADDRESS="fr_FR.UTF-8"
LC_TELEPHONE="fr_FR.UTF-8"
LC_MEASUREMENT="fr_FR.UTF-8"
LC_IDENTIFICATION="fr_FR.UTF-8"
LC_ALL=
```

Si pour la suite de ce document, vous préférez avoir des bases encodées LATIN9 ou LATIN1, il sera toujours possible de le faire. Pour avoir de plus amples précisions sur l'emploi de intidb, comme toujours man initdb

4.6 Autoriser les connexions TCP/IP et Démarrer/Arrêter le serveur

Il existe deux types de démarrage soit par la méthode manuelle ou en paramétrant PostgreSQL en tant que service sous Windows. Voyons en détail les deux méthodes. Il vous faudra choisir l'une des deux.

4.6.1 Autoriser les connexions TCP/IP

Avant la version 7.5 devel de PostgreSQL - version qui n'existe plus aujourd'hui -, l'ancienne façon pour autoriser les connexions TCP/IP était de fournir l'option **-o -i** à l'outil en ligne de commande pg_ctl. Depuis la version 8.0, cette méthode a été délaissée au profit d'une autre méthode. La manière la plus élégante pour autoriser les connexions TCP/IP est de modifier le paramètre listen_addresses du fichier sous Windows c :\PostgreSQL\8.2.1\data\postgresql.conf, respectivement /usr/local/postgresql/data/postgresql.conf pour GNU/Linux.

Ouvrez ce fichier et remplacez la ligne

```
#listen_addresses = 'localhost' # what IP interface(s) to listen on;
```

par

```
listen_addresses = '*' # what IP interface(s) to listen on;
```

4.6.2 Méthode 1 : Démarrage/Arrêt manuel

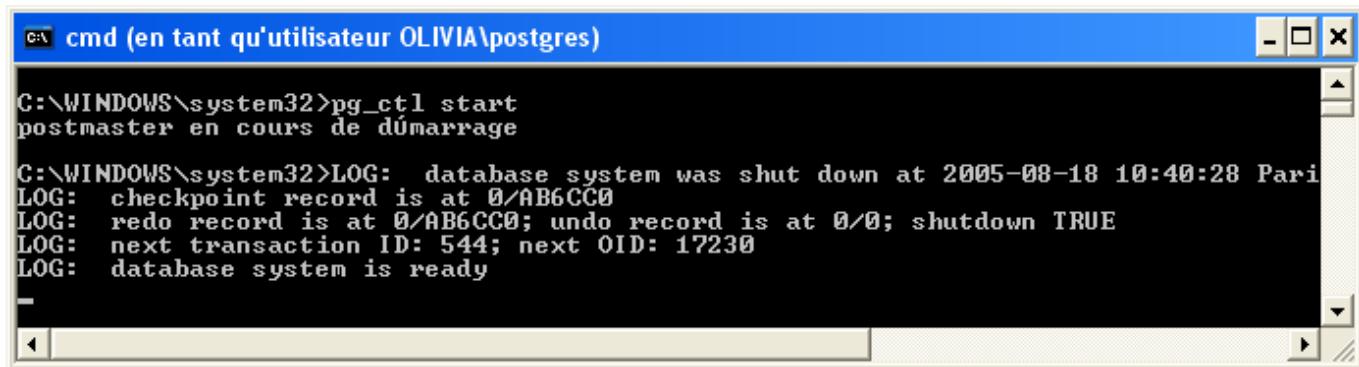
Le démarrage manuel impose que vous devez démarrer PostgreSQL à chaque démarrage physique de votre machine. Pour cela, il faut utiliser l'outil pg_ctl . Si tel est votre choix, pour démarrer PostgreSQL

```
# sous postgres:  
pg_ctl start
```

Pour arrêter le serveur, il suffit de taper :

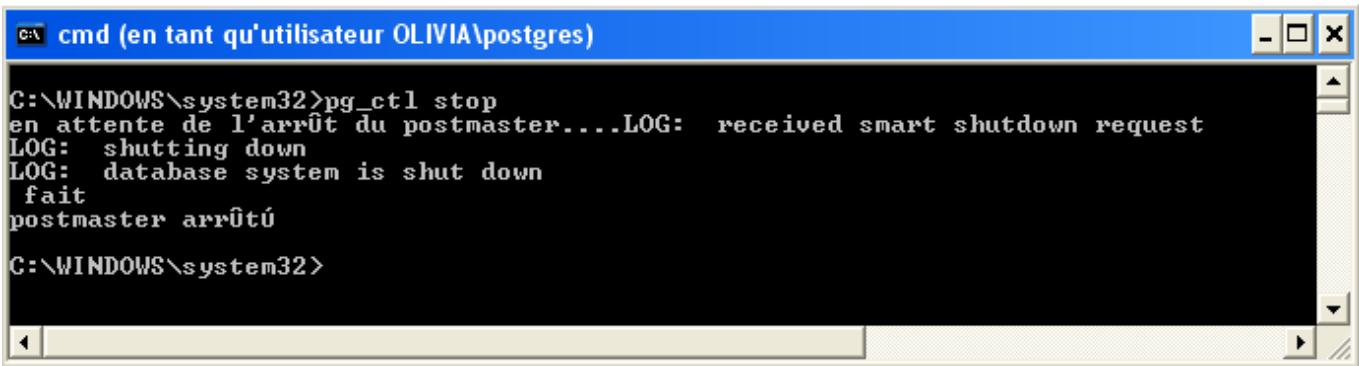
```
# sous postgres:  
pg_ctl stop
```

N'oubliez que si vous choisissez cette méthode, vous devez d'abord saisir la commande donnée avec runas si vous démarrez votre machine depuis votre session habituelle (cd. david ici).



```
C:\>pg_ctl start  
postmaster en cours de démarrage  
  
C:\>LOG: database system was shut down at 2005-08-18 10:40:28 Paris  
LOG: checkpoint record is at 0/AB6CC0  
LOG: redo record is at 0/AB6CC0; undo record is at 0/0; shutdown TRUE  
LOG: next transaction ID: 544; next OID: 17230  
LOG: database system is ready
```

FIG. 4.5 – Démarrage manuel de PostgreSQL



```
C:\>pg_ctl stop  
en attente de l'arrêt du postmaster....LOG: received smart shutdown request  
LOG: shutting down  
LOG: database system is shut down  
fait  
postmaster arrêté  
  
C:\>
```

FIG. 4.6 – Arrêt manuel de PostgreSQL

4.6.3 Méthode 2 : Démarrage/Arrêt automatique pour Windows

Avec cette méthode, vous n'aurez pas besoin de vous soucier de démarrer PostgreSQL à chaque démarrage de votre machine. En effet, nous allons ici procéder à l'installation du service Windows, livré avec PostgreSQL. C'est toujours pg_ctl que nous utiliserons avec l'option register

NOTE

Avant de procéder à l'installation du service, dans le cas éventuel où vous auriez testé le démarrage manuel(cf **pg_ctl start**), assurez-vous d'avoir arrêter le serveur par la commande **pg_ctl stop**

4.6.3.1 Installation du service

Nous allons créer le service nommé postgresqlwin32. C'est le super-utilisateur qui doit avoir le droit sous Windows d'installer les services. Donc nous ferons depuis une fenêtre DOS de david :

```
# sous david:  
pg_ctl register -N postgresqlwin32 -U postgres -P empr888
```

Le service est installé. Il faut maintenant le lancer. Ne pas oublier qu'ici «empr888» est le mot de passe de postgres.

ATTENTION AU PARE-FEU (FIREWALL) : L'utilisation express de service sous Windows implique que votre logiciel de pare-feu - si vous en avez un - autorise de lancer le service sur le port 5432 (cf. PGPORT) qui est le port sur lequel PostgreSQL écoute. Avant de continuer, configurez votre logiciel de pare-feu/firewall pour qu'il autorise cette opération.

4.6.3.2 Lancer le service

cela se fait en tapant

```
# sous david:  
net start postgresqlwin32
```

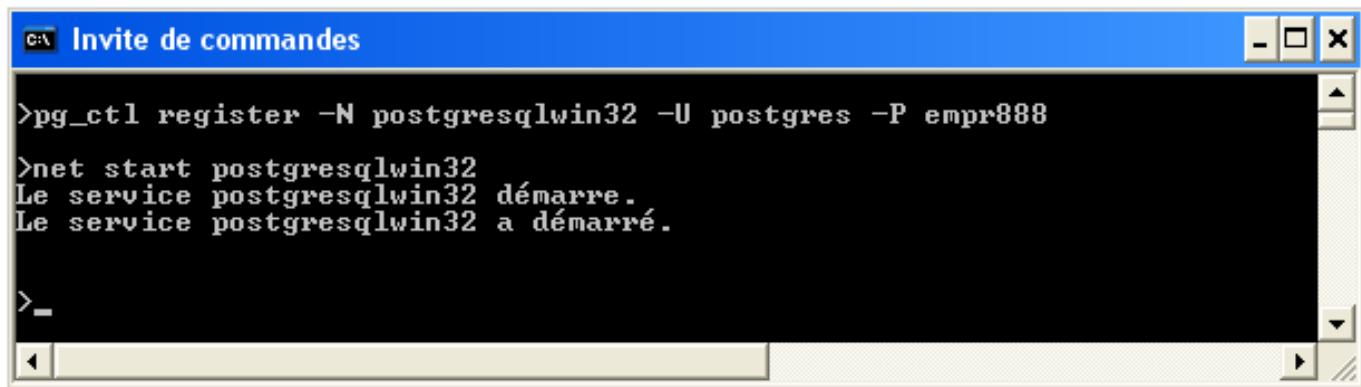


FIG. 4.7 – Installation et démarrage du service (postgresqlwin32)pour PostgreSQL

Rappel : ces deux états (installation et démarrage) du service ne sont à effectuer qu'une seule fois !

4.6.3.3 Désinstaller ou arrêter le service - Optionnel -

Dans les deux cas - au cas où un problème surviendrait ou pour une raison de confort (...) pour arrêter le service :

```
# sous david:  
net stop postgresqlwin32
```

pour désinstaller le service :

```
# sous david:  
pg_ctl unregister -N postgresqlwin32
```

4.6.4 Méthode 2 : Démarrage/Arrêt automatique pour GNU/Linux

Dans les sources de PostgreSQL, dans le sous-répertoire contrib/start-scripts/, vous trouverez un script nommé linux à partir duquel nous pouvons nous baser pour créer un script qui permettra l'instalalition du service

4.6.4.1 Modification du script source

Je vous donne ici le conetnu de mon script pour ma propre utilisation.

Le script nommé ici postgresql est à copier dans /etc/init.d/

Exemple 4.7 Script d'installation en service pour GNU/Linux

```
#! /bin/sh

# chkconfig: 2345 98 02
# description: PostgreSQL RDBMS

# This is an example of a start/stop script for SysV-style init, such
# as is used on Linux systems. You should edit some of the variables
# and maybe the 'echo' commands.
#
# Place this file at /etc/init.d/postgresql (or
# /etc/rc.d/init.d/postgresql) and make symlinks to
#   /etc/rc.d/rc0.d/K02postgresql
#   /etc/rc.d/rc1.d/K02postgresql
#   /etc/rc.d/rc2.d/K02postgresql
#   /etc/rc.d/rc3.d/S98postgresql
#   /etc/rc.d/rc4.d/S98postgresql
#   /etc/rc.d/rc5.d/S98postgresql
# Or, if you have chkconfig, simply:
# chkconfig --add postgresql
#
# Proper init scripts on Linux systems normally require setting lock
# and pid files under /var/run as well as reacting to network
# settings, so you should treat this with care.

# Original author: Ryan Kirkpatrick <pgsql@rkirkpat.net>

# $PostgreSQL: pgsql/contrib/start-scripts/linux,v 1.7 2004/10/01 18:30:21 tgl Exp $

## EDIT FROM HERE

# Installation prefix
prefix=/usr/local/pgsql

# Data directory [ ! ! ! LIGNE A MODIFIER ! ! ! ]
PGDATA="/usr/local/pgsql/data"

# Who to run the postmaster as, usually "postgres". (NOT "root")
PGUSER=postgres

# Where to keep a log file [ ! ! ! LIGNE A MODIFIER - A COMMENTER ! ! ! ]
#PGLOG="$PGDATA/serverlog"

## STOP EDITING HERE

# Check for echo -n vs echo \c
if echo '\c' | grep -s c >/dev/null 2>&1 ; then
    ECHO_N="echo -n"
    ECHO_C=""
else
    ECHO_N="echo"
    ECHO_C='\c'
fi

# The path that is to be used for the script
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# What to use to start up the postmaster (we do NOT use pg_ctl for this,
# as it adds no value and can cause the postmaster to misrecognize a stale
# lock file)
```

Exemple 4.8 Script d'installation en service pour GNU/Linux (suite)

```
DAEMON="$prefix/bin/postmaster"

# What to use to shut down the postmaster
PGCTL="$prefix/bin/pg_ctl"

set -e

# Only start if we can find the postmaster.
test -x $DAEMON || exit 0

# Parse command line parameters.
case $1 in
  start)
    $ECHO_N "Starting PostgreSQL: "$ECHO_C
    su - $PGUSER -c "$DAEMON -D '$PGDATA' &" 2>&1
    echo "ok"
    ;;
  stop)
    echo -n "Stopping PostgreSQL: "
    su - $PGUSER -c "$PGCTL stop -D '$PGDATA' -s -m fast"
    echo "ok"
    ;;
  restart)
    echo -n "Restarting PostgreSQL: "
    su - $PGUSER -c "$PGCTL stop -D '$PGDATA' -s -m fast -w"
    su - $PGUSER -c "$DAEMON -D '$PGDATA' &" 2>&1
    echo "ok"
    ;;
  reload)
    echo -n "Reload PostgreSQL: "
    su - $PGUSER -c "$PGCTL reload -D '$PGDATA' -s"
    echo "ok"
    ;;
  status)
    su - $PGUSER -c "$PGCTL status -D '$PGDATA' "
    ;;
  *)
    # Print help
    echo "Usage: $0 {start|stop|restart|reload|status}" 1>&2
    exit 1
    ;;
esac

exit 0
```

4.6.4.2 Installation

Pour l'installer, il faut pouvoir créer les liens symboliques pour que le service soit démarré aux bons niveaux d'exécution. Il faut commencer par restreindre les droits sur ce fichier ; En tant que root, tapez donc

```
chmod 744 /etc/init.d/postgresql
```

upda-rc.d va s'occuper de créer les divers liens symboliques attendus (toujours en tant que root)

```
update-rc.d postgresql defaults
```

Pour démarrer/arrêter le serveur, il suffira donc de saisir (toujours en tant que root)

```
/etc/init.d/postgresql {start|stop}
```

4.7 Devenir soi-même super-utilisateur de PostgreSQL

Le fait d'utiliser PostgreSQL en ayant besoin du super-utilisateur peut devenir à la longue un handicap. En effet, chaque fois que sous votre session habituelle, vous aurez envie d'utiliser les utilitaires de PostgreSQL, vous serez obligé de saisir l'option -U postgres. Le plus simple est de vous rendre super-utilisateur de PostgreSQL.

Il est vrai que je suis en contradiction avec ce que j'ai mentionné dans la section du super-utilisateur (voir seconde propriété) et que normalement vous devriez ne pas pouvoir le faire. Mais je mentionnerais ici la commande qui le permet de manière à vous simplifier la vie et pour vous éviter de trimballer le -U postgres en question à chaque occasion rencontrée. Que Saint-PostgreSQL me pardonne !!! Le but avoué ici est d'autoriser celui qui a utilisé MinGW - qui a tout installé - de devenir super-utilisateur de PostgreSQL.

Il vous suffira de saisir

```
# sous postgres:  
createuser -s david
```

L'option -s permet à david non seulement de créer à son tour de nouveaux utilisateurs mais aussi de créer de nouvelles bases de données.

Voilà, vous êtes maître absolu de votre PostgreSQL. Vous ne serez plus obligé de saisir la commande avec runas depuis votre session habituelle !

4.8 Autoriser les machines du réseau intranet/extranet à se connecter au serveur

Nous envisageons ici le cas où votre machine est le serveur PostgreSQL principal du réseau intranet. Si quelqu'un du réseau souhaite se connecter au serveur depuis un poste client, le meilleur outil que je puisse conseiller pour pouvoir administrer les bases de manière distante est d'installer sur le poste client l'outil PgAdmin qui tourne magnifiquement bien sous Windows (mais aussi linux).

Commencez par arrêter le serveur selon le mode de démarrage que vous avez choisi précédemment :

1. MODE MANUEL :

```
# sous postgres:  
pg_ctl stop
```

2. MODE AUTOMATIQUE :

```
# sous david:  
net stop postgresqlwin32
```

4.8.1 Intranet : Authentification par méthode de confiance (trusting) et connexion à toutes les bases

Déterminer le jeu de chiffrement IP de votre réseau intranet. Nous allons supposer ici que ce sera 192.168.134.X où X est un numéro quelconque du réseau. Nous allons autoriser toutes les machines du réseau à se connecter au serveur PostgreSQL.

Éditez le fichier **c :\PostgreSQL\8.2.1\data\pg_hba.conf** et ajoutez la ligne suivante dans ce fichier :

```
host      all          all      192.168.134.0      255.255.255.0      trust
```

Le zéro dans la chaîne 192.168.134.0 est là pour dire que toute la plage du réseau disponible est autorisée.

NOTE

Restreindre la plage de connexions des IP au serveur :

Si on souhaite restreindre la plage des connexions clientes - par exemple pour les IP allant de 192.168.134.1 à 192.168.134.12, on peut aussi essayer

```
host      all      all 192.168.134.1/12 trust
```

Il suffit ensuite de redémarrer le serveur en remplaçant le mot-clé 'stop' par 'start'. Autre astuce, au lieu de démarrer le serveur, il suffit normalement de saisir **pg_ctl reload** pour prendre en compte les modifications apportées à ce fichier.

Du côté client, pour pouvoir se connecter, l'utilisateur pourra par exemple se connecter en précisant juste un nom de super-utilisateur de PostgreSQL du serveur ainsi que son IP :

```
psql -h SERVEUR_IP -U SERVEUR_USER -d BASE_DE_DONNEES ...
```

Si par exemple, l'hôte serveur a pour IP 192.168.134.7 et que le nom du super-utilisateur est postgres et qu'on veuille se connecter à la base mabasededonnees

```
psql -h 192.168.134.7 -U postgres -d mabasededonnees ...
```

4.8.2 Extranet : Authentification par MD5, connexion à une seule et unique base de données pour un utilisateur unique de PostgreSQL - Optionnel -

Le but ici est de créer un utilisateur normal - qui n'est pas super-utilisateur - de PostgreSQL. Celui doit pouvoir se connecter à une seule base quelque soit son IP (interne ou externe au réseau) ;
se connecter à une seule et unique base de données ;

la base de données est une base de données déjà créée par un des super-utilisateurs. Il aura le droit de modifier/accéder librement aux données déjà existantes et d'en rajouter de nouvelles. Il devient donc propriétaire de la base ;
sur cette base, il ne pourra pas ajouter d'utilisateur supplémentaires, il lui est aussi impossible de créer des bases supplémentaires.

Dans l'attente des objectifs ci-dessus, nous savons que postgres et david - voir les sections précédentes - sont super-utilisateurs. Nous appellerons l'utilisateur en question **keizer** - on reconnaît là les fans de Usual Suspects ! LOL - et la base à laquelle il se connectera nous l'appellerons **testgis**. Nous supposerons donc que david a créé cette base, puisque cette base doit être déjà existante.

4.8.2.1 Création de l'utilisateur (createuser)

Pour la création de cet utilisateur, nous aurons recours à l'outil en ligne de commande **createuser** dont la documentation succincte est obtenue en saisissant

```
createuser --help
```

Pour les objectifs que nous visons, nous noterons les options suivantes qui nous seront utiles

- e : restitue à l'écran les instructions envoyées à PostgreSQL pour savoir ce que fait exactement le serveur ;
- D : comportement par défaut de createuser pour spécifier que l'utilisateur n'est pas autorisé à créer des bases de données ;
- R : comportement par défaut de createuser pour spécifier que l'utilisateur n'est pas autorisé à créer/ajouter des rôles supplémentaires, droit qui pourrait pû l'éléver au rang de super-utilisateur ;
- P : appel à une invite pour pouvoir saisir le mot de passe du nouvel utilisateur ;
- E : demande que le mot de passe soit crypté.
- S : le rôle ne peut pas être un super-utilisateur

Créons donc notre nouvel utilisateur avec comme mot de passe soze. Comme souvent dans le monde de GNU/Linux, au lieu de saisir la ligne suivante quand les options ne contiennent que "-unelettre"

```
createuser -D -e -P -E -R -S keizer
```

Nous saisirons plutôt

```
createuser -DePERS keizer
```

qui nous renverra en sortie sur l'écran par exemple depuis MinGW :¹

¹On peut aussi effectuer cela depuis une fenêtre DOS

```
david@OLIVIA ~
$createuser -DePERS keizer
Entrez le mot de passe pour le nouvel utilisateur : soze
Entrez-le de nouveau : soze
CREATE ROLE keizer ENCRYPTED PASSWORD 'azer' NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT ←
    LOGIN;
CREATE ROLE
```

L'avant-dernière ligne ci-dessus est la requête SQL envoyé au serveur et affiché par createuser. Elle confirme les droits attendus pour l'utilisateur créé (NOCREATEDB, NOCREATEUSER). Il nous faut maintenant configurer PostgreSQL pour que keizer puisse accéder à la base de données. Le paramètre LOGIN est important dans la mesure où il commande que l'utilisateur keizer puisse se connecter à la base. En effet sans ce paramètre, keizer ne peut pas se connecter à la base.

NOTE

Modifier le mot de passe :

Si par la suite, on souhaite par exemple pouvoir modifier le mot de passe "soze" de keizer en "kobayashi", on pourra utiliser la requête suivante

```
ALTER USER keizer PASSWORD 'kobayashi';
```

4.8.2.2 Configuration de PostgreSQL

Le fichier pg_hba.conf est le fichier qui pilote PostgreSQL pour les connexions réseau. Éditez le fichier suivant

c:\PostgreSQL\8.2.1\data\pg_hba.conf

et ajoutez la ligne suivante dans le fichier à la fin :

```
host      testgis      keizer      0.0.0.0      0.0.0.0      md5
```

La méthode d'authentification choisie est md5 (cf. option -E de createuser). Les adresses et masques de sous-réseaux ici donnés en absolu ('0.0.0.0') autorisent à se connecter depuis l'extérieur du réseau interne, à condition que le port 5432 de PostgreSQL soit accessible depuis l'extérieur du réseau !.

Nous n'oublierons pas de faire

```
pg_ctl reload
```

pour que le serveur recharge à la volée la nouvelle configuration réseau ajoutée dans le fichier pg_hba.conf.

4.8.2.3 Accorder au nouvel utilisateur les droits d'accès aux données déjà existantes dans la base

Pour notre dernier objectif à atteindre, il nous faut rendre keizer propriétaire de la base déjà existante ainsi que des tables déjà existantes. En effet, nous rappelons à cette fin que ces dernières ont été créées par le super-utilisateur david. Ces droits sont accordés en ayant recours aux deux requêtes suivantes. david est donc propriétaire de la base et des tables déjà existantes :

pour la base testgis :

```
ALTER DATABASE testgis OWNER TO keizer
```

pour chaque table de la base :

```
ALTER TABLE table OWNER TO keizer
```

Pour connaître les nom des tables contenues dans la base tesgis, on peut utiliser la requête suivante :

```
SELECT tablename FROM pg_tables WHERE (tablename NOT LIKE 'pg_%')
AND (tablename NOT LIKE 'sql_%')
```

et tout en sachant que l'on peut exécuter une requête dynamiquement grâce au moniteur interactif psql (voir prochain chapitre) pour exécuter une requête :

```
psql -d BASE_DE_DONNEES -c "REQUETE"
```

Il suffit alors d'avoir recours au petit script shell suivant pour effectuer tout le travail demandé côté serveur :

```
# à copier dans un script par exemple altertable.sh
psql -d testgis -c "ALTER DATABASE testgis OWNER TO keizer"
psql -d testgis -c "psql -F \" \" -Atd testgis -c \"select 'ALTER TABLE '\
||tablename||' OWNER TO keizer;' from pg_tables where \
(tablename not like 'pg_%') and (tablename not like 'sql_%') order by tablename '\""
```

En effet, la commande

```
psql -F " " -Atd testgis -c "select 'ALTER TABLE '\
||tablename||' OWNER TO keizer;' from pg_tables where \
(tablename not like 'pg_%') and (tablename not like 'sql_%') order by tablename"
```

renvoie textuellement les requêtes suivantes

```
ALTER TABLE <nom_de_la_premiere_table> OWNER TO keizer;
ALTER TABLE <nom_de_la_seconde_table> OWNER TO keizer;
ALTER TABLE <nom_de_la_troisieme_table> OWNER TO keizer;
etc...
```

Copiez ce code dans un script nommé altertable.sh et, depuis MinGW, tapez simplement pour l'exécuter :

```
altertable.sh
```

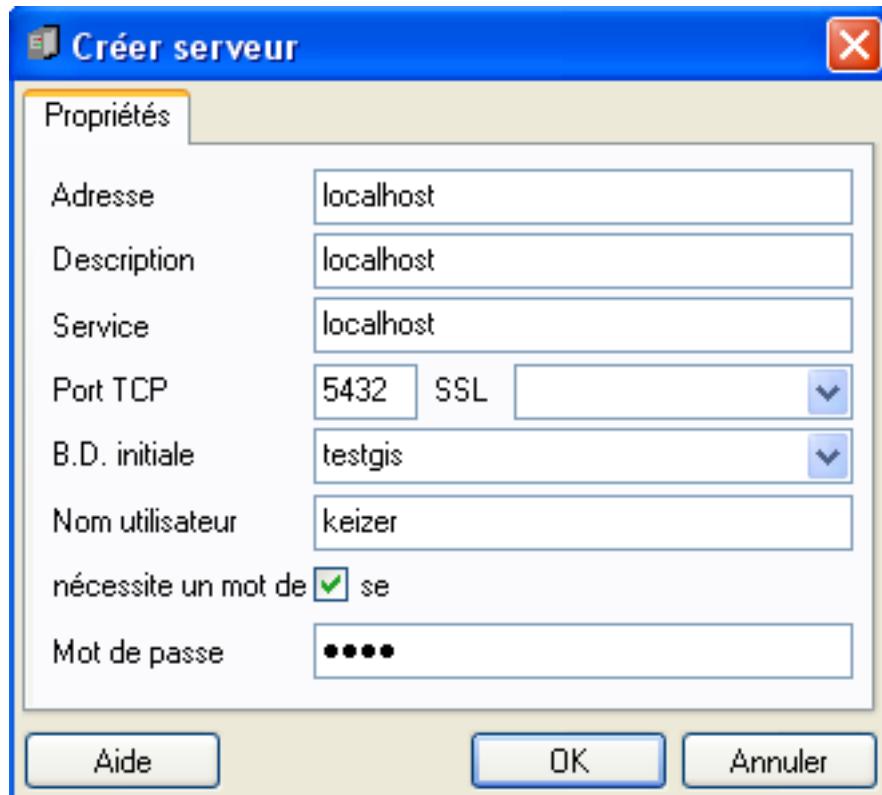


FIG. 4.8 – Exemple de connexion dans PgAdmin III

4.9 Gestion des fichiers de logs

Garder une trace des requêtes qui ont été effectuées, par qui et sur quelle base est une des possibilités qu'offre PostgreSQL. Celà permet aussi d'archiver les commandes qui ont été effectués sur le serveur. Tout celà se paramètre depuis le fichier de configuration à savoir C : \PostgreSQL\8.2.1\data\postgresql.conf.

4.9.1 Crédit du répertoire pour les logs

Dans le répertoire C : \PostgreSQL\8.2.1\data, nous allons commencer par définir un sous-répertoire pour y stocker nos fichiers de log. Appelons-le data. Pour le créer, utilisez la création de répertoire usuelle sous Windows ou depuis une fenêtre DOS tapez la commande

Exemple 4.9 Crédit du répertoire pour les logs sous WIndows

```
mkdir C:\PostgreSQL\8.2.1\data\pg_log
```

Exemple 4.10 Crédit du répertoire pour les logs sous GNU/Linux

```
mkdir /usr/local/pgsql/data/pg_log
```

4.9.2 Activation des paramètres

Ouvrez maintenant le fichier postgresql.conf et rendez-vous aux alentours de la ligne 214 et activez les paramètres redirect_stderr, log_directory et log_filename :

```
...
...
#-----
# ERROR REPORTING AND LOGGING
#-----

# - Where to Log -

log_destination = 'stderr'      # Valid values are combinations of
                                # stderr, syslog and eventlog,
                                # depending on platform.

# This is used when logging to stderr:
redirect_stderr = on            # Enable capturing of stderr into log
                                # files

# These are only used if redirect_stderr is on:
log_directory = 'pg_log'        # Directory where log files are written
                                # Can be absolute or relative to PGDATA
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log' # Log file name pattern.
                                # Can include strftime() escapes
log_truncate_on_rotation = off  # If on, any existing log file of the same
                                # name as the new log file will be
...
...
```

NOTE

Lisez donc les commentaires qui accompagnent les paramètres pour une meilleure compréhension. Lorqu'on installe PostgreSQL en prenant l'installeur fourni sur le site officiel, l'activation de ces paramètres est une option proposée par défaut.

Pour aller un peu plus loin, nous allons aussi activer des paramètres supplémentaires qui vont nous permettre de savoir qui a effectué une requête, depuis quelle IP, sur quelle base. Pour cela, rendez-vous de la ligne 298 à la ligne 316 pour activer les paramètres log_line_prefix, log_statement et log_hostname comme suit :

```
log_line_prefix = '[UTILISATEUR=%u HOTE=%h BASE=%d HEURE=%t COMMANDE=%i]'  
log_statement = 'all'  
log_hostname = on  
  
log_line_prefix = '[UTILISATEUR=%u HOTE=%h BASE=%d HEURE=%t COMMANDE=%i]'  
    # Special values:  
    #   %u = user name  
    #   %d = database name  
    #   %r = remote host and port  
    #   %h = remote host  
    #   %p = PID  
    #   %t = timestamp (no milliseconds)  
    #   %m = timestamp with milliseconds  
    #   %i = command tag  
    #   %c = session id  
    #   %l = session line number  
    #   %s = session start timestamp  
    #   %x = transaction id  
    #   %q = stop here in non-session  
    #       processes  
    #   %% = '%'  
    # e.g. '<%u%%%d> '  
log_statement = 'all'      # none, mod, ddl, all  
log_hostname = on
```

Si vous voulez voir l'intérêt des paramètres activés ici, rallumez donc votre serveur au moins deux fois - en prenant bien sûr qu'une personne ne soit connecté dessus ;-). Regardez le contenu du sous-répertoire data en question. Effectuez quelques requêtes sur le serveur et examinez le contenu du tout dernier fichier de log.



Avertissement

Faire le ménage de temps en temps

Plus le serveur sera actif et plus le sous-répertoire en question grossira. Pensez à faire le ménage de temps en temps pour ne pas trop remplir votre sous-répertoire en mettant en place par exemple une politique d'effacement des derniers fichiers de logs qui ne sont plus utiles par exemple.

Quatrième partie

PostGIS

Chapitre 5

Tutoriaux

Mon but ici n'est pas de m'étendre sur l'utilisation de PostGIS mais juste montrer comment créer une base avec PostGIS et un éventail de requêtes possibles avec PostGIS et un cas pratique d'utilisation avec MapServer. Nous commencerons donc par voir comment créer une base ayant les fonctionnalités de PostGIS.

Deux séries de requêtes seront ensuite abordées. La première consiste en un survol rapide des possibilités de PostGIS. Elle a plutôt un aspect ludique -LOL - La deuxième tente de voir plusieurs aspects possibles : importation de données, comportement de PostGIS, requêtes jugées pertinentes. Pour ces deux séries, les index spatiaux Gist ne seront pas abordés -car ici le nombre de données n'est pas assez volumineux-. Des clins d'oeil à MapServer sont fournis dans la deuxième série parfois. Ces derniers sont mis ici en illustration afin de montrer la requête adéquate pour faire la liaison entre PostGIS et MapServer.

L'utilisation avec MapServer - qui clôture ce chapitre - est un cas pratique où l'on doit dans un premier temps importer des données SIG réelles et extraire de ces données une nouvelle table qui puisse être exploitable par MapServer. Dans un second temps, il est demandé de créer de nouvelles données à partir des premières données, de les afficher ensuite dans MapServer. cela est fait dans l'optique de montrer le travail qui peut attendre un administrateur PostgreSQL/PostGIS.

5.1 Créeer une base avec PostGIS

La création d'une base - par exemple nommée madatabase - se fait en tapant depuis votre session habituelle sous DOS :

Exemple 5.1 Création d'une base PostGIS sous WIndows

```
createdb madatabase
createlang plpgsql madatabase
psql -d madatabase -f C:\PostgreSQL\8.2.1\share\contrib\lwpostgis.sql
psql -d madatabase -f C:\PostgreSQL\8.2.1\share\contrib\spatial_ref_sys.sql
```

et respectivement sous GNU/Linux nous ferons tout simplement

Exemple 5.2 Création d'une base PostGIS sous GNU/Linux

```
su postgres
createdb madatabase
createlang plpgsql madatabase
psql -d madatabase -f /usr/local/pgsql/share/contrib/lwpostgis.sql
psql -d madatabase -f /usr/local/pgsql/share/contrib/spatial_ref_sys.sql
```

Examinons en détail les diverses commandes utilisées :

1. La première (**createdb** ...) est la commande utilisée pour créer une base madatabase. Par défaut, au vue des options choisies lors de l'initialisation, l'encodage de la base sera du SQL_ASCII.
2. PostGIS est écrit en C/C++ Les fonctions spatiales sont stockées dans une librairie dynamique (.dll) libwgeom.dll. PostgreSQL accède à ces fonctions à condition de lui spécifier le langage PL/PGSQL. La seconde commande permet donc de doter notre base de ce langage.
3. Les définitions des diverses fonctions spatiales sont stockées dans le fichier lwpostgis.sql que doit accepter notre base. On charge les fonctions spatiales de PostGIS grâce à la troisième commande. On assure donc un pont entre notre base de données et la librairie libwgeom.dll
4. Il est souvent utile de pouvoir passer d'un système de projection à un autre et même impératif de se doter des divers systèmes de projections connus (Lambert I Carto, Lambert II Etendu ...). La quatrième commande nous permet de se doter des divers systèmes. Ces derniers sont stockés dans une table par le biais du chargement du fichier spatial_ref_sys.sql - nom que portera la table des systèmes de projection -.

NOTE

Nous aurion pu aussi utiliser la variable d'environnement PGHOME - qui ici vaut pour rappel C :\PostgreSQL\8.2.1 (respectivement /usr/local/pgsql/-).

5.2 Effectuer des requêtes : le moniteur interactif psql de PostgreSQL

psql - contenu dans toute distibrution même celle que nous avons compilé et installé - est le moniteur interactif de PostgreSQL. Il permet en outre de faire des requêtes adresser directement à un serveur PostgreSQL. On se connecte à une base mabasededonnees en faisant

```
psql mabasededonnees
```

NOTE

Pour sortir du moniteur interactif, il suffit de saisir \q

5.3 Effectuer des requêtes : PgAdmin III

PgAdmin est de loin le meilleur outil lire qui soit pour effectuer ses requêtes sous PostgreSQL. Mais il peut faire bien mieux. Vous pouvez le télécharger directement à l'adresse suivante <http://www.pgadmin.org>

NOTE

Pour ma part c'est avec psql qu'aura lieu les requêtes de ce chapitre.

5.4 Exemples de requêtes spatiales I

5.4.1 Crédation de la base et d'une table

Ici nous allons commencer par créer une base de données testgis que nous allons ensuite peupler par quelques données basiques. Pour la création de la base, reportez-vous à la première section de ce chapitre, ce qui depuis une fenêtre DOS devrait nous donner comme commandes à exécuter

Exemple 5.3 Création de la base test avec PostGIS sous Windows

```
createdb testgis
createlang plpgsql testgis
psql -d testgis -f C:\PostgreSQL\8.2.1\share\contrib\lwpostgis.sql
psql -d testgis -f C:\PostgreSQL\8.2.1\share\contrib\spatial_ref_sys.sql
```

Connectons-nous maintenant à notre base testgis grâce à la commande suivante :

```
psql testgis
```

Peuplons maintenant notre base par quelques données génériques. Pour cela, créons une petite table test en saisissant

```
CREATE TABLE test (id serial PRIMARY KEY, genre text);
```

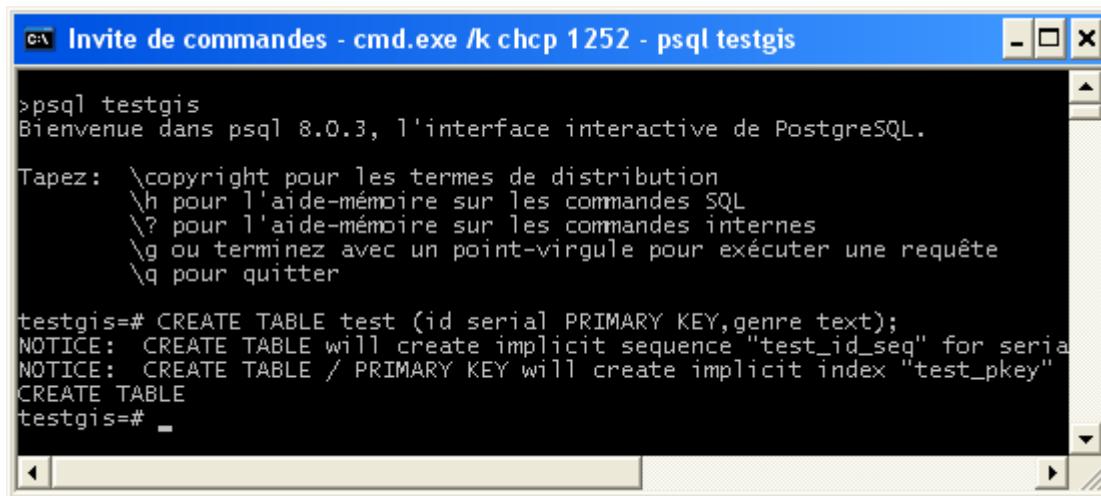


FIG. 5.1 – Le moniteur psql - Connexion à la base testgis.

5.4.2 Ajout de la colonne géométrique à la table - AddGeometryColumn()

Pour l'instant notre table est vide, nous allons lui ajouter une troisième colonne où seront stockées nos données géométriques. Nous aurons recours pour cela à la fonction `AddGeometryColumn()` de PostGIS dont la synthaxe générale est

```
AddGeometryColumn( [Table],
                    [Colonne_Geometrique],
                    [SRID],
                    [Type_Geometrie],
                    [Dimension]);
```

où

[Table] est le nom de la table à laquelle doit être ajoutée la colonne géométrique ;

[Colonne_Geometrique] est le nom de la colonne géométrique ;

[Type_Geometrie] est le type de géométrie possible :

[SRID] est l'identifiant spatial de projection selon le système de projection choisi. A titre d'exemple pour le Lambert II Carto Etendu, srid=27852 ;

[Dimension] est la dimension des objets géométriques 2D ou 3D ou 4D ;

Dans notre cas, nous saisirons

```
SELECT AddGeometryColumn( 'test', 'geom', -1, 'GEOMETRY', 2 );
```

Nous avons avec cette requête ajouté la colonne géométrique geom ([Colone_Geometrique]) à la table test ([Table]). Comme nous utilisons ici aucun système de projection référencé, - nous restons donc dans le plan 2D orthonormal par défaut - nous précisons juste srid=-1([SRID]). Le type de données géométriques que nous souhaitons enregistrer peut-être de n'importe quel type. Nous préciserons donc par défaut GEOMETRY ([Type_Geometrie]). Nous sommes en 2D ([Dimension]).

NOTE

Il aurait été tout à fait possible de créer la table test en faisant directement

```
CREATE TABLE test (id serial PRIMARY KEY,genre TEXT,geom GEOMETRY);
```

Mais il y aurait alors eu un perte concernant les métadonnées sur les objets géométriques concernant le type, le srid, la dimension. Il est donc préférable d'avoir recours à la fonction AddGeometryColumn() pour pouvoir tirer profit de ces métadonnées.

```
c:\ Invité de commandes - cmd.exe /k chcp 1252 - psql testgis
>psql testgis
Bienvenue dans psql 8.0.3, l'interface interactive de PostgreSQL.

Tapez: \copyright pour les termes de distribution
      \h pour l'aide-mémoire sur les commandes SQL
      \? pour l'aide-mémoire sur les commandes internes
      \g ou terminez avec un point-virgule pour exécuter une requête
      \q pour quitter

testgis=# CREATE TABLE test (id serial PRIMARY KEY,genre text);
NOTICE: CREATE TABLE will create implicit sequence "test_id_seq" for serial
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "test_pkey"
CREATE TABLE
testgis=# SELECT AddGeometryColumn( 'test', 'geom', -1, 'GEOMETRY', 2 );
           addgeometrycolumn
-----
 public.test.geom SRID:-1 TYPE:GEOMETRY DIMS:2
 geometry_column fixed:0
(1 ligne)

testgis=#

```

FIG. 5.2 – psql : Fonction AddGeometryColumn() de PostGIS

5.4.3 Objets géométriques spécifiés par l'O.G.C dans PostGIS

PostGIS supporte amplement les objets géométriques définis par l'O.G.C (Open GIS COnsortium). A titre d'exemple, nous citerons donc :

POINT;
LINESTRING;
POLYGON;

MULTIPOINT;
MULTILINESTRING;
MULTIPOLYGON;
GEOMETRYCOLLECTION

Insérons maintenant quelques objets dans notre table

5.4.4 Exemples d'objets géométriques

Dans le tableau suivant, les objets sont ici présentés sous le format WKT

5.4.5 Insertion d'objets géométriques - GeometryFromText()

L'insertion d'objets géométriques peut par exemple avoir lieu en ayant recours à la fonction `GeometryFromText()` dont la synthaxe générale est

```
GeometryFromText( [Objet_Geometrique] ,  
                  [SRID] )
```

où

[Objet_Geometrique] est l'objet à insérer est qui est "humainement lisible", j'entends par humainement lisible au sens qu'il est au format WKT (Well-Known Text) ;

[SRID] l'identifiant de projection qui doit être conforme à celui choisi la colonne géométrique (ici colonne geom avec srid=-1).

NOTE

Il existe aussi une autre définition de la fonction `GeometryFromText()`. Il est aussi possible d'utiliser les fonctions `GeomFromText()` ou `GeomFromEWKT()` etc... Par exemple pour insérer l'objet 'POINT(10 70)' ayant comme identifiant de projection SRID=-1, les 4 fonctions suivantes sont équivalentes

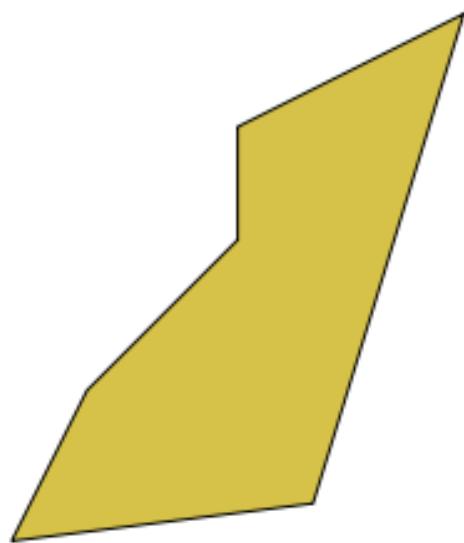
```
SELECT GeometryFromText( 'POINT(10 70)', -1 );  
SELECT GeometryFromText('SRID=-1;POINT(10 70)');  
SELECT GeomFromText('SRID=-1;POINT(10 70)');  
SELECT GeomFromEWKT('SRID=-1;POINT(10 70)');
```

5.4.6 Insertion des données dans la table

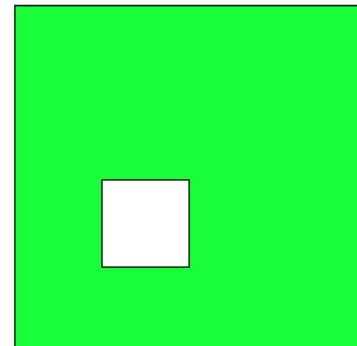
Insérons maintenant quelques objets géométriques. Ici les commandes sont à saisir au fur et à mesure pour chaque commande de type `INSERT INTO...`;

```
INSERT INTO test VALUES ( 1,  
'pieton 1', GeometryFromText( 'POINT(10 70)', -1 ) );  
  
INSERT INTO test VALUES ( 2,  
'pieton 2', GeometryFromText( 'POINT(30 30)', -1 ) );  
  
INSERT INTO test VALUES ( 3,  
'batiment 1', GeometryFromText( 'POLYGON((10 10,40 20,35 8,12 4,10 10))', -1 ) );  
  
INSERT INTO test VALUES ( 4,  
'batiment 2',  
GeometryFromText( 'POLYGON((10 40,20 30,30 40,40 35,50 60,35 80,20 60,10 40))', -1 ) );  
  
INSERT INTO test VALUES ( 5,  
'batiment 3', GeometryFromText( 'POLYGON((10 95,20 95,20 135,10 135,10 95))', -1 ) );
```

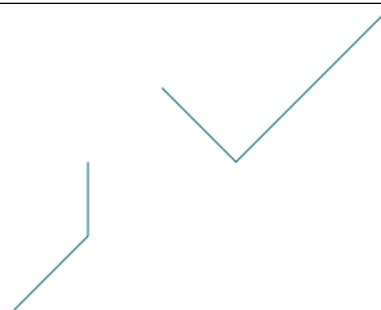
POLYGON((1 1,5 1.5,7 8,4 6.5,4 5,2 3,1 1))



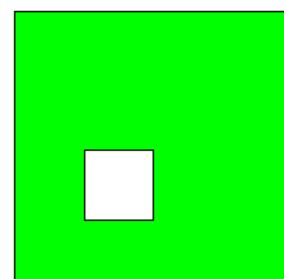
POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)) avec
un trou



MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))



MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)),((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))



```
INSERT INTO test VALUES ( 6,
'pieton 3', GeometryFromText( 'POINT(35 70)', -1 ) );

INSERT INTO test VALUES ( 7,
'pieton 4', GeometryFromText( 'POINT(35 60)', -1 ) );

INSERT INTO test VALUES ( 8,
'bordure 1 route', GeometryFromText( 'LINESTRING(1 85,50 85)', -1 ) );

INSERT INTO test VALUES ( 9, 'bordure 2 route', GeometryFromText( 'LINESTRING(1 92,50 92)', -1 ) );
```

```
testgis=# INSERT INTO test VALUES ( 1,
testgis(# 'pieton 1', GeometryFromText( 'POINT(10 70)', -1 ) );
INSERT 23470 1
testgis=#
testgis=# INSERT INTO test VALUES ( 2,
testgis(# 'pieton 2', GeometryFromText( 'POINT(30 30)', -1 ) );
INSERT 23471 1
testgis=#
testgis=# INSERT INTO test VALUES ( 3,
testgis(# 'batiment 1', GeometryFromText( 'POLYGON((10 10,40 20,35 8,12 4,10 10))' ), -1 );
INSERT 23472 1
testgis=#
testgis=# INSERT INTO test VALUES ( 4,
testgis(# 'batiment 2',
testgis(# GeometryFromText( 'POLYGON((10 40,20 30,30 40,40 35,50 60,35 80,20 40,10 40))' ), -1 );
INSERT 23473 1
testgis=#
testgis=# INSERT INTO test VALUES ( 5,
testgis(# 'batiment 3', GeometryFromText( 'POLYGON((10 95,20 95,20 135,10 135,10 95))' ), -1 );
INSERT 23474 1
testgis=#
testgis=# INSERT INTO test VALUES ( 6,
testgis(# 'pieton 3', GeometryFromText( 'POINT(35 70)', -1 ) );
INSERT 23475 1
testgis=#
testgis=# INSERT INTO test VALUES ( 7,
testgis(# 'pieton 4', GeometryFromText( 'POINT(35 60)', -1 ) );
INSERT 23476 1
testgis=#
testgis=# INSERT INTO test VALUES ( 8,
testgis(# 'bordure 1 route', GeometryFromText( 'LINESTRING(1 85,50 85)', -1 ) );
INSERT 23477 1
testgis=#
testgis=# INSERT INTO test VALUES ( 9, 'bordure 2 route', GeometryFromText( 'LINESTRING(1 92,50 92)', -1 ) );
testgis=#
```

FIG. 5.3 – psql : Insertion des données dans la table test

Le visuel de la table test est le suivant

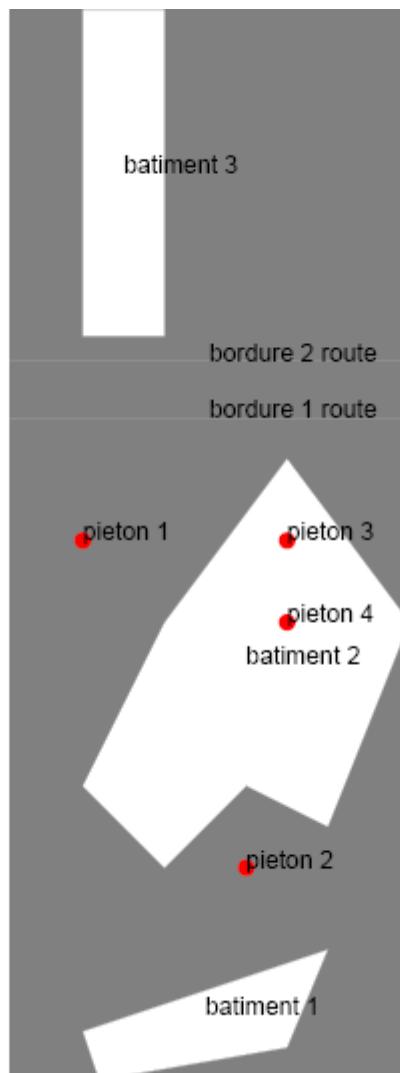


FIG. 5.4 – Visualisation de la table test dans la base testgis

Les instructions/requêtes suivantes doivent être saisie dans le terminal de psql.

NOTE

Il est tout à fait possible de saisir les requêtes sans entrer dans le terminal de psql. Pour cela, il suffit de saisir

```
psql -d testgis -c "VOTRE_REQUETE"
```

5.4.7 Question : Quelles sont les aires des objets ? - Area2d() -

La fonction Area2d() renvoie les aires des objets et la

```
testgis=# SELECT id, genre, Area2d(geom) FROM test;
 id |      genre      | area2d
----+-----+-----
 1 | pieton 1       |      0
 2 | pieton 2       |      0
 3 | batiment 1     |    228
```

```
4 | batiment 2      |    1050
5 | batiment 3      |     400
6 | pieton 3        |      0
7 | pieton 4        |      0
8 | bordure 1 route |      0
9 | bordure 2 route |      0
(9 rows)
```

5.4.8 Question : Quel sont les types géométriques des objets ? - GeometryType() -

Il faut pour cela utiliser la fonction GeometryType()

```
testgis=# SELECT id,genre,geometrytype(geom) FROM test;
 id |      genre      | geometrytype
----+-----+-----
 1 | pieton 1       | POINT
 2 | pieton 2       | POINT
 3 | batiment 1     | POLYGON
 4 | batiment 2     | POLYGON
 5 | batiment 3     | POLYGON
 6 | pieton 3       | POINT
 7 | pieton 4       | POINT
 8 | bordure 1 route | LINESTRING
 9 | bordure 2 route | LINESTRING
(9 rows)
```

5.4.9 Question : Qui est dans le bâtiment 2 ? - Distance() -

Ici, il suffira par exemple de trouver les objets dont le champs genre commence par pieton... et de déterminer les objets dont la distance à ce bâtiment est nulle :

```
testgis=# SELECT genre AS pietons_dans_batiment_2 FROM test
WHERE
          Distance((SELECT geom FROM test WHERE genre LIKE 'batiment 2'),test.geom)=0
AND
          genre LIKE 'pieton%';
pietons_dans_batiment_2
-----
pieton 3
pieton 4
(2 rows)
```

5.4.10 Question : Qui est dans le bâtiment 2 ? - Within() -

Ici, je propose par rapport à la première façon de faire d'utiliser la fonction Within(A,B) qui permet de savoir si A est contenu dans B

```
testgis=# SELECT genre AS pietons_dans_batiment_2 FROM test
WHERE
          Within(test.geom,(SELECT geom FROM test WHERE genre LIKE 'batiment 2'))
AND
          genre like 'pieton%';
pietons_dans_batiment_2
-----
pieton 3
pieton 4
(2 rows)
```

5.4.11 Question : Quel est l'objet géométrique le plus proche du piéton 2 ? - Min(), Distance() -

Pour cette question, nous aurons recours à la fonction `Min()` et à la fonction `Distance()`

```
testgis#SELECT q.genre FROM test z,test q WHERE z.genre LIKE 'pieton 2' AND
Distance(z.geom,q.geom)=(SELECT min(foo.valeur) FROM (SELECT h.genre AS nom,distance(t.geom ←
,h.geom)
AS valeur FROM test t, test h WHERE t.genre LIKE 'pieton 2' AND h.genre<>'pieton 2') foo);
genre
-----
batiment 2
(1 row)
```

Plusieurs formulations sont possibles pour les requêtes, on aurait ici pu aussi proposer la requête suivante

```
SELECT b.genre,Distance(a.geom,b.geom) FROM test a,test b
WHERE a.genre='pieton 2'
AND a.genre!=b.genre
ORDER BY distance ASC limit 1;
```

qui donnera le résultat suivant attendu

genre	distance
batiment 2	7.07106781186548

(1 ligne)

5.5 Exemples de requêtes spatiales II

Je ne propose ici que des exemples de bases de requêtes spatiales. Pour tirer un meilleur profit possible des fonctions spatiales, je ne serais vous conseiller la documentation sur le site de PostGIS réalisé par Paul RAMSEY. Pour de plus amples informations sur PostgreSQL, n'hésitez pas à regarder dans le répertoire **c :\PostgreSQL\8.2.1\doc\postgresql\html**.

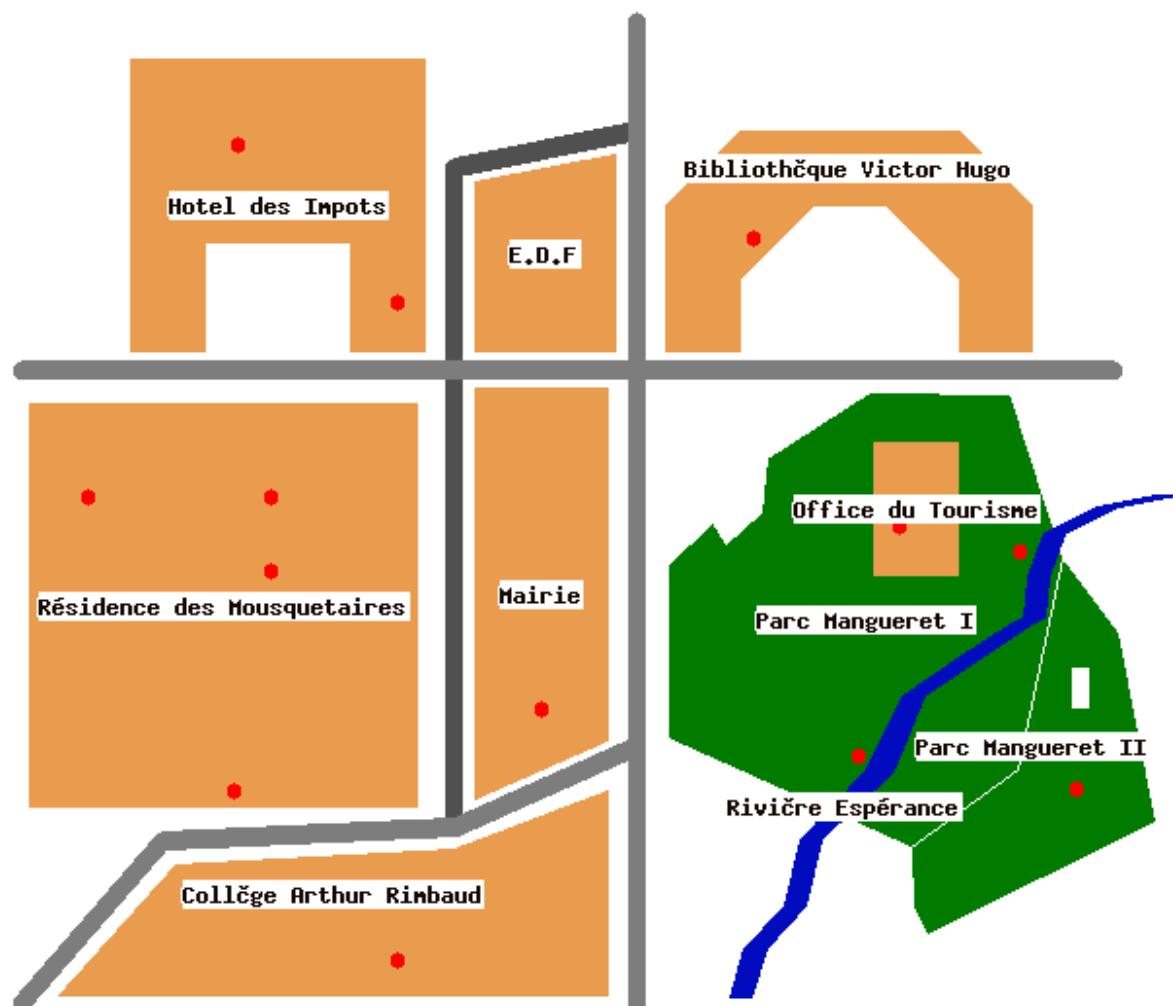


FIG. 5.5 – Visualisation des bâtiments (tables buildings et parcs et rivers).

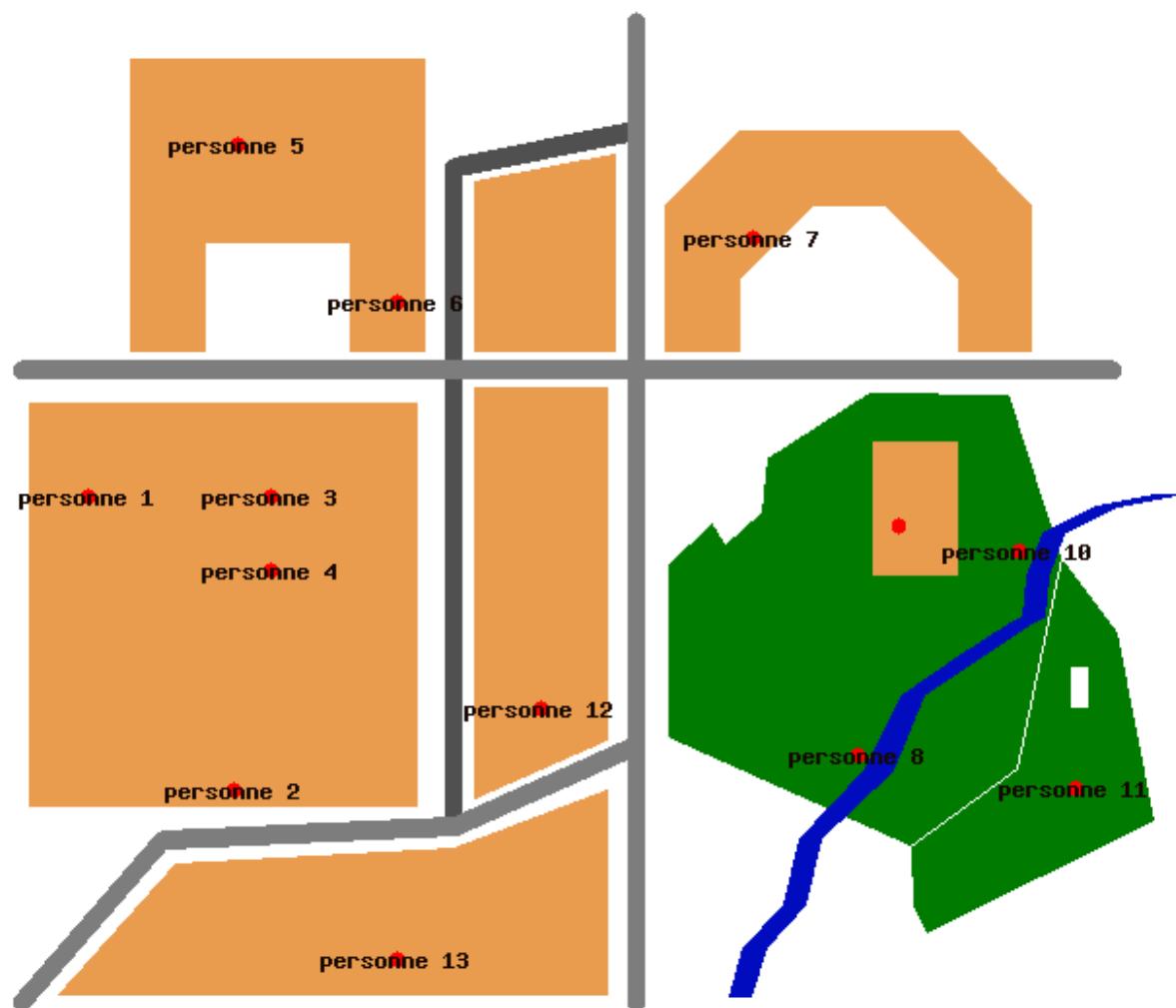


FIG. 5.6 – Visualisation des personnes (table personnes).

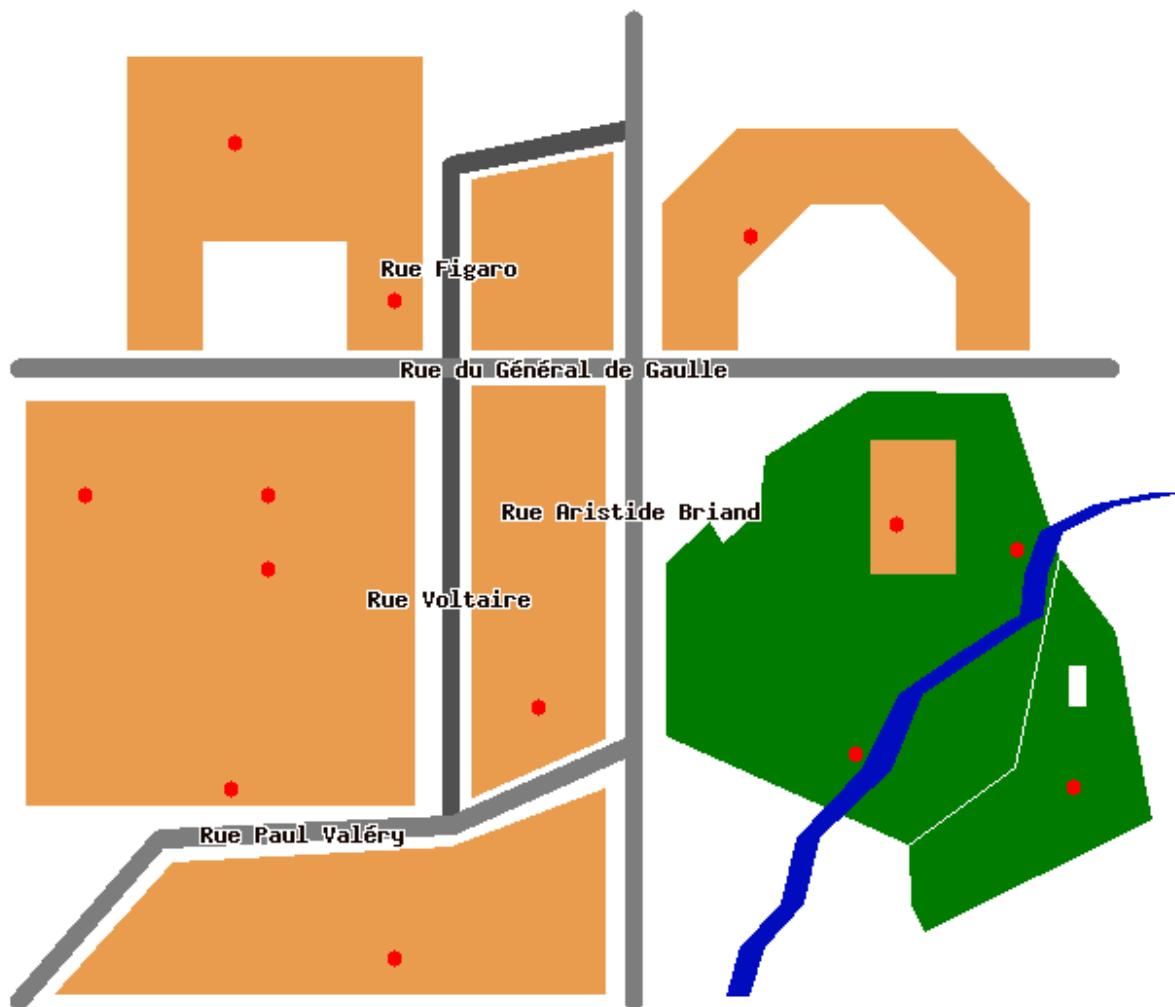


FIG. 5.7 – Visualisation des petites et grandes routes (tables small_roads et great_roads).

5.5.1 Démo de quelques requêtes en ligne avec SVG

J'ai mis certaines des requêtes de cette section en ligne selon le format de sortie possible de :

1. **Rendu en SVG** : vous pouvez essayer l'URL suivante <http://www.davidgis.fr/svg/>

5.5.2 Chargement des données par SQL

Pour charger le contenu du fichier sql suivant, copiez-le dans un fichier madatabase.sql et depuis MinGW ou DOS, tapez

```
psql -d madatabase -f madatabase.sql
```

Le contenu du script madatabase.sql¹ est

¹Ce script est aussi disponible à l'adresse <http://www.postgis.fr/download/win32/madatabase.zip>

```
/*
=====
TECHER Jean David

Script SQL 'madatabase.sql' à utiliser avec la base madatabase

Toutes les tables ont la même struture à savoir les champs suivants

- id : identifiant pour l'objet

- data: qui est la donnée attributaire de l'objet

- the_geom: qui est l'objet spatial

Il y a en tout 6 tables (voir commentaires pour chaque table).
=====*/
/*
=====
- commentaires:..... .... -
=====*/
/*
=====
Cette fonction écrite en SQL permet de tester si une table existe et de
l'effacer le cas échéant.

=====
CREATE OR REPLACE FUNCTION drop_table_if_exists(text, bool) RETURNS text AS '
DECLARE
opt text;
rec record;
BEGIN
IF $2 THEN
opt := '' CASCADE'';
ELSE
opt := '';
END IF;

IF NULLVALUE ($1) THEN
    RETURN ''ATTENTION: Table non trouvée'';
ELSE
    SELECT INTO rec tablename FROM pg_tables WHERE tablename like $1;

    IF FOUND THEN
        EXECUTE ''DROP TABLE '' || $1 || opt;
        RETURN ''Effacement de la table ''|| $1 || ''...OK'';
    END IF;

END IF;
    RETURN ''ATTENTION: Table ''|| $1 || '' non trouvée'';
END;
```

```
' LANGUAGE 'plpgsql';
/*
=====
On utilise la fonction pour effacer les tables éventuelles.
Fonction utile au cas où on devrait recharger ce fichier.
On évitera aussi d'effacer les tables geometry_columns et spatial_ref_sys
=====*/
SELECT tablename,drop_table_if_exists(tablename,false)
FROM pg_tables
WHERE (tablename NOT LIKE 'pg_%')
    AND (tablename NOT LIKE 'spatial_ref_sys' )
    AND (tablename NOT LIKE 'sql%' )
    AND (tablename NOT LIKE 'geom%' );
/*
=====

Table pietions

Certains des piétons sont dans certains des bâtiments.

*/
CREATE TABLE personnes (
    id serial PRIMARY KEY,
    data text);
SELECT AddGeometryColumn('','personnes','the_geom','-1','POINT',2);

INSERT INTO personnes VALUES ( 1, 'personne 1', GeometryFromText( 'POINT(10 70)', -1 ) );
INSERT INTO personnes VALUES ( 2, 'personne 2', GeometryFromText( 'POINT(30 30)', -1 ) );
INSERT INTO personnes VALUES ( 3, 'personne 3', GeometryFromText( 'POINT(35 70)', -1 ) );
INSERT INTO personnes VALUES ( 4, 'personne 4', GeometryFromText( 'POINT(35 60)', -1 ) );
INSERT INTO personnes VALUES ( 5, 'personne 5', GeometryFromText( 'POINT(30.54 118.28)', -1 ↔
) );
INSERT INTO personnes VALUES ( 6, 'personne 6', GeometryFromText( 'POINT(52.36 96.73)', -1 ↔
) );
INSERT INTO personnes VALUES ( 7, 'personne 7', GeometryFromText( 'POINT(100.94 105.44)', ↔
-1 ) );
INSERT INTO personnes VALUES ( 8, 'personne 8', GeometryFromText( 'POINT(115.16 34.81)', -1 ↔
) );
INSERT INTO personnes VALUES ( 9, 'personne 9', GeometryFromText( 'POINT(120.89 66.23)', -1 ↔
) );
INSERT INTO personnes VALUES ( 10, 'personne 10', GeometryFromText( 'POINT(137.40 62.56)', ↔
-1 ) );
INSERT INTO personnes VALUES ( 11, 'personne 11', GeometryFromText( 'POINT(144.97 30.23)', ↔
-1 ) );
INSERT INTO personnes VALUES ( 12, 'personne 12', GeometryFromText( 'POINT(72.04 41.23)', ↔
-1 ) );
INSERT INTO personnes VALUES ( 13, 'personne 13', GeometryFromText( 'POINT(52.32 6.84)', -1 ↔
) );
/*
=====

Table buildings:

J'ai opté pour des noms classiques de bâtiments administratifs
=====
*/
```

```
CREATE TABLE buildings (
    id serial PRIMARY KEY,
    data text);
SELECT AddGeometryColumn('','buildings','the_geom','-1','POLYGON',2);

INSERT INTO buildings VALUES ( 1, 'Collège Arthur Rimbaud',
GeometryFromText( 'POLYGON((6 2,81 2,81 30,60 22,22 20,6 2))', -1 ) );

INSERT INTO buildings VALUES ( 2, 'Résidence des Mousquetaires',
GeometryFromText( 'POLYGON((2 83,55 83,55 28,2 28,2 83))', -1 ) );

INSERT INTO buildings VALUES ( 3, 'Hôtel des Impots',
GeometryFromText( 'POLYGON((16 90,26 90,26 105,46 105,
46 90,56 90,56 130,16 130,16 90))', -1 ) );

INSERT INTO buildings VALUES ( 4,'E.D.F',
GeometryFromText( 'POLYGON((63 90,82 90,82 117,63 113.15,63 90))', -1 ) );

INSERT INTO buildings VALUES ( 5,'Bibliothèque Victor Hugo',
GeometryFromText( 'POLYGON((89 90,99 90,99 100,109 110,119 110,129 100,
129 90,139 90,139 110,129 120,99 120,89 110,89 90))', -1 ) );

INSERT INTO buildings VALUES ( 6, 'Mairie',
GeometryFromText( 'POLYGON((63 85,81 85,81 37,63 29,63 85))', -1 ) );

INSERT INTO buildings VALUES ( 7, 'Office du Tourisme',
GeometryFromText( 'POLYGON((117.36 77.6,128.71 77.60,128.71 59.49,
117.36 59.49,117.36 77.6))', -1 ) );
/*
=====
Tables small_roads:

Il n'y en a que deux.

=====
*/
CREATE TABLE small_roads (
    id serial PRIMARY KEY,
    data text);
SELECT AddGeometryColumn('','small_roads','the_geom','-1','LINESTRING',2);

INSERT INTO small_roads VALUES (1, 'Rue Figaro',
GeometryFromText( 'LINESTRING(60 87.5,60 115,85 120)', -1 ) );

INSERT INTO small_roads VALUES (2, 'Rue Voltaire',
GeometryFromText( 'LINESTRING(60 87.5,60 25)', -1 ) );
/*
=====
```

Tables great_roads

Il y en a 3. Les grandes routes gardent leur nom malgré un ou plusieurs croisement avec d'autres routes

```
/*
CREATE TABLE great_roads (
    id serial PRIMARY KEY,
    data text);
SELECT AddGeometryColumn('','great_roads','the_geom','-1','LINESTRING',2);
```

```
INSERT INTO great_roads VALUES (1, 'Rue Paul Valéry',
GeometryFromText( 'LINESTRING(1 1,20 23,60 25,85 36)', -1 ) );
INSERT INTO great_roads VALUES ( 2, 'Rue du Général de Gaulle',
GeometryFromText( 'LINESTRING(1 87.5,150 87.5)', -1 ) );
INSERT INTO great_roads VALUES ( 3, 'Rue Aristide Briand',
GeometryFromText( 'LINESTRING(85 1,85 135)', -1 ) );
/*
=====

```

Table parc

Il y en a deux portant le nom générique de Mangueret. Le premier parc est traversé par la rivière et conteint le bâtiment de l'Office du Tourisme. Le deuxième parc contient une aire administrative en réaménagement rachétée par la mairie. Le temps de la construction cette aire ne fait donc pas partie du parc. Soit un trou dans le polygone modélisant ce parc.

```
/*
CREATE TABLE parcs (
    id serial PRIMARY KEY,
    data text);
SELECT AddGeometryColumn('','parcs','the_geom','-1','POLYGON',2);

INSERT INTO parcs VALUES (1,'Parc Mangueret I',
GeometryFromText( 'POLYGON((89.19 37.11,89.19 60.73,
95.31 66.56,97.23 63.68,102.03 68,102.75 75.44,116.91 84.5,136 84.08,
143.08 62,140.92 50,137.08 32.95,122.43 22.39,89.19 37.11))',-1));
INSERT INTO parcs VALUES (2,'Parc Mangueret II',
difference(
GeometryFromText( 'POLYGON((143.08 62,150.94 51.41,155.96 25.66,
124.64 10.38,122.78 14.09,122.43 22.39,137.08 32.95,143.08 62))',-1),
GeometryFromText( 'POLYGON((144.55 46.64,146.68 46.64,146.68 41.46,
144.55 41.46,144.55 46.64))',-1)));
/*
=====
```

Table rivers

```
/*
CREATE TABLE rivers (
    id serial PRIMARY KEY,
    data text);
SELECT AddGeometryColumn('','rivers','the_geom','-1','POLYGON',2);

INSERT INTO rivers VALUES (1,'Rivière Espérance',
GeometryFromText( 'POLYGON((97.71 1.98,99.63 8.46,105.15 14.23,107.31 23.35,
115.95 32.71,121.23 43.03,129.15 48.31,
135.64 52.64,137.80 53.60,138.28 59.36,
140.44 65.12,147.88 68.72,155.80 70.40,158.80 70.40,
150.88 68.72,143.44 65.12,141.28 59.36,140.80 53.60,138.64 52.64,
132.15 48.31,124.23 43.03,119.95 32.71,110.31 23.35,108.15 14.23,
102.63 8.46,100.71 1.98,97.71 1.98))',-1));
/*
=====
```

- Fonction permettant de créer des index spatiaux -

On va commencer par vérifier quel est le nom de la colonne géométrique

dans la table demandée (par défaut = the_geom) ... Il faut bien sûr que ce nom soit préciser dans la table geometry_columns.

Cette fonction permet surtout d'alléger la commande de création d'indexation spatiale

```
=====
*/
CREATE OR REPLACE FUNCTION Create_Gis_Index(text) RETURNS text AS '
DECLARE
rec record;
BEGIN

SELECT INTO rec f_geometry_column FROM geometry_columns WHERE f_table_name like $1;

    IF FOUND THEN
EXECUTE ''CREATE INDEX '' || $1 || ''_index_spatial ON ''|| $1 ||
'' USING gist(''|| rec.f_geometry_column || '' gist_geometry_ops)'';
        RETURN ''Index Spatial sur '' || $1 || ''...OK'';
    END IF;

    RETURN ''Impossible de créer index spatial sur '' || $1 ;
END;
' LANGUAGE 'plpgsql';
/*
=====

    - Création des index spatiaux -

=====
*/
SELECT Create_Gis_Index(tablename)
FROM pg_tables
WHERE (tablename NOT LIKE 'pg_%')
    AND (tablename NOT LIKE 'spatial_ref_sys' )
    AND (tablename NOT LIKE 'sql%' )
    AND (tablename NOT LIKE 'geom%' )
    ORDER BY tablename;
/*
=====

    - Vérification de la création des index spatiaux -

=====
*/
\di *_index_spatial
/*
=====

    - Table de permettant de gérer MapServer -

=====
*/
CREATE TABLE mapserver_desc (
    ms_gid serial PRIMARY KEY,
    ms_table text,
    ms_libelle text,
    ms_name text,
    ms_labelitem text,
    ms_color text,
    ms_outlinecolor text,
```

```
        ms_symbol text
);

INSERT INTO mapserver_desc values (0,'small_roads', 'Petite routes',
'small_roads','data','80 80 80','80 80 80',NULL);
INSERT INTO mapserver_desc values (1,'great_roads','Grandes routes',
'great_roads','data','125 125 125','125 125 125',NULL);
INSERT INTO mapserver_desc values (2,'parcs','Parcs publiques',
'parcs','data','0 123 0','0 123 0',NULL);
INSERT INTO mapserver_desc values (3,'rivers','Rivière',
'rivers','data','0 12 189','0 12 189',NULL);
INSERT INTO mapserver_desc values (4,'buildings', 'Bâtiments',
'buildings','data','234 156 78','234 156 78',NULL);
INSERT INTO mapserver_desc values (5,'personnes', 'personnes',
'personnes','data',NULL,'0 0 255','circle');
```

5.5.3 Chargement de données par ESRI Shapefiles (shp2pgsql)

Nous allons supposer dans cette sous-sections que nous disposons des mêmes sources de données que celles ci-dessus. Ces dernières cette fois-ci au lieux d'être stockées au format sql sont ici stockées au format .shp. Chaque table est ainsi stockées dans un fichier .shp - avec les fichiers auxiliaires adéquates (.dbf, .shx....).

NOTE

Les fichiers en question sont disponibles à <http://www.postgis.fr/download/win32/shp.zip>. Décompressez ce fichier et double-cliquez sur le fichier batch import.bat pour les charger dans la base. N'hésitez pas à ouvrir pour l'étudier par rapport aux explications qui suivent sur shp2pgsql.exe. Si vous n'avez pas pour le moment importé le fichier madatabase.sql (cf. "Chargement en SQL") ouvrez alors ce fichier et remplacez l'option "-dD" en "-D".

PostGIS est livré avec un convertisseur **shp2pgsql.exe** qui permet d'importer directement - à la volée - les données fournies dans un fichier .shp. Pour de meilleures informations sur ce convertisseur, il suffit de saisir

```
shp2pgsql --help
```

qui vous renverra en sortie

```
shp2pgsql: illegal option -- -
RCSDID: $Id: shp2pgsql.c,v 1.104 2005/11/01 09:25:47 strk Exp $
USAGE: shp2pgsql [<options>] <shapefile> [<schema>.]<table>

OPTIONS:
  -s <srid>  Set the SRID field. If not specified it defaults to -1.

  (-d|a|c|p) These are mutually exclusive options:
    -d  Drops the table , then recreates it and populates
        it with current shape file data.
    -a  Appends shape file into current table, must be
        exactly the same table schema.
    -c  Creates a new table and populates it, this is the
        default if you do not specify any options.
    -p  Prepare mode, only creates the table

  -g <geometry_column> Specify the name of the geometry column
        (mostly useful in append mode).

  -D  Use postgresql dump format (defaults to sql insert
        statements.
```

```
-k Keep postgresql identifiers case.  
-i Use int4 type for all integer dbf fields.  
-I Create a GiST index on the geometry column.  
-w Use wkt format (for postgis-0.x support - drops M - drifts coordinates).  
-N <policy> Specify NULL geometries handling policy (insert,skip,abort)
```

Avec shp2pgsql, la ligne de commande la plus utilisée est

```
shp2pgsql -D -I shapefile nom_table | psql base_de_donnees
```

où l'option

-D permet d'importer les données sous forme de COPY au lieu de INSERT. Ce qui est beaucoup plus rapide au niveau de l'importation des données

-I permet de créer automatiquement un index spatial pour les données géométriques importées

shapefile est le chemin d'accès complet vers le shapefile

nom_table est le nom que vous souhaiteriez donner à votre table

Dans notre cas nous allons supposer que tous nos shapefiles sont dans le répertoire c :\Mes données SIG\shp et que les noms respectives des tables seront le même que celui des fichiers .shp. Le contenu en fichier .shp du répertoire en question est obtenu par la commande **ls** sous MinGW

```
david@OLIVIA ~  
$ cd /c/Mes\ donnees\ SIG/shp/  
david@OLIVIA /c/Mes donnees SIG/shp  
$ ls *.shp  
buildings.shp great_roads.shp parcs.shp personnes.shp rivers.shp small_roads.shp
```

Pour nos importations, il nous suffira de faire depuis MinGW ou DOS

pour MinGW :

```
for i in `ls *.shp`;\n> do \  
> table='basename $i | cut -d '.' -f 1';\n> shp2pgsql -dDI $i $table | psql madatabase;\n> done
```

ou en un peu plus subtil :

```
for i in $(find . | grep shp);do shp2pgsql -dDI $i $(basename $i .shp) | psql ←  
madatabase;done
```

pour DOS :

```
for /F "usebackq" %i in ('dir /B *.shp') do shp2pgsql -dDI %i %~ni | psql madatabase
```

qui nous renverra

```
Shapefile type: Polygon  
Postgis type: MULTIPOLYGON[2]  
dropgeometrycolumn  
-----  
public.buildings.the_geom effectively removed.  
(1 row)  
  
DROP TABLE  
BEGIN
```

```
NOTICE: CREATE TABLE will create implicit sequence "buildings_gid_seq" for serial column "←
       buildings.gid"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "buildings_pkey" for table "←
       buildings"
CREATE TABLE
              addgeometrycolumn
-----
public.buildings.the_geom SRID:-1 TYPE:MULTIPOLYGON DIMS:2
geometry_column fixed:0
(1 row)

CREATE INDEX
COMMIT
.
.
.

Shapefile type: Arc
Postgis type: MULTILINESTRING[2]
              dropgeometrycolumn
-----
public.small_roads.the_geom effectively removed.
(1 row)

DROP TABLE
BEGIN
NOTICE: CREATE TABLE will create implicit sequence "small_roads_gid_seq" for serial column ←
       "small_roads.gid"
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "small_roads_pkey" for table ←
       "small_roads"
CREATE TABLE
              addgeometrycolumn
-----
public.small_roads.the_geom SRID:-1 TYPE:MULTILINESTRING DIMS:2
geometry_column fixed:0
(1 row)

CREATE INDEX
COMMIT
```

Le message CREATE INDEX confirme la création des index spatiaux Gist, 'addgeometrycolumn' confirme aussi l'insertion des données spatiales.

Avant d'aborder les fonctions usuelles de PostGIS, commençons cette section par un rapide tutoriel sur des commandes de PostgreSQL.

5.5.4 Question-Pratique : Qu'elle est la version de PostgreSQL ?

Il suffit pour cela d'utiliser la fonction interne de PostgreSQL : `Version()`

```
select version()
```

qui nous renvoit

```
version
-----
PostgreSQL 8.2.1 on i686-pc-mingw32, compiled by GCC gcc.exe (GCC) 3.4.2 (mingw-special)
(1 ligne)
```

On peut aussi utiliser la requête suivante

```
SHOW server_version
```

qui nous renvoit

```
server_version
-----
8.2.1
(1 ligne)
```

NOTE

Le numéro de version peut-être aussi obtenu en tapant depuis DOS ou MinGW :

```
pg_config --version
```

5.5.5 Question-Pratique : Où se trouve notre répertoire de bases de données ? - PGDATA -

La requête suivante

```
SHOW data_directory
```

confirme ce que nous avons choisi comme valeur pour la variable d'environnement PGDATA²

```
data_directory
-----
C:/PostgreSQL/8.2.1/data
(1 ligne)
```

5.5.6 Question-Pratique : Qui sont les utilisateurs de PostgreSQL ?

Je ne fais pas ici de distinction entre les super-utilisateurs et les utilisateurs courants de PostgreSQL. En un mot, je ne m'occupe-rais pas ici des droits attribués (création de base, ajout d'utilisateur etc...). Pour simplifier, on dira que la requête est

```
select usename from pg_user order by usename
```

qui nous renvoit

```
usename
-----
david
keizer
postgres
(3 lignes)
```

On obtient de meilleures informations en faisant directement depuis MinGW ou DOS la commande suivante :

```
psql -d template1 -c "\du"
```

Liste des rôles					
Nom du rôle	Superutilisateur	Crée un rôle	Crée une base	Connexions	Membre de
david	oui	oui	oui	sans limite	
keizer	non	non	non	sans limite	
postgres	oui	oui	oui	sans limite	
(3 lignes)					

²voir pour cela le chapitre 'Paramétriser PostgreSQL'

5.5.7 Question-Pratique : Quelles sont les infos sur les outils compilés pour PostGIS ?

La requête suivante

```
select postgis_full_version();
```

nous renverra comme réponse

```
postgis_full_version
-----
POSTGIS="1.2.0" GEOS="2.2.3-CAPI-1.1.1" PROJ="Rel. 4.5.0, 22 Oct 2006" USE_STATS
(1 ligne)
```

Ce qui prouve bien que PostGIS 1.2.0, Geos 2.2.3 et Proj 4.5.0 ont bien compilés.

NOTE

Il existe aussi d'autres fonctions qui permettent d'obtenir des informations : `postgis_geos_version()`, `postgis_proj_version()`,... Vous obtiendrez d'autres fonctions en faisant directement depuis psql :

```
\df *postgis*
```

`\df` est le raccourci de psql pour obtenir le détail sur des fonctions

5.5.8 Question-Pratique : Quel est le listing des bases de PostgreSQL ?

cela se fait en faisant la requête suivante

```
SELECT pg_database.datname
      FROM pg_database, pg_user
     WHERE pg_database.datdba = pg_user.useryesid
       AND pg_database.datname not
        LIKE 'template%' order by datname
```

qui ici renvoie

```
datname
-----
madatabase
testgis
(2 lignes)
```

On obtient aussi les mêmes informations en faisant `psql -l` depuis MinGW ou DOS.

5.5.9 Question-Pratique : Quelles sont les tables contenues dans la base ?

La requête suivante

```
select tablename as nom_table from pg_tables where (tablename not like 'pg_%') and
(tablename not like 'spatial_ref_sys') and (tablename not like 'sql_%')
and (tablename not like 'geom%') order by tablename; ;
```

nous renverra comme réponse

```
nom_table
-----
buildings
great_roads
mapserver_desc
```

```
parcs
personnes
rivers
small_roads
(7 rows)
```

Le listing des tables peut facilement être obtenu en tapant simplement '\dt' dans psql le moniteur interactif de PostgreSQL.

NOTE

Je pars ici du principe que toutes les tables sont contenues dans un même schéma et soient contenus sur un même espace logique de PostgreSQL. Mais si vous êtes sûr que toutes vos tables sont dans le même schéma, vous pouvez par exemple exploiter la requête

```
select tablename as nom_table from pg_tables where schemaname='public'
and tablename not in ('spatial_ref_sys','geometry_columns') order by tablename;
```

5.5.10 Question-Pratique : Utiliser une vue pour simplifier la recherche du listing des tables.

PostgreSQL offre un concept fort appréciable concernant les SGDBR, celui des vues. Les vues permettent parfois de se simplifier la vie avec des requêtes parfois bien longue et que l'on ne pourrait pas toujours pouvoir se souvenir. Pour la requête concernant le listing des tables contenues dans une base de données - vu précédemment - on peut avoir recours aux vues en créant la la vue suivante

```
create view liste_table as select tablename as nom_table from pg_tables where
(tablename not like 'pg_%') and (tablename not like 'spatial_ref_sys' )
and (tablename not like 'sql_%' ) and (tablename not like 'geom%' ) order by tablename;
```

On aura alors le listing attendu en faisant directement

```
select * from liste_table
```

5.5.11 Question-Pratique : Avec psql, comment obtenir rapidement un bref rappel de la synthaxe des commandes SQL de PostgreSQL ?

Supposons que l'on ne se souvienne plus de la synthaxe pour la commande ALTER TABLE. Rien de plus simple avec psql ! Faites précéder votre commande de \h

```
testgis=\h ALTER TABLE
```

qui renverra

```
Commande :      ALTER TABLE
Description : modifier la définition d'une table
Syntaxe :
ALTER TABLE [ ONLY ] nom [ * ]
    action [, ... ]
ALTER TABLE [ ONLY ] nom [ * ]
    RENAME [ COLUMN ] colonne TO nouvelle_colonne
ALTER TABLE nom
    RENAME TO nouveau_nom
ALTER TABLE nom
    SET SCHEMA nouveau_schéma
```

où action fait partie de :

```
    ADD [ COLUMN ] colonne type [ contrainte_colonne [ ... ] ]
```

```
DROP [ COLUMN ] colonne [ RESTRICT | CASCADE ]
ALTER [ COLUMN ] colonne TYPE type [ USING expression ]
ALTER [ COLUMN ] colonne SET DEFAULT expression
ALTER [ COLUMN ] colonne DROP DEFAULT
ALTER [ COLUMN ] colonne { SET | DROP } NOT NULL
ALTER [ COLUMN ] colonne SET STATISTICS integer
ALTER [ COLUMN ] colonne SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
ADD contrainte_table
DROP CONSTRAINT nom_contrainte [ RESTRICT | CASCADE ]
DISABLE TRIGGER [ nom_déclencheur | ALL | USER ]
ENABLE TRIGGER [ nom_déclencheur | ALL | USER ]
CLUSTER ON nom_index
SET WITHOUT CLUSTER
SET WITHOUT OIDS
OWNER TO nouveauPropriétaire
SET TABLESPACE nouvelEspaceLogique
```

On peut aussi essayer \h ALTER

NOTE

Si on veut obtenir l'intégralité des diverses synthaxe possibles, depuis une fenêtre DOS, faites

```
psql -d template1 -c "\h" > memento.sql
```

commande qui redirigera la sortie vers le fichier memento.sql qu'il suffira ensuite d'ouvrir pour obtenir l'aide attendue.

Pour approfondir vos connaissances avec PostgreSQL, n'hésitez pas comme je l'ai déjà mentionné à consulter la documentation française

la documentation française

les sites français <http://www.postgresqlfr.org>, les ressources et tutoriaux comme <http://postgresql.developpez.com> etc...

la FAQ fournie en Annexe de ce document

etc..

5.5.12 Question : Où sont stockées les informations relatives aux données spatiales (métadonnées) des tables avec PostGIS ? - table geometry_columns -

Ces informations (srid, type, nom de la colonne géométrique) sont stockées dans la table geometrycolumns. La requête

```
select * from geometry_columns
```

nous renvoie les informations suivantes

f_table_catalog	f_table_schema	f_table_name	f_geometry_column	coord_dimension	
srid	type				
public	personnes	the_geom		2	↵
-1	POINT				
public	buildings	the_geom		2	↵
-1	POLYGON				
public	small_roads	the_geom		2	↵
-1	LINESTRING				
public	great_roads	the_geom		2	↵
-1	LINESTRING				
public	parcs	the_geom		2	↵
-1	POLYGON				
public	rivers	the_geom		2	↵
-1	POLYGON				

(6 rows)

Les informations utiles sont

- f_table_name** qui fournit le nom de la table ;
- f_geometry_column** qui donne le nom de la colonne géométrique ;
- srid** fournit le srid - identifiant de système de projection- ;
- type** qui fournit le type d'objet contenu dans la table.

**AVERTISSEMENT**

Certains logiciels SIG - au sens large du terme - comme MapServer, JUMP ou QGIS exploitent cette table pour effectuer leurs requêtes attendues ou lister l'ensemble des tables exploitables. On comprend ainsi mieux son importance. Cette table ainsi que certains champs de cette table sont aussi spécifiés par l'O.G.C (Open GIS Consortium).

5.5.13 Question-Pratique : Comment créer une fonction en PLP/PGSQL qui puisse faire le différentiel entre les tables référencées par geometry_columns et toutes les tables contenus dans le schéma public de la base (tables non géospatiales) ?

Comme il a été fait mention dans la sous-section précédente, la table `geometry_columns` contient toutes les métadonnées nécessaires à la gestion des tables géospatiales de notre base. Or il se peut qu'un jour nous ayons besoin d'une fonction qui puisse faire le différentiel entre les tables géospatiales de notre base avec les tables non géospatiales.

Nous allons commencer par créer un type comme celà est possible avec PostgreSQL

```
CREATE TYPE nom_de_table AS (tablename name);
```

Il est alors possible d'utiliser la fonction suivante

```
CREATE OR REPLACE FUNCTION diff_geometry_columns() RETURNS SETOF nom_de_table AS $$  
DECLARE  
j nom_de_table;  
BEGIN  
FOR j IN SELECT tablename AS nom_table FROM pg_tables  
WHERE tablename NOT IN  
(  
SELECT f_table_name FROM geometry_columns)  
UNION (SELECT 'spatial_ref_sys') UNION (select 'geometry_columns')  
)  
AND pg_tables.schemaname='public' ORDER BY 1  
LOOP  
-- cette syntaxe est nécessaire pour retourner plusieurs lignes  
RETURN NEXT j;  
END LOOP;  
END;  
$$ LANGUAGE plpgsql STABLE;
```

Ainsi l'appel de cette fonction par

```
select * from diff_geometry_columns();
```

j'obtiens

```
diff_geometry_columns  
-----  
mapserver_desc  
(1 ligne)
```

qui est bien la seule table non géospatiale - à part spatial_ref_sys - de ma base.

NOTE

Pour de plus amples informations sur PL/PGSQL, merci de consulter la documentation française de PostgreSQL à <http://traduc.postgresqlfr.org/pgsql-8.2.1-fr/plpgsql.html>

5.5.14 Question : Comment sont stockées les données géométriques avec PostGIS ?

Avant la version 1.0.0 de PostGIS, les données étaient stockées sous forme WKT (Well-Known Text), c'est-à-dire un format lisible par le commun des mortels. PostGIS prenait aussi en compte les formats géométriques (sémantique, grammaire, toléopologie...) des objets définis par l'OGC Open GIS Consortium. Pour des raisons d'économie sur le disque de dur et pour un accès plus rapide, depuis la version 1.0.0, les formats acceptés par PostGIS sont les formats EWKB /EWKT. Depuis cette version, les formats supportés par PostGIS forment une extension des définitions des formats définis par l'OGC qui ne s'intéressaient qu'aux objets définis en 2D. On peut donc avoir des données 3d. Ainsi EWKB/EWKT sont une extension du WKB/WKT. WKB/WKT sont donc acceptés par PostGIS en tant que EWKB/EWKT en dimension 2D.

NOTE

Pour de plus amples informations, veuillez consulter la documentation de PostGIS.

L'accès direct aux objets géométriques en EWKB ne permet pas de lire leur contenus des données pour le commun des mortels. Je rappelle au passage qu'un objet spatial dans une base de données est défini par son type, l'ensemble ou les sous-ensembles de points qui le composent ainsi qu'au système de projection auquel il est attaché, et à la dimension (2d,3d ou 4d) auquel il appartient. PostgreSQL de son côté selon les processus internes de fonctionnement et les format d'entrée des objets spatiaux en mode ascii ou binaire choisi - ascii : HEXEWKB/EWT et binaire : EWKB - stockera respectivement en ascii : HEXEWKB et pour le binaire en EWKB.

Pour pouvoir respectivement lire les données aux formats EWKT - format lisible -, il faut avoir recours aux fonctions AsText() ou AsEWKT(), et Srid() pour connaître leur identifiant de système de projection. Par exemple pour la table personnes dont la colonne géométrique est the_geom, affichons ces diverses informations grâce à la requête suivante

```
SELECT AsText(the_geom), Srid(the_geom), the_geom FROM personnes
```

qui nous renvoie

On voit bien ici le stockage réalisé en HEXEWKB des objets géométriques. En effet, si nous effectuons la requête suivante par exemple relatif au premier enregistrement pour le point POINT(35 70) ayant SRID=-1, on a comme attendu en utilisant la fonction GeomFromEWKT()

```
SELECT GeomFromEWKT('SRID=-1; POINT (35 70)');
```

ou la requête équivalente

```
SELECT 'SRID=-1; POINT(35 70)' ::geometry;
```

on obtient

```
geomfromewkt
```

PostgreSQL lit en entrée (mode ascii) du EWKT et restitue en stockage HEXEWKB (= EWKB sous forme hexadécimale)

Cet objet stocké en HEXEWKB a la forme suivante en EWKT, restitué grâce à la fonction AsEWKT() :

```
select AsEWKT('01010000000000000000008041400000000000805140');
```

asewkt

POINT (35 70)
(1 ligne)

et en EWKB restitué grâce à AsEWKB ()

```
select AsEWKB ('010100000000000000000008041400000000000805140');
```

et comme srid, restitué grâce à la fonction Srid()

```
select Srid('010100000000000000000000000000080414000000000000000805140',::geometry);
```

srid

-1
(1 ligne)

En mode d'entrée, PostGIS accepte aussi le HEXEWKB directement et PostgreSQL le stockera tout simplement en HEXEWKB. Ainsi dans une table test dont la structure serait par exemple

```
CREATE TABLE test (...,
                  the_geom GEOMETRY);
```

les deux insertions suivantes sont équivalentes

```
INSERT INTO test VALUES (...,
                           GeomFromEWKT('SRID=4326;POINT(35 70)'))
```

```
INSERT INTO test VALUES (...,
                      '010100000000000000008041400000000000805140',
                      );
```

Ici il n'est pas besoin de préciser le type (`:geometry`) puisque la colonne `the_geom` est de type `geometry`.

5.5.15 Question : Quelles sont les aires et les périmètres des bâtiments ?

La requête suivante

```
select data as batiment,  
       cast(area2d(the_geom) as decimal(15,2))||' m carre' as Aire,  
       cast(perimeter(the_geom) as decimal(15,2))||' m' as Perimetre  
    from buildings
```

nous renverra comme réponse

batiment	aire	perimetre
Collège Arthur Rimbaud	1370.00 m carre	187.61 m
Résidence des Mousquetaires	2915.00 m carre	216.00 m
Hotel des Impots	1300.00 m carre	190.00 m
E.D.F	476.43 m carre	88.54 m
Bibliothèque Victor Hugo	900.00 m carre	176.57 m
Mairie	936.00 m carre	141.70 m
Office du Tourisme	205.55 m carre	58.92 m
(7 rows)		

5.5.16 Question : Qui est dans le bâtiment Résidence des Mousquetaires ?

La requête suivante

```
select personnes.data as personnes_dans_batiment_2 from personnes,buildings  
      where within(personnes.the_geom,buildings.the_geom)  
        and buildings.data = 'Résidence des Mousquetaires';
```

nous renverra comme réponse

personnes_dans_batiment_2
personne 1
personne 2
personne 3
personne 4
(4 rows)

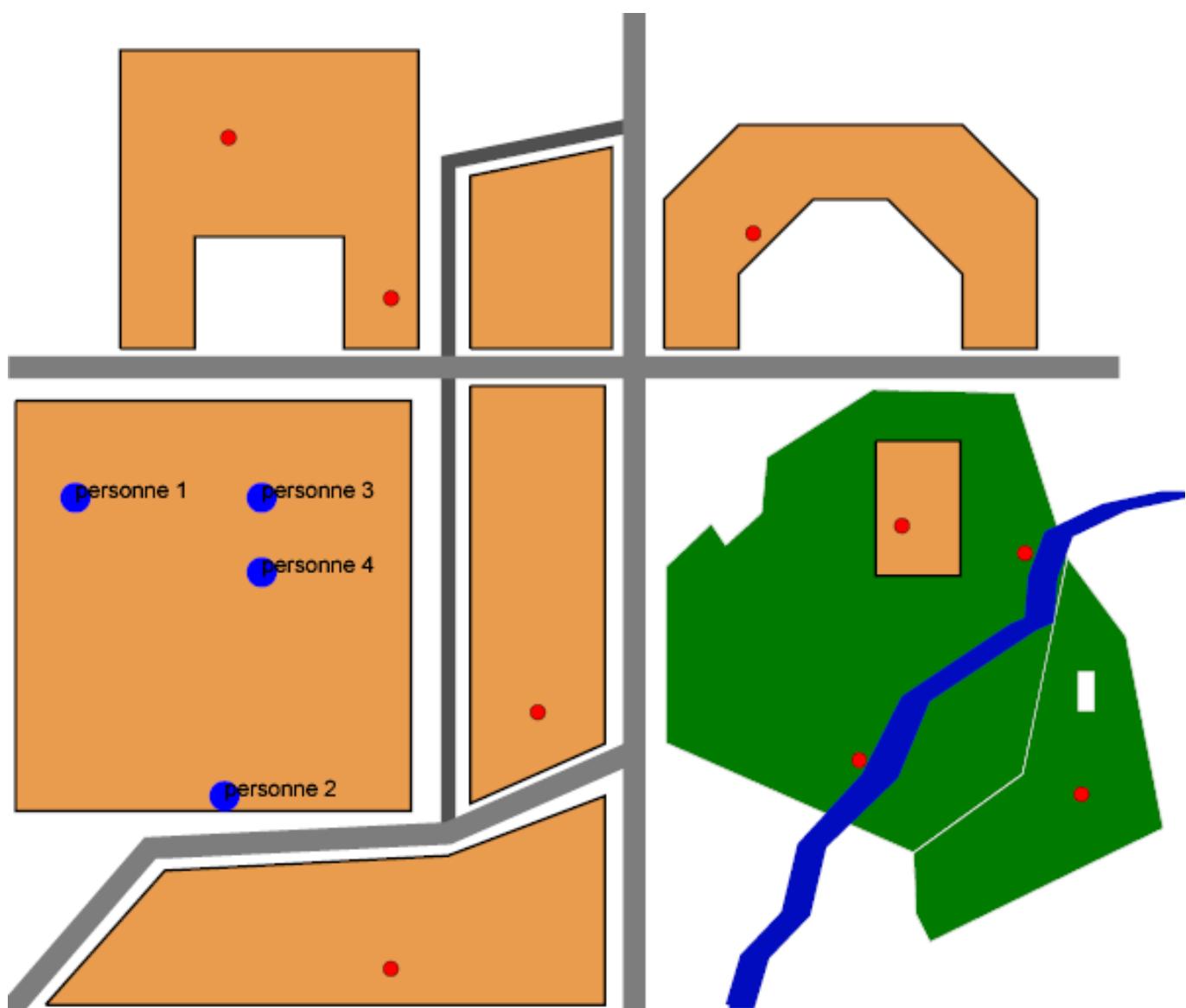


FIG. 5.8 – Personnes présentes dans le bâtiment Résidence des Mousquetaires

5.5.17 Question : Quelles distances séparent les bâtiments ?

La requête suivante

```
select h.data as batiment_1,t.data as batiment_2,  
cast(Distance(t.the_geom,h.the_geom) as decimal(15,2))||' m'  
as distance_entre_batiment from buildings t, buildings h where h.data!=t.data;
```

nous renverra comme réponse

batiment_1	batiment_2	distance_entre_batiment
Résidence des Mousquetaires	Collège Arthur Rimbaud	6.25 m
Hotel des Impots	Collège Arthur Rimbaud	64.97 m
E.D.F	Collège Arthur Rimbaud	60.00 m
Bibliothèque Victor Hugo	Collège Arthur Rimbaud	60.53 m

Mairie	Collège Arthur Rimbaud	5.47 m
Office du Tourisme	Collège Arthur Rimbaud	46.82 m
Collège Arthur Rimbaud	Résidence des Mousquetaires	6.25 m
Hotel des Impots	Résidence des Mousquetaires	7.00 m
E.D.F	Résidence des Mousquetaires	10.63 m
Bibliothèque Victor Hugo	Résidence des Mousquetaires	34.71 m
Mairie	Résidence des Mousquetaires	8.00 m
Office du Tourisme	Résidence des Mousquetaires	62.36 m
Collège Arthur Rimbaud	Hotel des Impots	64.97 m
Résidence des Mousquetaires	Hotel des Impots	7.00 m
E.D.F	Hotel des Impots	7.00 m
Bibliothèque Victor Hugo	Hotel des Impots	33.00 m
Mairie	Hotel des Impots	8.60 m
Office du Tourisme	Hotel des Impots	62.60 m
Collège Arthur Rimbaud	E.D.F	60.00 m
Résidence des Mousquetaires	E.D.F	10.63 m
Hotel des Impots	E.D.F	7.00 m
Bibliothèque Victor Hugo	E.D.F	7.00 m
Mairie	E.D.F	5.00 m
Office du Tourisme	E.D.F	37.47 m
Collège Arthur Rimbaud	Bibliothèque Victor Hugo	60.53 m
Résidence des Mousquetaires	Bibliothèque Victor Hugo	34.71 m
Hotel des Impots	Bibliothèque Victor Hugo	33.00 m
E.D.F	Bibliothèque Victor Hugo	7.00 m
Mairie	Bibliothèque Victor Hugo	9.43 m
Office du Tourisme	Bibliothèque Victor Hugo	12.40 m
Collège Arthur Rimbaud	Mairie	5.47 m
Résidence des Mousquetaires	Mairie	8.00 m
Hotel des Impots	Mairie	8.60 m
E.D.F	Mairie	5.00 m
Bibliothèque Victor Hugo	Mairie	9.43 m
Office du Tourisme	Mairie	36.36 m
Collège Arthur Rimbaud	Office du Tourisme	46.82 m
Résidence des Mousquetaires	Office du Tourisme	62.36 m
Hotel des Impots	Office du Tourisme	62.60 m
E.D.F	Office du Tourisme	37.47 m
Bibliothèque Victor Hugo	Office du Tourisme	12.40 m
Mairie	Office du Tourisme	36.36 m

(42 rows)

5.5.18 Question : Combien de points composent chaque objet de la table great_roads ? - NumPoints()

Nous aurons ici recours à la fonction NumPoint () qui permet de déterminer le nombre de points qui compose un objet :

```
SELECT data, AsText(the_geom), NumPoints(the_geom) FROM great_roads
```

nous renvoie effectivement

data	astext	numpoints
Rue Paul Valéry	LINESTRING(1 1,20 23,60 25,85 36)	4
Rue du Général de Gaulle	LINESTRING(1 87.5,150 87.5)	2
Rue Aristide Briand	LINESTRING(85 1,85 135)	2
(3 lignes)		

5.5.19 Question : Dans la table great_roads, quels sont les premier et dernier point de la Rue Paul Valéry ? - StartPoint(), EndPoint() -

StartPoint () et EndPoint () permettent facilement de répondre à cette question. D'où la requête suivante

```
SELECT data, AsText(the_geom), AsText(StartPoint(the_geom)), AsText(EndPoint(the_geom))
FROM great_roads where data like '%Valéry%'
```

data	astext	astext	astext
Rue Paul Valéry	LINESTRING(1 1,20 23,60 25,85 36)	POINT(1 1)	POINT(85 36)

(1 ligne)

5.5.20 Question : Quels sont les coordonnées des centres des bâtiments (buildings) ? - Centroid() -

Nous allons ici utiliser la fonction Centroid()

```
SELECT data AS "Bâtiment", AsText(Centroid(the_geom)) AS "Centre", Distance(the_geom, Centroid ←
(the_geom)) from buildings;
```

qui nous renverra

Bâtiment	Centre	distance
Collège Arthur Rimbaud	POINT(49.6851581508516 12.0973236009732)	0
Résidence des Mousquetaires	POINT(28.5 55.5)	0
Hotel des Impots	POINT(36 112.884615384615)	0
E.D.F	POINT(72.7431040212695 102.562130275839)	0
Bibliothèque Victor Hugo	POINT(114 107.222222222222)	2.777777777777777
Mairie	POINT(71.7692307692308 58.9487179487179)	0
Office du Tourisme	POINT(123.035 68.545)	0

(7 lignes)

En fonction de leur convexité/concavité, les POLYGON peuvent avoir leur centre situé à l'extérieur. C'est ce qui se passe ici par exemple pour le bâtiment "Bibliothèque Victor Hugo". Ce qui nous amène à la question suivante

5.5.21 Question : Comment garantir de toujours avoir un point sur un POLYGON autre que son centre en dépit de sa convexité/concavité ? - PointOnSurface() -

Le mieux est d'utiliser la fonction PointOnSurface () qui renvoit toujours un tel point

```
SELECT data AS "Bâtiment", AsText(PointOnSurface(the_geom)) AS "Point Trouvé", Distance( ←
the_geom, PointOnSurface(the_geom)) from buildings;
```

confirmé par le résultat

Bâtiment	Point Trouvé	distance
Collège Arthur Rimbaud	POINT(49.72222222222222 16)	0
Résidence des Mousquetaires	POINT(28.5 55.5)	0
Hotel des Impots	POINT(36 110)	0
E.D.F	POINT(72.5 103.5)	0
Bibliothèque Victor Hugo	POINT(96.5 105)	0
Mairie	POINT(72 57)	0
Office du Tourisme	POINT(123.035 68.545)	0

(7 lignes)

Vous pouvez essayez cette requête autant de fois que vous voulez, PointOnSurface vous renverra toujours le même point.

NOTE

PointOnSurface() est une fonction bien utile quand on veut par exemple pourvoir afficher un point sur un POLYGON. Il peut par exemple s'agir du nom de commune, le centre d'un camembert pour des statistiques etc...

5.5.22 Question : Quels sont les points d'intersection entre les petites routes (small_roads) et les grandes routes (great_roads) ?

On pourra utiliser les fonctions respectives AsText() pour avoir les points en WKT et Intersects() pour les tests d'intersection

```
SELECT s.data,g.data,AsText(Intersection(s.the_geom,g.the_geom))
FROM small_roads s, great_roads g
WHERE Intersects(s.the_geom,g.the_geom)
```

renvoyant comme résultat

data		data		astext
Rue Figaro		Rue du Général de Gaulle		POINT(60 87.5)
Rue Figaro		Rue Aristide Briand		POINT(85 120)
Rue Voltaire		Rue Paul Valéry		POINT(60 25)
Rue Voltaire		Rue du Général de Gaulle		POINT(60 87.5)
(4 lignes)				

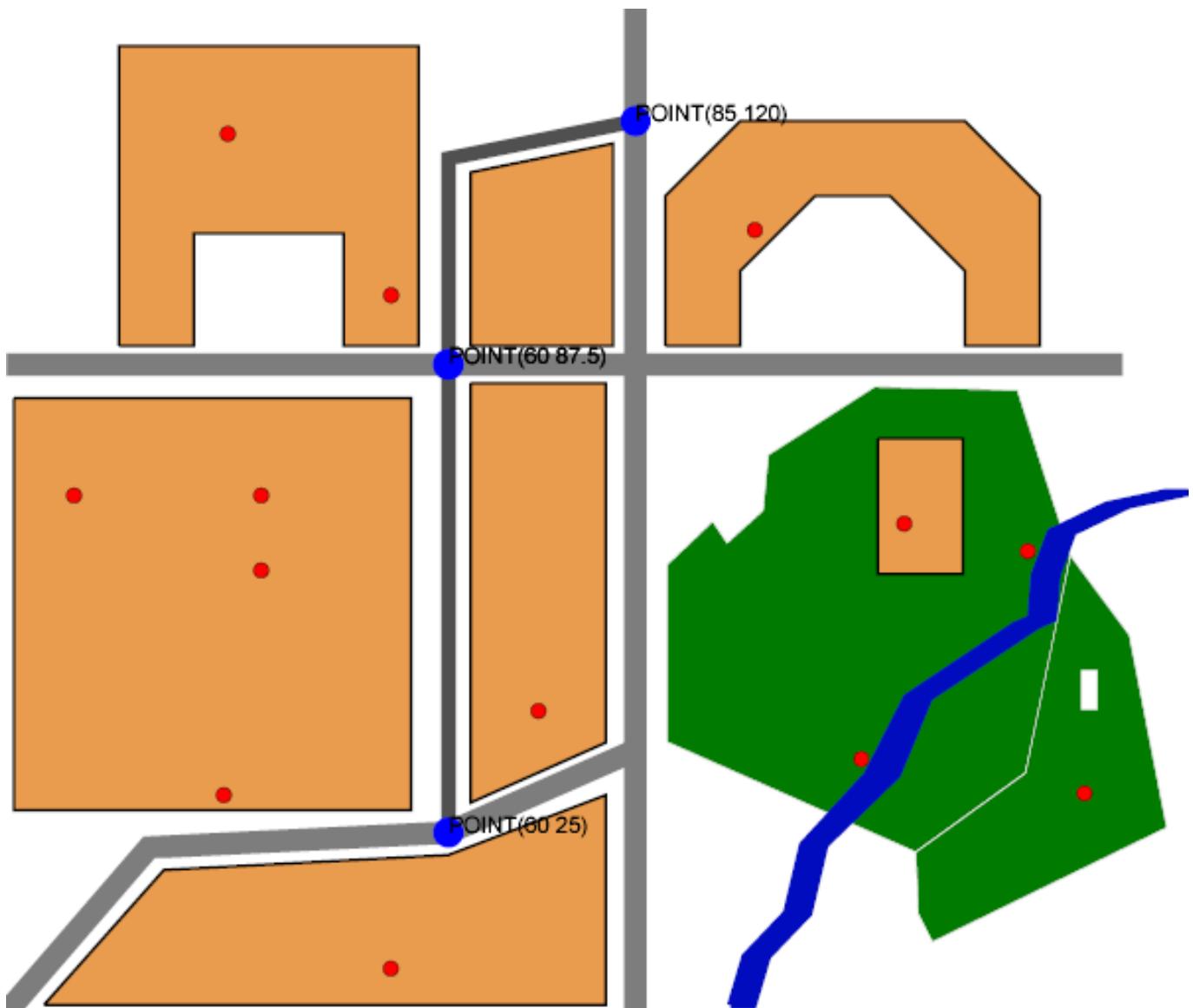


FIG. 5.9 – Points d'intersection entre les tables small_roads et great_roads

5.5.23 Quelle distance (relative au tracé de la rue Paul Valéry) dois-je couvrir si je pars de l'entrée la Rue Paul Valéry (table great_roads) jusqu'à son point de rencontre (intersection) avec la Rue Voltaire (table small_roads) ? - `line_locate_point()`,`line_interpolate_point()` -

Pour répondre à cette question, il est par exemple intéressant d'utiliser la fonction `line_locate_point()` qui retourne pour une LINESTRING une valeur entre 0 et 1 (relatif à un pourcentage) relative à la longueur 2D de la LINESTRING en question selon la position d'un point sur ou à l'extérieur de la LINESTRING.

NOTE

Pour les besoins de cet exemple, je me sers directement du résultat trouvé antérieurement à la sous-section précédente, à savoir que mon point d'intersection est `POINT(60 25)`. On peut tout aussi bien prendre un autre point.

A partir de la fonction `Line_locate_point()`, je formule ma requête comme suit

```
SELECT Line_Locate_Point(the_geom, GeomFromText('SRID=-1;POINT(60 25)')) FROM great_roads ←  
WHERE data = 'Rue Paul Valéry';
```

qui me sort comme résultat

```
line_locate_point  
-----  
0.716763707222655  
(1 ligne)
```

Ainsi je dois couvrir plus au moins 71% de la Rue Paul Valéry pour arriver au point POINT(60 25). La réponse à ma question est obtenue à partir du résultat de la requête suivante

```
SELECT Line_Locate_Point(the_geom, GeomFromText('SRID=-1;POINT(60 25)'))*length2d(the_geom) ←  
FROM great_roads WHERE data = 'Rue Paul Valéry';
```

soit que je dois parcourir

```
?column?  
-----  
69.1188524964988  
(1 ligne)
```

L'affichage sera donc le suivant

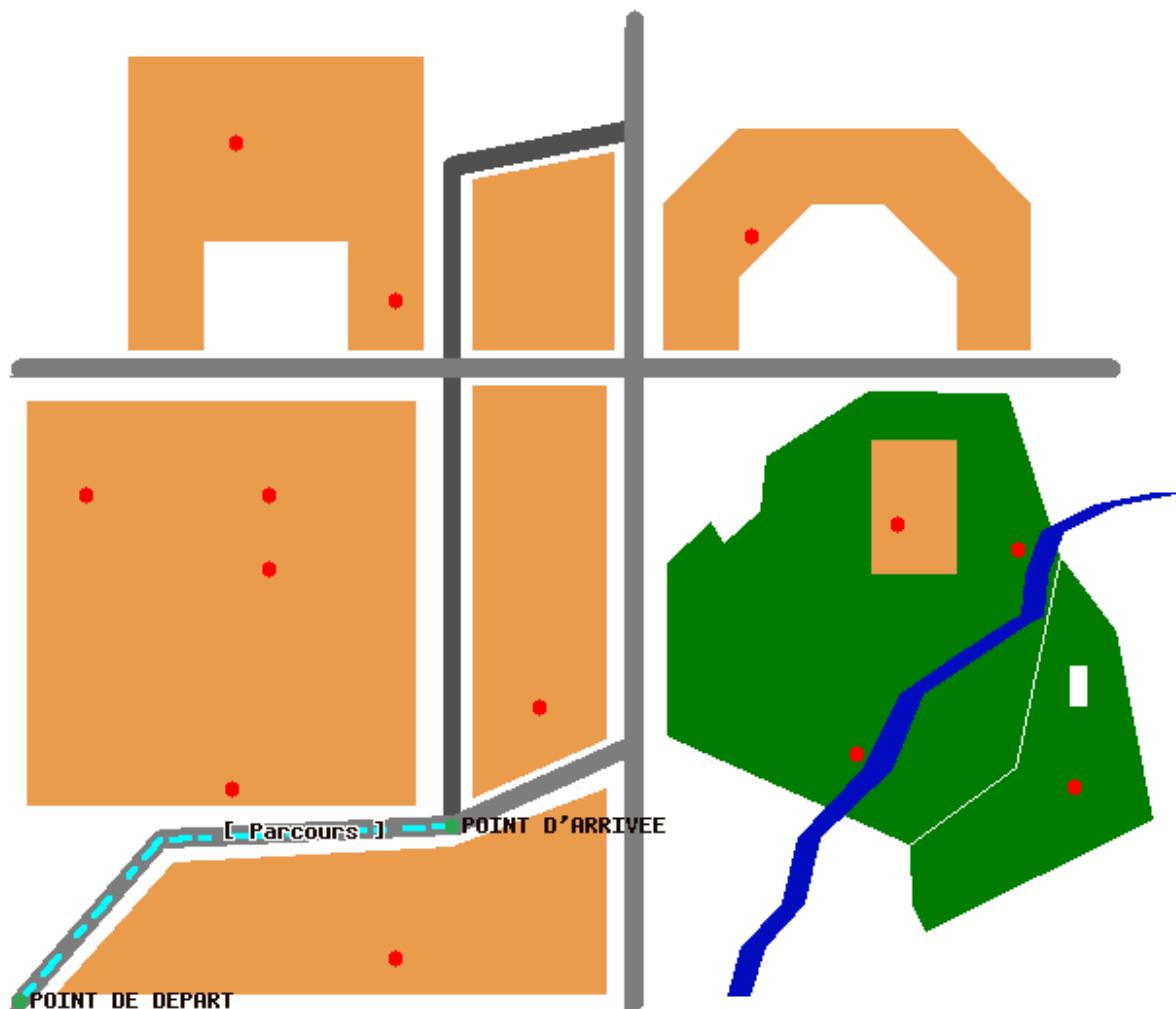


FIG. 5.10 – Parcours effectué sur 71% de la rue Paul Valéry

Au lieu de préciser un POINT sur une LINESTRING pour obtenir le pourcentage que cela représente au niveau de la longueur de la LINESTRING, il est également possible d'effectuer l'opération inverse. Je veux dire par là qu'il existe la fonction `line_interpolate_point(linestring, location)` qui prend en entrée le pourcentage voulu et restitue en sortie le point attendu.

Pour le résultat traité ici, on aura

```
select astext(line_interpolate_point(the_geom, 0.716763707222655)) from great_roads where ←  
  data='Rue Paul Valéry';  
  astext  
-----  
  POINT(60 25)  
(1 ligne)
```

Voyons déjà une exemple de MapFile - fichier qui pilote MapServer - comment faire pour afficher cette image. Pour celà, nous pourrons par exemple en 3 layers

1. Afficher le parcours ainsi que le message "Parcours"

```
LAYER
  CONNECTION "user=david dbname=madatabase host=localhost"
  CONNECTIONTYPE POSTGIS
  DATA "line_substring from (select '[ Parcours ]'::text as id,line_substring(←
    the_geom,0,0.716763707222655) from great_roads where data like'%Paul%') as foo ←
    USING UNIQUE id USING SRID=-1"
  LABELITEM "id"
  METADATA
  END
  NAME "parcours"
  SIZEUNITS PIXELS
  TOLERANCE 0
  TYPE LINE
  SIZEUNITS PIXELS
  STATUS ON
  TOLERANCEUNITS PIXELS
  UNITS METERS
  CLASS
    METADATA
    END
  LABEL
    SIZE MEDIUM
    TYPE BITMAP
    BUFFER 0
    COLOR 22 8 3
    FORCE FALSE
    MINDISTANCE -1
    MINFEATURESIZE -1
    OFFSET 0 0
    OUTLINECOLOR 255 255 255
    PARTIALS TRUE
    POSITION CC
  END
  STYLE
    ANGLE 360
    OUTLINECOLOR 0 255 255
    SIZE 3
    SYMBOL "dash"
  END
END
END
```

2. Pour afficher le point de départ ainsi que le message "POINT DE DEPART"

```
LAYER
  CONNECTION "user=david dbname=madatabase host=localhost"
  CONNECTIONTYPE POSTGIS
  DATA "geometryfromtext from (select 1 as id,'POINT DE DEPART'::text as data, ←
    geometryfromtext('SRID=-1;POINT(1 1)') ) as foo USING UNIQUE id USING SRID=-1"
  LABELITEM "data"
  METADATA
  END
  NAME "depart"
  SIZEUNITS PIXELS
  STATUS DEFAULT
  TOLERANCE 0
  TOLERANCEUNITS PIXELS
  TYPE POINT
  UNITS METERS
  CLASS
    LABEL
      SIZE MEDIUM
```

```
TYPE BITMAP
BUFFER 0
COLOR 22 8 3
FORCE FALSE
MINDISTANCE -1
MINFEATURESIZE -1
OFFSET 0 0
PARTIALS TRUE
POSITION CR
END
METADATA
END
STYLE
    ANGLE 360
        COLOR 45 167 78
        SIZE 8
        SYMBOL "circle"
    END
END
END
```

3. Pour le point d'arrivée et pour afficher le message "POINT D'ARRIVEE", je crées le même layer en changeant le titre du LAYER et les coordonnées du point POINT(60 25)

Voyons dans la prochaine sous-section, un autre opérateur sur les objets de type linéaires

5.5.24 Extraire de la Rue Paul Valéry la portion de route joignant le premier cinquième 1/5 au quatrième cinquième (4/5) de cette rue - line_substring()

On utilisera pour ce faire la fonction line_substring() avec comme requête possible

```
SELECT AsText(Line_SubString(the_geom,1/5::real,4/5::real)) FROM great_roads WHERE data = ↪
'Rue Paul Valéry';
```

retournant comme résultat

```
astext
-----
LINESTRING (13.6059550593843 15.5963690161291,20 23,60 25,67.3468950572286 ↪
28.2326338251806)
(1 ligne)
```

Au niveau de l'affichage, on obtient

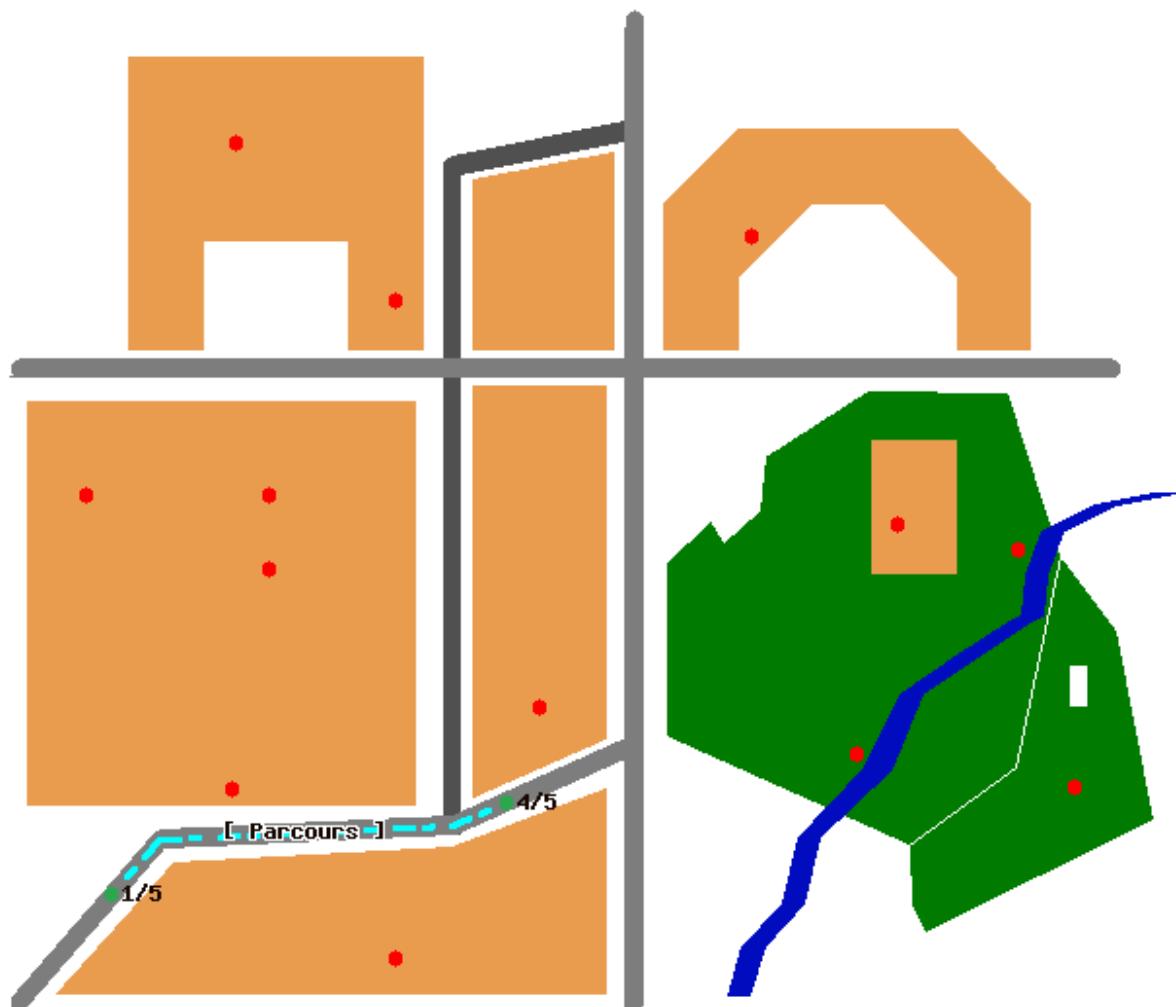


FIG. 5.11 – Portion de route allant du 1/5 au 4/5 de la Rue Paul Valéry

A titre d'information, par rapport à la sous-section précédente et respectivement par rapport aux divers layers antérieurement présentées, on changera juste les valeurs des DATA et respectifs en faisant

1. LAYER 1 :

```
DATA "line_substring from (select '[ Parcours ]'::text as id,line_substring(the_geom ←
    ,1/5::real,4/5::real) from great_roads
    where data like '%Paul%') as foo USING UNIQUE id USING SRID=-1"
```

2. LAYER 2 :

```
DATA "startpoint from (select 1 as id,'1/5'::text as data,startpoint(line_substring( ←
    the_geom,1/5::real,4/5::real)) from great_roads
    where data like '%Paul%') as foo USING UNIQUE id USING SRID=-1"
```

3. LAYER 3 : en utilisant la fonction endpoint() pour extraire le dernier point de la portion de route

```
DATA "endpoint from (select 1 as id,'4/5'::text as data,endpoint(line_substring( ←
    the_geom,1/5::real,4/5::real)) from great_roads
where data like '%Paul%') as foo USING UNIQUE id USING SRID=-1"
```

5.5.25 Question : Quel bâtiment est le plus proche de la personne 2 ?

Plusieurs formulations sont possibles. Les requêtes suivantes

```
select q.data from personnes z,buildings q
where z.data like 'personne 2' and
Distance(z.the_geom,q.the_geom)=
(select min(foo.valeur) from (select h.data as nom,distance(t.the_geom,h.the_geom)
as valeur from personnes t, buildings h where t.data like 'personne 2') foo);
```

et celle-ci

```
SELECT data from(
SELECT a.data,Distance(a.the_geom,b.the_geom) FROM buildings a,personnes b
 WHERE b.data='personne 2'
 ORDER BY distance ASC limit 1)
as foo;
```

nous renvoient comme réponse

```
data
-----
Résidence des Mousquetaires
(1 ligne)
```

5.5.26 Question : Quel bâtiment ne contient aucune personne ?

La requête suivante

```
select b.data from buildings b,
       (select geomunion(the_geom) from personnes) gp
      where
        not intersects(gp.geomunion,b.the_geom);
```

nous renverra comme réponse

```
data
-----
E.D.F
(1 row)
```

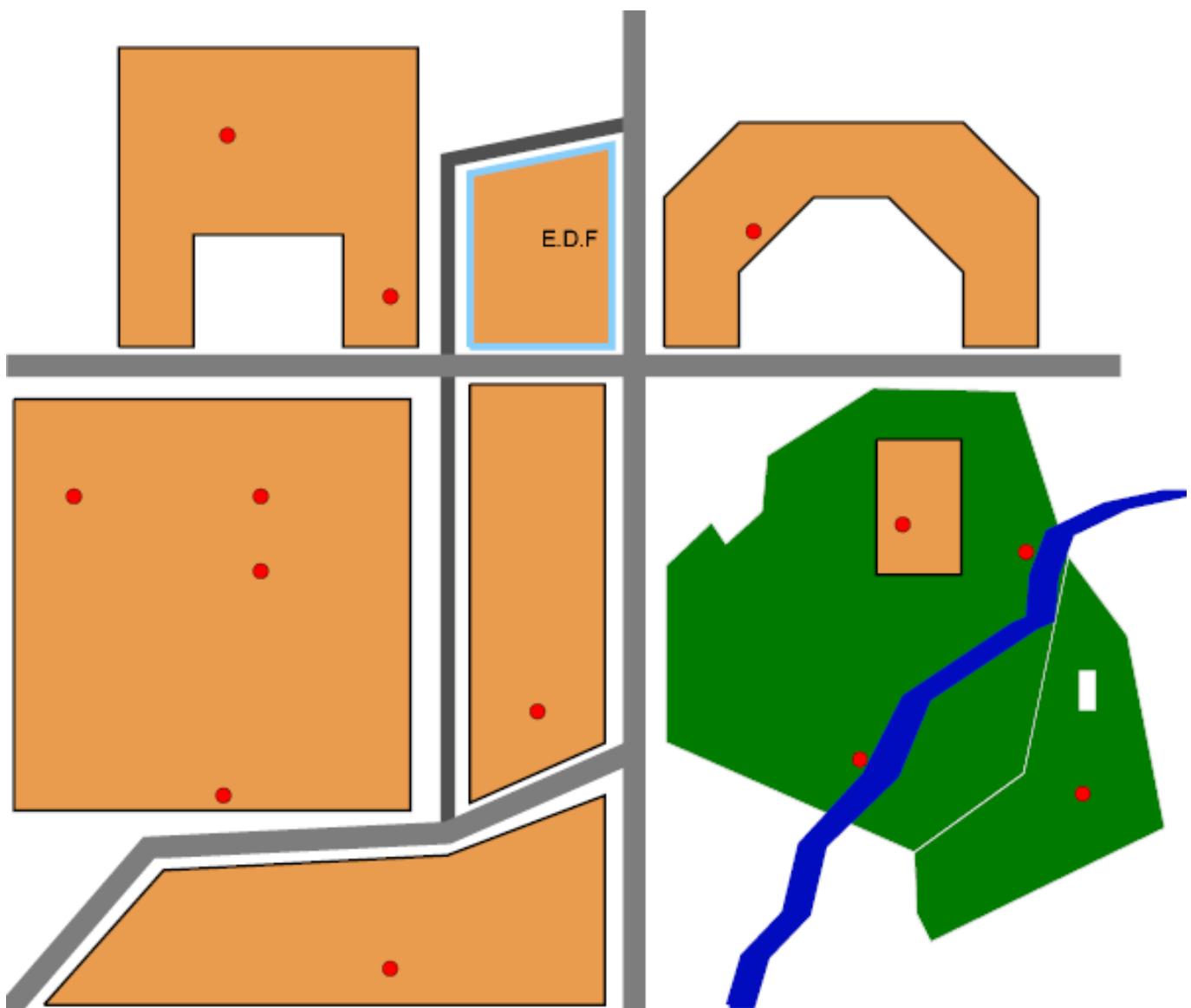


FIG. 5.12 – Bâtiment ne contenant aucune personne : EDF

5.5.27 Question : Quels sont les personnes présentes dans les bâtiments ?

On utilisera pour cela la requête suivante

```
select p.data,b.data from personnes p, buildings b where intersects(p.the_geom,b.the_geom)
```

qui nous renverra

data		data
personne 1		Résidence des Mousquetaires
personne 2		Résidence des Mousquetaires
personne 3		Résidence des Mousquetaires
personne 4		Résidence des Mousquetaires
personne 5		Hotel des Impots
personne 6		Hotel des Impots

```
personne 7 | Bibliothèque Victor Hugo
personne 9 | Office du Tourisme
personne 12 | Mairie
personne 13 | Collège Arthur Rimbaud
(10 lignes)
```

5.5.28 Question : Combien y-a-t-il de personnes par bâtiments ?

Cette requête est un peu plus poussée que la précédente car il se peut que ce nombre puisse valoir zéro. On peut par exemple utiliser la négation de la fonction de Contains() pour obtenir le nombre de personnes non présentes par bâtiment. Ce dernier nombre est alors soustrait du nombre total de personnes - nombre que l'on peut obtenir par le fonction Count() de PostgreSQL. Ce qui donnerait par exemple comme requête

```
SELECT (SELECT Count(*) FROM personnes) -Count(p.data) AS nombre_de_personnes,b.data as batiment
FROM
buildings b , personnes p
WHERE NOT Contains(b.the_geom,p.the_geom)
GROUP BY b.data ORDER BY b.data
```

nous renvoyant comme résultat

nombre_de_personnes	batiment
1	Bibliothèque Victor Hugo
1	Collège Arthur Rimbaud
0	E.D.F
2	Hotel des Impots
1	Mairie
1	Office du Tourisme
4	Résidence des Mousquetaires

(7 rows)

5.5.29 Question : Quel est l'aire d'intersection entre la rivière et les parcs ?

La requête suivante

```
select cast(area2d(intersection(r.the_geom,p.the_geom)) as decimal(15,1)) ||' m carre' as Aire
      from rivers r,
           parcs p
      where intersects(r.the_geom,p.the_geom);
```

nous renverra comme réponse

```
aire
-----
123.1 m carre
(1 row)
```

L'affichage avec MapServer peut être obtenu en utilisant par exemple le LAYER suivant dans une mapfile

```
LAYER
  CONNECTION "user=david dbname=madatabase host=localhost"
  CONNECTIONTYPE POSTGIS
  DATA "intersection FROM (select r.gid,intersection(r.the_geom,p.the_geom) FROM rivers r
        ,parcs p
       WHERE intersects(r.the_geom,p.the_geom)) foo USING UNIQUE gid USING SRID=-1"
```

```
NAME "requete"
TYPE POLYGON
STATUS DEFAULT
CLASS
STYLE
  ANGLE 360
  OUTLINECOLOR 0 0 0
  COLOR 134 100 0
  SIZE 5
  SYMBOL 0
END
END
END
```

Au niveau du visuel, on obtiendrait

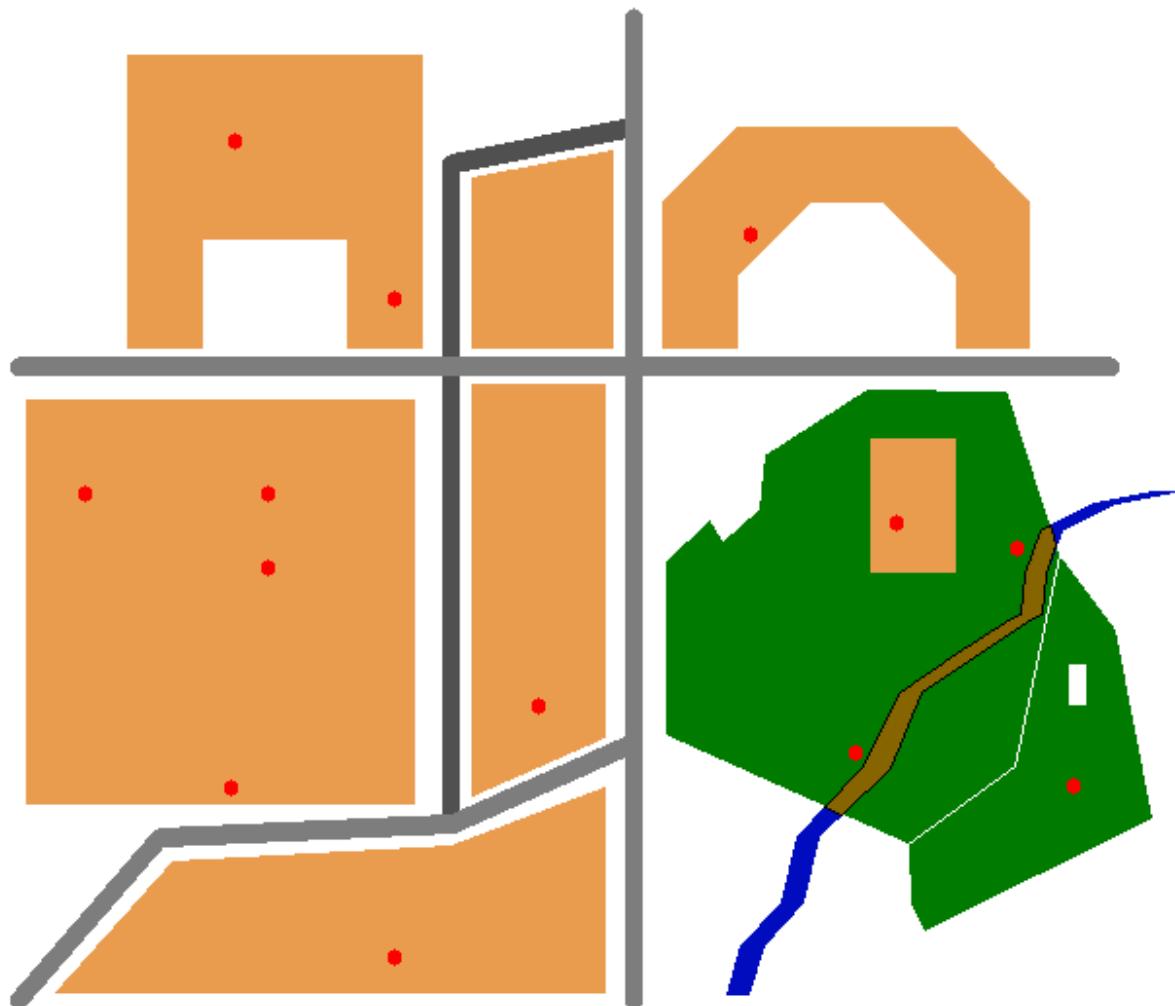


FIG. 5.13 – Intersection entre la rivière et les parcs

5.5.30 Question :Quel bâtiment est contenu dans le parc Mangueret I ? - Contains() -

La fonction Contains(A,B) permet de savoir si l'objet A contient l'objet B. De ce fait, la requête

```
select b.data from buildings b,parcs where  
Contains(parcs.the_geom,b.the_geom)
```

renvoie le résultat immédiat

```
data  
-----  
Office du Tourisme  
(1 row)
```

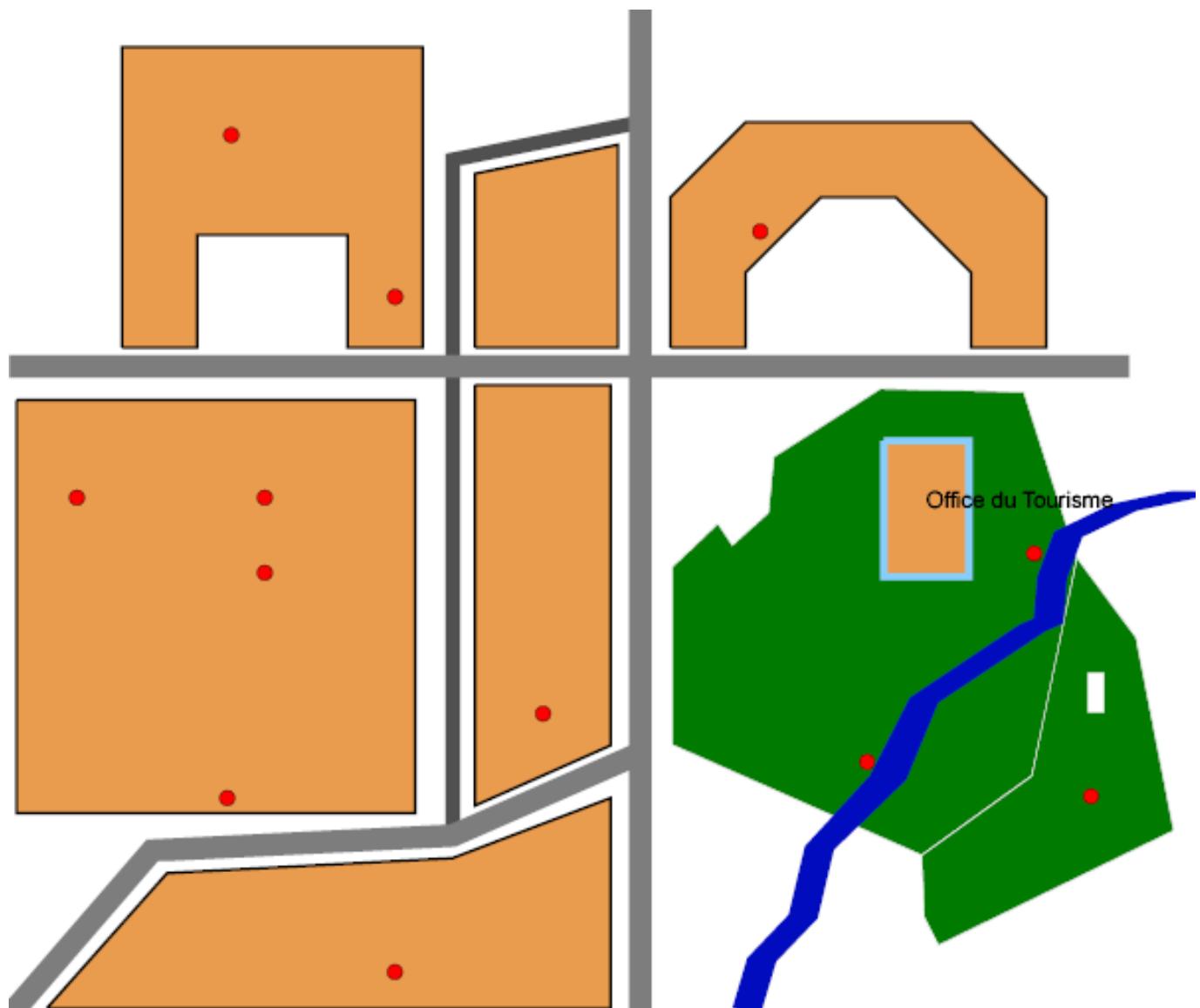


FIG. 5.14 – Bâtiment contenu dans le parc Mangueret I : Office du Tourisme

5.5.31 Question :Quelles sont les personnes proches de la rivière dans un rayon de 5 mètres ? - Buffer () -

Ici, nous allons utiliser une combinaison des fonctions Contains() que nous avons vu précédemment et de la fonction Buffer(A, r) où A est l'objet géométrique auquel on applique un buffer de rayon r . Pour rappel , Buffer(A,r) est l'objet - ensemble de points - dont la distance à l'objet A est inférieure ou égale à r. La requête

```
select p.data from personnes p,rivers  
where contains(buffer(rivers.the_geom,5),p.the_geom);
```

nous renvoie alors comme réponse

```
data  
-----  
personne 8  
personne 10  
(2 rows)
```

Au niveau de MapServer pour un rendu visuel, il faut deux layers pour obtenir les résultats souhaités. Il en faut un pour afficher le buffer en question et l'autre pour afficher les résultats de la requête précédente.

1. Pour afficher le buffer, on pourra utiliser le layer suivant :

```
LAYER  
  CONNECTION "user=david dbname=madatabase host=localhost"  
  CONNECTIONTYPE POSTGIS  
  DATA "buffer from (select gid,buffer(rivers.the_geom,5) from rivers) foo  
        USING UNIQUE gid USING SRID=-1"  
  NAME "requete_1"  
  TYPE LINE  
  STATUS DEFAULT  
  CLASS  
    STYLE  
      OUTLINECOLOR 20 156 78  
      SIZE 10  
      SYMBOL 0  
    END  
  END  
END
```

2. Pour afficher les données attendues, on pourra proposer :

```
LAYER  
  CONNECTION "user=david dbname=madatabase host=localhost"  
  CONNECTIONTYPE POSTGIS  
  DATA "the_geom from (select * from personnes where  
        contains(buffer(rivers.the_geom,5),personnes.the_geom)) foo  
        USING UNIQUE gid USING SRID=-1"  
  NAME "requete_2"  
  LABELITEM "data"  
  STATUS DEFAULT  
  TYPE POINT  
  CLASS  
    LABEL  
      COLOR 22 8 3  
    END  
    STYLE  
      COLOR 255 0 0  
      SIZE 8  
      SYMBOL "circle"  
    END  
  END  
END
```

NOTE

Il est aussi possible de n'avoir recours qu'à un seul layer pour afficher à la fois l'objet et les personnes attendues ! Ce sera le cas lors de la prochaine section - sur les requêtes concernant une certaine table communes_avoisinantes -.

Le visuel sera alors le suivant

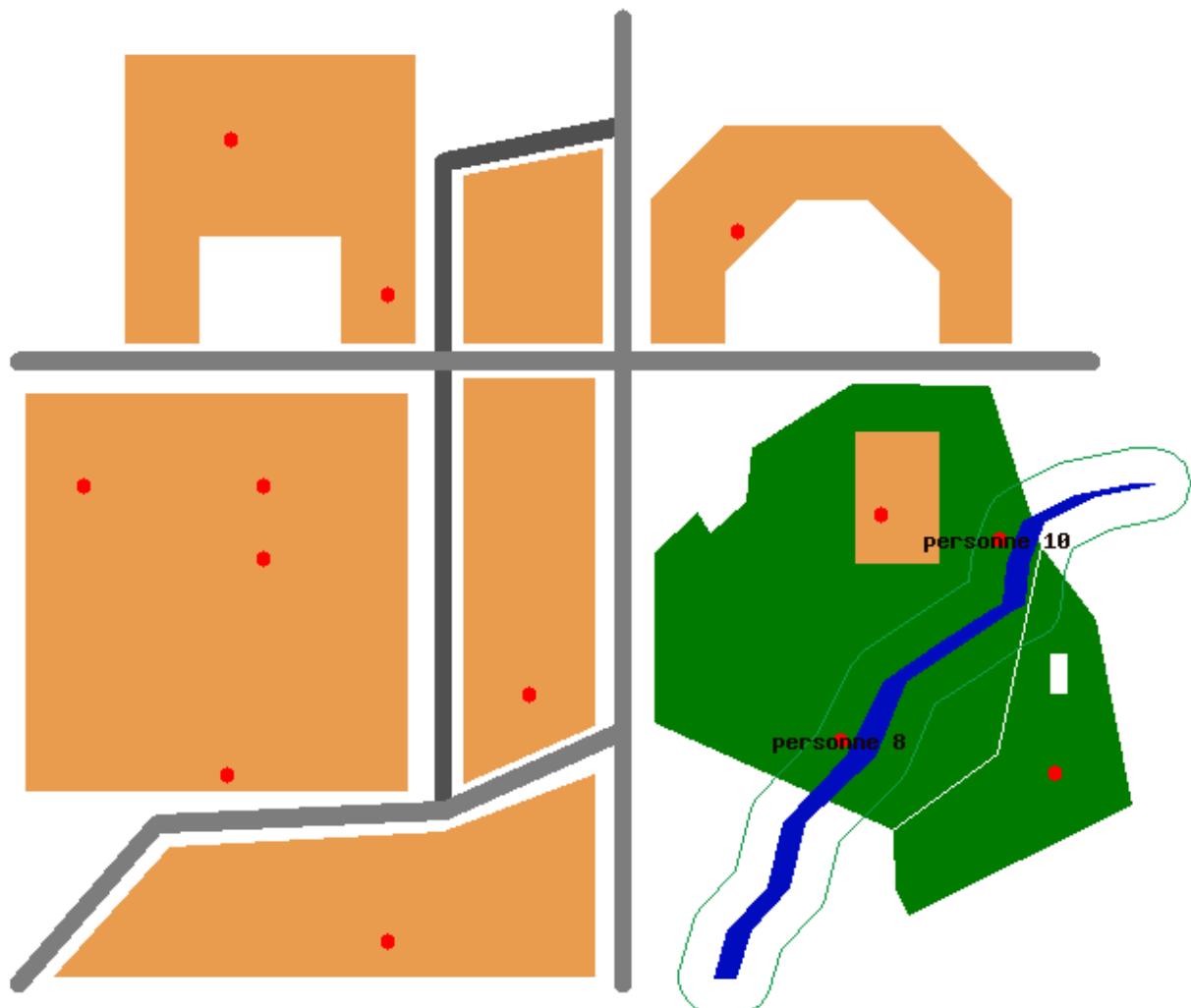


FIG. 5.15 – Buffer de 5 mètres sur la rivière : les personnes 8 et 10 y sont présentes

5.5.32 Question : Quel parc contient un "trou" ? - Nrings() -

La requête suivante avec l'emploi de la fonction Nrings(geometry)

```
select data from parcs where nrings(the_geom)>1;
```

nous renvoie comme réponse

```
data
```

```
-----
```

```
Parc Mangueret II  
(1 ligne)
```

En effet, cette fonction permet par exemple dans le cas d'un POLYGON ou d'un MULTIPOLYGON de déterminer le nombre de polygones faisant partie de la collection constituant l'objet :

```
select data,nrings(the_geom) from parcs;
```

```
data      | nrings
-----+-----
Parc Mangueret I |    1
Parc Mangueret II |   2
(2 lignes)
```

5.5.33 Question : Quels sont les bâtiments que rencontrent la ligne qui relie la personne 5 à la personne 13 ? - MakeLine() -

Pour répondre à cette question, il faut commencer par construire cette ligne en utilisant la fonction MakeLine(geometry,geometry) qui permet de relier entre eux deux points :

```
select astext(makeline(a.the_geom,b.the_geom)) from personnes a,personnes b
where a.data='personne 5' and b.data='personne 13';
```

```
astext
-----
LINESTRING (30.54 118.28,52.32 6.84)
(1 ligne)
```

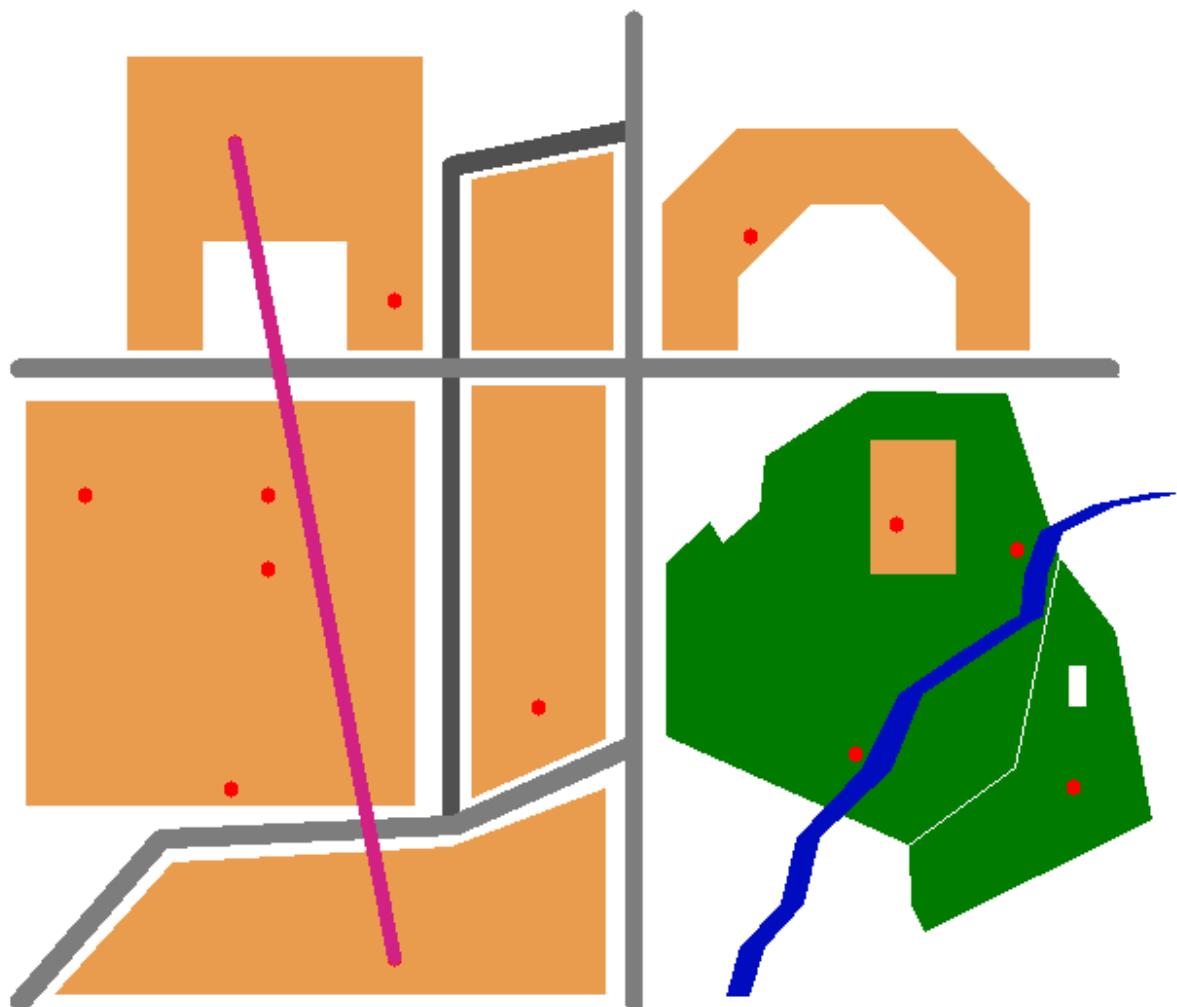


FIG. 5.16 – Ligne reliant les points désignant les personnes 5 et 13

Il en résultera donc la requête suivante

```
select build.data from buildings build where
intersects(
build.the_geom,
(select makeline(a.the_geom,b.the_geom) from personnes a,personnes b where a.data='personne ↔
      5'
and b.data='personne 13')
);
```

data

Collège Arthur Rimbaud
Résidence des Mousquetaires
Hotel des Impots
(3 lignes)

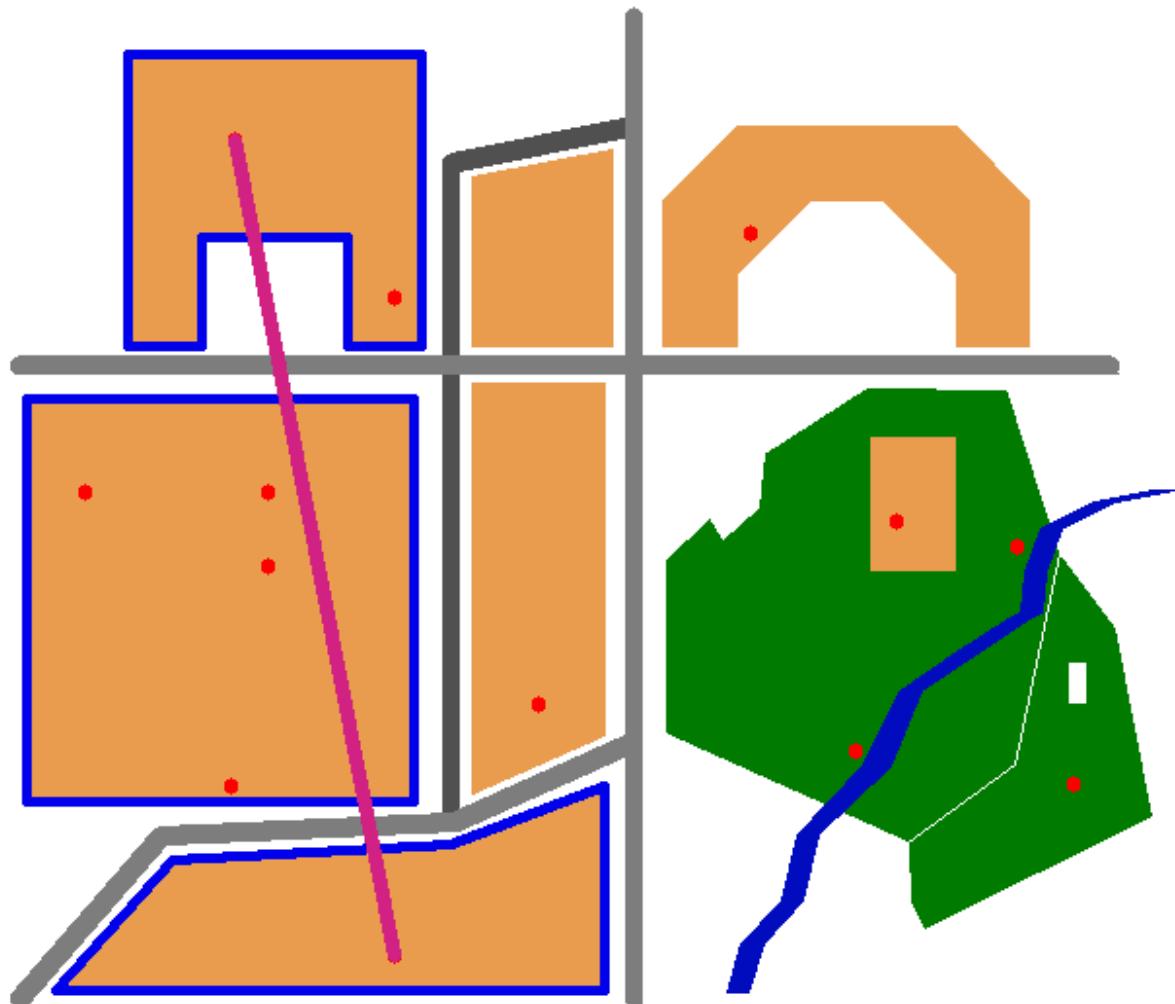


FIG. 5.17 – Bâtiments (table buildings) que rencontre la ligne reliant les points désignant les personnes 5 et 13

5.5.34 Question : Comment arrondir la position des personnes (table personnes) avec un chiffre après la virgule ? -SnapToGrid()-

Le mieux est d'utiliser la fonction SnapToGrid() comme suit

```
madatabase=# SELECT data, astext(the_geom), astext(snapToGrid(the_geom,0.1)) FROM personnes;
   data    |      astext      |      astext
-----+-----+-----+
personne 1 | POINT(10 70)    | POINT(10 70)
personne 2 | POINT(30 30)    | POINT(30 30)
personne 3 | POINT(35 70)    | POINT(35 70)
personne 4 | POINT(35 60)    | POINT(35 60)
personne 5 | POINT(30.54 118.28) | POINT(30.5 118.3)
personne 6 | POINT(52.36 96.73) | POINT(52.4 96.7)
personne 7 | POINT(100.94 105.44) | POINT(100.9 105.4)
personne 8 | POINT(115.16 34.81) | POINT(115.2 34.8)
```

```
personne 9 | POINT (120.89 66.23) | POINT (120.9 66.2)
personne 10 | POINT (137.4 62.56) | POINT (137.4 62.6)
personne 11 | POINT (144.97 30.23) | POINT (145 30.2)
personne 12 | POINT (72.04 41.23) | POINT (72 41.2)
personne 13 | POINT (52.32 6.84) | POINT (52.3 6.8)
(13 lignes)
```

5.5.35 Application : Utiliser les déclencheurs (triggers) en PL/PGSQL de PostgreSQL pour suivre à la trace la personne 7 quand elle se déplace. Selon sa position, savoir quel est le bâtiment qui lui est le plus proche ou le bâtiment dans lequel elle se trouve ?

5.5.35.1 Fonction, table et trigger nécessaires

Le langage PL/PGSQL sert tirer profit des procédures déclencheurs. Nous allons ici créer une table `suivi_de_la_personne_7` qui enregistrera à tout moment demandé enregistrera les informations suivantes :

utilisateur correspondant au nom de l'utilisateur sous de PostgreSQL qui aura fait un update sur les données spatiales de la "personne 7" ;
date correspondant à l'heure à laquelle a eu lieu le déplacement ;
suivi qui précisera si la personne est "dans" ou "proche" du bâtiment ;
position qui donnera la position de la personne.

Nous aurons aussi besoin d'une fonction `get_info_on_personne_7()` écrite en PL/PGSQL appelé par le déclencheur qui nous renverra le bâtiment qui contient la personne 7 et dans le cas contraire le premier bâtiment qui lui est le plus proche.

```
/*
   Rappel pour effacer un trigger
   drop trigger suivi_7_date on personnes;
*/



/*
   Table qui contiendra le suivi de la
   personne 7 lors de ses déplacements
   avec effacement éventuel avant création
*/
select drop_table_if_exists('suivi_de_la_personne_7',false);

create table suivi_de_la_personne_7(
utilisateur text,
date text,
suivi text,
position text
);
/*
   Fonction appelée par le trigger
*/
create or replace function get_info_on_personne_7() returns trigger as $$
declare
j record;
begin
if (TG_OP = 'UPDATE') then
    SELECT into j data,distance,astext(the_geom) from(
        SELECT a.data,Distance(a.the_geom,b.the_geom),b.the_geom FROM buildings a, ←
        personnes b
        WHERE b.data='personne 7'
        ORDER BY distance ASC limit 1)
    as foo;
    if j.distance=0 then
        insert into suivi_de_la_personne_7 values
```

```
(current_user,now()),'La personne 7 est dans le batiment '''||j.data::text||'''',j ←
    .astext::text);

        RAISE NOTICE 'La personne 7 est dans le batiment ''%''',j.data;
        else
            insert into suivi_de_la_personne_7 values
        (current_user,now(),'La personne 7 est proche du batiment '''||j. ←
            data::text||'''',j.astext::text);

        RAISE NOTICE 'La personne 7 est proche du batiment ''%''',j.data;
    end if;
end if;
return new;
end;

$$ language plpgsql;
/*
Création du trigger
*/
CREATE TRIGGER suivi_7_date AFTER UPDATE ON personnes
    EXECUTE PROCEDURE get_info_on_personne_7();
```

5.5.35.2 Modifications de la position par la commande UPDATE et GeometryFromText

Procédons maintenant à quelques exemples de mise à jour des données spatiales de la personne 7 :

```
madatabase=# update personnes set the_geom=geometryfromtext('POINT(100.94 105.44)',-1) ←
    where data='personne 7';
INFO: La personne 7 est dans le batiment 'Bibliothèque Victor Hugo'
UPDATE 1
madatabase=# update personnes set the_geom=geometryfromtext('POINT(100 70)',-1) where data ←
    ='personne 7';
INFO: La personne 7 est proche du batiment 'Office du Tourisme'
UPDATE 1
madatabase=# update personnes set the_geom=geometryfromtext('POINT(120 66)',-1) where data ←
    ='personne 7';
INFO: La personne 7 est dans le batiment 'Office du Tourisme'
UPDATE 1
madatabase=# update personnes set the_geom=geometryfromtext('POINT(0 0)',-1) where data=' ←
    personne 7';
INFO: La personne 7 est proche du batiment 'Collège Arthur Rimbaud'
UPDATE 1
madatabase=# update personnes set the_geom=geometryfromtext('POINT(60 87.5)',-1) where data ←
    ='personne 7';
INFO: La personne 7 est proche du batiment 'E.D.F'
UPDATE 1
```

5.5.35.3 Suivi des déplacements

La simple requête suivante nous renvoie les divers informations obtenues lors du déplacement de la personne 7

```
testgis=# select * from suivi_de_la_personne_7 ;
utilisateur | date | position | suivie
-----+-----+-----+-----
postgres | 2006-01-12 12:05:01.989762+01 | La personne 7 est dans le batiment ' ←
    Bibliothèque Victor Hugo' | POINT(100.94 105.44)
```

```
postgres | 2006-01-12 12:05:20.824226+01 | La personne 7 est proche du batiment 'Office ↵
du Tourisme' | POINT(100 70)
postgres | 2006-01-12 12:05:41.106236+01 | La personne 7 est dans le batiment 'Office ↵
du Tourisme' | POINT(120 66)
postgres | 2006-01-12 12:06:05.024382+01 | La personne 7 est proche du batiment ' ↵
Collège Arthur Rimbaud' | POINT(0 0)
postgres | 2006-01-12 12:07:31.859971+01 | La personne 7 est proche du batiment 'E.D.F' ↵
| POINT(60 87.5)
(6 lignes)
```

C'est le genre de petite application qui peut par exemple servir lors de suivi par une application-cliente (front-end) par le Web par exemple : interface cartographique en SVG

5.5.35.4 Affichage avec MapServer

Afin d'obtenir un rendu adéquate, on va commencer par stocker les diverses positions dans une petite table que nous appelerons ici mouvement :

```
create table mouvement(id serial primary key);
select addgeometrycolumn('mouvement','the_geom',-1,'POINT',2);
insert into mouvement(the_geom) values(geometryfromtext('POINT(100.94 105.44)',-1));
insert into mouvement(the_geom) values(geometryfromtext('POINT(100 70)',-1));
insert into mouvement(the_geom) values(geometryfromtext('POINT(120 66)',-1));
insert into mouvement(the_geom) values(geometryfromtext('POINT(0 0)',-1));
insert into mouvement(the_geom) values(geometryfromtext('POINT(60 87.5)',-1)) ;
```

Afin d'afficher les diverses positions successives, on peut par exemple opter pour un layer comme suit

```
LAYER
  CONNECTION "user=david dbname=madatabase host=localhost"
  CONNECTIONTYPE POSTGIS
  DATA "the_geom from mouvement"
  LABELITEM "id"
  METADATA
    END
  NAME "personnes"
  SIZEUNITS PIXELS
  STATUS DEFAULT
  TOLERANCE 0
  TOLERANCEUNITS PIXELS
  TYPE POINT
  UNITS METERS
  CLASS
    LABEL
      SIZE LARGE
      TYPE BITMAP
      BUFFER 0
      COLOR 0 0 255
      FORCE FALSE
      MINDISTANCE -1
      MINFEATURESIZE -1
      OFFSET 0 0
      PARTIALS TRUE
      POSITION CC
    BACKGROUNDCOLOR 255 255 255
    END
    METADATA
    END
    STYLE
      ANGLE 360
      COLOR 255 0 0
```

```
    SIZE 2
    SYMBOL "circle"
  END
END
END
```

On peut si on le souhaite joindre aussi les diverses positions successives entre elles

```
LAYER
  CONNECTION "user=david dbname=madatabase host=localhost"
  CONNECTIONTYPE POSTGIS
  DATA "makeline from (select 1 as id,makeline(the_geom) from mouvement) as foo USING ↪
        UNIQUE id USING SRID=-1"
  METADATA
  END
  NAME "requete_0"
  SIZEUNITS PIXELS
  STATUS ON
  TOLERANCE 0
  TOLERANCEUNITS PIXELS
  TYPE LINE
  UNITS METERS
  CLASS
    LABEL
      SIZE MEDIUM
      TYPE BITMAP
      BUFFER 0
      COLOR 22 8 3
      FORCE FALSE
      MINDISTANCE -1
      MINFEATURESIZE -1
      OFFSET 0 0
      PARTIALS TRUE
      POSITION CC
    END
    METADATA
    END
    STYLE
      ANGLE 360
      COLOR 209 33 130
      SIZE 3
      SYMBOL "dash"
    END
  END
END
```

En ayant pris soin de définir quelque part dans la mapfile, le symbole "dash" pour une ligne discontinue

```
SYMBOL
  NAME "dash"
  TYPE ELLIPSE
  POINTS 1 1 END
  FILLED TRUE
  STYLE 10 5 5 10 END
END
```

L'affichage sera alors le suivant

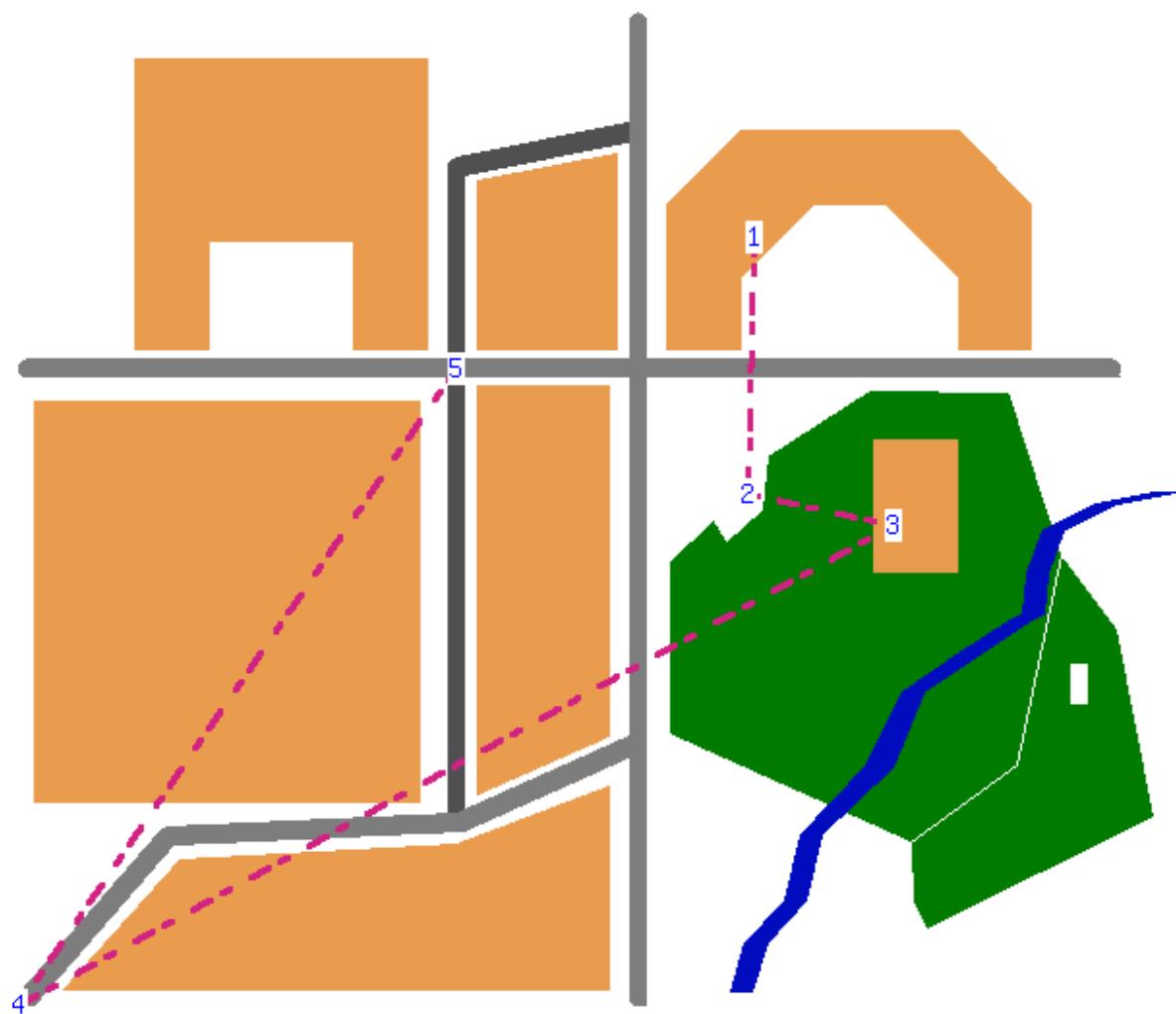


FIG. 5.18 – Déplacement successifs de la personne 7

Chapitre 6

Etudes de cas

Après avoir passé en revue quelques unes des fonctionnalités de PostGIS, nous allons ici dans ce chapitre passer en revue quelques études de cas avec des données réelles.

6.1 Cas pratique I : Etude détaillée, manipulation de données, cas pratique avec MapServer

Nous allons exposer ici des exemples de requêtes sur des données SIG réelles. Les situations citées ici sont tirées de notre travail quotidien en tant qu'utilisateur de PostGIS et de MapServer.

6.1.1 Importation des communes du Languedoc-Roussillon

Ici les communes de cette région sont "fournies" au format ESRI Shapefiles dans le fichier `communes_lr.shp`. Ces données sont déjà identifiées dans le système de projection français Lambert II Carto Etendu. Sachant que dans ce cas précis, l'identifiant de projection (`srid`) vaut 27582, pour les importer dans notre base madatabase, il nous suffira de faire

```
shp2pgsql -s 27582 -DI communes_lr.shp communes_lr | psql madatabase
```

Reportez-vous à la documentation de `shp2pgsql` pour les options ici utilisées. Nous noterons au passage que la région du Languedoc-Roussillon est composée de 1545 communes soit 1545 enregistrements dans notre table.

6.1.2 Afficher les informations relatives au Lambert II Etendu depuis la table spatial_ref_sys - get_proj4_fr

Pour avoir le maximum d'information sur la structure de cette table, je ne saurais vous conseiller la documentation en ligne sur le site de PostGIS. Comme dit dans ce document, c'est la table `spatial_ref_sys` de PostGIS qui contient les informations relatives aux divers systèmes de projection - table requise selon les spécifications de l'O.G.C (Open GIS Consortium). Exécutez la requête suivante par exemple pour voir les divers types de projections relatives à la France.

```
select srid, auth_name, auth_srid, srtext from spatial_ref_sys  
where srtext like '%France%'
```

La ligne qui nous intéresse est celle pour laquelle `srid` vaut 27582. Son champs `srtext` vaut

```
PROJCS["NTF (Paris) / France II",GEOGCS["NTF (Paris)",astng",600000](...)AUTHORITY ["EPSG ←  
","27582"]]
```

Si l'on veut des détails supplémentaires, on pourra aussi faire

```
SELECT get_proj4_from_srid(27582);
```

qui renverra

```
get_proj4_from_srid ↴  
-----  
+proj=lcc +lat_1=46.8 +lat_0=46.8 +lon_0=0 +k_0=0.99987742 +x_0=600000 +y_0=2200000 +a ↴  
=6378249.2 +b=6356515 +towgs84=-168,-60,320,0,0,0 +pm=paris +units=m +no_defs  
(1 ligne)
```

6.1.3 Question : Comment faire pour effacer la colonne géométrique sans effacer les données attributaires ? - DropGeometryColumn() -

Il faut utiliser la fonction `DropGeometryColumn()`. Cette fonction est une fonction surchargée avec PostgreSQL. On pourra par exemple utiliser la requête suivante

```
SELECT DropGeometryColumn('communes_lr','the_geom');
```

qui renverra

```
dropgeometrycolumn  
-----  
public.communes_lr.the_geom effectively removed.  
(1 ligne )
```

Cette fonction nettoie effectivement la table `geometry_columns`, en effaçant l'enregistrement concernant la table `communes_lr` ici ainsi que la colonne géométrique de cette dernière. Les données attributaires (= non géométriques) sont maintenues.

6.1.4 Question : Comment faire pour effacer une table géométrique, avant une réimportation si les métadonnées à utiliser vont changer ?

La façon la plus simple de faire serait de faire

```
DROP TABLE ma_table;
```

Mais cela ne permet pas d'effacer les métadonnées concernant `ma_table` dans `geometry_columns`. Imaginez la situation où vous voudriez réimporter les données et que vous ayant besoin de spécifier un SRID différent ou un non de colonne différent pour la colonne géométrique ! Sinon vous risquez de vous retrouver avec le

Pour une telle situation, le mieux est de faire

```
SELECT DropGeometryTable('matable');
```

On fera appel à cette fonction dans le cas où votre table se trouve dans le schéma courant de votre session cliente au serveur. En effet, cette fonction est une fonction surchargée, pour laquelle vous avez la fonction suivante aussi

```
SELECT DropGeometryTable('monschema','matable');
```

Si votre table se trouve dans un schéma différent du schéma courant.

6.1.5 Question : Comment faire si on a oublié de préciser l'identifiant de système de projection ? - `UpdateGeometrySrid()` -

Dans le cas où les données ont déjà été importées et que lors de l'utilisation de shp2pgsql on a oublié d'utiliser l'option -s pour préciser le système de projection, PostGIS considère qu'il s'agit de données plannes du plan orthonormal. Il précise alors par défaut srid=1. Ce qui peut-être contraignant par exemple si par la suite, nous soyons amenés à utiliser les fonctions Distance() ou Area2f() qui selon le système de projection peuvent donner des résultats faussées.

Pour préciser le système de projection, deux solutions possibles :

solution 1 : recharger à nouveau les données en utilisant shp2pgsql avec les options respectives -s pour l'identifiant de projection et l'option -d qui permet d'effacer la table avant de recharger le fichier .shp d'origine. La commande sera donc

```
shp2pgsql -s 27582 -dDI communes_lr.shp communes_lr | psql madatabase
```

solution 2 : garder les données déjà importées et modifier le srid à la volée. Pour cela, il faut avoir recours à la fonction `UpdateGeometrySrid` dont la synthaxe dans notre cas sera

```
SELECT UpdateGeometrySrid('communes_lr','the_geom',27582);
```

Dans les deux cas, les données sont mises à jours ainsi que les méta-données les concernant dans la table `geometry_columns`.



AVERTISSEMENT

La fonction `UpdateGeometrySrid()` ne fait que mettre à jour comme précisé les méta-données des données spatiales MAIS EN AUCUN CAS, ELLE N'EFFECTUE AUCUN TRAVAIL DE REPROJECTION!!!

6.1.6 Question : Si on s'est trompé dans le système de projection, comment faire pour reprojeter dans un autre système de manière définite ? - `Transform()` -

Le mieux ici est d'utiliser la fonction `Transform()`. Supposons par exemple que nous avons importé nos données en Lambert III Carto (27583) au lieu de Lambert II Etendu (27582)

```
BEGIN;
-- 
-- On met à jour l'identifiant du système de projection.
-- On passe donc du 27583 au 27582
-- 
SELECT UpdateGeometrySrid('communes_lr','the_geom',27582);
-- 
-- On met à jour la table
-- 
UPDATE communes_lr SET the_geom = foo.transform FROM (SELECT gid,Transform(the_geom,27582) ←
    from communes_lr ORDER BY gid ASC ) AS foo WHERE communes_lr.gid = foo.gid;
-- 
-- On se crée une vue de manière à ne pas se trainer
-- un ORDER BY avec nous sur la colonne "nom"
CREATE VIEW liste_communnes_lr AS ( SELECT * from communes_lr ORDER BY NOM ASC );
-- 
-- Ok ... La transaction a réussie? On la clotûre.
-- 
END ;
```

Je me suis ici permis de faire une simple vue sur la table `communes_lr`, de manière à ne pas me trainer un ORDER BY sur la colonne nom. Je rappelle au passage qu'un UPDATE sur une table ne fait référence à aucune notion d'ordre puisqu'en SQL, une table (relation) n'est pas un ensemble ordonné. D'où la raison personnelle ici, pourquoi j'utilise la vue `liste_communnes` pour ne pas me trainer un ORDER BY si je veux la liste des noms de communes dans l'ordre croissant.

6.1.7 Crédation d'index spatiaux Gist, Vacuum de la base

La table que nous avons créée contient pas moins de 1545 enregistrements. Normalement la création d'index spatiaux a lieu en ayant recours par exemple

```
CREATE INDEX [index_spatial_table] ON [table]
USING gist([colonne_géométrique] gist_geometry_ops);
```

Or l'option -I de shp2pgsql permet de créer automatiquement un index spatial - suffixé _the_geom_gist - en pour chacun de ces enregistrements et évite d'avoir à se soucier de leur création par l'emploi de cette requête. Par exemple ici, cette option fait alors directement appel ici pour la table communes_lr à la requête d'indexation spatiale suivante

```
CREATE INDEX communes_lr_the_geom_gist ON communes_lr
USING gist(the_geom gist_geometry_ops);
```

La création des index spatiaux peut s'avérer "gourmand". Pour que le planificateur de requêtes de PostgreSQL dispose des informations statistiques nécessaires et adéquates pour savoir quand avoir recours aux index spatiaux selon la fonction spatial demandée, il est nécessaire parfois de faire un VACUUM ANALYZE sur la base en cours

Nous reviendrons sur cet aspect plus tard. Pour résumer, nous dirons juste que à la suite d'une ou plusieurs importation(s) de données importante pour lesquelles il y a eu besoin de création d'index spatiaux Gist, il est recommandé d'utiliser la requête VACUUM ANALYZE

```
VACUUM ANALYZE
```

NOTE

Pour PostgreSQL 8.x.x, cette commande permet de collecter les informations statistiques de l'emploi des index spatiaux

6.1.8 Question : Qu'elle est l'étendue géographique/emprise de la table communes_lr ? - Extent() -

Par définition, je rappelle que l'étendue géographique d'un objet spatial dans une base de données est la plus petite fenêtre (=rectangle) qui le contienne. Ce dernier est défini par le quadruplet (Xmin,Ymin,Xmax,Ymax) . (Xmin,Ymin) (respectivement (Xmax,Ymax)) est le sommet inférieur gauche (respectivement le sommet supérieur droit) du rectangle dans tout système de projection orienté de gauche à droite horizontalement et de bas en haut verticalement.

Pour connaître ce quadruplet, la requête suivante

```
SELECT Xmin(foo.extent),
       Ymin(foo.extent),
       Xmax(foo.extent),
       Ymax(foo.extent)
  FROM
    (SELECT Extent(the_geom) FROM communes_lr) AS foo
```

nous renvoie comme résultat

xmin	ymin	xmax	ymax
547293	1703209.875	801423	1997767.125

(1 row)

Ce dernier implique donc que tous les objets de la colonne géométrique the_geom sont contenus dans le quadruplet ci-dessus obtenu.

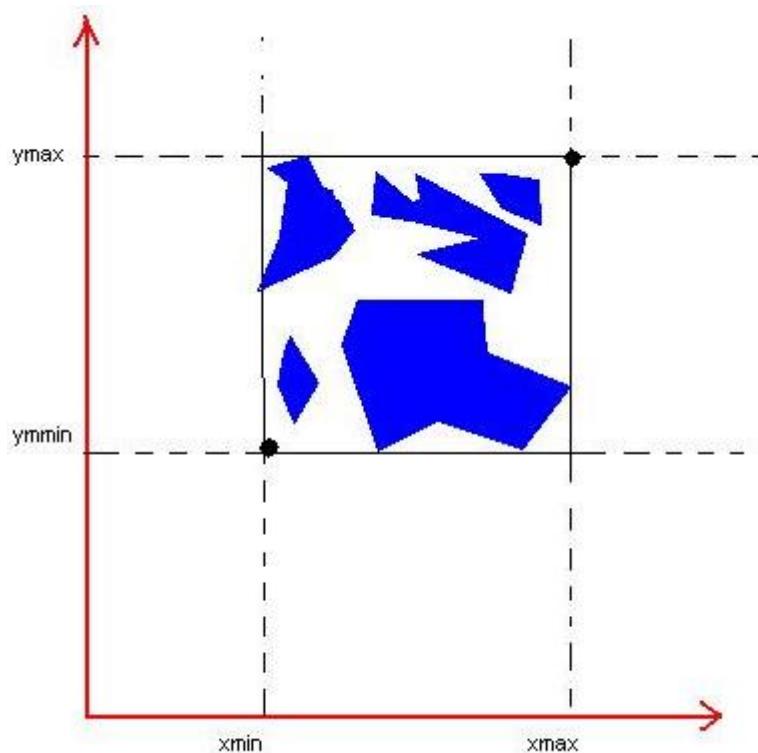


FIG. 6.1 – Extent avec PostGIS. Ce qu'on appelle aussi emprise en S.IG.

NOTE

Connaître l'étendue géographique peut s'avérer utile, notamment avec Mapserver dont les fichiers mapfiles possèdent un paramètre EXTENT qui correspond à l'étendue géographique. Ici il s'agira de fournir ce quadruplet. Ce que j'appelle **étendue géographique** est aussi ce qu'on appelle **emprise** en SIG classique

6.1.9 Visualisation des données avec MapServer

Cette section suppose que l'emploi de MapServer vous soit familier. Je ne passerais pas ici en revue les divers fonctionnements de MapServer. Je profite de l'occasion juste pour détailler de quoi est composé la plus petite couche - LAYER au sens de MapServer - pour PostGIS. Je pars ici du principe que vous avez Apache, Php et PhpMapScript d'installés sur votre machine.

NOTE

Pour disposer de tous ces outils, le mieux est de recourir au paquet MS4W (MapServer For Windows) disponible sur le site <http://www.maptools.org>

Pour construire mon image sur les communes du Languedoc-Roussillon, le premier paramètre à fournir est bien l'EXTENT. Or selon une des dimensions de l'image que je me fixe (longueur ou hauteur), il me faut aussi savoir qu'elle sera la valeur de l'autre dimensions.

6.1.9.1 Pour une image de longueur 500 pixels, quelle doit être la hauteur de l'image en fonction de l'étendue géographique ?

Il suffit pour cela d'effectuer une simple règle de trois que me donne la requête suivante inspirée de la requête de la section précédente :

```
SELECT
    500*(abs(Ymax(foo.extent)-Ymin(foo.extent))/abs(Xmax(foo.extent)-Xmin(foo.extent)))
    AS hauteur
FROM (SELECT Extent(the_geom) FROM communes_lr) AS foo
```

qui me renvoie

```
    hauteur
-----
579.540491087239
(1 ligne)
```

6.1.9.2 Mapfile et Script PHP

Pour afficher ma table communes_lr, ayant l'étendue géographique et la hauteur de l'image, je peux par exemple avoir comme mapfile appelée communes_lr.map

```
MAP
EXTENT 547293 1703209.875 801423 1997767.125
IMAGECOLOR 255 255 255
IMAGETYPE png
SIZE 500 579.540491087239

WEB
IMAGEPATH "c:/wamp/www/tutorial/tmp/"
IMAGEURL "/tutorial/tmp/"
END
#
# Couche des communes
#
LAYER
NAME "communes_lr"
CONNECTION "user=david dbname=madatabase host=localhost"
CONNECTIONTYPE POSTGIS
DATA "the_geom from communes_lr"
STATUS DEFAULT
TYPE POLYGON
CLASS
STYLE
COLOR 255 255 255
OUTLINECOLOR 0 0 0
END
END
END

END
```

avec comme paramètres au niveau générale

EXTENT qui prend comme paramètres le quadruplet Xmin Ymin Xmax Ymax ;

IMAGECOLOR prend comme valeur un RGB (Red Green Blue) pour définir le fond de couleur. Ici 255 255 255 pour avoir du blanc ;

IMAGETYPE png pour préciser que l'image produite sera au format png. cela suppose que MapServer est été compilé avec l'option --with-png pr défaut comme OUTPUTFORMAT ;

SIZE qui prend comme paramètres Longueur Hauteur

au niveau du WEB :

IMAGEPATH pour préciser le chemin d'accès absolue sur le disque où seront générées les images produites ;

IMAGEURL url relative à la valeur donnée par IMAGEPATH

au niveau de la couche - LAYER - pour la table communes_lr

CONNECTION suivi des paramètres de connection à PostgreSQL ;

CONNECTIONTYPE POSTGIS pour préciser qu'il s'agit d'une connection à une base de données PostGIS ;

DATA dont la pseudo-valeur doit être "colonne_geometrique from table". On met donc ici "the_geom from communes_lr" sans le mot-clé SELECT.

TYPE qui précise la nature géométrique des objets à afficher. TYPE peut prendre la valeur POINT, LINE ou POLYGON.

le reste au niveau du CLASS/STYLE concerne l'habillage pour l'affichage des objets. Ici un contour noir (OUTLINECOLOR 0 0 0 et un intérieur blanc (COLOR 255 255 255)

Au niveau de php grâce à - l'extension phpmapscript si vous l'avez installé - pour générer l'image on peut utiliser le script suivant

```
<html>
<body>
<?php
$sw_MapFile = "./mapfiles/communes_lr.map";
$map = ms_newMapObj( $sw_MapFile );
$image = $map->draw();
$image_url = $image->saveWebImage(MS_PNG,1,1,0);

echo "<IMG
BORDER=0
SRC='".$image_url.'
width='".$map->width."' height='".$map->height."' />
<BR>";
?>
</body>
</html>
```

On obtiendra ainsi le visuel suivant



FIG. 6.2 – Affichage des communes du Languedoc-Roussillon (MapServer+PhpMapScript)

6.1.10 Question : Quelles sont les communes avoisinantes de Montpellier ?, Utilité des index spatiaux - Distance(), && -

Dans le cas de certaines fonctions de PostGIS, parmi lesquelles nous pouvons citer Distance() ; WithIn(), Intersects() [...], il est parfois nécessaire selon la résultat escompté de les coupler à **&&** := opérateur de chevauchement des étendues géographiques.

En effet, celui-ci sait tirer profit des index spatiaux - en interne de son implémentation, nous dirons qu'il les exploite-. Comme son nom l'indique, il teste si les étendues géographiques de deux objets géométriques A et B se chevauchent.

A titre d'exemple, pour avoir la liste des communes qui avoisinent Montpellier, il s'agit simplement des communes dont la distance à Montpellier est nulle, ce qui sous-entend donc que leurs étendues géographiques se chevauchent- nous aurons recours à l'utilisation de l'opérateur **&&** pour accélérer le temps nécessaire à la requête. D'où la requête suivante

```
SELECT b.nom FROM communes_lr a, communes_lr b
WHERE
    a.nom LIKE 'MONTPELLIER'
AND
    b.nom NOT LIKE 'MONTPELLIER'
AND
    Distance(a.the_geom,b.the_geom)=0
AND
    a.the_geom && b.the_geom
ORDER BY b.nom
```

qui nous renvoie

```
nom
-----
CASTELNAU-LE-LEZ
CLAPIERS
GRABELS
JUVIGNAC
LATTES
LAVERUNE
MAUGUIO
MONTFERRIER-SUR-LEZ
SAINT-AUNES
SAINT-CLEMENT-DE-RIVIERE
SAINT-JEAN-DE-VEDAS
(11 rows)
```

Sans l'utilisation ici de **a.the_geom && b.the_geom**, le temps demandé par la requête s'avère long - j'en témoigne pour 1544 enregistrements à tester ! -.

NOTE

Il est également possible d'utiliser une fonction plus performante que la fonction Distance avec la condition **Distance(A,B)=0** . On aurait pu avoir recours à la fonction **Touches(A,B)** qui permet de savoir si A et B se touchent puisqu'ici nous savons que les communes ne se "touchent" que par leur frontière

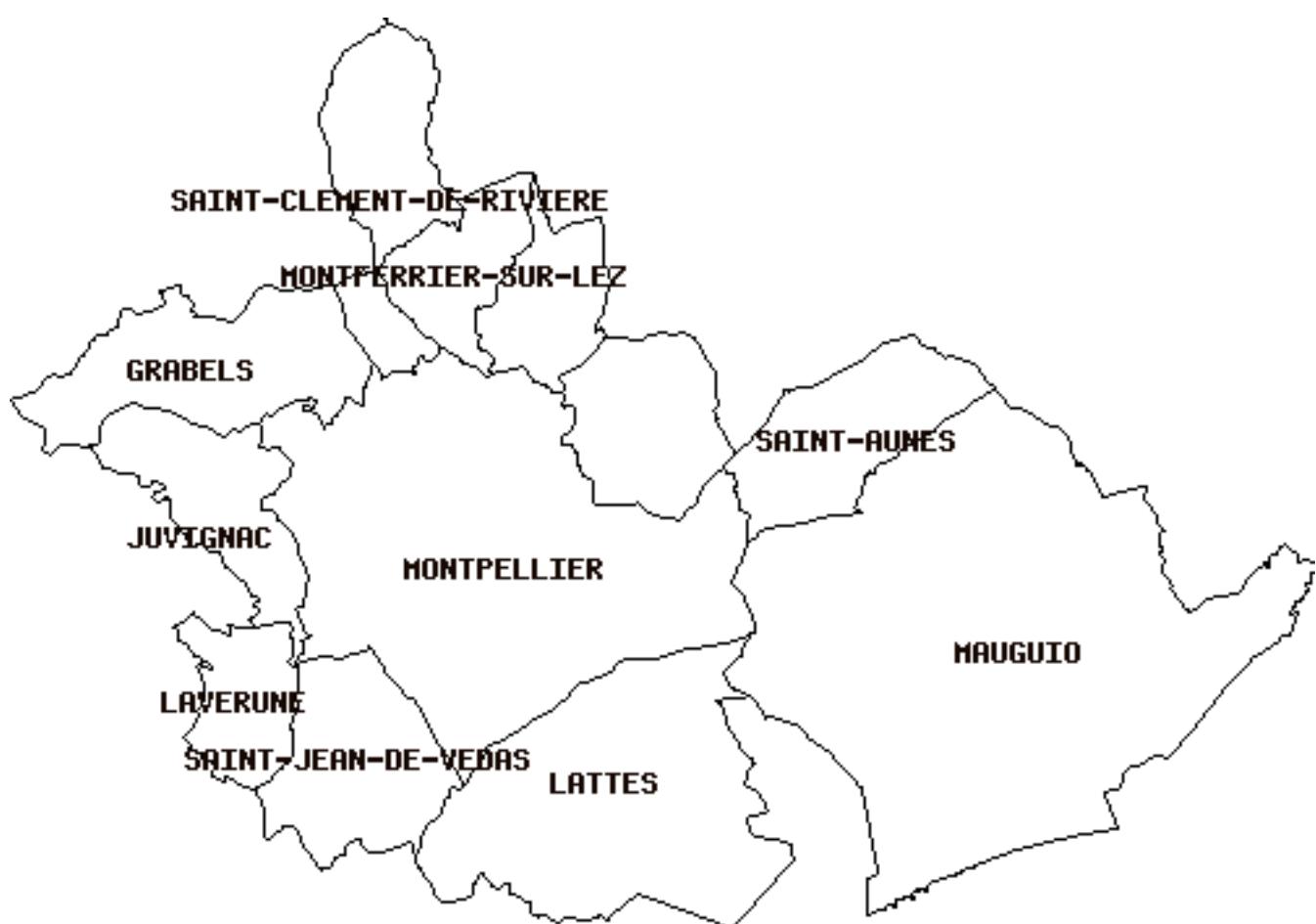


FIG. 6.3 – Les communes avoisinantes de Montpellier

6.1.11 Utilité des index spatiaux - temps demandé pour exécuter la requête

Pour mettre en évidence le temps mis par PostgreSQL pour exécuter la requête de la section précédente avec et sans la condition **a.the_geom && b.the_geom**, le mieux serait donc de reprendre la requête en la combinant aux mot-clés **EXPLAIN ANALYZE**

NOTE

Je n'exposerais pas ici les explications faisant appel au planificateur de tâches et à l'optimisateur de requêtes de PostgreSQL. Le mieux à mon sens est de se référer à la documentation officielle de PostgreSQL.

6.1.11.1 Avec la condition pour &&

La requête suivante

```
EXPLAIN ANALYZE
SELECT b.nom  FROM communes_lr a, communes_lr b
WHERE
    a.nom LIKE 'MONTPELLIER'
AND
    b.nom NOT LIKE 'MONTPELLIER'
AND
    Distance(a.the_geom,b.the_geom)=0
AND
```

```
a.the_geom && b.the_geom  
ORDER BY b.nom
```

nous renvoit comme réponse

```
-----  
QUERY PLAN  
-----  
  
Sort  (cost=131.34..131.35 rows=1 width=32) (actual time=932.000..932.000 rows=11 loops=1)  
  Sort Key: b.nom  
    -> Nested Loop  (cost=0.00..131.33 rows=1 width=32) (actual time=230.000..921.000 rows =<  
      =11 loops=1)  
      Join Filter: (distance("outer".the_geom, "inner".the_geom) = 0::double precision)  
      -> Seq Scan on communes_lr a  (cost=0.00..83.31 rows=8 width=32) (actual time =<  
        =10.000..10.000 rows=1 loops=1)  
          Filter: ((nom)::text ~~ 'MONTPELLIER'::text)  
      -> Index Scan using communes_lr_the_geom_gist on communes_lr b  (cost=0.00..5.98 =<  
        rows=1 width=64) (actual time=10.000..20.000 rows=17 loops=1)  
          Index Cond: ("outer".the_geom && b.the_geom)  
          Filter: ((nom)::text !~~ 'MONTPELLIER'::text)  
Total runtime: 932.000 ms  
(10 lignes)
```

soit un temps de 0.9 secondes (cf Total runtime : 932.000 ms). On voit bien d'après les résultats que l'index spatiaux est utilisé (cf "Index Scan using communes_lr_the_geom_gist on communes_lr", index créé automatiquement lors de l'importation des données par l'option -I de shp2pgsql.)

6.1.11.2 Sans la condition pour &&

La même requête mais sans la contrainte

```
EXPLAIN ANALYZE  
SELECT b.nom  FROM communes_lr a, communes_lr b  
WHERE  
  a.nom LIKE 'MONTPELLIER'  
AND  
  b.nom NOT LIKE 'MONTPELLIER'  
AND  
  Distance(a.the_geom,b.the_geom)=0  
ORDER BY b.nom
```

renvoie comme résultat

```
-----  
QUERY PLAN  
-----  
  
Sort  (cost=476.08..476.23 rows=62 width=32) (actual time=152159.000..152159.000 rows=11 =<  
  loops=1)  
  Sort Key: b.nom  
    -> Nested Loop  (cost=83.32..474.23 rows=62 width=32) (actual time =<  
      =74577.000..152159.000 rows=11 loops=1)  
      Join Filter: (distance("inner".the_geom, "outer".the_geom) = 0::double precision)  
      -> Seq Scan on communes_lr b  (cost=0.00..83.31 rows=1538 width=64) (actual time =<  
        =0.000..50.000 rows=1544 loops=1)  
          Filter: ((nom)::text !~~ 'MONTPELLIER'::text)  
      -> Materialize  (cost=83.32..83.40 rows=8 width=32) (actual time=0.006..0.013 =<  
        rows=1 loops=1544)  
        -> Seq Scan on communes_lr a  (cost=0.00..83.31 rows=8 width=32) (actual =<  
          time=10.000..20.000 rows=1 loops=1)  
            Filter: ((nom)::text ~~ 'MONTPELLIER'::text)  
Total runtime: 152189.000 ms
```

(10 lignes)

ce qui s'apparente à un temps d'exécution d'environ 2 minutes 30 (cf. Total runtime : 152189.000 ms). Ce qui est énorme !!!
Ici, comme attendu il n'est pas fait usage des index spatiaux - par rapport à l'opérateur de chevauchement && -.

Ici par rapport au résultat avec l'opérateur && de la section précédente, la ligne pour la table 'communes_lr b'

```
-> Seq Scan on communes_lr b [... ] (actual time=0.000..50.000 rows=1544 loops=1) [...]
```

implique que 1544 enregistrements (cf rows=1544) de la table communes_lr ont été testés. Or si on enlève l'enregistrement qui correspond à celui de MONTPELLIER ('condition de la requête a.nom LIKE 'MONTPELLIER'), cela fait 1545 enregistrements. Soit le total d'enregistrement contenu dans la table communes_lr. Ce qui suppose donc que tous les enregistrements de la table ont été testés ! Ce qui est justement l'opposé du bénéfice des index en bases de données.

En effet, avec l'opérateur && le coût de la table 'communes_lr b' s'élève à 17 enregistrements qui sont testés !

```
-> Index Scan using communes_lr_the_geom_gist on communes_lr b [...]  
(actual time=10.000..20.000 rows=17 loops=1)
```

Résultat qui exploite les index spatiaux !

6.1.12 Crée une table communes_avoisinantes correspondant aux communes avoisinantes de MONTPELLIER, extraite et conforme à la structure de la table communes_lr, exploitable par MapServer.

On pourrait se contenter d'entrée de dire que la réponse directe serait de faire à la volée

```
CREATE TABLE communes_avoisinantes AS  
(  
    SELECT b.* FROM communes_lr a, communes_lr b  
    WHERE  
        a.nom LIKE 'MONTPELLIER'  
    AND  
        Distance(a.the_geom,b.the_geom)=0  
    AND  
        a.the_geom && b.the_geom  
ORDER BY b.nom  
)
```

Le premier souci qui se pose est que l'on n'a pas ici les métadonnées concernant la nouvelle table référencées dans la table geometry_columns. On peut s'arrêter là si on n'a pas besoin d'utiliser cette table pour un affichage ultérieur avec MapServer.

Or dans le cas contraire d'un affichage avec MapServer, il est nécessaire que les métadonnées concernant la nouvelle table soient fournies dans la tables geometry_columns. En effet, quand il s'agit de données PostGIS à afficher, MapServer se sert de la fonction find_srid() qui elle s'appuie sur la table geometry_columns pour connaître l'identifiant de projection à utiliser.

Le mieux est alors d'avoir recours à quelques requêtes basique.s. On pourra essayer les requêtes successives décrites ici pour répondre à tous ces impératifs :

```
/*  
 * On commence par effacer la table communes_avoisinantes  
 * si elle existe déjà.  
 * Ceci au cas où nous serions amenés à recharger ce fichier.  
 * On peut aussi utiliser la commande: DROP TABLE communes_avoisinantes  
 */  
SELECT drop_table_if_exists('communes_avoisinantes',false);  
/*  
 * On crée une table vide communes_avoisinantes  
 * pour cela, il suffit de demander de retourner la table communes_lr avec 0 lignes.  
 * On obtient ainsi la même structure que la table communes_lr  
 */
```

```
CREATE TABLE communes_avoisinantes AS
    SELECT * FROM communes_lr LIMIT 0;
/*
  On efface la colonne géométrique 'the_geom' dans la table communes_avoisinantes
  car pour l'instant celle-ci n'est pas référencée comme il faut
  dans la table geometry_columns
*/
ALTER TABLE communes_avoisinantes DROP COLUMN the_geom;
/*
  On ajoute la colonne 'the_geom' proprement en utilisant AddGeometryColumn()
  Ici, on joue le jeu que l'on ne connaît de la table
  communes_lr que son nom. On ignore donc le type
  géométrique, son srid...Toutes ces informations sont stockées
  dans la table geometry_columns.
*/
SELECT AddGeometryColumn('communes_avoisinantes'::varchar,
                           foo.f_geometry_column::varchar,
                           foo.srid::integer,
                           foo.type::varchar,
                           foo.coord_dimension::integer
)
FROM (
    SELECT * FROM geometry_columns WHERE
        f_table_name LIKE 'communes_lr'
) AS foo;
/*
  On insère maintenant les données attendues dans la table
*/
INSERT INTO communes_avoisinantes
(
    SELECT b.* FROM communes_lr a, communes_lr b
    WHERE
        a.nom LIKE 'MONTPELLIER'
    AND
        Distance(a.the_geom,b.the_geom)=0
    AND
        a.the_geom && b.the_geom
ORDER BY b.nom
);
```

NOTE

A partir de l'étude réalisée pour la table communes_lr, il est alors aisément de pouvoir visualiser le contenu de cette table dans MapServer.

6.1.13 Requête 1 : Qu'elle est l'intersection entre MONTPELLIER et les communes de LATTES et de JUVIGNAC ?- Intersection()- Que vaut cette géométrie en SVG ? - AsSVG(),

Nous allons utiliser la table communes_avoisinantes. Toutes ces résultats peuvent être obtenus grâce à la requête suivante :

```
SELECT foo.nom,
       AsText(Intersection(foo.the_geom,foo.the_geom)),
       AsSVG(Intersection(foo.the_geom,foo.the_geom))
FROM (
    SELECT the_geom FROM communes_avoisinantes
    WHERE nom LIKE 'MONTPELLIER'
) AS foo
,
```

```
(  
    SELECT nom, the_geom FROM communes_avoisinantes  
    WHERE nom LIKE 'JUVIGNAC' OR nom LIKE 'LATTES'  
) AS fooo ;
```

ou sinon la requête suivante possible

```
SELECT b.nom,  
       AsText(Intersection(a.the_geom,b.the_geom)),  
       AssVG(Intersection(a.the_geom,b.the_geom))  
FROM communes_avoisinantes a,communes_avoisinantes b  
WHERE a.nom='MONTPELLIER'  
AND (b.nom='LATTES' OR b.nom='JUVIGNAC');
```

Je n'afficherais pas ici les résultats obtenus car ces derniers prennent trop de place en affichage. Je me contenterais de les résumés ainsi

```
nom      |          intersection   ↪  
          |          |   ↪  
          |          assvg  
-----+-----  
JUVIGNAC | MULTILINESTRING ((719652.990195505 1844379.97900231,[...]718841.967178608 ↪  
1849682.046271555)) | M 719652.9901955052 -1844379.9790023109 [...] 718841.96717860829 ↪  
-1849682.0462715544  
LATTES   | MULTILINESTRING ((729577.052318637 1845119.97019685,[...],723311.014479515 ↪  
1841729.00265142)) | M 729577.05231863656 -1845119.9701968494 [...] 723311.01447951456 ↪  
-1841729.0026514169  
(2 lignes)
```

Au niveau de l'affichage on obtient

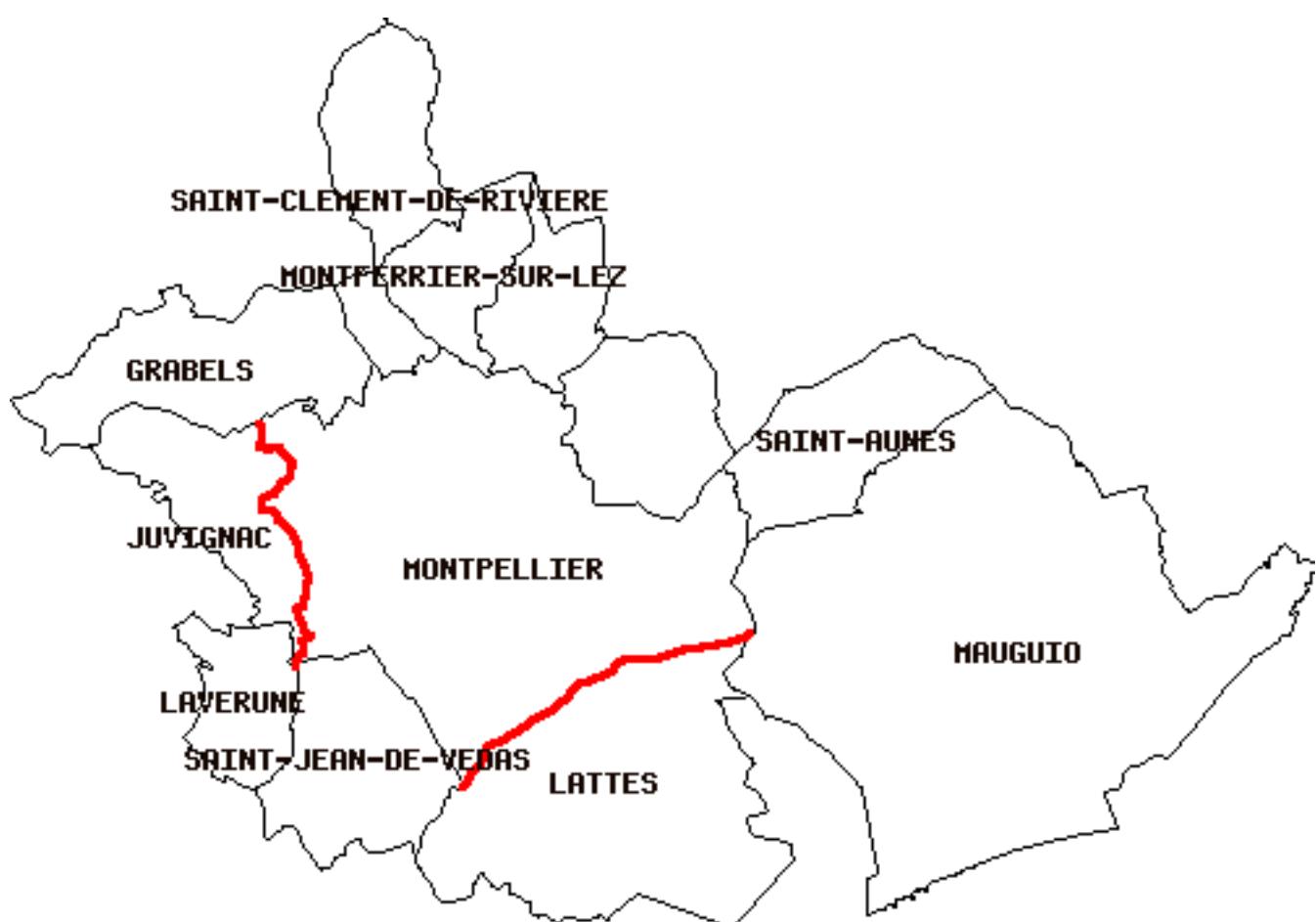


FIG. 6.4 – Intersection entre MONTPELLIER et les communes de LATTES et de JUVIGNAC

6.1.14 Requête 2 : Qu'elle est la commune ayant la plus petite aire ?

Il suffit par exemple d'utiliser la fonction Min() et Area2d() pour avoir comme requête :

```
SELECT nom FROM communes_avoisinantes
  WHERE Area2d(the_geom) = (SELECT Min(Area2d(the_geom)) FROM communes_avoisinantes)
```

nous renvoyant comme réponse

```
nom
-----
LAVERUNE
(1 row)
```

Le visuel sera alors le suivant

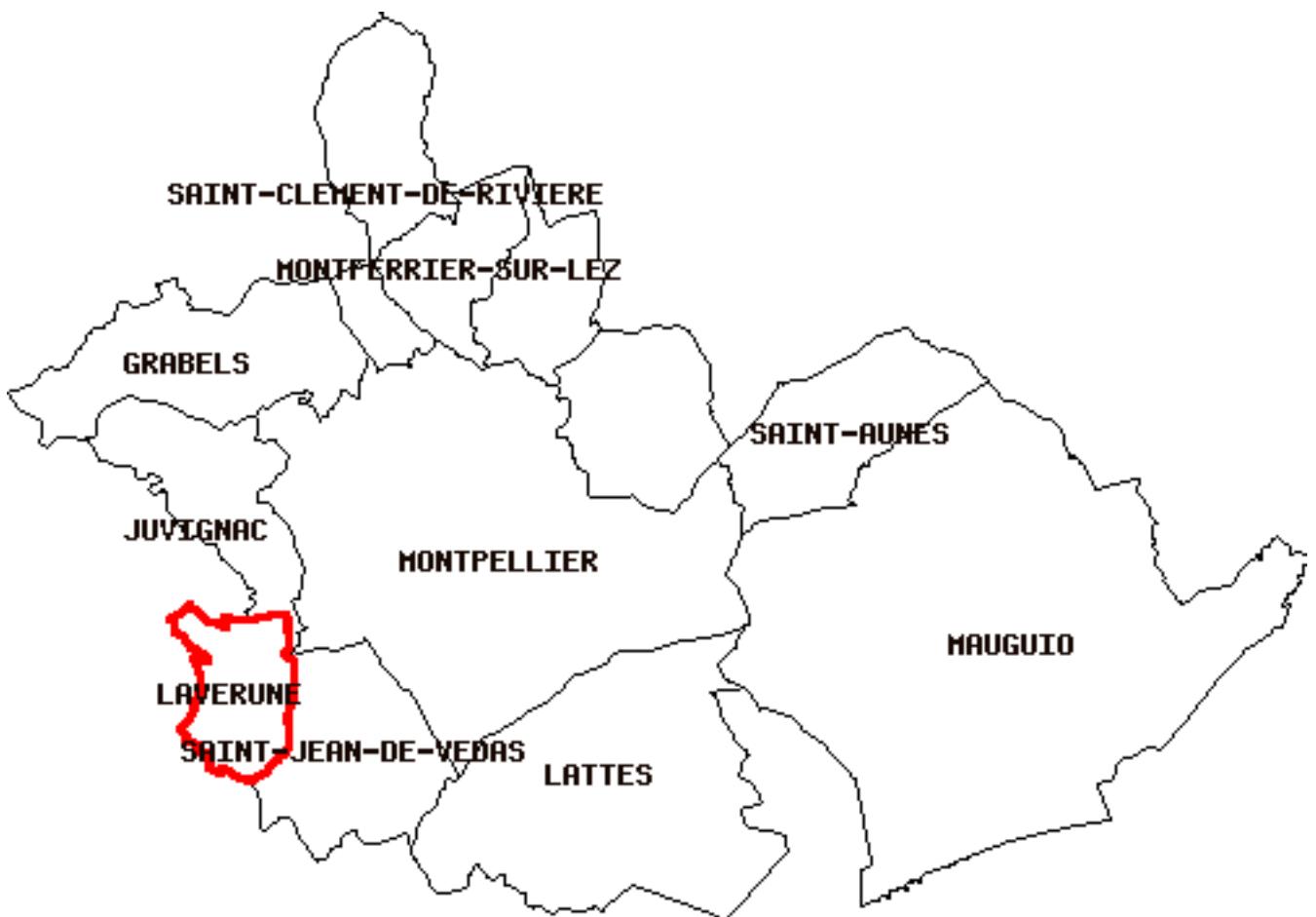


FIG. 6.5 – LATTES : la commune ayant la plus petite aire.

6.1.15 Mapfile générale pour la table communes_avoisinantes et les deux requêtes précédentes

Je fournis ci après la mapfile qui permet d'afficher la table communes_avoisinantes, ainsi que les deux requêtes précédemment poser. Il est à noter surtout la formulation des requêtes - paramètre DATA - pour pouvoir afficher les objets attendus.

```
MAP
  EXTENT 713413.9375 1838659.875 741999.0625 1858572.125
  FONTSET "c:\wamp\www\tutorial\etc\fonts.txt"
  SYMBOLSET "c:\wamp\www\tutorial\etc\symbols.sym"
  IMAGECOLOR 255 255 255
  IMAGETYPE png
  SIZE 500 348.297409929115
# STATUS ON

WEB
  IMAGEPATH "c:/wamp/www/tutorial/tmp/"
  IMAGEURL "/tutorial/tmp/"
END
=====
# La table communes_avoisinantes
=====
LAYER
  NAME "communes_avoisinantes"
  CONNECTION "user=david dbname=madatabase host=localhost"
  CONNECTIONTYPE POSTGIS
```

```
DATA "the_geom from communes_avoisinantes"
STATUS DEFAULT
TYPE POLYGON
LABELITEM "nom"
CLASS
LABEL
SIZE MEDIUM
TYPE BITMAP
BUFFER 0
COLOR 22 8 3
FORCE FALSE
MINDISTANCE -1
MINFEATURESIZE -1
OFFSET 0 0
PARTIALS TRUE
POSITION CC
END
STYLE
COLOR 255 255 255
OUTLINECOLOR 0 0 0
END
END
END
=====
# Requête 1: Intersection entre Montpellier et les communes de LATTES et JUVIGNAC
=====
LAYER
NAME "requete_1"
CONNECTION "user=david dbname=madatabase host=localhost"
CONNECTIONTYPE POSTGIS
DATA "intersection from
(SELECT
    intersection(foo.the_geom,fooo.the_geom),fooo.nom,fooo.gid
FROM (
    SELECT * FROM communes_avoisinantes
    WHERE nom LIKE 'MONTPELLIER'
) AS foo
,
(
    SELECT * FROM communes_avoisinantes
    WHERE nom LIKE 'JUVIGNAC' OR nom LIKE 'LATTE'
) AS fooo
) AS uo USING UNIQUE gid USING SRID=27582"
STATUS OFF
TYPE LINE
LABELITEM "nom"
CLASS
STYLE
OUTLINECOLOR 255 0 0
SIZE 3
SYMBOL "circle"
END
END
END
=====
# Requête 2: La plus petite commune autour de Montpellier
=====
LAYER
NAME "requete_2"
CONNECTION "user=david dbname=madatabase host=localhost"
CONNECTIONTYPE POSTGIS
DATA "the_geom from
```

```
(SELECT the_geom,nom,gid FROM communes_avoisinantes
WHERE Area2d(the_geom) = (SELECT Min(Area2d(the_geom)) FROM
communes_avoisinantes))AS uo USING UNIQUE gid USING SRID=27582"
STATUS DEFAULT
TYPE LINE
LABELITEM "nom"
CLASS
STYLE
    OUTLINECOLOR 255 0 0
    SIZE 3
    SYMBOL "circle"
END
END
END

END
```

6.1.16 Exercice : Obtenir une table départements_lr qui contient les contours départementaux du Languedoc-Roussillon à partir de la table communes_lr

La région du Languedoc-Roussillon est composée de 5 départements dont voici la liste

Aude 11;
Hérault 34;
Gard 30;
Lozère 48;
Pyrénées Orientales 66.

Normalement lorsqu'on dispose des limites communales d'une région on doit aussi pouvoir disposer des limites départementales de cette dernière. Nous partirons ici du principe que nous ne disposons que de la table communes_lr. C'est à partir de cette table que nous allons créer nos limites départementales. Nous allons mettre les contours départementaux dans une table **departements_lr** ayant la structure suivante

```
CREATE TABLE departements_lr(id serial,nom text,numero integer)WITH OIDS;
```

où nom désignera le nom du département et numero son numéro. Comme pour l'instant, je ne sais pas quel est la nature des objets géométrique, je fais juste

```
SELECT AddGeometryColumn('departements_lr','the_geom',27582,'GEOMETRY',2);
```

Avant de remplir ma table, je dois préciser que je dois pour la suite faire des requêtes du genre

```
INSERT INTO departements_lr ( select 1 as id , 'Herault':: text as nom ,34 as numero ,
geomunion (the_geom) as the_geom from communes_lr where insee like '34%' );
```

insertion que je vais utiliser 5 fois. Il apparaît clairement ici que j'utilise la condition "**where insee like 'constante%'"** . Or PostgreSQL pour les champs de colonne pour lesquels on exige un modèle de recherche du style "WHERE mon_champs LIKE 'chaine_de_caractère_fixe%" " sait tirer profit des index pour ce genre de modèle. Comme la locale de notre installation n'est pas en 'C', il va falloir utiliser un autre type de modèle que le B-Tree proposé par défaut par PostgreSQL Avant toute chose, je dois donc connaître le type de données du champs insee de ma table communes_lr :

```
SELECT column_name,data_type FROM information_schema.columns WHERE table_name ='communes_lr' -->
  ' AND table_schema ='public' AND column_name = 'insee' AND table_catalog = 'madatabase';
```

qui me renvoit

column_name	data_type
insee	character varying

(1 ligne)

Je crée donc l'index adéquate sur la colonne insee qui est de type varchar de ma table communes_lr en faisant

```
CREATE INDEX communes_lr_insee ON communes_lr(insee varchar_pattern_ops);
```

Vériﬁons bien que l'index sera utilisé sur la requête d'insertion donnée avant

```
EXPLAIN ANALYZE SELECT 1 AS id , 'Hérault' :: text AS nom , 34 AS numero , geomunion ( ←  
the_geom) AS the_geom from communes_lr WHERE insee LIKE '34%'
```

qui me renvoit

```
-----  
                                     QUERY PLAN  
-----  
  
Aggregate  (cost=71.00..71.01 rows=1 width=3957) (actual time=43989.893..43989.898 rows=1 ←  
loops=1)  
  -> Bitmap Heap Scan on communes_lr  (cost=4.43..70.40 rows=238 width=3957) (actual time ←  
= 0.163..5.447 rows=343 loops=1)  
        Filter: ((insee)::text ~ '34%'::text)  
        -> Bitmap Index Scan on communes_lr_insee  (cost=0.00..4.43 rows=238 width=0) (←  
actual time=0.138..0.138 rows=343 loops=1)  
              Index Cond: (((insee)::text ~>= '34'::character varying) AND ((insee)::text ←  
~<~ '35'::character varying))  
Total runtime: 44001.960 ms  
(6 lignes)
```

La portion de texte "Bitmap Index Scan on communes_lr_insee" met en évidence l'utilisation de l'index communes_lr_insee dans cette requête pour le modèle de recherche WHERE mon_champs like 'chaine_constante%';

NOTE

Ici ce qui coûte cher c'est la fonction GeomUnion() pour faire la réunion géométrique. Certes il est vrai que pour une table de 1545 lignes, cela peut paraître dérisoire que de faire un index. Mon but ici est de montrer comment créer l'index lorsque la local n'est pas en 'C' pour la CLAUSE WHERE énoncée ici. Je ne dis pas non plus qu'il faut impérativement créer un index pour tous les champs mais si la table avait contenu par exemple au moins 20000 lignes cela peut s'avérer utile.

Je remplis maintenant la table pour tous les départements

```
INSERT INTO departements_lr  
(  
select 1 as id,'Hérault' :: text as nom,34 as numero,geomunion(the_geom) as the_geom from  
communes_lr where insee like '34%'  
);  
INSERT INTO departements_lr  
(  
select 2 as id,'Gard' :: text as nom,30 as numero,geomunion(the_geom) as the_geom from  
communes_lr where insee like '30%'  
);  
INSERT INTO departements_lr  
(  
select 3 as id,'Lozère' :: text as nom,48 as numero,geomunion(the_geom) as the_geom  
from communes_lr where insee like '48%'  
);  
INSERT INTO departements_lr  
(  
select 4 as id,'Aude' :: text as nom,11 as numero,geomunion(the_geom) as the_geom from  
communes_lr where insee like '11%'  
);  
INSERT INTO departements_lr  
(  
select 5 as id,'Pyrénées Orientales' :: text as nom,66 as numero,geomunion(the_geom)
```

```
as the_geom from communes_lr where insee like '66%'  
);
```

Je ne cache pas qu'ici le peuplement de la table m'a pris 7 à 8 minutes quand même ! Compréhensible dans le sens où faire/calculer la réunion géométrique de plusieurs objets géométriques demande du temps.

Pour savoir quel est le type de données géométriques adéquates, il faut utiliser la commande

```
select nom, geometrytype(the_geom) from departements_lr
```

qui me renvoit

nom	geometrytype
Hérault	MULTIPOLYGON
Gard	POLYGON
Lozère	POLYGON
Aude	POLYGON
Pyrénées Orientales	POLYGON

(5 lignes)

Je réordonne les tables geometry_columns en mettant le type de departements_lr à MULTIPOLYGON et je convertis tous les départements en MULTIPOLYGON grâce à la fonction Multi() de PostGIS.

```
update geometry_columns set type='MULTIPOLYGON' where f_table_name='departements_lr';  
update departements_lr set the_geom=(multi(the_geom::text));
```

Il faut ensuite créer l'index spatial sur la table departements_lr et faire un vacuum analyze sur la base.

```
create index departements_lr_the_geom_gist on departements_lr using gist(the_geom ↔  
gist_geometry_ops);  
vacuum analyze;
```

Au niveau de la mapfile, il faudrait ajouter le layer suivant

```
#=====  
# Couche des départements  
#=-----  
LAYER  
  NAME "departements_lr"  
  CONNECTION "user=david dbname=madatabase host=localhost"  
  CONNECTIONTYPE POSTGIS  
  DATA "the_geom from departements_lr"  
  STATUS DEFAULT  
  TYPE POLYGON  
  LABELITEM "nom"  
  CLASS  
    LABEL  
    SIZE MEDIUM  
    TYPE BITMAP  
    BUFFER 0  
    COLOR 22 8 3  
    FORCE FALSE  
    MINDISTANCE -1  
    MINFEATURESIZE -1  
    OFFSET 0 0  
    PARTIALS TRUE  
    POSITION CC  
  END  
  STYLE  
    OUTLINECOLOR 255 0 0  
    SIZE 3
```

```
    SYMBOL "circle"
  END
END
END
```

À niveau de l'affichage, on obtient alors

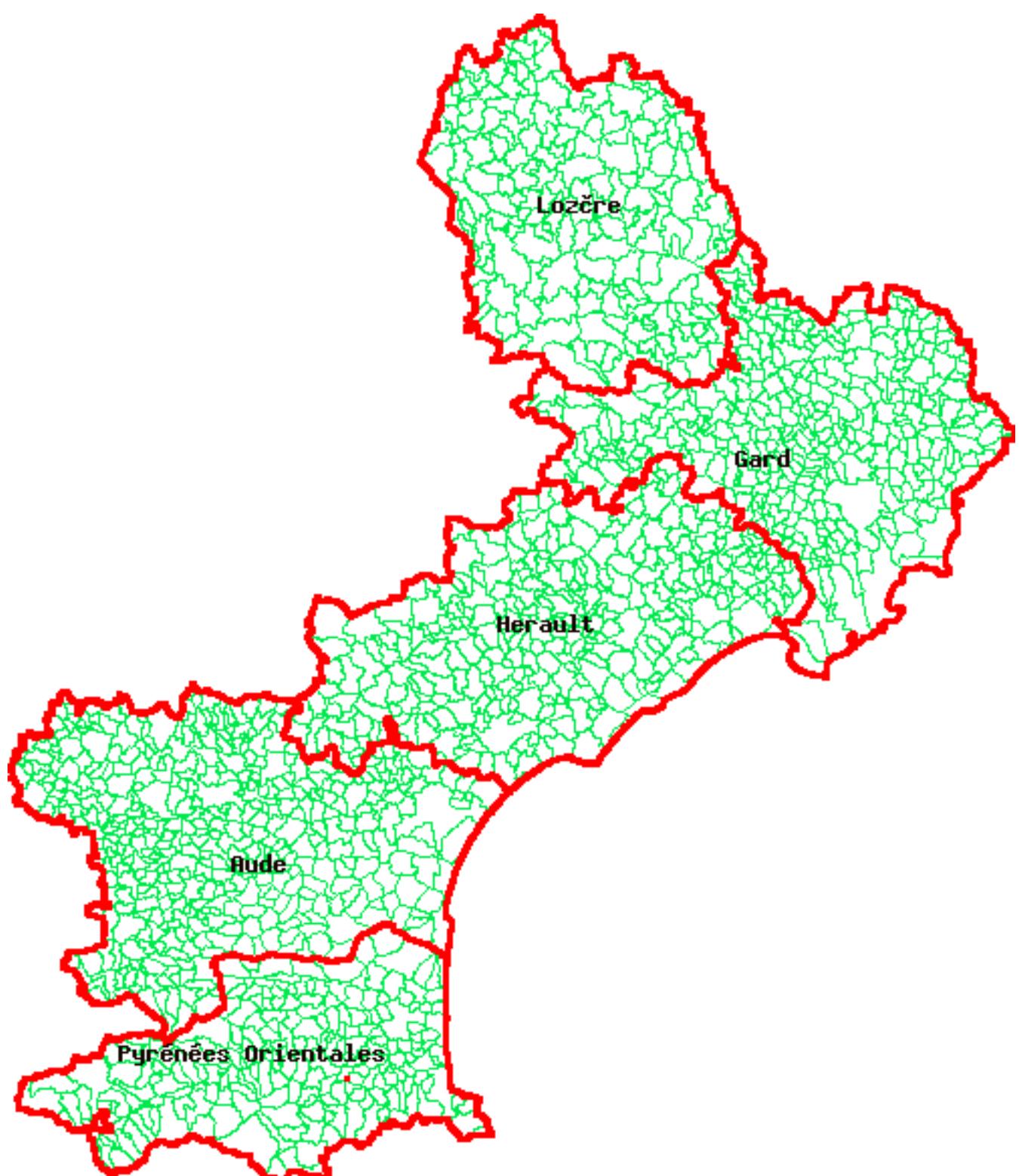


FIG. 6.6 – Affichage des départements du Languedoc-Roussillon

6.1.17 Exercice : Trouver les communes du Gard et de l'Aude qui sont limitrophes à l'Hérault et les afficher grâce à MapServer.

Comme je sais ici que les communes des départements du Gard et de l'Aude commencent respectivement par 30 et 11, je vais pour cela utiliser la fonction Touches de PostGIS.

Au niveau de la mapfile, je ferais simplement

```
#=====
# requete
#=====
LAYER
  NAME "requete"
  CONNECTION "user=david dbname=madatabase host=localhost"
  CONNECTIONTYPE POSTGIS
  DATA "the_geom from (
    select c.* from departements_lr d,communes_lr c
    where touches(d.the_geom,c.the_geom)
    and d.the_geom && c.the_geom
    and d.nom='Hérault'
    and (c.insee like '11%' or c.insee like '30%')
  ) as foo USING UNIQUE gid USING SRID=27582"
  STATUS DEFAULT
  TYPE POLYGON
  CLASS
    LABEL
    SIZE MEDIUM
    TYPE BITMAP
    BUFFER 0
    COLOR 22 8 3
    FORCE FALSE
    MINDISTANCE -1
    MINFEATURESIZE -1
    OFFSET 0 0
    PARTIALS TRUE
    POSITION CC
  END
  STYLE
    OUTLINECOLOR 0 0 255
    SIZE 2
    SYMBOL "circle"
  END
END
END
```

Au niveau de l'affichage avec MapServer, j'obtiens

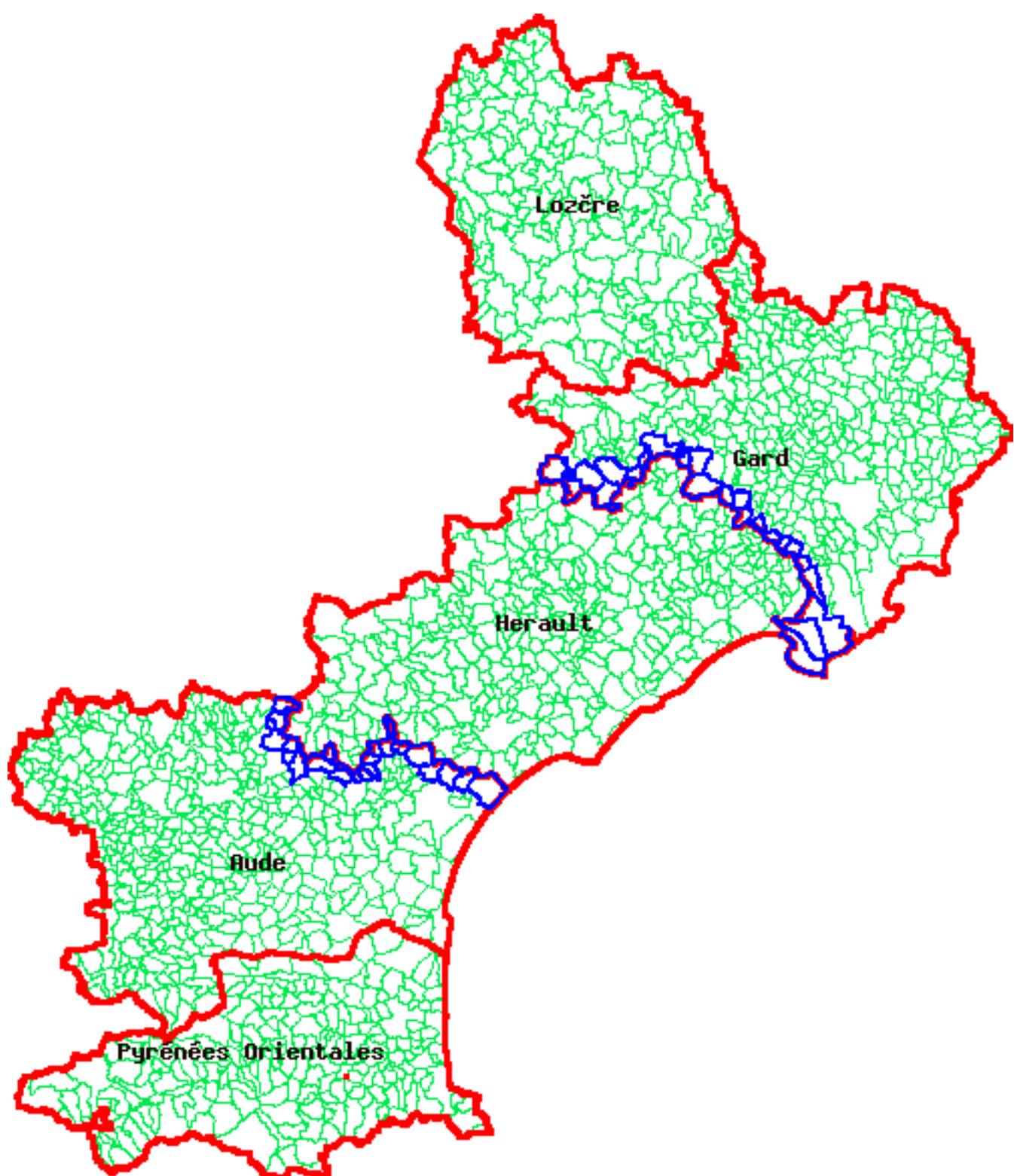


FIG. 6.7 – MapServer : Communes du Gard (30) et de l'Aude (11) limitrophes à l'Hérault (34)

6.1.18 QGIS : Affichage des tables précédentes

A la rédaction de ce document, depuis le 7 septembre 2005 la version 0.7 de QGIS est disponible pour diverses plateformes dont Windows. PostGIS 1.0.X est accepté par ce viewer - qui est bien plus qu' un viewer ! -. QGIS 0.7 est disponible à cette adresse

<http://qgis.org>

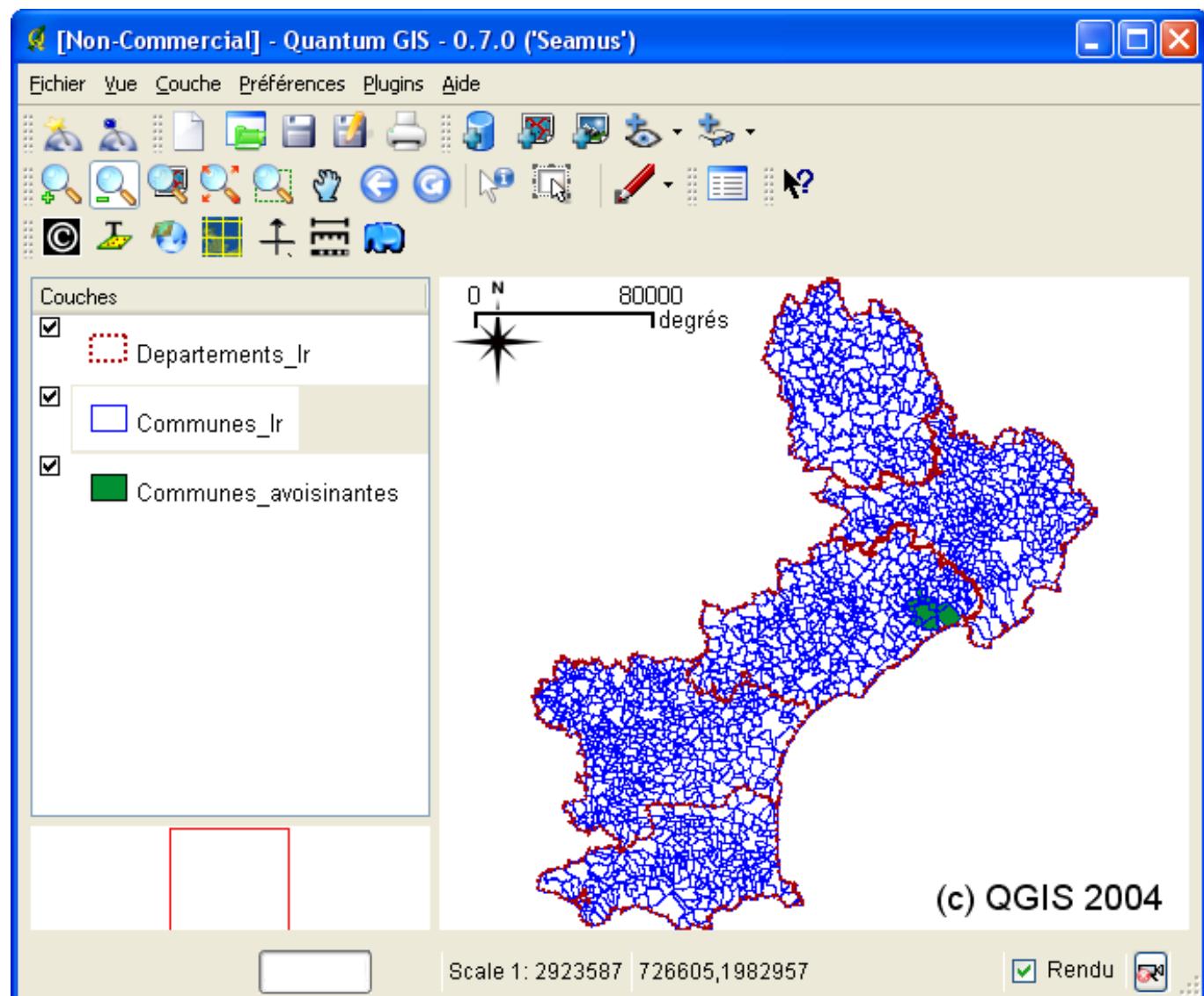


FIG. 6.8 – QGIS : Affichage des tables communes_lr, departements_lr et communes_avoisinantes

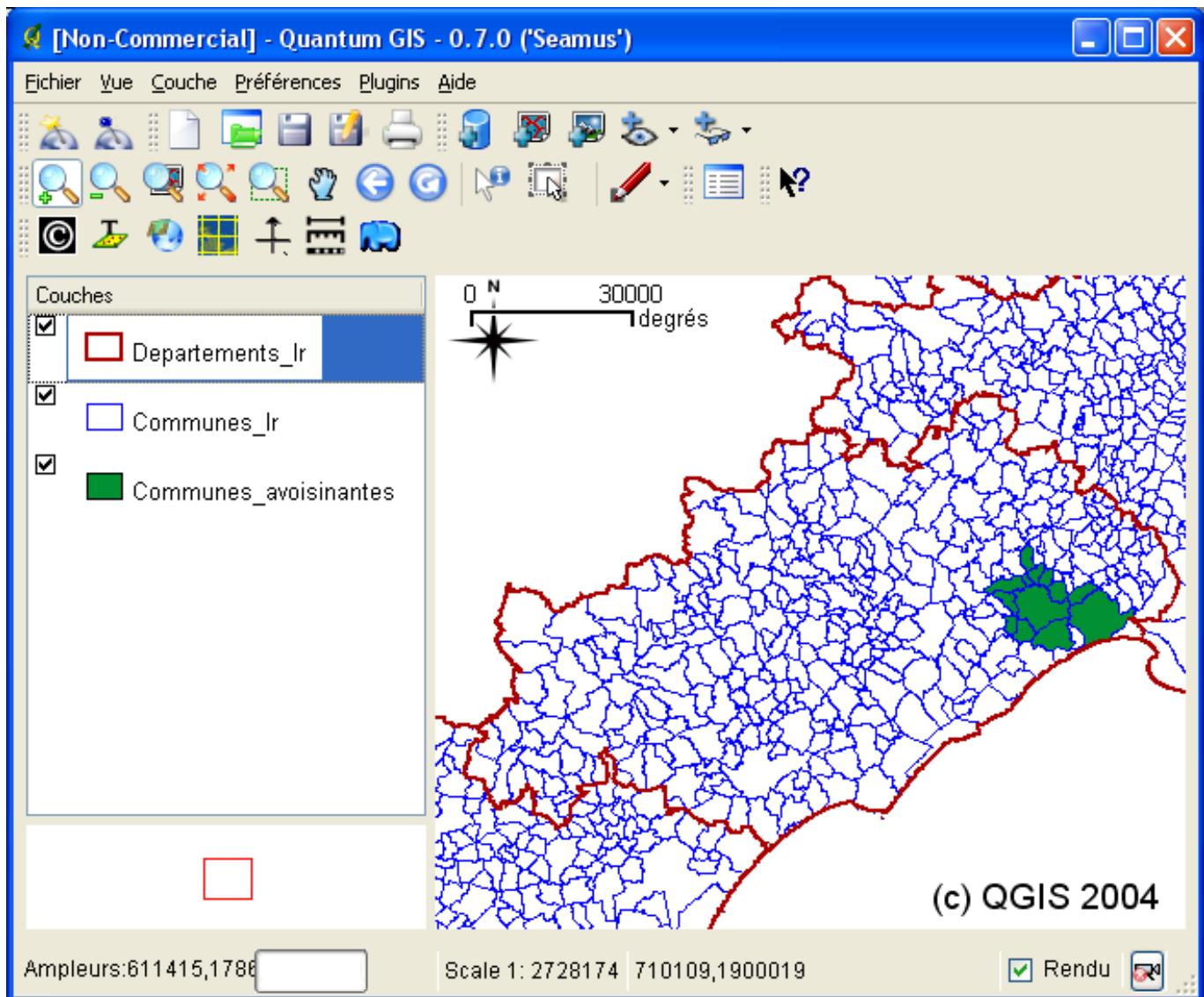
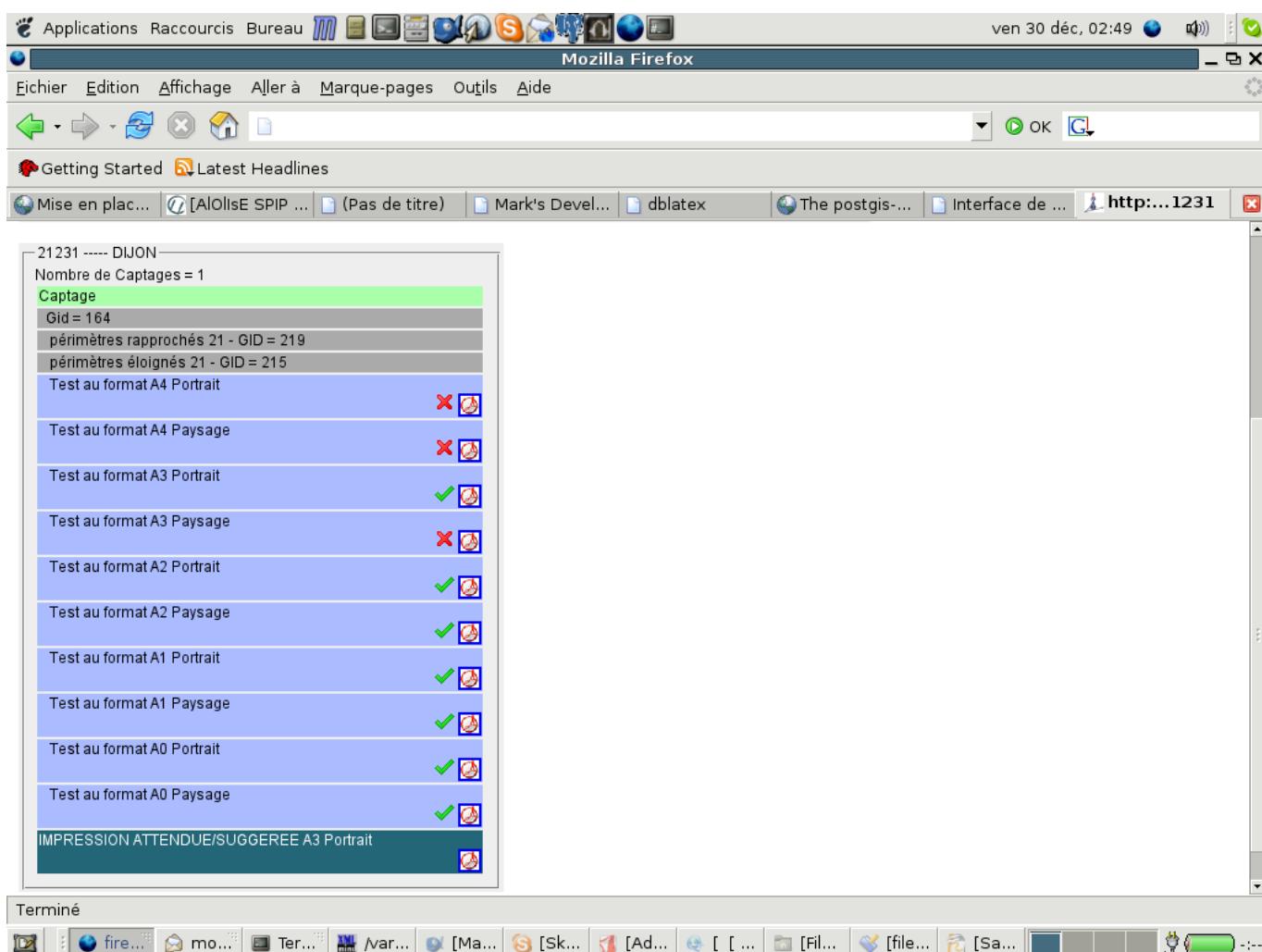


FIG. 6.9 – QGIS : Zoom sur le département de l'Hérault

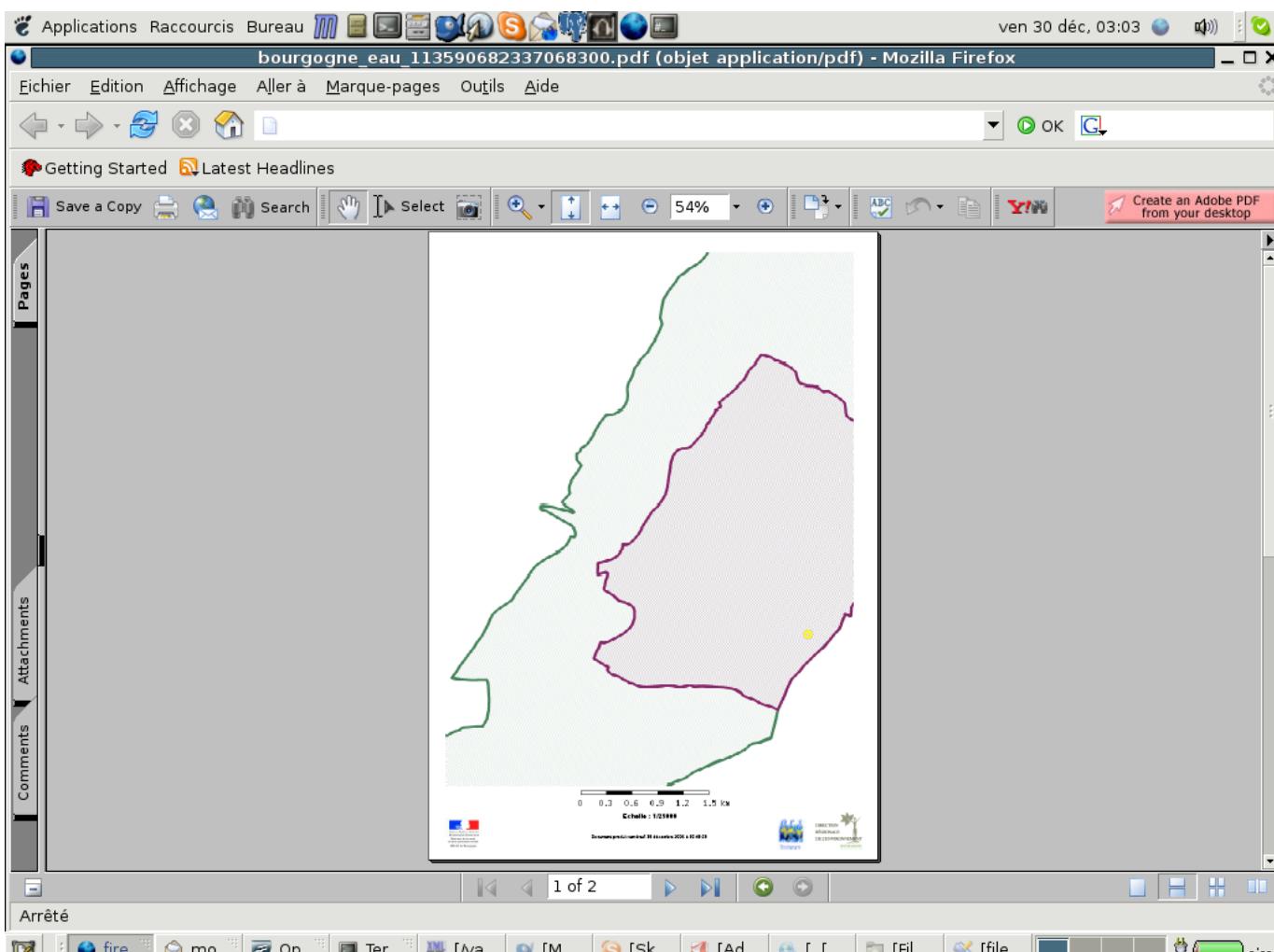
6.1.19 Exemple de projet sous GNU/Linux avec MapServer/PostgreSQL/PostGIS : savoir si à une échelle donnée, quel sera le meilleur format d'impression de A4 à A0 pour savoir si un polygone ne débordera du cadre de la carte

Travaillant personnellement sous Linux, je me permettrais ici uniquement de citer un exemple de projet réalisé sous GNU/Linux pour lequel on souhaiterait savoir à l'échelle du 25000ème quel serait pour un polygone rattaché à un point dans la ville de DIJON le meilleur format possible allant du A4 Portrait au A0 Paysage.

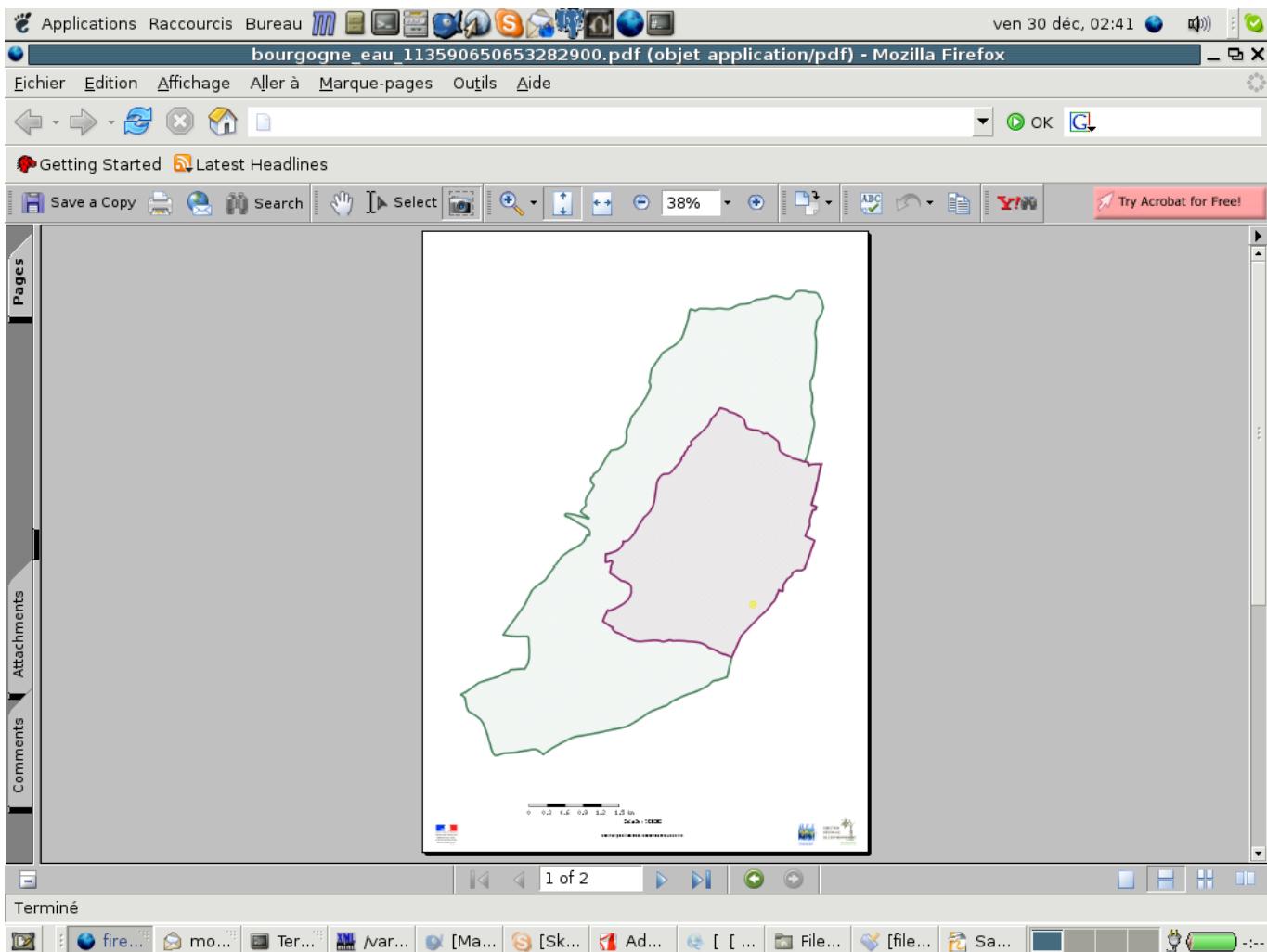
Sur l'image suivante, une croix rouge précise que le format est déconseillé, alors que le symbole en vert propose que le format étudié est possible. On cherche à savoir par exemple si le polygone en vert (voir les images) débordera ou non de la carte.



D'après cette image, on s'aperçoit donc que le format A4 Portrait ne convient pas :



alors que le format A3 convient parfaitement



En utilisant le wrapper PHP pour PostGIS réalisé par Paul SCOTT, un extrait de code possible pour traiter un tel projet serait par exemple :

```
/**  
 * Function IsAGoodFormat  
 * Permet de tester si pour un format d'impressions sera parfait  
 * selon le captage en question  
 */  
function IsAGoodFormat($Connexion,  
                      $mapfile,  
                      $gidcaptage,  
                      $couches,  
                      $gids,  
                      $Dept,  
                      $format  
                      )  
{  
//  
// Preparation des requetes  
//  
//echo $couches;echo $gids;  
$tables = array();$tables = explode(",",$couches);  
$tables_gid = array();$tables_gid = explode(",",$gids); //echo count($tables);  
if ( (count($tables)==1) && (count($tables_gid)==1) )  
{  
    $result= 0;  
}
```

```
else
{ //Debut du traitement general
if (count($tables)==2)
{
    $table = $tables[1];
    $Requete = "select the_geom from ".$table." where ".$table.".gid=".$tables_gid[1];
}
else
{
    $Requete = "";
    for($ItTable=1;$ItTable<count($tables);$ItTable++)
    { // Debut de la boucle general
        $table = $tables[$ItTable];
        $gid = $tables_gid[$ItTable];
        $Requete .= "(select the_geom from ".$table." where gid =".$gid.") union ";
    } // Fin de la boucle general
    $Requete = substr($Requete,0,strlen($Requete)-7);
}
//echo $format." --- ".$Requete."<br>";
$Connexion->exec("SELECT Xmin(foo.extent),Ymin(foo.extent),Xmax(foo.extent), ←
                    Ymax(foo.extent)
                     FROM
                     (select extent(fo.the_geom) from (".$Requete.") as fo) ←
                     AS foo");
if($Connexion->numRows() > 0)
{
    $Connexion->nextRow();$rs = $Connexion->fobject();
    $sw_MapFile = $mapfile;
    $goMap = ms_newMapObj( $sw_MapFile );
    $goMap->setextent($rs->xmin,$rs->ymin,$rs->xmax,$rs->ymax);
    $goMap->set("width",GetImageWidth($format));$goMap->set("height",GetImageHeight ←
($format));
    $oPixelPos = ms_newpointobj();
    $WidthPix = 400;
    $HeightPix = 300;
    $MapScale = GetScale();
    $oPixelPos->setxy($WidthPix/2, $HeightPix/2);
    $oGeorefExt = ms_newrectobj();
    $oGeorefExt->setextent($rs->xmin,$rs->ymin,$rs->xmax,$rs->ymax);
    $zoomc = $goMap->zoomscale($MapScale, $oPixelPos, $WidthPix, $HeightPix, ←
    $oGeorefExt);
    $Connexion->exec( "select GeomFromText('MULTIPOINT("$.rs->xmin." ".$rs->ymin ←
        .",".$rs->xmax." ".$rs->ymax.")',-1) @ GeomFromText('MULTIPOINT("$.goMap-> ←
        extent->minx." ".$goMap->extent->miny.", ".$goMap->extent->maxx." ".$goMap-> ←
        extent->maxy.")',-1) as affichage");
    $Connexion->nextRow();$rs = $Connexion->fobject();
    $result = (($rs->affichage=="t")? 1:0);
    $GoodMap = $goMap;
}
}

/// Fin du traitement general

return array($result,$GoodMap);
}
```

6.2 Cas pratique II : Réunifier des tronçons d'un réseau hydrologique d'un département

Ici nous exposons un second cas auquel nous sommes parfois confronté

6.2.1 Objectifs

Supposons que dans nous disposions d'une table contenant les tronçons des rivières, canaux, fleuves (...) traversent la DROME (26). Nous appellerons cette table troncons_rivieres_26. Nous souhaiterions pouvoir réunifier les tronçons de manière à pouvoir reconstituer le réseau hydrologique de la Drôme. Ici nous ne ferons pas de distinction entre les diverses natures des tronçons. Je veux dire par là que nous ne tiendrons pas contre à savoir si un tronçon fait parti d'une rivière, canal etc...

Dans la table troncons_rivieres_26, nous avons à notre disposition deux champs toponyme et cgenelin qui nous permettent de réunifier les tronçons en fonction du toponyme qui les regroupent entre eux. Par exemple, nous pourrions avoir

```
SELECT DISTINCT count(*), toponyme, cgenelin FROM troncons_rivieres_26 GROUP BY 2,3 ORDER BY 1 DESC
```

qui nous renverra

count	toponyme	cgenelin
...
172	fleuve le rhône	XXXXXX
132	rivière la roanne	XXXXXX
129	rivière la lyonne	XXXXXX
127	rivière l'isère	XXXXXX
116	rivière l'ouèze	XXXXXX
116	ruisseau le chalon	XXXXXX
112	rivière la galaure	XXXXXX
111	rivière la méouge	XXXXXX
...

On voit par exemple ici que le fleuve le rhône qui traverse la Drôme est constitué de 172 tronçons, la roanne de 132 etc...etc...

Notre but ici est donc de reconstituer le réseau hydrologique à partir de ces deux champs

6.2.2 Mise en oeuvre

Nous ferons appel ici à une transaction de PostgreSQL

```
BEGIN;  
--  
-- Début de la transaction  
--  
-- select dropgeometrytable('reseau_hydro_26');  
--  
-- Sur la table reseau_hydro_26 à créer, activation des oids  
--  
SET default_with_oids=true;  
--  
-- Création de la table en récupérant  
-- le toponyme, le cgenelin et en faisant  
-- la réunion géométrique des tronçons  
-- grâce à geomunion()  
--  
CREATE TABLE reseau_hydro_26 AS  
  (SELECT DISTINCT toponyme, cgenelin, geomunion(the_geom) FROM troncons_rivieres_26 GROUP BY toponyme, cgenelin);  
--  
-- Ajout de la colonne géométrique  
--  
SELECT addgeometrycolumn('reseau_hydro_26','the_geom',-1,'MULTILINESTRING',2);  
--  
-- La colonne the_geom est vide,  
-- on la met à jour en la remplaçant par la colonne geomunion
```

```
--  
UPDATE reseau_hydro_26 SET the_geom=geomunion;  
--  
-- Nous n'avons plus besoin de la colonne geomunion  
--  
ALTER TABLE reseau_hydro_26 DROP COLUMN geomunion;  
--  
-- fin de la transaction  
--  
END;
```

On obtiendra alors le réseau comme sur l'image suivante

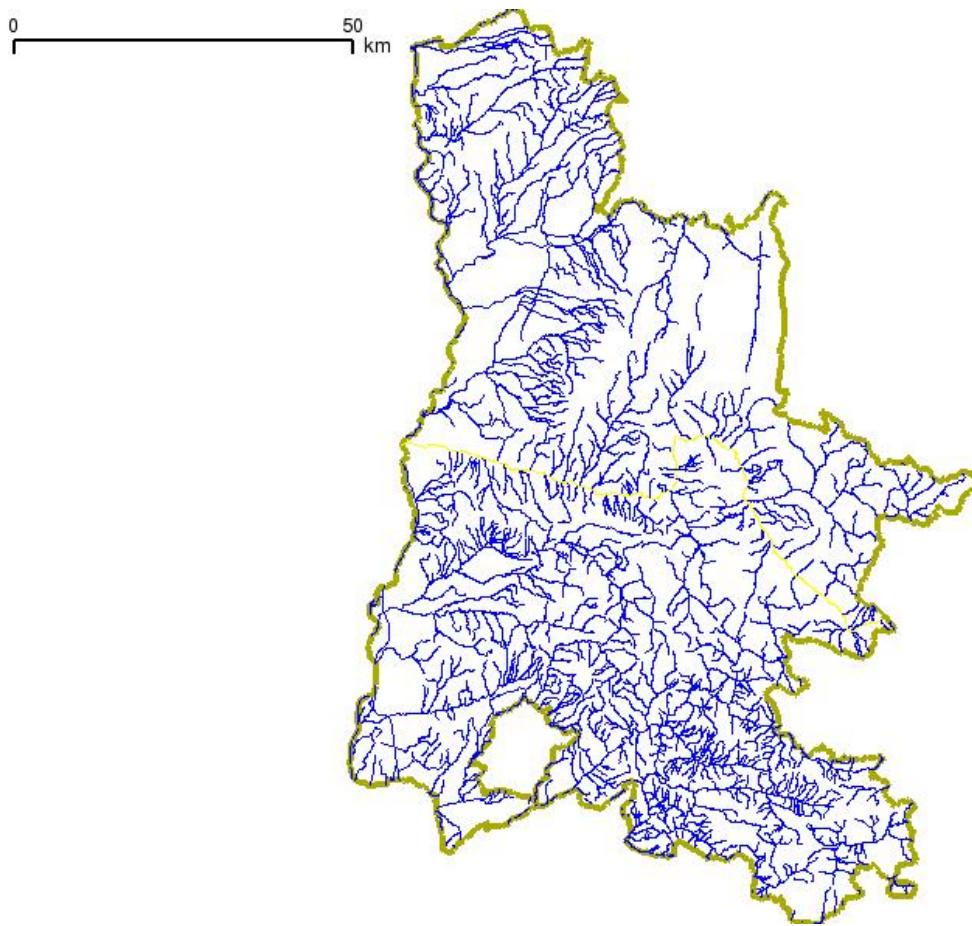


FIG. 6.10 – Reconstitution du réseau hydrolique de la DROME (26). La drôme est en jaune sur l'image.

6.2.3 Questions

Puisque notre table est créée, On va essayer ici d'ajouter des questions utiles de PostGIS qui permettent de faire le tour sur d'autre fonctions possibles

6.2.3.1 De combien d'objets est constituée la rivière de la Drôme ? - NumGeometries() -

Nous utiliserons ici la fonction NumGeometries () qui peut-être utilisée car les objets sont de type MULTI.

```
SELECT toponyme, numgeometries(the_geom) FROM reseau_hydro_26 WHERE topohy_sign='drôme, la (↔  
rivièr)e)';
```

qui nous renverra

toponyme	numgeometries
drôme, la (rivièr)e)	206

(1 ligne)

La "drôme" est donc la réunion géométrique de 206 tronçons.



AVERTISSEMENT

L'utilisation de la fonction NumGeometries() ne peut se faire que si l'on s'est d'avance que la nature de l'objet à interroger est de type MULTI (POLYGON, LINESTRING ou POINT).

Chapitre 7

PostGIS et les langages

Dans ce chapitre, je vais tenter de mettre à profit les divers languages que j'ai peu testé sous Win32 avec PostGIS.

7.1 avec du C -interface client de PostgreSQL -

Les sources de PostGIS sont fournis avec un petit exemple de code en C que vous trouverez dans postgis-1.2.0\extra\wkb_reader. Nous allons ici adapter le code de certains fichiers à Windows pour pouvoir les utiliser pour un petit exemple que nous avons traiter dans le précédent chapitre.

7.1.1 Modifications des fichiers

Sous Win32 pour MinGW, on ne dispose pas du fichier d'en-tête endian.h, ouvrez donc le fichier wkbtest.h et au début du fichier remplacez la ligne

```
...
#include <endian.h>
...
```

par

```
...
#ifndef __ENDIAN_H__
#define __ENDIAN_H__
#include <sys/param.h>
#endif /* __ENDIAN_H__ */
...
```

Ouvrez ensuite le fichier readwkb.c et remplacer son contenu par le suivant -que j'ai ici adapté à notre exemple -

```
#include "wkbtest.h"

/* example set-up
```

Nous allons ici utiliser le premier exemple de données rencontrées au cours du chapitre précédent

Ne pas oublier de définir convenablement les variables d'environnement PGDATABASE et PGUSER avant d'utiliser ce programme

Also, change the "declare cursor" command so it returns the columns you want, and converts the geometry

```
into wkb.  
  
*/  
  
void  
exit_nicely(PGconn *conn)  
{  
    PQfinish(conn);  
    exit(1);  
}  
  
void dump_bytes( char *a, int numb)  
{  
    int t;  
  
    for (t=0; t<numb; t++)  
    {  
        printf(" + Byte # %i has value %i (%x)\n",t,a[t],a[t]);  
    }  
}  
  
// need to find the OID corresponding to the GEOMETRY type  
//  
// select OID from pg_type where typname = 'wkb';  
  
int find_WKB_typeid(PGconn *conn)  
{  
    PGresult *dbresult;  
    char *num;  
  
    if (PQstatus(conn) == CONNECTION_BAD)  
    {  
        fprintf(stderr, "no connection to db\n");  
        exit_nicely(conn);  
    }  
  
    dbresult = PQexec(conn, "select OID from pg_type where typname = 'bytea';");  
  
    if (PQresultStatus(dbresult) != PGRES_TUPLES_OK)  
    {  
        fprintf(stderr, "couldnt execute query to find oid of geometry type");  
        exit_nicely(conn); //command failed  
    }  
  
    if( PQntuples(dbresult) != 1)  
    {  
        fprintf(stderr, "query to find oid of geometry didnt return 1 row!");  
        exit_nicely(conn);  
    }  
  
    num = PQgetvalue(dbresult, 0, 0); // first row, first field  
  
    PQclear(dbresult);  
  
    return ( atoi(num) );  
}  
  
main()  
{
```

```
char      *pghost,
        *pgport,
        *pgoptions,
        *pgtty;
char      *dbName;
int       nFields;
int       row,
        field;
PGconn   *conn;
PGresult *res;
int      junk,j;
char    *field_name;
int      field_type;
int      WKB_OID;
char    conn_string[255];

bool     *bool_val;
int      *int_val;
float   *float_val;
double  *double_val;
char    *char_val;
char    *wkb_val;
char    *table_name = "test";
char    query_str[1000];

/* make a connection to the database */
conn = PQconnectdb("");

/*
 * check to see that the backend connection was successfully made
 */
if (PQstatus(conn) == CONNECTION_BAD)
{
    fprintf(stderr, "%s", PQerrorMessage(conn));
    exit_nicely(conn);
}

//what is the geometry type's OID #?
WKB_OID = find_WKB_typeid(conn);

/* start a transaction block */
res = PQexec(conn, "BEGIN");
if (!res || PQresultStatus(res) != PGRES_COMMAND_OK)
{
    fprintf(stderr, "%s", PQerrorMessage(conn));
    PQclear(res);
    exit_nicely(conn);
}

/*
 * should PQclear PGresult whenever it is no longer needed to avoid
 * memory leaks
 */
PQclear(res);

/*
 * fetch rows from the pg_database, the system catalog of
 * databases
 */
sprintf(query_str, "DECLARE mycursor BINARY CURSOR FOR select text(genre), astext(geom) ←
                  as astext,geometrytype(geom) as type_geometrique,srid(geom),assvg(geom) as svg,text(←
                  as text");


```

```
area2d(geom)) as aire from %s", table_name);

printf(query_str); printf("\n");

res = PQexec(conn, query_str);

if (!res || PQresultStatus(res) != PGRES_COMMAND_OK)
{
    fprintf(stderr, "DECLARE CURSOR command failed\n");
    fprintf(stderr, "%s", PQerrorMessage(conn));
    PQclear(res);
    exit_nicely(conn);
}
PQclear(res);

res = PQexec(conn, "FETCH ALL in mycursor");
if (!res || PQresultStatus(res) != PGRES_TUPLES_OK)
{
    fprintf(stderr, "FETCH ALL command didn't return tuples properly\n");
    fprintf(stderr, "%s", PQerrorMessage(conn));
    PQclear(res);
    exit_nicely(conn);
}
j=0;
for (row=0; row< PQntuples(res); row++)
{
    j++;
    printf("-----Ligne %i -----\\n", j);
    //not so efficient, but...
    for (field =0 ; field< PQnfields(res); field++)
    {
        field_type = PQftype(res,field);
        field_name = PQfname(res, field);

        //we just handle a few of the popular type since this is just an example

        if (field_type ==16)// bool
        {
            bool_val = (bool *) PQgetvalue(res, row, field);
            if (*bool_val)
                printf("%s: TRUE\\n",field_name);
            else
                printf("%s: FALSE\\n",field_name);
        }
        else if (field_type ==23 )//int4 (int)
        {
            int_val = (int *) PQgetvalue(res, row, field);
            printf("%s: %i\\n",field_name,*int_val);
        }
        else if (field_type ==700 )//float4 (float)
        {
            float_val = (float *) PQgetvalue(res, row, field);
            printf("%s: %g\\n",field_name,*float_val);
        }
        else if (field_type ==701 )//float8 (double)
        {
            double_val = (double *) PQgetvalue(res, row, field);
            printf("%s: %g\\n",field_name,*double_val);
        }
        else if ( (field_type ==1043 ) || (field_type==25) )//varchar
        {
            char_val = (char *) PQgetvalue(res, row, field);
```

```
    printf("%s: %s\n", field_name, char_val);
}
else if (field_type == WKB_OID) //wkb
{
    char_val = (char *) PQgetvalue(res, row, field);
    printf("%s: ", field_name);
    // skip 4 bytes varlena size
    decode_wkb(char_val, &junk);
    printf("\n");
}

}

PQclear(res);

/* close the cursor */
res = PQexec(conn, "CLOSE mycursor");
PQclear(res);

/* commit the transaction */
res = PQexec(conn, "COMMIT");
PQclear(res);

/* close the connection to the database and cleanup */
PQfinish(conn);

return 0;
}
```

On va aussi adapter le Makefile de ce répertoire à notre exemple en remplaçant

```
CFLAGS=-I`pg_config --includedir` -L`pg_config --libdir` -lpq
```

par

```
CFLAGS=-I`pg_config --includedir` `pg_config --libdir`/libpq.dll
```

7.1.2 Sortie et exécution

Depuis MinGW, il ne reste plus qu'à préciser les variables de PostgreSQL nécessaire à l'utilisation du programme

```
export PGDATABASE=testgis
export PGUSER=$USERNAME
export PGHOST=localhost
```

où ici **PGUSER** correspond à votre utilisateur de MinGW. **\$USERNAME** est une variable interne de MiNGW. Il ne reste plus qu'à faire¹

```
make
readwkb.exe
```

qui nous renverra

```
DECLARE mycursor BINARY CURSOR FOR select text(genre), astext(geom) as astext, geometrytype(geom) as type_geometrique, srid(geom), assvg(geom) as svg, text(area2d(geom)) as aire from test
```

¹On se sera assurer ici que l'utilisateur sous MinGW soit un super-utilisateur dans PostgreSQL. Pour cela, consultez le chapitre 3 : "Devenir soi-même super-utilisateur de PostgreSQL (pour l'utilisateur de MinGW)"

```
-----Ligne 1 -----
text: pieton 1
astext: POINT(10 70)
type_geometrique: POINT
srid: -1
svg: cx="10" cy="-70"
aire: 0
-----Ligne 2 -----
text: pieton 2
astext: POINT(30 30)
type_geometrique: POINT
srid: -1
svg: cx="30" cy="-30"
aire: 0
-----Ligne 3 -----
text: batiment 1
astext: POLYGON((10 10,40 20,35 8,12 4,10 10))
type_geometrique: POLYGON
srid: -1
svg: M 10 -10 40 -20 35 -8 12 -4 10 -10
aire: 228
-----Ligne 4 -----
text: batiment 2
astext: POLYGON((10 40,20 30,30 40,40 35,50 60,35 80,20 60,10 40))
type_geometrique: POLYGON
srid: -1
svg: M 10 -40 20 -30 30 -40 40 -35 50 -60 35 -80 20 -60 10 -40
aire: 1050
-----Ligne 5 -----
text: batiment 3
astext: POLYGON((10 95,20 95,20 135,10 135,10 95))
type_geometrique: POLYGON
srid: -1
svg: M 10 -95 20 -95 20 -135 10 -135 10 -95
aire: 400
-----Ligne 6 -----
text: pieton 3
astext: POINT(35 70)
type_geometrique: POINT
srid: -1
svg: cx="35" cy="-70"
aire: 0
-----Ligne 7 -----
text: pieton 4
astext: POINT(35 60)
type_geometrique: POINT
srid: -1
svg: cx="35" cy="-60"
aire: 0
-----Ligne 8 -----
text: bordure 1 route
astext: LINESTRING(1 85,50 85)
type_geometrique: LINESTRING
srid: -1
svg: M 1 -85 50 -85
aire: 0
-----Ligne 9 -----
text: bordure 2 route
astext: LINESTRING(1 92,50 92)
type_geometrique: LINESTRING
srid: -1
svg: M 1 -92 50 -92
```

```
aire: 0
```

Pour de plus amples informations sur l'interface client libpq en C de PostgreSQL, merci de vous reportez à la documentation de PostgreSQL. Consultez notamment

<http://traduc.postgresqlfr.org/pgsql-8.2.1-fr/client-interfaces.html>

7.2 avec du PHP - un peu de SVG -

Ici mon exemple est basé sur celui du chapitre 4 "Exemples de requêtes spatiales II"

On va commencer par définir la page index.php qui va contenir une liste déroulante de quelques requêtes spatiales qui seront afficher depuis une liste déroulante

```
<html>
<body bgcolor='#AAEEFC'>
<!--
    Formulaire générale
-->
<form method='get' action='<?php $PHP_SELF;?>'>
<!--
    Insertion des requetes possibles
-->
<select name='id'>
<option value='0'>Affichage normale</option>
<option value='1'>Afficher les personnes contenus dans le bâtiment 'Résidence des Mousquetaires'</option>
<option value='2'>Afficher les bâtiments qui ne contiennent aucune personne</option>
<option value='3'>Afficher les bâtiments contenus dans les parcs</option>
<option value='4'>Afficher les points d'intersection entre les petites et les grandes routes</option>
<option value='5'>Afficher l'intersection entre la rivière et les parcs</option>
<option value='6'>Afficher les personnes présentes autour de la rivière dans une rayon de 5 mètres</option>
</select>
<input type='submit' value='recharger'>
<br>
<?php
/*
    On commence par récupérer la valeur de id passer au script
*/
$id = (( isset($_GET["id"]) ) ? $_GET["id"] : "0");
?>
<!--
    insertion du svg
-->
<embed name='SVG' type='image/svg+xml' src='madatabase.php?id=<?php echo $id;?>' width='600 px' height='509.505693569px' />
<!--
    Fin du formulaire
-->
</form>
</body>
</html>
```

Lors de la soumission du formulaire ce script qui s'appelle lui-même fait appel au script madatabase.php suivant :

```
<?php
header("Content-type: image/svg+xml");
```

```
echo '<?xml version="1.0" encoding="iso-8859-1"?>';
$id = (( isset($_GET["id"]) ) ? $_GET["id"] : "0");
if ( extension_loaded('pgsql') != 1)
{
    switch (PHP_OS)
    {

        case "WINNT": if (!extension_loaded('pgsql')) dl("php_pgsql.dll");
                        break;
        default:   if (!extension_loaded('pgsql')) dl("php_pgsql.so");
                        break;
    }
}
?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN" "http://www.w3.org/TR/2001/REC-SVG ←
-20010904/DTD/svg10.dtd">
<?php $pg_hote = "localhost";
      $pg_base_de_donnees = "madatabasse";
      $pg_utilisateur = "postgres";
      $pg_mot_de_passe = " ";
      $pg_srid="-1";
$db_handle = pg_connect("host=".$pg_hote."
                           dbname=".$pg_base_de_donnees."
                           user=".$pg_utilisateur."
                           password=".$pg_mot_de_passe."");

$widthsvg = 600;
$Requete_MS_Tables = "SELECT * from mapserver_desc";

$Resultat_MS_Tables = pg_exec( $db_handle, $Requete_MS_Tables );
// tableaux nécessaire pour calculer l'extent
$Xmin = array(); $Xmax = array();
$Ymin = array(); $Ymax = array();
while( $MS_Ligne = pg_fetch_object( $Resultat_MS_Tables ) )
{
    /*
        calcul de l'Extent pour chaque table
    */

    $colonne_geometrique = "the_geom";
    $Requete_Extent = "select
                      xmin(extent(\".$colonne_geometrique.\")),
                      ymin(extent(\".$colonne_geometrique.\")),
                      xmax(extent(\".$colonne_geometrique.\")),
                      ymax(extent(\".$colonne_geometrique.\")) from
                      \"$MS_Ligne->ms_table\";

$Resultat_Extent = pg_exec( $db_handle, $Requete_Extent );

    while ( $Row_Extent = pg_fetch_object( $Resultat_Extent ) )
    {
        $Xmin[] = $Row_Extent->xmin;
        $Ymin[] = $Row_Extent->ymin;
        $Xmax[] = $Row_Extent->xmax;
        $Ymax[] = $Row_Extent->ymax;
    }
}

/*
*/
```

Calcul de l'Extent générale et des dimensions de l'image

```
/*
$xmin = min($Xmin);
$xmax = max( $Xmax );
$ymin = min($Ymin );
$ymax = max ($Ymax);
$width= abs($xmax-$xmin);
$height= abs($ymax-$ymin);
$rapport= $width / $height;

echo "<svg id='racine' width='".$widthsvg."px'
           height='".$widthsvg/$rapport."px'
           viewBox='".$xmin." ". -$1*$ymax." ".$width." ".$height."'>";
echo "
<style type='text/css'><![CDATA[
.StyleFondsmallroadsOn { fill:none; stroke:rgb(80,80,80); stroke-width:2}
.StyleFondsmallroadsOff { fill:none; stroke:rgb(80,80,80); stroke-width:2}
.StyleFondgreatroadsOn { fill:none; stroke:rgb(125,125,125); stroke-width:3}
.StyleFondgreatroadsOff { fill:none; stroke:rgb(80,80,80); stroke-width:3}
.StyleFondparcsOn { fill:rgb(0,123,0); stroke:rgb(255,255,255); stroke-width:0.1}
.StyleFondparcsOff { fill:none; stroke:rgb(255,255,255); stroke-width:0.1}
.StyleFonddriversOn { fill:rgb(0,12,189); stroke:rgb(0,12,189); stroke-width:1}
.StyleFonddriversOff { fill:none; stroke:rgb(80,80,80); stroke-width:11}
.StyleFondbuildingsOn { fill:rgb(234,156,78); stroke:black; stroke-width:0.4 }
.StyleFondbuildingsOff { fill:rgb(0,30,0); stroke:black; stroke-width:0.4 }
.StyleFondpersonnesOn { fill:rgb(255,0,0); stroke:black; stroke-width:0.1 }
.StyleFondpersonnesOff { fill:rgb(255,0,0); stroke:black; stroke-width:0.1 }
]]>
</style>";
/*
    On liste les tables à afficher ici

*/
$Table_Geom = array();
$Table_Geom[] = "small_roads";
$Table_Geom[] = "great_roads";
$Table_Geom[] = "parcs";
$Table_Geom[] = "rivers";
$Table_Geom[] = "buildings";
$Table_Geom[] = "personnes";
/*
    On ferme le fichier SVG
*/
for($Iter_Geom = 0;$Iter_Geom< count($Table_Geom);$Iter_Geom++)
{ // Début de la boucle sur les tables

    $table = $Table_Geom[$Iter_Geom];

    $Requete_SVG = "select AsSVG(the_geom,0) from ".$table;
    $Requete_Geom_Type = "select type from geometry_columns where
                           f_table_name like '". $table ."'";

    $Resultat_SVG = pg_exec($db_handle, $Requete_SVG);

    $Resultat_Geom_Type = pg_result(pg_exec($db_handle, $Requete_Geom_Type),
                                    0,
                                    0);

    $Nb_ligne_table_SVG = pg_numrows( $Resultat_SVG );
```

```
for ($It_SVG=0; $It_SVG < $Nb_ligne_table_SVG; $It_SVG++)
{

echo "<g id='id_".$table."_".$It_SVG."'"
      class='StyleFond'.str_replace("_","", $table)."On'>";

switch    ($Resultat_Geom_Type)
{

case "POINT": echo "<circle ".pg_result($Resultat_SVG,$It_SVG,0)." r='1' />";
                break;

default:
    echo "<path id='id_path_".$table."_".$It_SVG."' d='".$pg_result($Resultat_SVG, ←
        $It_SVG,0)."' />";
    break;
}
echo "</g>";

}

} // Fin de la boucle sur les tables
/*
$Table_Geom = array();
$Table_Geom[] = "small_roads";
$Table_Geom[] = "great_roads";
$Table_Geom[] = "parcs";
$Table_Geom[] = "rivers";
$Table_Geom[] = "buildings";
$Table_Geom[] = "personnes";
for ($Iter_Geom = 0; $Iter_Geom < count($Table_Geom); $Iter_Geom++)
{// Début de la boucle sur les tables
$table = $Table_Geom[$Iter_Geom];

$Requete_Gid = "select gid from ".$table;

$Resultat_Gid = pg_exec($db_handle, $Requete_Gid);

$Nb_ligne_table_Gid = pg_numrows( $Resultat_Gid );

for ($It_SVG=0; $It_SVG < $Nb_ligne_table_Gid; $It_SVG++)
{
    if (($table=="great_roads") || ($table=="small_roads"))
    {
        $Requete_Affichage_buildings = "select assvg(translate(pointonsurface(the_geom ←
            ), -12, 0), 1), data from ".$table." where gid=".intval($It_SVG+1);
        $Resultat_Affichage_buildings = pg_exec($db_handle, ←
            $Requete_Affichage_buildings);

        $my_SVG->ecrire("<text ".pg_result($Resultat_Affichage_buildings,0,0)." fill='←
            black'
font-size='3' font-color='black'
>".pg_result($Resultat_Affichage_buildings,0,1)."←
</text>");

    }
}
```

```
        }
    }
}

/*
   Traitement des cas

*/
switch ($id)
{

    case 1: $Requete_Affichage = "select AsSvg(personnes.the_geom,0),
                                AsSvg(PointOnSurface(personnes.the_geom),1),
                                personnes.data from personnes,buildings
                                where within(personnes.the_geom,buildings.the_geom)
                                and buildings.data like 'Résidence des Mousquetaires';
                $Resultat_Affichage = pg_exec($db_handle, $Requete_Affichage);
                for($It_SVG=0;$It_SVG< pg_numrows($Resultat_Affichage);$It_SVG++)
                {
                    echo "<circle fill='blue' ".pg_result($Resultat_Affichage,$It_SVG,0)." r='2'>
                        <animate attributeName='fill' values='blue;lightskyblue' dur='1.5s' ←
                            repeatCount='indefinite' begin='0s' calcMode='discrete' />
                    </circle>";
                    echo "<text ".pg_result($Resultat_Affichage,$It_SVG,1)." fill='black'
                        font-size='3' font-color='white'
                    >".pg_result($Resultat_Affichage,$It_SVG,2)."
                    </text>";
                }
                break;
    case 2: $Requete_Affichage = "select AsSvg(the_geom,0),
                                AsSvg(PointOnSurface(the_geom),1),
                                b.data from buildings b,
                                (select geomunion(the_geom) from personnes) gp
                                where
                                not intersects(gp.geomunion,b.the_geom)";
                $Resultat_Affichage = pg_exec($db_handle, $Requete_Affichage);
                for($It_SVG=0;$It_SVG< pg_numrows($Resultat_Affichage);$It_SVG++)
                {
                    echo "<path fill='rgb(234,156,78)' d='".pg_result($Resultat_Affichage,$It_SVG ←
                        ,0)."'>
                        <animate attributeName='stroke' values='white;lightskyblue' dur='1.5s' ←
                            repeatCount='indefinite' begin='0s' calcMode='discrete' />
                    </path>";
                    echo "<text ".pg_result($Resultat_Affichage,$It_SVG,1)." fill='black'
                        font-size='3' font-color='white'
                    >".pg_result($Resultat_Affichage,$It_SVG,2)."
                    </text>";
                }
                break;
    case 3: $Requete_Affichage = "select AsSvg(the_geom,0),
                                AsSvg(PointOnSurface(the_geom),1),
                                b.data from buildings b where
                                Contains(parcs.the_geom,b.the_geom)";
                $Resultat_Affichage = pg_exec($db_handle, $Requete_Affichage);
                for($It_SVG=0;$It_SVG< pg_numrows($Resultat_Affichage);$It_SVG++)
                {
                    echo "<path fill='rgb(234,156,78)' d='".pg_result($Resultat_Affichage,$It_SVG ←
                        ,0)."'>
                        <animate attributeName='stroke' values='white;lightskyblue' dur='1.5s' ←
                            repeatCount='indefinite' begin='0s' calcMode='discrete' />
```

```
        </path>";
echo "<text ".pg_result($Resultat_Affichage,$It_SVG,1)." fill='black'
font-size='3' font-color='white'
>".pg_result($Resultat_Affichage,$It_SVG,2)."
</text>";
}

break;

case 4:      $Requete_Affichage = "select
assvg(intersection(g.the_geom,s.the_geom),1),
astext(intersection(g.the_geom,s.the_geom)),
assvg(intersection(g.the_geom,s.the_geom),0)
from great_roads g, small_roads s
where intersects(g.the_geom,s.the_geom)";
$Resultat_Affichage = pg_exec($db_handle, $Requete_Affichage);
for($It_SVG=0;$It_SVG< pg_numrows($Resultat_Affichage);$It_SVG++)
{
echo "<circle fill='blue' ".pg_result($Resultat_Affichage,$It_SVG,2)." r='2'>
<animate attributeName='fill' values='blue;lightskyblue' dur='1.5s' ←
repeatCount='indefinite' begin='0s' calcMode='discrete' />
</circle>";
echo "<text ".pg_result($Resultat_Affichage,$It_SVG,0)." fill='black'
font-size='3' font-color='black'
>".pg_result($Resultat_Affichage,$It_SVG,1)."
</text>";
}
break;
case 5: $Requete_Affichage = "select
AsSvg(Intersection(r.the_geom,p.the_geom),0)
from rivers r, parcs p
where intersects(r.the_geom,p.the_geom)";
$Resultat_Affichage = pg_exec($db_handle, $Requete_Affichage);
for($It_SVG=0;$It_SVG< pg_numrows($Resultat_Affichage);$It_SVG++)
{
echo "<path d='".pg_result($Resultat_Affichage,$It_SVG,0)."'>
<animate attributeName='fill' values='blue;lightskyblue' dur='1.5s' ←
repeatCount='indefinite' begin='0s' calcMode='discrete' />
</path>";
}
break;
case 6:      $Requete_Affichage = "select AsSvg(p.the_geom,0),
AsSvg(PointOnSurface(p.the_geom),1),
p.data from personnes p
where contains(buffer(rivers.the_geom,5),p.the_geom)";
$Resultat_Affichage = pg_exec($db_handle, $Requete_Affichage);
for($It_SVG=0;$It_SVG< pg_numrows($Resultat_Affichage);$It_SVG++)
{
echo "<circle fill='blue' ".pg_result($Resultat_Affichage,$It_SVG,0)." r='2'>
<animate attributeName='fill' values='blue;lightskyblue' dur='1.5s' ←
repeatCount='indefinite' begin='0s' calcMode='discrete' />
</circle>";
echo "<text ".pg_result($Resultat_Affichage,$It_SVG,1)." fill='black'
font-size='3' font-color='white'
>".pg_result($Resultat_Affichage,$It_SVG,2)."
</text>";
}
$Requete_Affichage_Buffer = "select AsSvg(buffer(the_geom,5),0) from rivers";
$Resultat_Affichage_Buffer = pg_exec($db_handle, ←
$Requete_Affichage_Buffer);
for($It_SVG=0;$It_SVG< pg_numrows($Resultat_Affichage_Buffer);$It_SVG++)
```

```
{  
    echo "<path fill='none' d='".$pg_result($Resultat_Affichage_Buffer,$It_SVG,0) ."'>  
        <animate attributeName='stroke' values='red;lightskyred' dur='1.5s'  ←  
            repeatCount='indefinite' begin='0s' calcMode='discrete' />  
    </path>";  
}  
break;  
  
default: break;  
  
}// Fin des test pour id  
/*  
    Connexion à PostgreSQL => Fermeture  
*/  
pg_close ($db_handle);  
?>  
</svg>
```

On aura par exemple l'image suivante

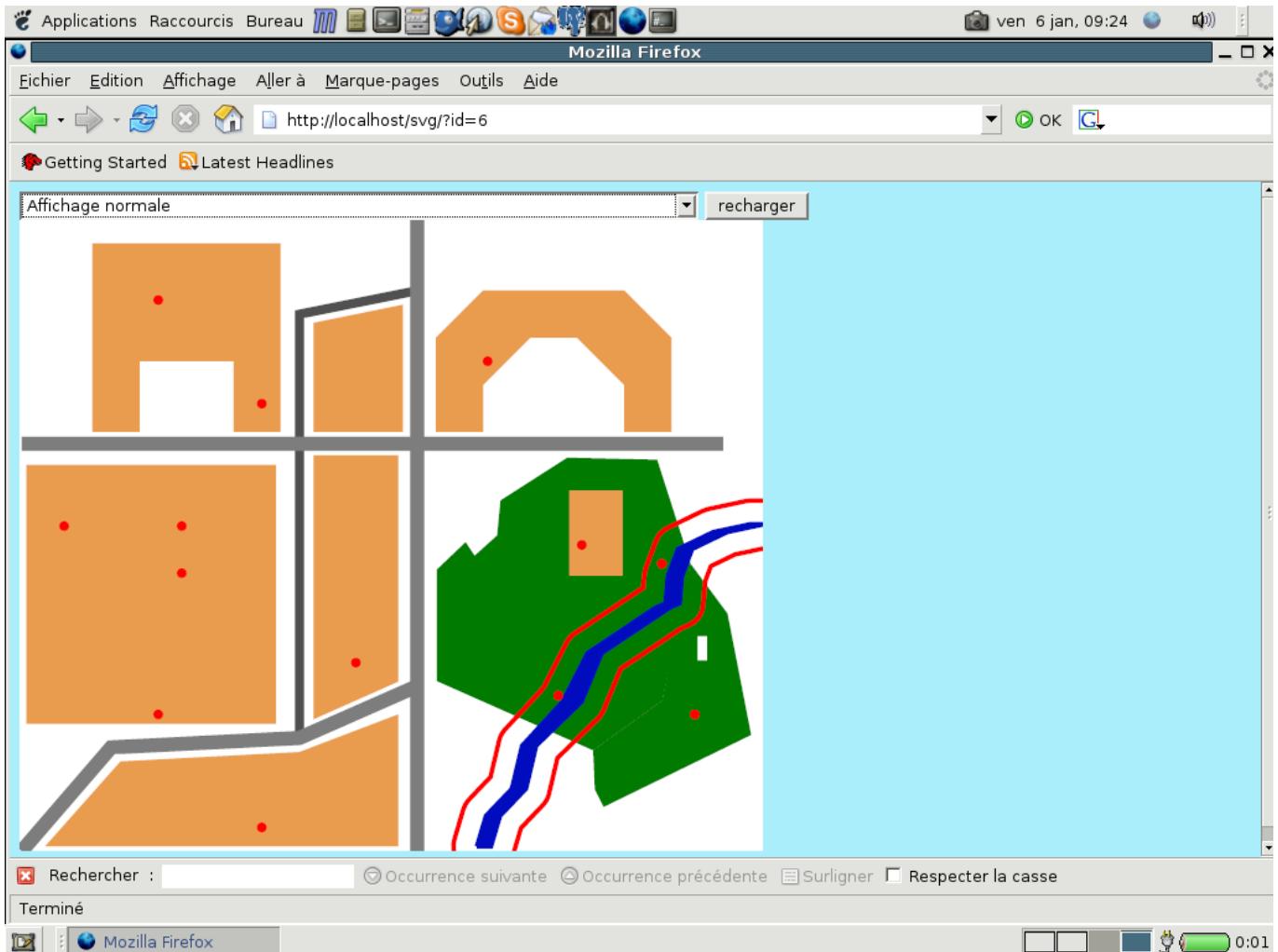


FIG. 7.1 – Sortie en SVG sur quelques requêtes de la section "Exemples de requêtes spatiales II" du chapitre 4

Chapitre 8

Conclusion

Dans ce document, nous nous sommes efforcés de montrer le travail qui puisse attendre un administrateur PostgreSQL/PostGIS sous Windows. Cet administrateur en effet joue un rôle important notamment dans la mise en place d'une solution WebSIG.

En amont, l'un des premiers points forts notifiés aura été de décortiquer tout le travail nécessaire en amont pour pouvoir travailler avec ses outils préférés. Ce qui fut le cas notamment des deux quatrres premier chapitres. En effet, la mis en place d'un tel environnement peut paraître au premier abord déroutant. Mais il est quand même important de signaler qu'avant le printemps 2004, l'utilisation de PostgreSQL/PostGIS sous Windows relevait du parcours du combattant. Il fallait avant tout installer cygwin. Ce qui n'était pas une mince affaire (énormément de choses à installer) !

En aval, dans le chapitres 4, nous avons vu comment installer PostgreSQL manuellement. Aujourd'hui depuis l'hivers 2004, l'équipe de Magnus HAGANDER fournit un installateur pour PostgreSQL qui ne cesse d'évoluer depuis la première version en 8.0.0 beta 1. Je me suis efforcé dans ce chapitre de montrer ce qui se passe derrière l'installateur fournit par Magnus. L'installateur suit les diverses instructions proposées à certaines subtilités prêt. Cette habitude de montrer comment tout installer manuellement vient de l'expérience acquise bien avant l'arrivée de l'installleur de l'équipe de PGInstaller - habitude que j'essaie ici de partager autant que possible - !

Dans les chapitres 5 et 6, par rapport à la documentation officielle de PostGIS réalisée par Sandro SANTILLI et Paul RAMSEY, je me suis efforcé comme un professeur de proposer un chapitre évolutif sur l'apprentissage de PostGIS. Parfois autant que possible, j'ai essayé de décortiquer certains points qui restent quand même assez succints dans la documentation officielle. On fait ce qu'on peut :-) pour y arriver en espérant que ce long chapitre aura trouver /prouver toute son utilité :-).

En sortant du cadre restrictif ici de cette documentation pour Windows, l'enjeu majeur réside dans la coeur même d'une solution cartographique qu'est la base de données spatiale, voire la problématique imposée pour le choix d'une solution logiciel de cartographie : choix du système (Mac OS X, Linux Windows...), choix humain, compétences techniques de toute une équipe etc...C'est surtout la raison pour laquelle je me suis efforcée de fournir autant que possible toutes ces requêtes et les commandes/instructions de ce document autour desquelles gravitent toute une solution à laquelle il faut se rattacher et peaufiner dans le temps ! Le reste des décisions une fois le choix de la base spatiale effectuée est l'affaire des programmeurs/décideurs - peu importe le langage dirais-je - une fois le choix de l'interface cartographique applicative choisie par analogie avec 'une API faisant abstraction du code interne et du langage

Cinquième partie

Annexes

Notes sur les annexes

Ici sont regroupés les diverses cas d'utilisation que nous rencontrons dans notre expérience quotidienne avec PostgreSQL et PostGIS. Nous vous les proposons afin porter à votre connaissance leur utilisation dans un cadre quotidien. Ces annexes ne peuvent pas à eux seul permettre de répondre/couvrir à toutes les attentes aussi bien pour PostgreSQL que pour PostGIS. Je m'efforce au contraire dans ces annexes de fournir des pistes de réflexion.

Annexe A

PgRouting pour le calcul d'itinéraire

Cité en exemple dans l'Avant-propos de la documentation, PgRouting est la librairie développée en étroite collaboration par Camptocamp (France/Suisse) et Orkney (Japon). PgRouting fait partie du projet PostLBS. Le site de PgRouting est <http://pgrouting.postgis.net/>. Gérald FENOY a proposé une traduction de la documentation de pgrouting disponible à <http://www.postgis.fr/book/print/360>

Je propose ici de montrer un exemple d'utilisation des fonctionnalités `shortest_path()`, `shortest_path_astar()` et la fonctionnalité `tsp()`. Comme les exemples ici proposés sont très minimalistes, je ne prends pas en considération l'usage des index ! Merci d'en prendre note.

A.1 Installation sous Windows

Pour PostgreSQL 8.2.X, il faut se procurer le fichier zippé http://files.orkney.jp/pgrouting/binaries/pgRouting-1.0.0.a-0_win32.exe. Il s'agit d'une version personnelle de PgRouting compilée.



FIG. A.1 – Installeur 1.0.0.a

A.2 Installation sous Ubuntu Dapper/Edgy

Conformément à la documentation sous Ubuntu, l'installation n'est pas des plus compliqués. Rapidement voyons les diverses étapes de l'installation

A.2.1 GAUL

Il faut se rendre à <http://prdownloads.sourceforge.net/gaul/gaul-devel-0.1849-0.tar.gz?download> pour télécharger gaul-devel-0.1849-0.tar.bz2. Ensuite

```
tar xzf gaul-devel-0.1849-0.tar.gz
cd gaul-devel-0.1849-0
./configure --enable-slang=no
make
make install
```

A.2.2 BGL

La commande suivante suffira amplement

```
apt-get install libboost-graph-dev libboost-graph1.33.1
```

A.2.3 CGAL

Il faut télécharger et l'installer

```
wget ftp://ftp.mpi-sb.mpg.de/pub/outgoing(CGAL/CGAL-3.2.1.tar.gz  
tar xvzf CGAL-3.2.1.tar.gz  
cd CGAL-3.2.1  
.install_cgal --prefix=/usr/local/cgal --with-boost=n --without-autofind -ni /usr/bin/g++
```

Puisque la librairie routing.so que nous allons installée sera liéé par la suite à libCGAL.so, prenons déjà les devants. En effet, lors de l'appel de la librairie routing.so par le biais des fichiers SQL routing.sql et routing_postgis.sql, cette librairie a besoin de savoir où se trouve libCGAL.so

```
echo /usr/local/cgal/lib/i686_Linux-2.6_g++-4.0.3/ >> /etc/ld.so.conf  
ldconfig
```

A.2.4 PgRouting

Nous ferons tous simplement

```
wget http://www.postlbs.org/postlbs-cms/files/downloads/pgRouting-0.9.9.tgz  
tar xvzf pgRouting-0.9.9.tgz  
cd routing/  
export CGAL_MAKEFILE=/usr/local/cgal/make/makefile_i686_Linux-2.6_g++-4.0.3  
.configure --with-cgal=/usr/local/cgal --with-gaul=/usr/local/  
make  
make install
```

A.3 Installation sous Mac OS X

L'installation sous cet OS nécessite quelques subtilités, fruit de test du travail de Nicolas RIBOT, que je salue ici au passage ;).

NOTE : Les notes de cette section appartiennent à leur auteur respectif Nicolas RIBOT.

A.3.1 FINK

Il vous faut installer Fink. Puis depuis Fink, faites "Fink -> preferences -> Fink -> Check : Use unstable packages"

A.3.2 GAUL

Télécharger la dernière version à <http://prdownloads.sourceforge.net/gaul/gaul-devel-0.1849-0.tar.gz?download>. Puis faites

```
export CFLAGS=  
.configure --enable-ccoptim=no --enable-slang=no  
make  
sudo make install
```

A.3.3 BGL

Télécharger BOOST libs : www.boost.org, version 1.33.1. Exécutez ensuite les commandes suivantes

```
./configure  
make
```

Pensez à aller prendre un café ! Puis faites

```
make install
```

Un deuxième café ne serait pas de trop. Il faut maintenant compiler bjam. Pour cela rendez-vous dans tools/build/jam_src. Faites ensuite

```
./build.sh
```

Il faut créer un lien symbolique depuis tools/build/jam_src/bin.macosxx86/bjam dans un folder contenu dans le path

```
cd /usr/bin; ln -s <path_to_bjam>
```

Puis exécutez

```
bjam install
```

Celà copiera les headers dans /usr/share/boost_1_33_1/. Exécutez ensuite

```
bjam stage
```

qui copiera les lib dans un seul endroit.

A.3.4 CGAL

Télécharger le DMG pour Mac http://www.cgal.org/cgi-bin/cgal_download.pl Installez-le en faisant

```
CXX=/usr/bin/g++ ./install_cgal -ni --BOOST_INCL_DIR /usr/local/include/boost-1_33_1 \  
--without-autofind --BOOST_LIB_DIR /Users/nicolas/download/boost_1_33_1/stage -with-BOOST
```

A.3.5 PgRouting

Mettre à jour les chemins vers CGAL, BOOST, GAUL. créer un lien symbolique pour BOOST dans /usr/local :

```
ln -s boost-1_33_1/boost .
```

Créer un lien symbolique dans /usr/share vers CGAL :

```
cd /usr/share/; ln -s /Users/Shared/CGAL-3.2.1 CGAL
```

Revenir ensuite aux sources de pgRouting. Puis faites

```
./configure --with-boost=/usr/local --with-cgal=/Users/Shared/CGAL-3.2.1 --with-gaul=/usr/ ↵  
local  
make  
sudo make install
```

En cas de problème avec le make

```
--Unwind_Resume  
collect2: ld returned 1 exit status  
make: *** [librouting.0.0.so] Error
```

Éditer le `makefile.in` et remplacer la ligne

```
SHLIB_LINK += -lstdc++ $(TSP_LIBS) $(ALPHA_LIBS)
```

par

```
SHLIB_LINK += -lstdc++ $(TSP_LIBS) $(ALPHA_LIBS) -fexceptions
```

Ensuite

```
make clean  
make  
sudo make install
```

A.4 Chargement des fonctionnalités de PgRouting

Pour créer une base routing, ayant les fonctionnalités de PgRouting, il nous suffira de faire

```
createdb routing  
createlang plpgsql routing  
psql -d routing -f [chemin_d_acces_vers]lwpostgis.sql  
psql -d routing -f [chemin_d_acces_vers]spatial_ref_sys.sql  
psql -d routing -f [chemin_d_acces_vers]routing.sql  
psql -d routing -f [chemin_d_acces_vers]routing_postgis.sql
```

où [chemin_d_acces_vers] est le chemin d'accès vers le fichier attendu en fonction de votre OS (GNU/Linux ou Windows)

A.5 Fonctionnalité Shortest_Path() - Dijkstra

A.5.1 Dijkstra : module de routing pour PostgreSQL pour la recherche du plus cours chemin

Nous allons voir ici un exemple d'utilisation du module de routing de PgDijkstra

A.5.1.1 Description

L'algorithme de Dijkstra est connu en informatique (théorie des graphes) pour permettre de trouver le parcours du meilleur chemin. Porté initialement comme une extension dans le projet CARTOWEB <http://www.cartoweb.org> sous le nom de pgdijkstra, il permet en autre de trouver sur un réseau routier par exemple de trouver le meilleur chemin à parcourir pour se rendre d'un point à un autre



AVERTISSEMENT

Pour obtenir un descriptif intéressant de Dijkstra, merci de vous reporter au fichier README.routing de pgrouting bien utile pour une meilleure compréhension. Je ne m'attarderais pas ici sur les explications à fournir. Merci !. D'autres liens intéressants existent aussi comme par exemple celui de http://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra

A.5.1.2 Création d'un réseau routier

Ici je vais me permettre de partir des données fournies dans les tables small_roads, et great_roads du chapitre "Tutoriaux", de tronçonner les rues et de les mettre dans une tables roads dont voici le contenu

```
BEGIN;
CREATE TABLE "roads" (gid serial, "source_id" int4, "target_id" int4,"orientation" varchar) ← ;
SELECT AddGeometryColumn('','roads','the_geom',-1,'MULTILINESTRING',2);
-- Rue Figaro en tronçons
-- INSERT INTO roads values (0,1,2,'double sens',GeometryFromText('MULTILINESTRING((60 ←
-- 87.5,60 115))',-1));
INSERT INTO roads values (1,2,3,'double sens',GeometryFromText('MULTILINESTRING((60 115,85 ←
-- 120))',-1));
-- Rue Voltaire
-- INSERT INTO roads values (2,1,4,'sens inverse',GeometryFromText('MULTILINESTRING((60 ←
-- 87.5,60 25))',-1));
-- Rue Paul Valéry en tronçons
-- INSERT INTO roads values (3,5,6,'double sens',GeometryFromText('MULTILINESTRING((1 1,20 ←
-- 23))',-1));
INSERT INTO roads values (4,6,4,'double sens',GeometryFromText('MULTILINESTRING((20 23,60 ←
-- 25))',-1));
INSERT INTO roads values (5,4,7,'double sens',GeometryFromText('MULTILINESTRING((60 25,85 ←
-- 36))',-1));
-- Rue du Général de Gaulle en tronçons
-- INSERT INTO roads values (6,8,1,'double sens',GeometryFromText('MULTILINESTRING((1 87.5,60 ←
-- 87.5))',-1));
INSERT INTO roads values (7,1,9,'double sens',GeometryFromText('MULTILINESTRING((60 ←
-- 87.5,85 87.5))',-1));
INSERT INTO roads values (8,9,10,'double sens',GeometryFromText('MULTILINESTRING((85 ←
-- 87.5,150 87.5))',-1));
-- Rue Aristide Briand en tronçons
-- INSERT INTO roads values (9,11,3,'double sens',GeometryFromText('MULTILINESTRING((85 ←
-- 135,85 120))',-1));
INSERT INTO roads values (10,3,9,'double sens',GeometryFromText('MULTILINESTRING((85 ←
-- 120,85 87.5))',-1));
INSERT INTO roads values (11,9,7,'double sens',GeometryFromText('MULTILINESTRING((85 ←
-- 87.5,85 36))',-1));
INSERT INTO roads values (12,7,12,'double sens',GeometryFromText('MULTILINESTRING((85 ←
-- 36,85 1))',-1));
ALTER TABLE ONLY "roads" ADD CONSTRAINT "roads_pkey" PRIMARY KEY (gid);
ALTER TABLE roads ADD COLUMN edge_id integer;
END;
```

Voici le contenu de ma table

```
SELECT gid,source_id,target_id,orientation,astext(the_geom) from roads;
```

gid	source_id	target_id	orientation	astext
0	1	2	double sens	MULTILINESTRING((60 87.5,60 115))
1	2	3	double sens	MULTILINESTRING((60 115,85 120))
2	1	4	sens inverse	MULTILINESTRING((60 87.5,60 25))
3	5	6	double sens	MULTILINESTRING((1 1,20 23))
4	6	4	double sens	MULTILINESTRING((20 23,60 25))

```
5 |      4 |      7 | double sens | MULTILINESTRING((60 25,85 36))  
6 |      8 |      1 | double sens | MULTILINESTRING((1 87.5,60 87.5))  
7 |      1 |      9 | double sens | MULTILINESTRING((60 87.5,85 87.5))  
8 |      9 |     10 | double sens | MULTILINESTRING((85 87.5,150 87.5))  
9 |     11 |      3 | double sens | MULTILINESTRING((85 135,85 120))  
10 |      3 |      9 | double sens | MULTILINESTRING((85 120,85 87.5))  
11 |      9 |      7 | double sens | MULTILINESTRING((85 87.5,85 36))  
12 |      7 |     12 | double sens | MULTILINESTRING((85 36,85 1))  
(13 lignes)
```

On va maintenant utiliser la fonction `assign_vertex_id()` en lui appliquant la distance de 0.01

```
SELECT Assign_Vertex_Id('roads', 0.01);
```

On crée maintenant notre graphe grâce à la fonction `create_graph_tables()` :

```
SELECT Create_Graph_Tables('roads', 'int4');
```

qui créera aussitôt deux tables supplémentaires `roads_vertices` et `roads_edges` dont les contenus respectifs sont

```
testgis=# SELECT * from roads_vertices;  
id | geom_id  
---+-----  
1 |      1  
2 |      2  
3 |      3  
4 |      4  
5 |      5  
6 |      6  
7 |      7  
8 |      8  
9 |      9  
10 |     10  
11 |     11  
12 |     12  
(12 lignes)
```

et

```
testgis=# SELECT * from roads_edges;  
id | source | target | cost | reverse_cost  
---+-----+-----+-----+-----  
1 |      1 |      2 |      |  
2 |      2 |      3 |      |  
3 |      1 |      4 |      |  
4 |      5 |      6 |      |  
5 |      6 |      4 |      |  
6 |      4 |      7 |      |  
7 |      8 |      1 |      |  
8 |      1 |      9 |      |  
9 |      9 |     10 |      |  
10 |     11 |      3 |      |  
11 |      3 |      9 |      |  
12 |      9 |      7 |      |  
13 |      7 |     12 |      |  
(13 lignes)
```

Pour l'instant, cette dernière table a sa colonne des coûts (colone `cost`) et aussi des coûts inversés (colonne `reverse_cost`) vides. Nous allons les remplir.

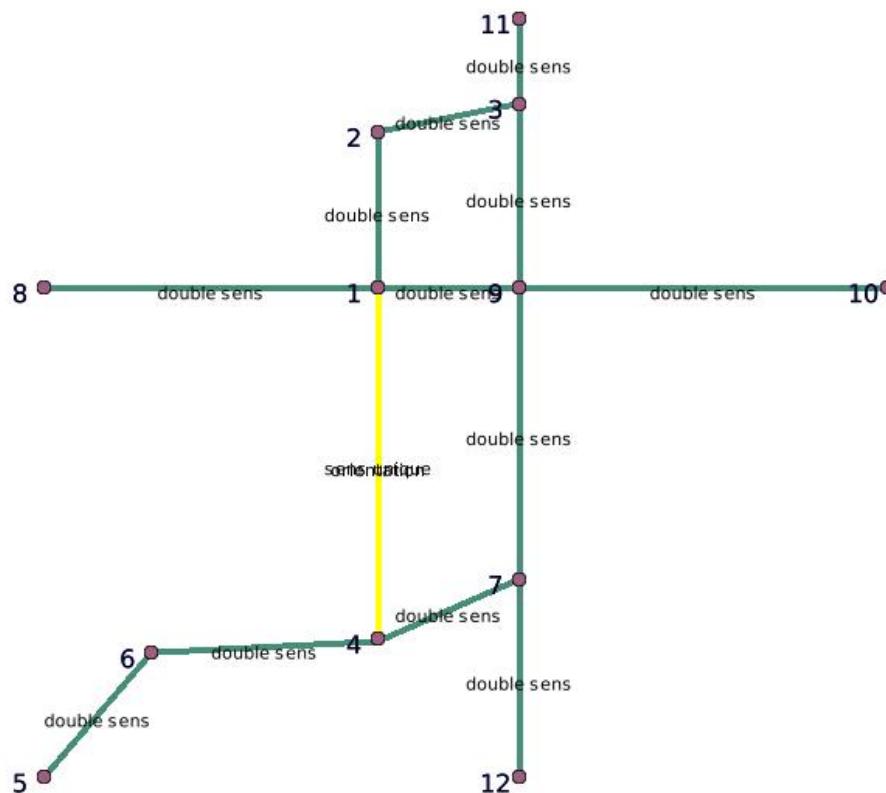


FIG. A.2 – Graphe orienté, la rue Voltaire est à sens unique (en jaune sur le graphe, du noeud 4 au noeud 1)

A.5.1.3 Calcul des coûts sur les noeuds du graphe pour rendre la Rue Voltaire à sens unique

Nous allons baser les coûts pour se rendre d'un noeud à un autre (voir figure plus loin dans le document) sur les distances qui les séparent. L'équipe de CartoWeb a ainsi prévu à cette fin, la fonction `update_cost_from_distance()` - repris par l'équipe de PgRouting -

```
SELECT update_cost_from_distance('roads');
```

Nous allons aussi estimé que notre graphe est orienté et que l'on peut aussi pour certains points hormis le 1 et le 4 aller d'un point à un autre.

```
UPDATE roads_edges SET reverse_cost=cost;
```

ce qui renverra donc

```
testgis=# SELECT * from roads_edges order by id;
 id | source | target |      cost       | reverse_cost
----+-----+-----+-----+-----+
  1 |     1 |     2 |      27.5 |      27.5
  2 |     2 |     3 | 25.4950975679639 | 25.4950975679639
  3 |     1 |     4 |      62.5 |      62.5
  4 |     5 |     6 | 29.0688837074973 | 29.0688837074973
  5 |     6 |     4 | 40.0499687890016 | 40.0499687890016
  6 |     4 |     7 | 27.3130005674953 | 27.3130005674953
```

7	8	1	59	59
8	1	9	25	25
9	9	10	65	65
10	11	3	15	15
11	3	9	32.5	32.5
12	9	7	51.5	51.5
13	7	12	35	35
(13 lignes)				

Pour que la Rue Voltaire soit à sens unique (de 4 --> 1), il faut lui appliquer un coût relativement élevé => cost = 1000000 : 4 ->1 sens inverse.

D'où la commande

```
UPDATE roads_edges SET cost=1000000 WHERE id=3
```

Ce qui donne pour la table roads_edges :

#	SELECT * from roads_edges ORDER BY 1;			
id	source	target	cost	reverse_cost
1	1	2	27.5	27.5
2	2	3	25.4950975679639	25.4950975679639
3	1	4	1000000	62.5
4	5	6	29.0688837074973	29.0688837074973
5	6	4	40.0499687890016	40.0499687890016
6	4	7	27.3130005674953	27.3130005674953
7	8	1	59	59
8	1	9	25	25
9	9	10	65	65
10	11	3	15	15
11	3	9	32.5	32.5
12	9	7	51.5	51.5
13	7	12	35	35
(13 lignes)				

A.5.1.4 Exemples d'utilisation

Nous utiliserons ici la fonction shortest_path(). Nous partirons du principe que le graphe est orienté et que nous utilisons aussi les couts inversés.

1. Pour aller de 1 à 4

```
SELECT * FROM shortest_path('SELECT id, source, target, cost, reverse_cost FROM ←
    roads_edges', 1,4,true,true);
```

qui nous renvoit

step	vertex_id	edge_id	cost
0	1	8	25
1	9	12	51.5
2	7	6	27.3130005674953
3	4	-1	0
(4 lignes)			

Selon la colonne vertex_id, le chemin effectué est bien celui attendu à savoir 1-->9-->7-->4

2. Pour aller de 4 à 1 :

```
SELECT * FROM shortest_path('SELECT id, source, target, cost, reverse_cost FROM ←
    roads_edges', 4,1,true,true);
```

qui nous renvoit

step	vertex_id	edge_id	cost
0	4	3	62.5
1	1	-1	0
(2 lignes)			

et le chemin est donc bien direct !

3. Pour aller de 6 à 8, avec MapServer l'affichage pourrait être le suivant

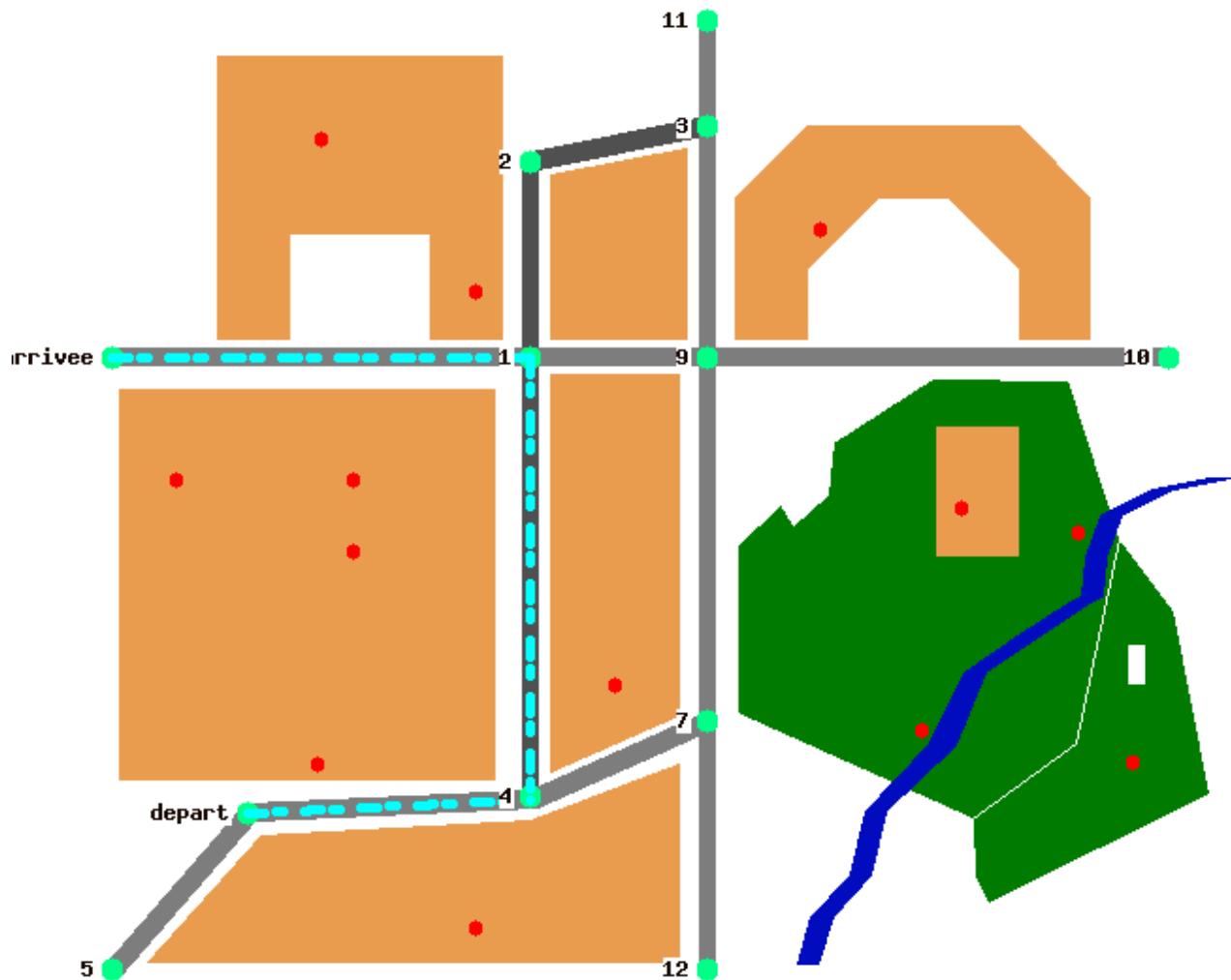


FIG. A.3 – Shortest_Path() : chemin le plus court pour aller de 6 à 8

4. Pour aller de 2 à 5, on peut aussi proposer l'affichage suivant

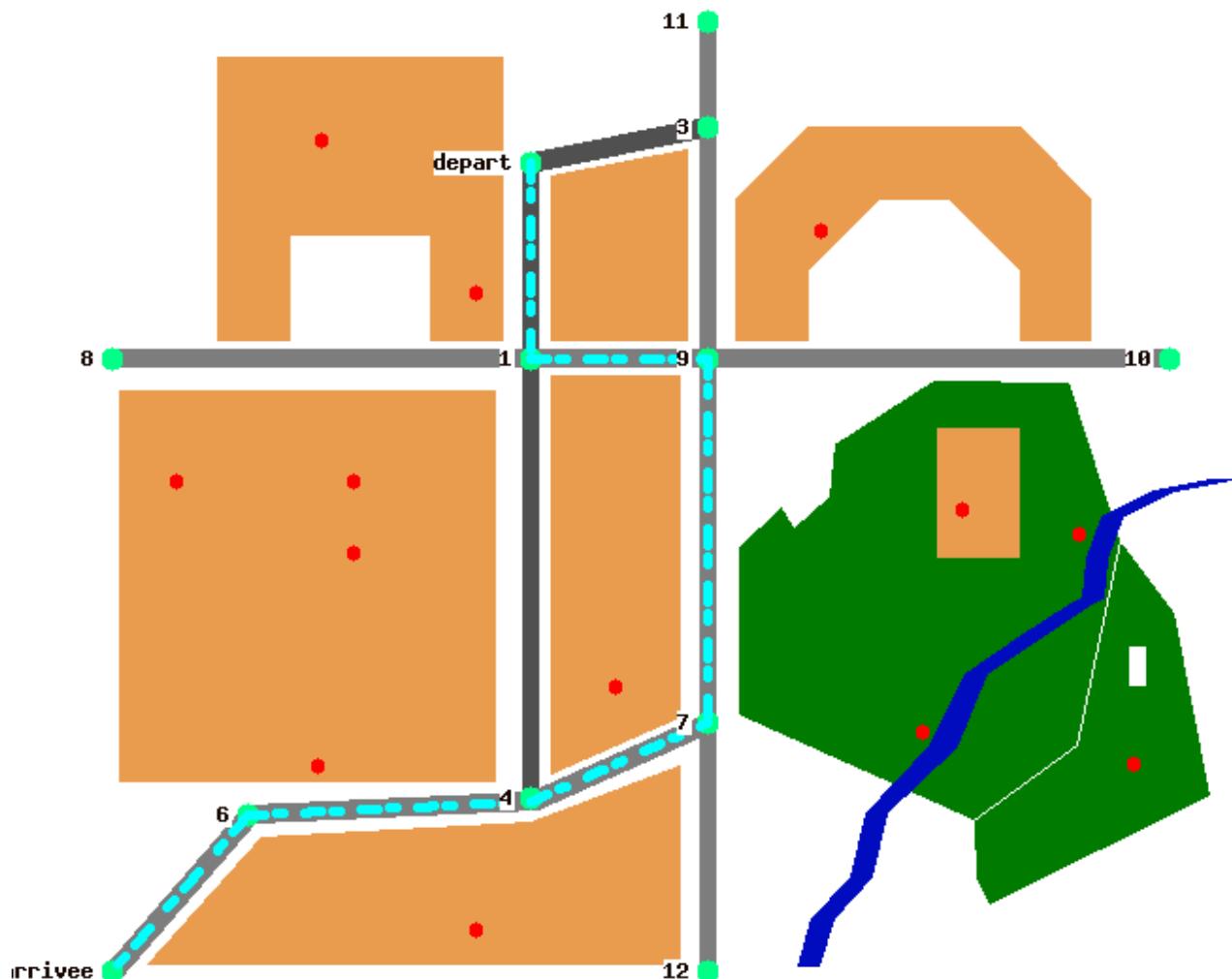


FIG. A.4 – Shortest_Path() : chemin le plus court pour aller de 2 à 5 (puisque la Rue Voltaire est à sens unique)

et avec dans la mapfile les trois layers suivants

```
# =====
# Layer pour afficher les points exceptés les points 2 et 5
# =====
LAYER
    CONNECTION "user=postgres host=192.168.0.8 dbname=testgis"
    CONNECTIONTYPE POSTGIS
    DATA "the_geom from (SELECT * from roads_vertices where id not in (2,5)) as
          foo using unique id using srid=-1"
    LABELITEM "id"
NAME "test"
    METADATA
        END
    SIZEUNITS PIXELS
    STATUS ON
    TOLERANCE 0
    TOLERANCEUNITS PIXELS
```

```
TYPE POINT
UNITS METERS
CLASS
LABEL
SIZE MEDIUM
TYPE BITMAP
BUFFER 0
COLOR 22 8 3
BACKGROUNDCOLOR 255 255 255
FORCE FALSE
MINDISTANCE -1
MINFEATURESIZE -1
OFFSET 1 1
PARTIALS TRUE
POSITION CL
END
METADATA
END
STYLE
ANGLE 360
COLOR 0 255 134
SIZE 12
SYMBOL "circle"
END
END
END

# =====
# Afficher les messages 'depart' et 'arrivee'
# =====
LAYER
CONNECTION "user=postgres host=192.168.0.8 dbname=testgis"
CONNECTIONTYPE POSTGIS
DATA "the_geom from (select id,case when id=2 then 'depart'::text else 'arrivee'::text end as nature,the_geom from roads_vertices where id in (2,5)) as foo using unique id using srid=-1"
LABELITEM "nature"
NAME "test"
METADATA
END
SIZEUNITS PIXELS
STATUS ON
TOLERANCE 0
TOLERANCEUNITS PIXELS
TYPE POINT
UNITS METERS
CLASS
LABEL
SIZE MEDIUM
TYPE BITMAP
BUFFER 0
COLOR 22 8 3
BACKGROUNDCOLOR 255 255 255
FORCE FALSE
MINDISTANCE -1
MINFEATURESIZE -1
OFFSET 1 1
PARTIALS TRUE
POSITION CL
END
METADATA
END
```

```
STYLE
  ANGLE 360
  COLOR 0 255 134
  SIZE 12
  SYMBOL "circle"
END
END
END
=====
# Tracé avec Shortest_Path()
=====
LAYER
  NAME "path"
  CONNECTIONTYPE postgis
    CONNECTION "user=postgres host=192.168.0.8 dbname=testgis"
    DATA "the_geom from (SELECT the_geom,gid from roads where edge_id IN (SELECT ←
      edge_id FROM shortest_path('SELECT * FROM roads_edges',2,5,true,true))) as
      foo using unique gid using srid=-1"
    TYPE LINE
    STATUS ON
    CLASS
      LABEL
        SIZE MEDIUM
        TYPE BITMAP
        BUFFER 0
        COLOR 22 8 3
        FORCE FALSE
        MINDISTANCE -1
        MINFEATURESIZE -1
        OFFSET 0 0
        PARTIALS TRUE
        POSITION CC
      END
      NAME "0"
      STYLE
        SYMBOL "dash"
        SIZE 5
        OUTLINECOLOR 0 255 255
      END
    END
  END
END
```

A.6 Importation d'un shapefile concernant des tronçons

Le shapefile que nous allons utiliser ici est fournit à l'adresse http://www.davidgis.fr/download/troncon_route.zip. Une fois le fichier dézippé, comme à l'accoutumée pour l'importer dans PostGIS,

```
shp2pgsql -DI troncon_route.shp troncon_route | psql routing
```

Les données géométriques du shapefile en question - visualisé dans QGIS- ressemblent à ca

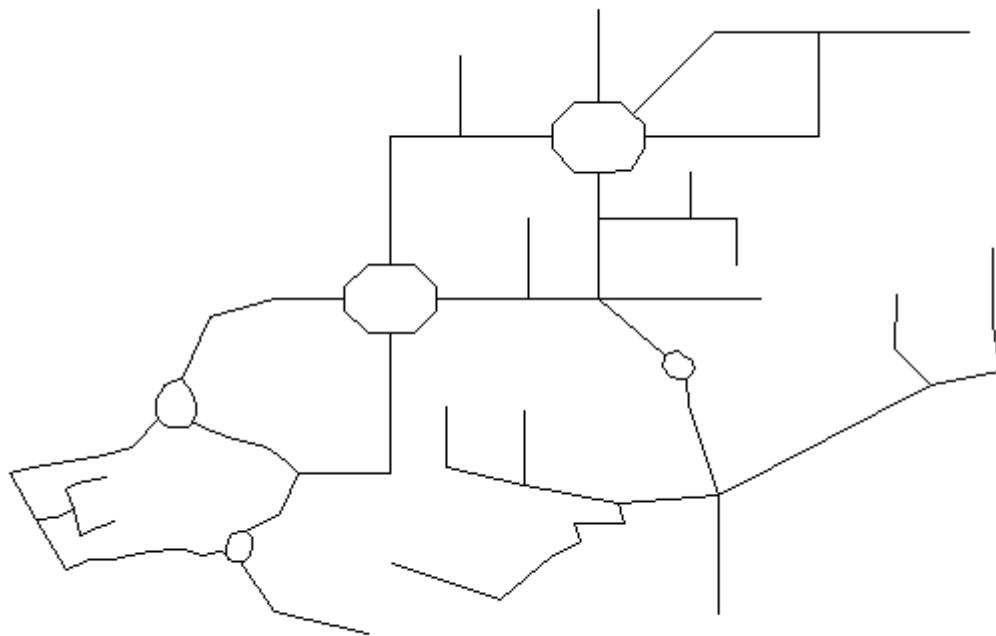


FIG. A.5 – Le réseau avec 5 rond-points.

Pour le moment, ma table ne contient rien de bien intéressant. Dans la colonne sens, sens="sens direct" désignent un tronçon d'un rond-point.

```
SELECT gid,sens,astext(the_geom) FROM troncon_route ORDER BY gid "
gid |      sens      |           astext
-----+-----+
 1 | double sens | MULTILINESTRING((1 0,5 0))
 2 | double sens | MULTILINESTRING((5 0,5 6))
 3 | double sens | MULTILINESTRING((0 7.5,3 7.5))
 4 | sens direct | MULTILINESTRING((3 7.5,3 7,4 6,5 6))
 5 | sens direct | MULTILINESTRING((5 6,6 6,7 7,7 7.5))
 6 | sens direct | MULTILINESTRING((7 7.5,7 8,6 9,5 9))
 7 | sens direct | MULTILINESTRING((5 9,4 9,3 8,3 7.5))
 8 | double sens | MULTILINESTRING((7 7.5,11 7.5))
 9 | double sens | MULTILINESTRING((11 7.5,11 11))
10 | double sens | MULTILINESTRING((11 7.5,14 7.5))
11 | double sens | MULTILINESTRING((14 7.5,21 7.5))
... | double sens | MULTILINESTRING((.....))
```

On aura compris que pour le moment, je ne dispose que des tronçons. Or pour la suite, il me faut définir les noeuds de mon réseau.

A.7 Obtention des noeuds du réseau

Les noeuds se présentent ainsi

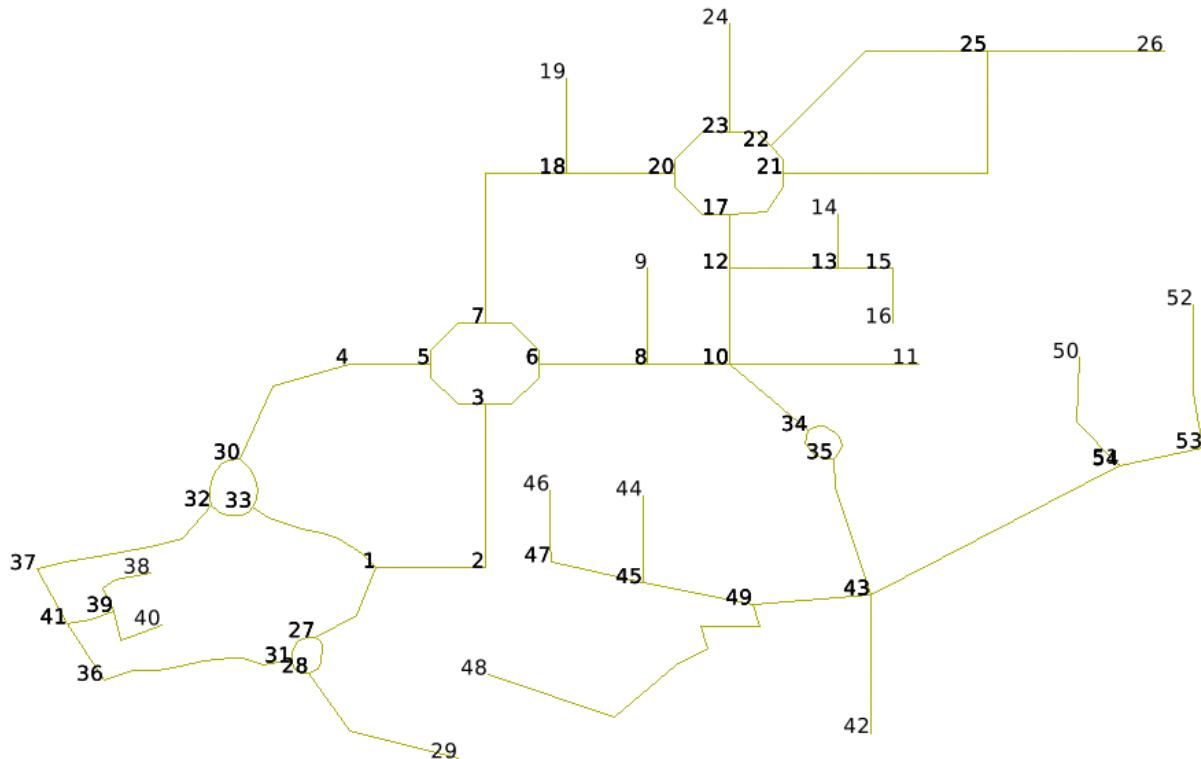


FIG. A.6 – Les noeuds du réseau

Ils ont été obtenus en utilisant les instructions SQL suivantes

```

BEGIN TRANSACTION;
-- ! ! LA LIGNE SUIVANTE EST A DECOMMENTER SI ON DOIT RECHARGER CE BLOC D'INSTRUCTIONS SQL ←
--SELECT drop_graph_tables('troncon_route');

/*
Ajouter les colonnes adéquates
*/
ALTER TABLE troncon_route ADD column source_id int4;
ALTER TABLE troncon_route ADD column target_id int4;
ALTER TABLE troncon_route ADD column edge_id int4;
/*
Mettre à jour le srid=1 sinon pgRouting gueule 8-
*/
SELECT updategeometrysrid('troncon_route','the_geom',-1);

SELECT assign_vertex_id('troncon_route',0.00001);
/*
Ok...Je crée mon graphe
*/
SELECT create_graph_tables('troncon_route', 'int4');

--SELECT UPDATE_cost_FROM_distance('troncon_route');
ALTER TABLE troncon_route_edges ADD column sens text;
ALTER TABLE troncon_route_edges ADD column x1 double precision;
ALTER TABLE troncon_route_edges ADD column y1 double precision;
ALTER TABLE troncon_route_edges ADD column x2 double precision;
ALTER TABLE troncon_route_edges ADD column y2 double precision;

```

```

ALTER TABLE troncon_route_edges ADD column edge_id int4;
/*
    Mise à jour des colonnes x1,y1,x2,y2 originaux par rapport aux données géométriques de ←
        la table troncon_route
    et mise à jour des colonnes sens et edge_id
*/
UPDATE troncon_route_edges SET cost=(SELECT length(the_geom) FROM troncon_route g WHERE g. ←
    edge_id=id GROUP BY id,g.the_geom);
UPDATE troncon_route_edges SET x1=(SELECT x(startpoint(the_geom)) FROM troncon_route g ←
    WHERE g.edge_id=id GROUP BY id,g.the_geom);
UPDATE troncon_route_edges SET y1=(SELECT y(startpoint(the_geom)) FROM troncon_route g ←
    WHERE g.edge_id=id GROUP BY id,g.the_geom);
UPDATE troncon_route_edges SET x2=(SELECT x(endpoint(the_geom)) FROM troncon_route g WHERE ←
    g.edge_id=id GROUP BY id,g.the_geom);
UPDATE troncon_route_edges SET y2=(SELECT y(endpoint(the_geom)) FROM troncon_route g WHERE ←
    g.edge_id=id GROUP BY id,g.the_geom);
UPDATE troncon_route_edges SET edge_id=(SELECT edge_id FROM troncon_route g WHERE g.edge_id ←
    =id GROUP BY id,g.edge_id);
UPDATE troncon_route_edges SET sens=(SELECT sens::text FROM troncon_route g WHERE g.edge_id ←
    =id GROUP BY id,g.sens);
SELECT AddGeometryColumn('troncon_route_edges', 'the_geom', -1, 'MULTILINESTRING', 2 );
UPDATE troncon_route_edges SET the_geom=(SELECT the_geom FROM troncon_route g WHERE g. ←
    edge_id=id GROUP BY id,g.the_geom);
/*
    Tout ce qui est à double sens je le garde
*/
UPDATE troncon_route_edges SET reverse_cost=cost;
/*
    Paramétriser le coût des tronçons à sens unique
*/
UPDATE troncon_route_edges SET reverse_cost=1000000 WHERE sens='sens direct';

END TRANSACTION;
VACUUM FULL ANALYZE ;

```

J'ai donc maintenant une table troncon_route_edges avec le contenu suivant.

```

SELECT id,sens,astext(the_geom),x1,y1,x2,y2,source,target,edge_id,cost,reverse_cost FROM ←
    troncon_route_edges ORDER BY id LIMIT 10

me renvoit
   id |      sens      |           astext           |   x1 |   y1 |   x2 |   y2 | source | ←
     target | edge_id |          cost          | reverse_cost
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
 1 | double sens | MULTILINESTRING((1 0,5 0)) | 1 | 0 | 5 | 0 | 1 | ←
 2 |      1 | 4 | 4 |
 2 | double sens | MULTILINESTRING((5 0,5 6)) | 5 | 0 | 5 | 6 | 2 | ←
 3 |      2 | 6 | 6 |
 3 | double sens | MULTILINESTRING((0 7.5,3 7.5)) | 0 | 7.5 | 3 | 7.5 | 4 | ←
 5 |      3 | 3 | 3 |
 4 | sens direct | MULTILINESTRING((3 7.5,3 7.4 6,5 6)) | 3 | 7.5 | 5 | 6 | 5 | ←
 3 |      4 | 2.91421356237309 | 1000000 |
 5 | sens direct | MULTILINESTRING((5 6,6 6,7 7,7 7.5)) | 5 | 6 | 7 | 7.5 | 3 | ←
 6 |      5 | 2.91421356237309 | 1000000 |
 6 | sens direct | MULTILINESTRING((7 7.5,7 8,6 9,5 9)) | 7 | 7.5 | 5 | 9 | 6 | ←
 7 |      6 | 2.91421356237309 | 1000000 |
 7 | sens direct | MULTILINESTRING((5 9,4 9,3 8,3 7.5)) | 5 | 9 | 3 | 7.5 | 7 | ←
 5 |      7 | 2.91421356237309 | 1000000 |
 8 | double sens | MULTILINESTRING((7 7.5,11 7.5)) | 7 | 7.5 | 11 | 7.5 | 6 | ←
 8 |      8 | 4 | 4 |
 9 | double sens | MULTILINESTRING((11 7.5,11 11)) | 11 | 7.5 | 11 | 11 | 8 | ←

```

```
      9 |      9 |      3.5 |      3.5
10 | double sens | MULTILINESTRING((11 7.5,14 7.5)) | 11 | 7.5 | 14 | 7.5 |
    10 |      10 |          3 |          3
(10 lignes)
```

A.8 Fonctionnalité shortest_path_astar()

Intéressons-nous en premier lieu à cette fonctionnalité du plus court chemin

A.8.1 Exemple pour les noeuds 38 et 48.

Je vais ici créer deux tables aller et retour

A.8.1.1 Pour l'aller (source=38 et target=48)

je fais

```
BEGIN TRANSACTION;

CREATE TABLE aller(gid int4) WITH oids;

SELECT AddGeometryColumn( 'aller', 'the_geom', -1, 'MULTILINESTRING', 2 );

INSERT INTO aller(the_geom)
(
  SELECT the_geom FROM troncon_route_edges WHERE edge_id IN
    (SELECT edge_id FROM shortest_path_astar('SELECT id,source,target,cost,
      reverse_cost, x1,y1,x2,y2 FROM troncon_route_edges',38,48,false,true)
   )
);

END TRANSACTION;
```

ce qui me donnera le résultat suivant

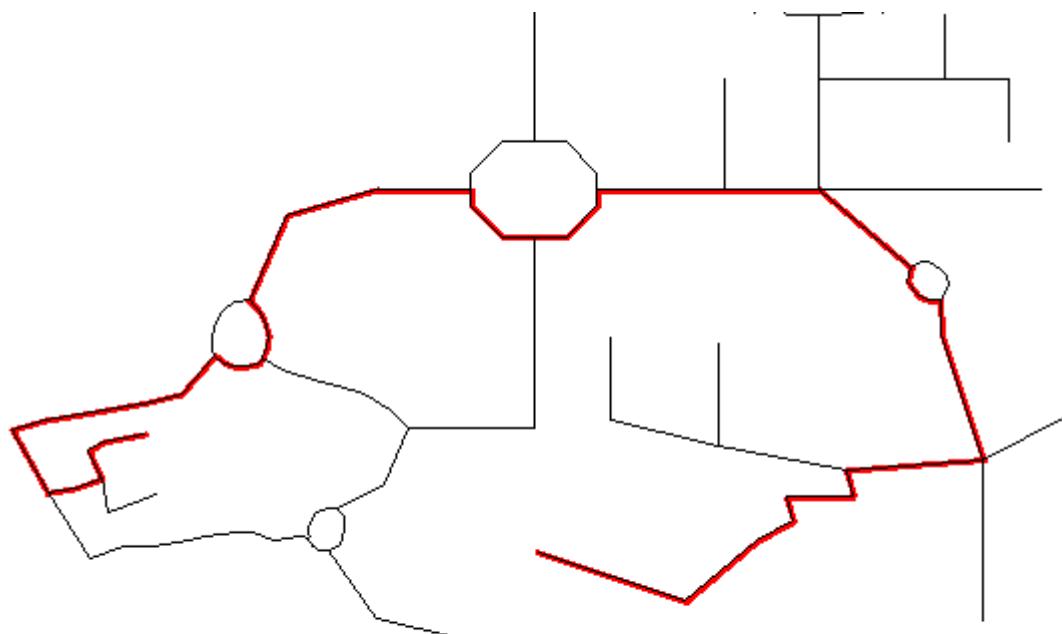


FIG. A.7 – Parcours à l'aller

A.8.1.2 Pour le retour (source=48 et target=38)

Je fais

```
BEGIN TRANSACTION;

CREATE TABLE retour(gid int4) WITH oids;
SELECT AddGeometryColumn( 'retour', 'the_geom', -1, 'MULTILINESTRING', 2 );
INSERT INTO retour(the_geom)
(
  SELECT the_geom FROM troncon_route_edges WHERE edge_id IN
    (SELECT edge_id FROM shortest_path_astar('SELECT id,source,target,cost,
      reverse_cost, x1,y1,x2,y2 FROM troncon_route_edges',48,38,false,true)
    )
);
END TRANSACTION;
```

Au niveau visualisation, on obtient

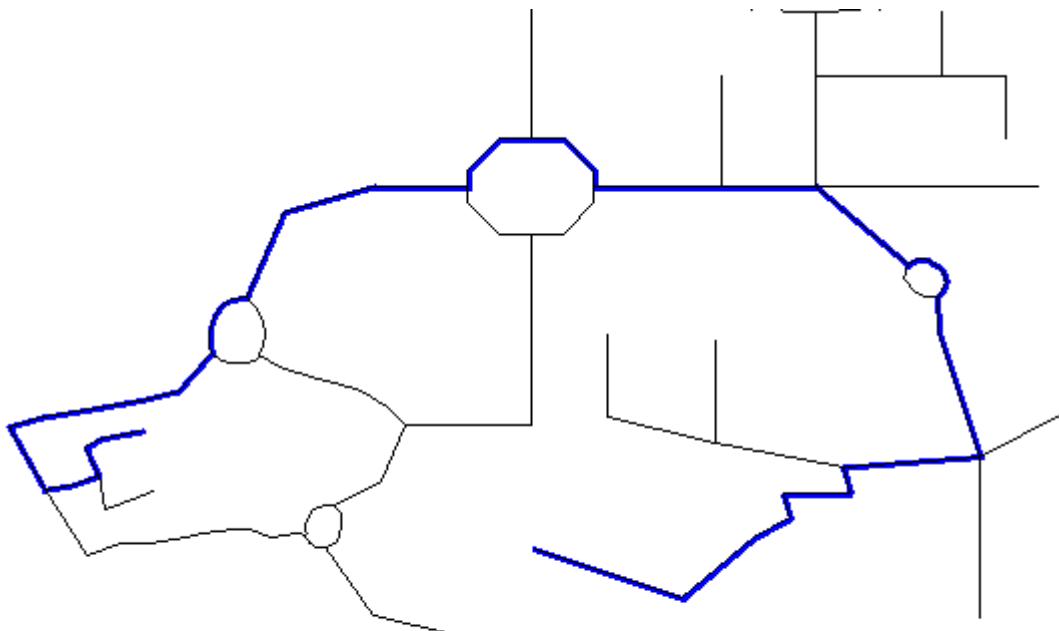


FIG. A.8 – Parcours au retour

A.8.2 Démo en ligne avec MapServer

Vous trouverez une démo en ligne de ce qui a été suggéré précédemment à l'adresse <http://www.davidgis.fr/routing/>

A.8.3 Tester soi-même la démo avec MapServer.

En pré-requis, il faut que vous ayez PhpMapScript et l'extension PostgreSQL pour Php d'activé. Vous trouverez un répertoire de ce tutoriel à http://www.davidgis.fr/download/pgRouting_demo.zip. Une fois le fichier téléchargé et dézippé, il suffit dans la mapfile mapfile/map.map d'adapter la portion de la partie WEB à votre configuration

```
WEB
  IMAGEPATH "/var/www/tutorial/routing/tmp/" <-- A adapter
  IMAGEURL "/tutorial/routing/tmp/" <- A adapter
  METADATA
  END
  QUERYFORMAT text/html
END
```

et pour chaque layer, il faudra adapter la partie

```
"user=postgres dbname=routing password=empr888 host=localhost"
```

à votre propre configuration de PostgreSQL. Le shapefile troncon_route.shp est dans le sous-répertoire data

A.8.4 Tester sur un jeu de données réelles : jeu de tests GEOROUTE IGN

Sur le site de l'IGN, on peut avoir un échantillon des données gratuits. En remplissant un simple formulaire, on peut recevoir un CD contenant un jeu de test en zone agglomération sur la ville d'Orléans (45). Les données sont fournies au format MapInfo

A.8.4.1 Conversion du fichier MapInfo vers PostGIS

A la racine du CD, il faut se rendre à DONNEES/GEOROUTE/NAVIGATION/RESEAU_ROUTIER. Je vais utiliser ogr2ogr pour convertir une première fois le fichier en shapefile, puis en table PostGIS ensuite

```
cd /media/cdrom/DONNEES/GEOROUTE/NAVIGATION/RESEAU_ROUTIER
cp TRONCON_ROUTE.* ~/
cd
ogr2ogr -f "ESRI Shapefile" TRONCON_ROUTE.SHP TRONCON_ROUTE.TAB
shp2pgsql -dDI TRONCON_ROUTE.SHP troncon_route|iconv -f LATIN1 -t UTF-8|psql routing
```

A.8.4.2 Instructions SQL

Je vais effectuer les requêtes SQL suivantes

```
BEGIN TRANSACTION;

SELECT drop_graph_tables('troncon_route');

/*
   Ajouter les colonnes adéquates
*/
ALTER TABLE troncon_route ADD column source_id int4;
ALTER TABLE troncon_route ADD column target_id int4;
ALTER TABLE troncon_route ADD column edge_id int4;
/*
   Mettre à jour le srid=1 sinon pgRouting gueule 8-
*/
SELECT updategeometrysrid('troncon_route','the_geom',-1);
UPDATE troncon_route SET the_geom=reverse(the_geom),sens='Sens direct' WHERE sens='Sens ←
    inverse';
SELECT assign_vertex_id('troncon_route',0.00001);
/*
   Ok...Je crée mon graphe
*/
SELECT create_graph_tables('troncon_route', 'int4');

--SELECT UPDATE_cost_FROM_distance('troncon_route');
ALTER TABLE troncon_route_edges ADD column sens text;
ALTER TABLE troncon_route_edges ADD column x1 double precision;
ALTER TABLE troncon_route_edges ADD column y1 double precision;
ALTER TABLE troncon_route_edges ADD column x2 double precision;
ALTER TABLE troncon_route_edges ADD column y2 double precision;
ALTER TABLE troncon_route_edges ADD column edge_id int4;
/*
   Mise à jour des colonnes x1,y1,x2,y2 originaux par rapport aux données géométriques de ←
       la table troncon_route
   et mise à jour des colonnes sens et edge_id
*/
SELECT update_cost_from_distance('troncon_route');

UPDATE troncon_route_edges SET x1=(SELECT x(startpoint(the_geom)) FROM troncon_route g ←
    WHERE g.edge_id=id GROUP BY id,g.the_geom LIMIT 1);

UPDATE troncon_route_edges SET y1=(SELECT y(startpoint(the_geom)) FROM troncon_route g ←
    WHERE g.edge_id=id GROUP BY id,g.the_geom LIMIT 1);

UPDATE troncon_route_edges SET x2=(SELECT x(endpoint(the_geom)) FROM troncon_route g WHERE ←
    g.edge_id=id GROUP BY id,g.the_geom LIMIT 1);
```

```
UPDATE troncon_route_edges SET y2=(SELECT y(endpoint(the_geom)) FROM troncon_route g WHERE g.edge_id=id GROUP BY id,g.the_geom LIMIT 1);

UPDATE troncon_route_edges SET edge_id=(SELECT edge_id FROM troncon_route g WHERE g.edge_id=id GROUP BY id,g.edge_id LIMIT 1);

UPDATE troncon_route_edges SET sens=(SELECT sens::text FROM troncon_route g WHERE g.edge_id=id GROUP BY id,g.sens LIMIT 1);

SELECT AddGeometryColumn( 'troncon_route_edges', 'the_geom', -1, 'MULTILINESTRING', 2 );

UPDATE troncon_route_edges SET the_geom=(SELECT the_geom FROM troncon_route g WHERE g.edge_id=id GROUP BY id,g.the_geom LIMIT 1);
/*
   Tout ce qui est à double sens je le garde
*/
UPDATE troncon_route_edges SET reverse_cost=cost;
/*
   Paramétriser le coût des tronçons à sens unique
*/
UPDATE troncon_route_edges SET reverse_cost=1000000 WHERE sens='Sens direct';

END TRANSACTION;
VACUUM FULL ANALYZE ;
```

J'exporte ensuite la table troncon_route_edges en Shapefile en faisant

```
pgsql2shp -h localhost -u postgres -f troncon_route_edges.shp routing troncon_route_edges
```

A.8.4.3 Exemple

Depuis QGIS, je charge le shapefile troncon_route_edges.shp en deux fois : un pour avoir les sources (en fonction de x1,y1) et l'autre pour les target (en fonction de x2,y2). Pour celà, je joue sur les propriétés d'affichage que propose QGIS : clic-droit->propriété, onglet "Etiquette" etc...Une fois tout celà fait, j'ai par exemple l'affichage suivant

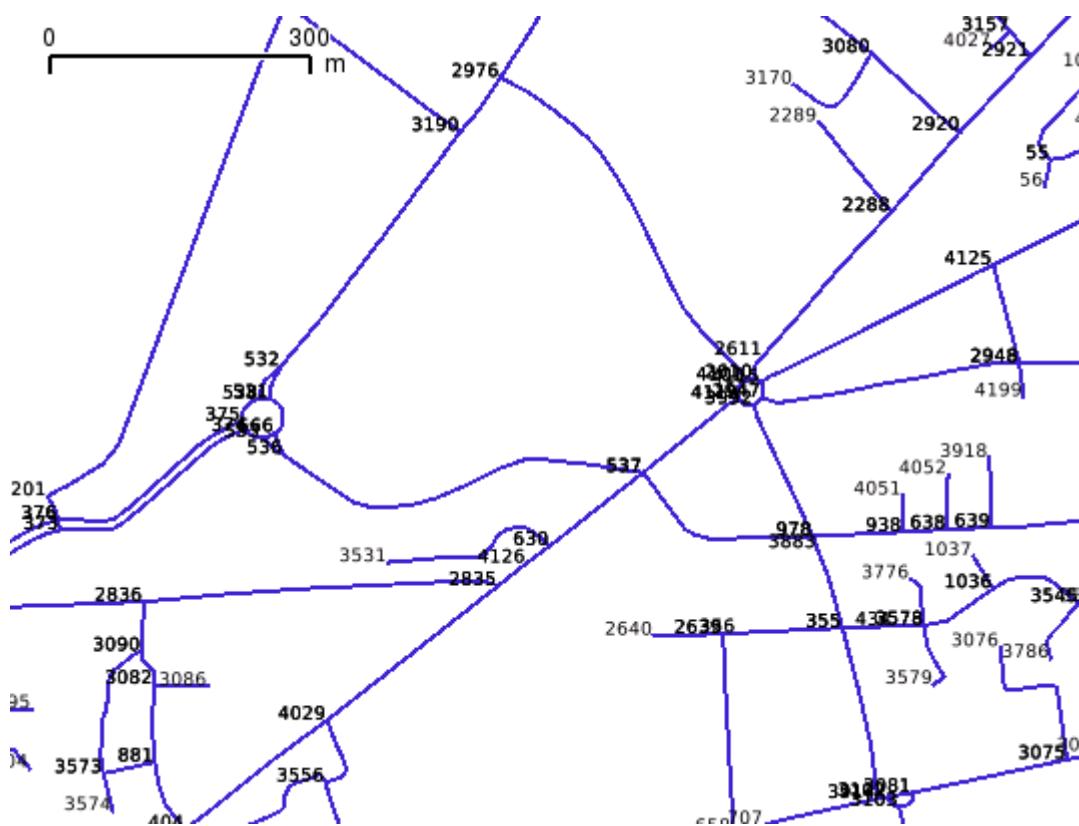


FIG. A.9 – Une portion des noeuds de mon réseau (sources+targets)

Je choisis ensuite un source et un target. Sur le CD, en me rendant à DONNEES/GEOROUTE_Raster/LZW/5K, je charge le fond gr_5k_ITagglo_v1.tif dans QGIS. Je crée ensuite les tables aller et retour comme fait précédemment. J'ai par exemple l'affichage suivant



FIG. A.10 – Parcours à l'aller



FIG. A.11 – Parcours au retour

A.9 Fonctionnalité TSP()

TSP signifie Traveling Salesman Problem, désignant ainsi le problème du voyageur du commerce. Pour une définition plus profane, il s'agit du problème du père Noë pour sa distribution de cadeaux, ou de l'agent commercial qui doit visiter des clients dans un trajet professionnel en essayant d'optimiser son parcours - un peu de théorie des graphes, celà ne fait pas de mal que les puristes de la discipline m'excusent -. Le graphe que j'utilise ici est toujours le même. Nous conviendrons donc même si l'exemple qui sera fourni n'est pas des plus convaincants. Le but est en fade réussir à optimiser le parcours qui permet en passant par différents points attendus. Pour ma part ici, je vous renvoie à la documentation réalisée par Gérald FENOY.

A.9.1 Exemple

Sur mon réseau, je prendrais source=7, comme point de départ. Je souhaite passer par les noeuds fournis ici dans un ordre quelconque 25,41,28,42.tsp() va me permettre de savoir quel est le meilleur ordre de parcours pour ces noeuds. J'execuerais donc la requête suivante

```
routing=# SELECT * FROM tsp('SELECT DISTINCT source AS source_id, x1 AS x, y1 AS y FROM troncon_route_edges WHERE source IN (25,7,41,28,42)', '25,7,41,28,42', 7);
```

qui me renverra

vertex_id	edge_id	cost
7	8	5.76058619309876e-269
41	8	5.76063240607483e-269
28	8	5.7606786190509e-269
42	8	6.83337886510596e-316
25	32	2.13684976166112e-311

(5 lignes)

Les colonnes edge_id et cost n'ont pas d'importance. Seul compte la colonne vertex_id. Selon l'ordre énuméré des valeurs des lignes de la colonne de vertex_id, il me faudra donc pour optimiser mon parcours aller de 7 à 41, de 41 à 28, de 28 à 42 etc...

Et donc de proche en proche, il me suffira par exemple d'utiliser la fonctionnalité shortest_path_astar() pour reconstruire mon parcours ou d'utiliser la fonctionnalité tsp_astar_as_geometry_internal_id_directed(). Le parcours sera donc le suivant

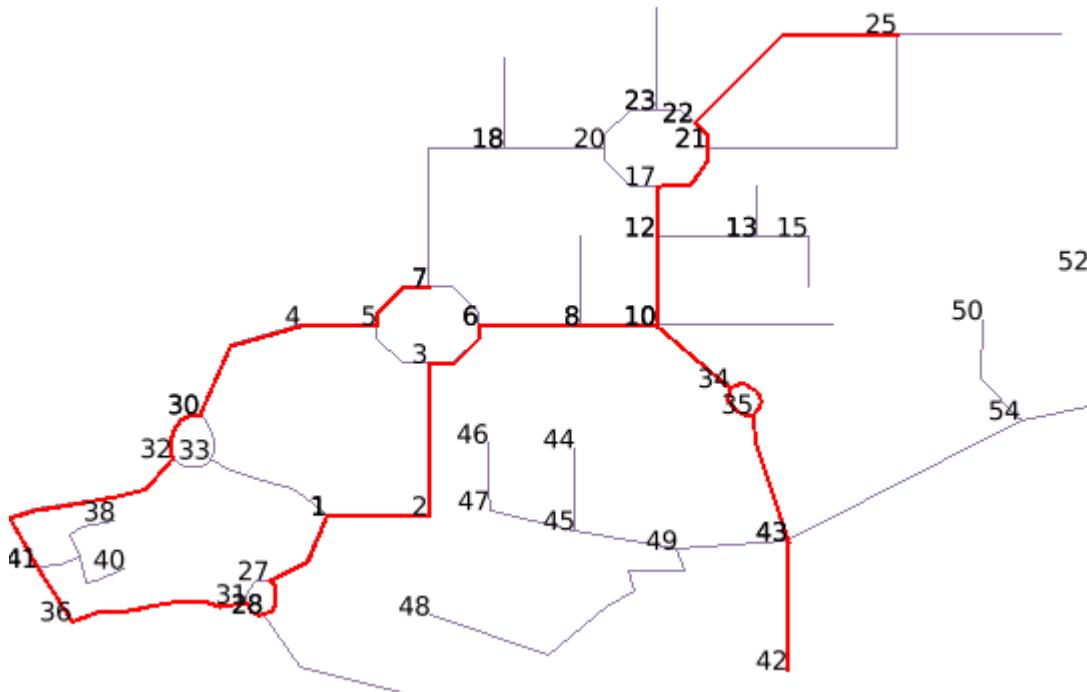


FIG. A.12 – fonctionnalité TSP() pour les noeuds 7,41,28,42 et 25

Pour le moment, nous allons nous limiter à la première solution proposée, à savoir utiliser shortest_path_astar() de proche en proche. Celà nous permettra de voir les limites d'une telle solution. Il en découlera l'utilisation plus intéressante de tsp_astar_as_geometry_internal_id_directed()

A.9.2 Programme en C pour les appels successifs à shortest_path_astar()

En se basant sur l'exemple précédent la seule tâche qui nous resterait à effectuer serait de pouvoir automatiser les appels successifs à shortest_path_astar(). Pour ce faire, nous allons utiliser un programme `tsp_trajet`. Ce programme est écrit en C en se basant sur l'interface de programmation libpq de PostgreSQL. Il prendra comme paramètre la connexion à notre base, la liste des points à parcourir et le point de départ. L'utilisation de `tsp_trajet` sera par exemple

```
tsp_trajet "dbname=routing user=postgres" "25,7,41,28,42,19" 7
```

Pour la simplification de l'exposé ici, le principe le plus simple sera de stocker dans une table trajet -que le programme se chargera d'alimenter - les divers parcours rentrés par shortest_path_astar(). Notre table aura donc la structure suivante

```
#SELECT source,target,astext(the_geom) FROM trajet
source | target | the_geom
-----+-----+-----
 7 |     41 | MULTILINESTRING((...))
 41 |     28 | MULTILINESTRING((...))
 28 |     42 | MULTILINESTRING((...))
 42 |     25 | MULTILINESTRING((...))
(4 lignes)
```

Par exemple, l'appel de shortest_path_astar() pour source=41 et target=28, correspondra à la requête

```
INSERT INTO trajet (source,target,the_geom) values
(41,28,(SELECT geomunion(the_geom) FROM troncon_route_edges WHERE edge_id IN
(SELECT edge_id FROM shortest_path_astar('SELECT id,source,target,cost,reverse_cost,
x1,y1,x2,y2 FROM troncon_route_edges',41,28,false,true))))
```

Le programme se chargera à chaque utilisation de recréer la table trajet. On notera ici l'emploi de GeomUnion() qui permet pour chaque couple

A.9.2.1 Le programme

On pourra proposer par exemple le programme suivant que l'on peut s'amuser à optimiser

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <libpq-fe.h>

int main(int argc, char * argv[])
{
    PGresult *result,*result2;
    PGconn *conn;
    char query1[900]=
"CREATE OR REPLACE FUNCTION drop_geometrystable_if_exists(text) RETURNS text AS $$ \
DECLARE rec record;\n
BEGIN \
IF NULLVALUE($1) THEN \
    RETURN 'ATTENTION: Table non trouvée'; \
ELSE \
    SELECT INTO rec tablename FROM pg_tables WHERE tablename = quote_ident($1);\
    IF FOUND THEN\
        EXECUTE 'SELECT DropGeometryTable('||quote_literal('public')||','||quote_literal($1) ←
        ||')';\
        RETURN 'Effacement de la table ...OK';\
    END IF; \
END IF; \
RETURN 'ATTENTION: Table non trouvée';END;$$ \
LANGUAGE plpgsql; \
SELECT drop_geometrystable_if_exists('trajet'); \
CREATE TABLE trajet(gid int4,source int,target int) WITH oids; \
SELECT AddGeometryColumn( 'trajet', 'the_geom', -1, 'MULTILINESTRING', 2 ); \
SELECT vertex_id::int from tsp('select distinct source as source_id\
, x1::double precision as x, y1::double precision as y from \
troncon_route_edges where source in (";

if (argc != 4)
{
    printf("Usage:%s \"dbname=XXX user=XXX host=XXX\ password=XXX\" \"liste de points\" \
startpoint\n\nExemple %s \"dbname=routing user=postgres\" \"43,30,18,41,1\" 30\n\n", argv ←
[0],argv[0]);
    return EXIT_SUCCESS;
}

conn = PQconnectdb(argv[1]);

if(PQstatus(conn) == CONNECTION_OK) {
    printf("Connexion OK\n");
    strcat(query1," ");
    strcat(query1,argv[2]);
    strcat(query1,"' '/'");
```

```
strcat(query1, argv[2]);
strcat(query1, "'");
strcat(query1, argv[3]);
strcat(query1, ")");
//printf("%s\n", query1);

result = PQexec(conn, query1);

switch(PQresultStatus(result))
{
    case PGRES_TUPLES_OK:
        {
            int r, n;
            int nrows = PQntuples(result);
            int nfields = PQnfields(result);
            printf("\n-----\n");
            printf(" %d insertions à effectuer dans la table projet", nrows);
            printf("\n-----\n");
            for(r = 1; r < nrows; r++)
            {
                for(n = 0; n < nfields; n++)
                {
                    char query2[800] = "INSERT INTO trajet (source,target,the_geom) values (";
                    strcat(query2, PQgetvalue(result, r-1, n));
                    strcat(query2, ",");
                    strcat(query2, PQgetvalue(result, r, n));
                    strcat(query2, "(");
                    strcat(query2, "SELECT geomunion(the_geom) FROM troncon_route_edges \
WHERE edge_id IN (SELECT edge_id FROM shortest_path_astar('SELECT id,source,target,cost, \
reverse_cost,\
x1,y1,x2,y2 FROM troncon_route_edges',"));
                    strcat(query2, PQgetvalue(result, r-1, n));
                    strcat(query2, ",");
                    strcat(query2, PQgetvalue(result, r, n));
                    strcat(query2, ",false,true)))");
                    //printf("%s\n", query2);
                    printf(" %s --> %s \n", PQgetvalue(result, r-1, n), PQgetvalue(result, r, n));
                    result2 = PQexec(conn, query2);
                }
            }
        }
    PQclear(result); PQclear(result2);
}
else
    printf("Echec de connexion: %s\n", PQerrorMessage(conn));

PQfinish(conn);
return EXIT_SUCCESS;
}
```

Pour la compilation, on pourra se baser sur le Makefile suivant à adapter selon vos besoins

```
INC='pg_config --includedir '
LIB='pg_config --libdir '

CFLAGS=-I$(INC)
LDLIBS=-L$(LIB) -lpq

ALL=tsp_trajet
```

```
all: $(ALL)

clean:
    @rm -f *.o *~ $(ALL)
```

A.9.3 Limites du programme

On l'aura compris de suite, ce genre de programme ne peut être utilisé dans un milieu de production et la gestion de la table trajet serait alors impossible dans un mode multi-utilisateur. Or PgRouting propose la fonctionnalité `tsp_astar_as_geometry_internal_id_directed()` qui permet de calculer à la volée les divers portions de parcours attendus.

A.10 Fonctionnalités `shortest_path_astar_as_geometry_internal_id_directed()` et `tsp_astar_as_geometry_internal_id_directed()`

Je profite ici de proposer deux exemples sur ces deux fonctionnalités. Avant de commencer, nous allons importer un nouveau jeu de données issus de NavTeq. Nous en profiterons pour voir nos fonctionnalités grâce à ce jeu.

A.10.1 Importation d'un jeu de données NavTeq

NavTeq on ne le présente plus. Tout le monde connaît. La société ADCI - <http://www.adci.com> - propose sur son site un jeu de données test sur le réseau routier à Washington DC. Pour se faire, il faut se rendre à <http://www.adci.com/products/navteq/index.html>. C'est surtout le jeu "NAVTEQ Premium - for routing applications" qu'il faut télécharger. Une simple inscription sur le site permet par la suite de pouvoir télécharger les données. Une fois téléchargées et décompressées, ce sera surtout le shapefile `streets.shp` que nous allons tester ici. Les données sont fournis dans le système GPS Mondial WGS 84

Le petit + c'est aussi la documentation qui accompagne les shapefiles. On y apprend comment NavTeq présente sa topologie, ses spécificités etc...Pour convertir le fichier en kml, rien de plus simple que

```
ogr2ogr -f KML ~/streets.kml streets.shp
```

Une petite importation du fichier `streets.kml` dans GoogleEarth nous renverra les screenshots suivantes

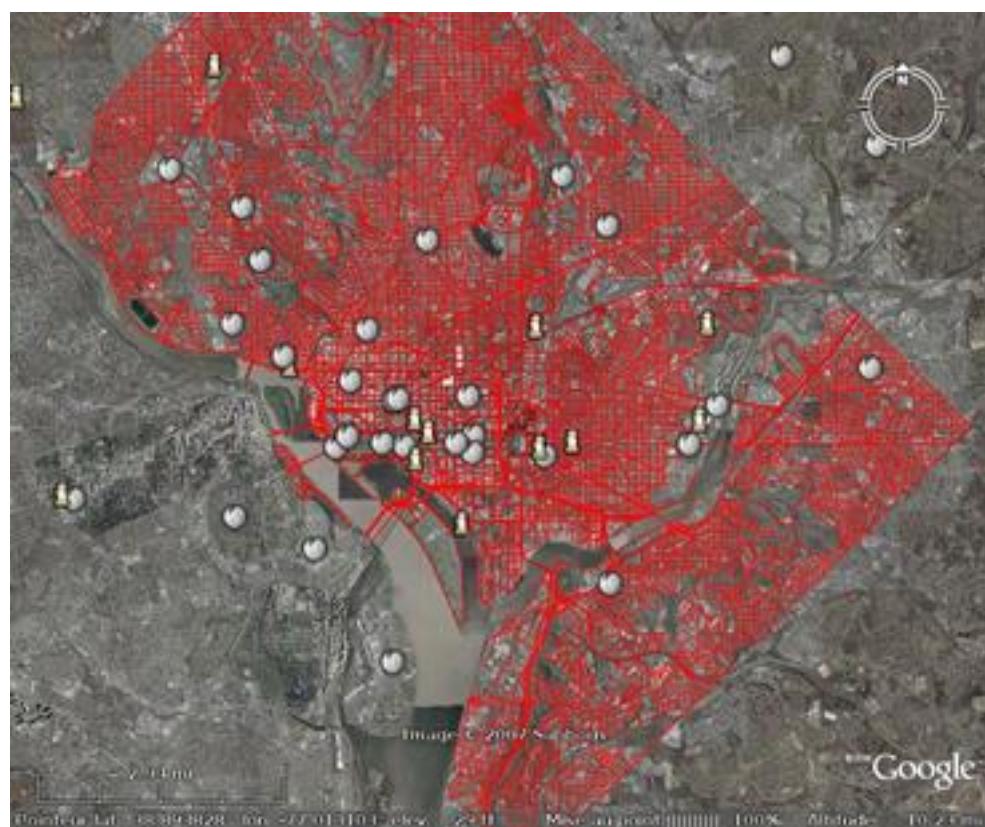


FIG. A.13 – GoogleEarth : réseau routier à Washington DC

ainsi que



FIG. A.14 – GoogleEarth : réseau routier à Washington DC

Pour l'importation des données, nous ferons tout simplement

```
shp2pgsql -DI streets.shp streets | psql routing
```

A.10.2 Noeuds du réseau et direction pour le routage

NavTeq propose déjà sa propre topologie. En vertu de la documentation qui accompagne les données, je pourrais bien me servir des champs ref_in_id et de nref_in_id pour les associer respectivement à mes champs source et target pour pgrouting. Or pour rester en conformité avec ma documentation, je préfère reconstruire les noeuds. De plus pour le sens de parcours des tronçons, il faut savoir que ma table streets contient le champ dir_travel (Direction Travel) qui a les spécificités suivantes

1. dir_travel='B' c'est une tronçon à double sens ;
2. dir_travel='F' c'est une tronçon à sens unique. Le sens de parcours de l'arc est le sens direct par rapport au noeud de référence (le point de départ est le premier point de mon arc) ;
3. dir_travel='T' c'est une tronçon à sens unique. Le sens de parcours de l'arc est le sens opposé par rapport au noeud de référence (le point de départ est le dernier point de mon arc) ;

Pour la suite, je vais donc appliquer le même principe que celui des instructions SQL que j'ai employé pour le jeu de données GEOROUTE de l'IGN. Par analogie, la colonne dir_travel jouera donc le même rôle que celle de la colonne sens du GEOROUTE : Les différences notables seront donc

```
BEGIN TRANSACTION;  
ALTER TABLE streets ADD column source_id int4;  
ALTER TABLE streets ADD column target_id int4;  
ALTER TABLE streets ADD column edge_id int4;  
SELECT updategeometrysrid('streets','the_geom',-1);  
UPDATE streets SET the_geom=reverse(the_geom),dir_travel='F' WHERE dir_travel='T';  
SELECT assign_vertex_id('streets',0.00001);
```

```
SELECT create_graph_tables('streets', 'int4');
ALTER TABLE streets_edges ADD column dir_travel text;
ALTER TABLE streets_edges ADD column x1 double precision;
ALTER TABLE streets_edges ADD column y1 double precision;
ALTER TABLE streets_edges ADD column x2 double precision;
ALTER TABLE streets_edges ADD column y2 double precision;
ALTER TABLE streets_edges ADD column edge_id int4;
SELECT update_cost_from_distance('streets');
UPDATE streets_edges SET x1=(SELECT x(startpoint(the_geom)) FROM streets g WHERE g.edge_id=id GROUP BY id,g.the_geom LIMIT 1);
UPDATE streets_edges SET y1=(SELECT y(startpoint(the_geom)) FROM streets g WHERE g.edge_id=id GROUP BY id,g.the_geom LIMIT 1);
UPDATE streets_edges SET x2=(SELECT x(endpoint(the_geom)) FROM streets g WHERE g.edge_id=id GROUP BY id,g.the_geom LIMIT 1);
UPDATE streets_edges SET y2=(SELECT y(endpoint(the_geom)) FROM streets g WHERE g.edge_id=id GROUP BY id,g.the_geom LIMIT 1);
UPDATE streets_edges SET edge_id=(SELECT edge_id FROM streets g WHERE g.edge_id=id GROUP BY id,g.edge_id LIMIT 1);
UPDATE streets_edges SET dir_travel=(SELECT dir_travel::text FROM streets g WHERE g.edge_id=id GROUP BY id,g.dir_travel LIMIT 1);
SELECT AddGeometryColumn('streets_edges', 'the_geom', -1, 'MULTILINESTRING', 2 );
UPDATE streets_edges SET the_geom=(SELECT the_geom FROM streets g WHERE g.edge_id=id GROUP BY id,g.the_geom LIMIT 1);
UPDATE streets_edges SET reverse_cost=cost;
UPDATE streets_edges SET reverse_cost=1000000 WHERE dir_travel='F';
END TRANSACTION;
```

A.10.3 Modifications nécessaires sur la table streets_edges

Il faut effectuer les commandes suivantes

```
ALTER TABLE street_edges RENAME id TO gid;
ALTER TABLE street_edges DROP COLUMN edge_id;
alter TABLE street_edges RENAME cost TO length;
```

Ces modifications sont nécessaires notamment pour l'emploi des fonctionnalités. On en profitera aussi pour créer les index sur les colonnes source et target

```
CREATE INDEX k1 on streets_edges(source);
CREATE INDEX k2 on streets_edges(target);
VACUUM FULL ANALYZE;
```

A.10.4 Exemple avec shortest_path_astar_as_geometry_internal_id_directed()

Comme mon graphe est orienté, pour aller par exemple du noeud source=5274 au noeud target=5488, il me faudra faire

```
SELECT * FROM shortest_path_astar_as_geometry_internal_id_directed('streets_edges', 5274, 5488, false, true);
```

qui renverra les champs gid et the_geom. Ce qui implique qu'elle puisse être directement utilisée par MapServer. Sur les deux images suivantes, les tronçons à double sens (respectivement à sens unique) sont en bleu (respectivement en vert).

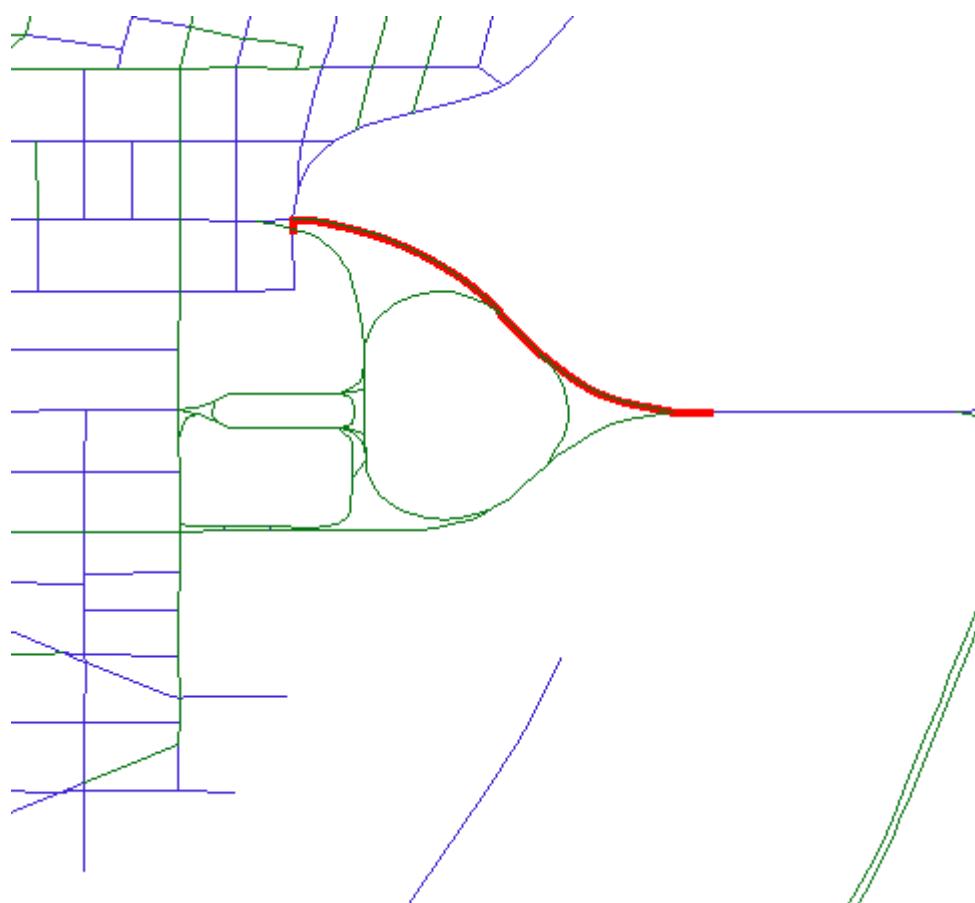


FIG. A.15 – Fonction shortest_path_astar_as_geometry_internal_id_directed() sur source=5274 et target=5488 (sens aller)



FIG. A.16 – Fonction shortest_path_astar_as_geometry_internal_id_directed() sur source=5274 et target=5488 (sens aller)

et pour le retour

```
SELECT * FROM shortest_path_astar_as_geometry_internal_id_directed('street_edges' ↔
    ,5488,5274,false,true);
```

qui renverra comme résultat

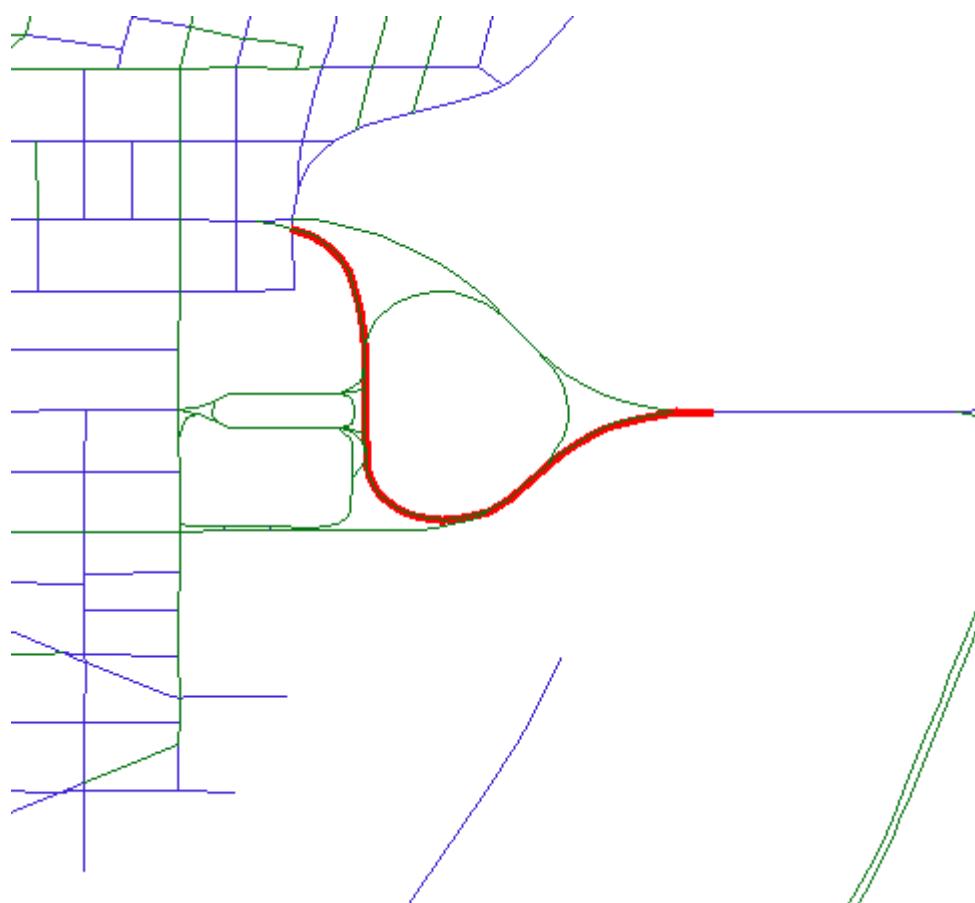


FIG. A.17 – Fonction shortest_path_astar_as_geometry_internal_id_directed() sur source=5488 et target=5274 (sens retour)



FIG. A.18 – Fonction `shortest_path_astar_as_geometry_internal_id_directed()` sur `source=5488` et `target=5274` (sens retour)

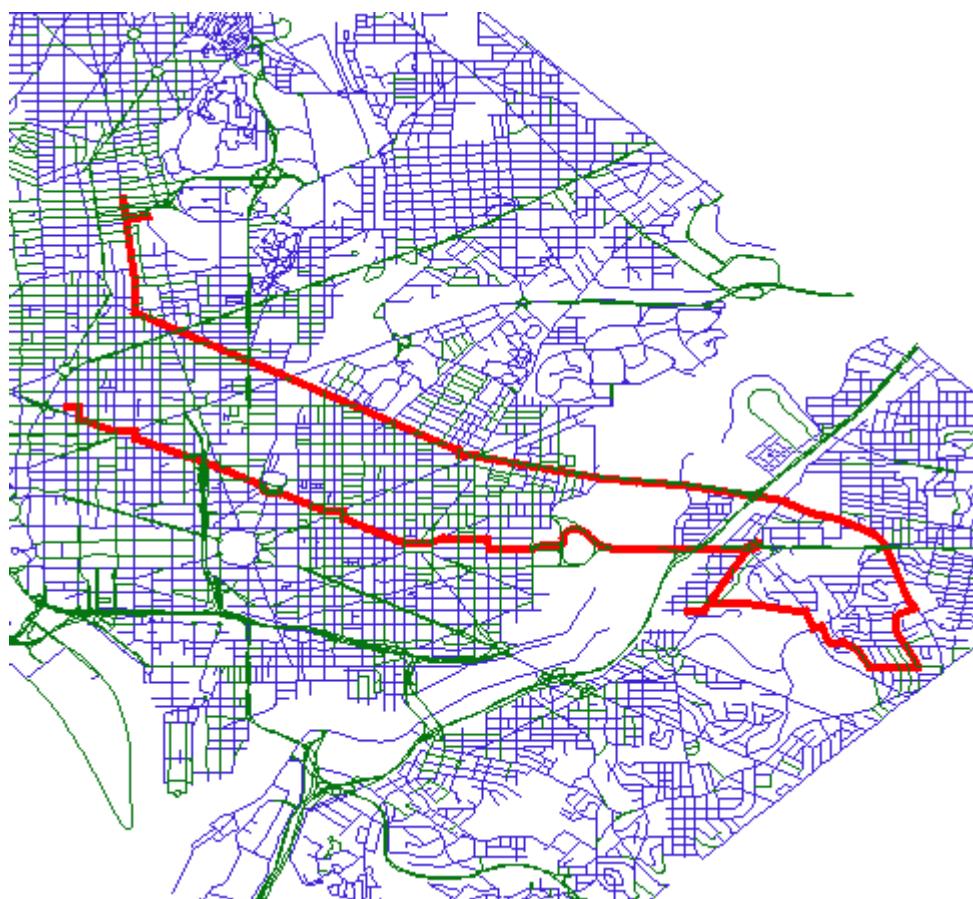
A.10.5 Exemple avec `tsp_astar_as_geometry_internal_id_directed()`

Comme mon graphe est orienté, je vais ici considérer les noeuds 5403,5822,338,7106,6043,1952. Je prendrais comme point de départ `source=1952`

Nous obtiendrons directement les enregistrements `the_geom` en faisant

```
SELECT the_geom FROM tsp_astar_as_geometry_internal_id_directed('streets_edges' ←
  ',5403,5822,338,7106,6043,1952',1952,0.03,false,true);
```

qui renverra les champs `gid` et `the_geom`. Ce qui implique qu'elle puisse être directement utilisée par MapServer.

FIG. A.19 – Fonction `tsp_astar_as_geometry_internal_id_directed()`

Une exportation de cette requête dans un fichier kml par

```
ogr2ogr -f KML ~/parcours.kml PG:'host=localhost dbname=routing user=postgres' \
-sql "SELECT (dump).geom FROM (SELECT dump(the_geom) FROM \
tsp_astar_as_geometry_internal_id_directed('streets_edges' <-
,'5403,5822,338,7106,6043,1952',1952,.03,false,true)) AS foo"
```

me renverra l'image suivante dans GoogleEarth

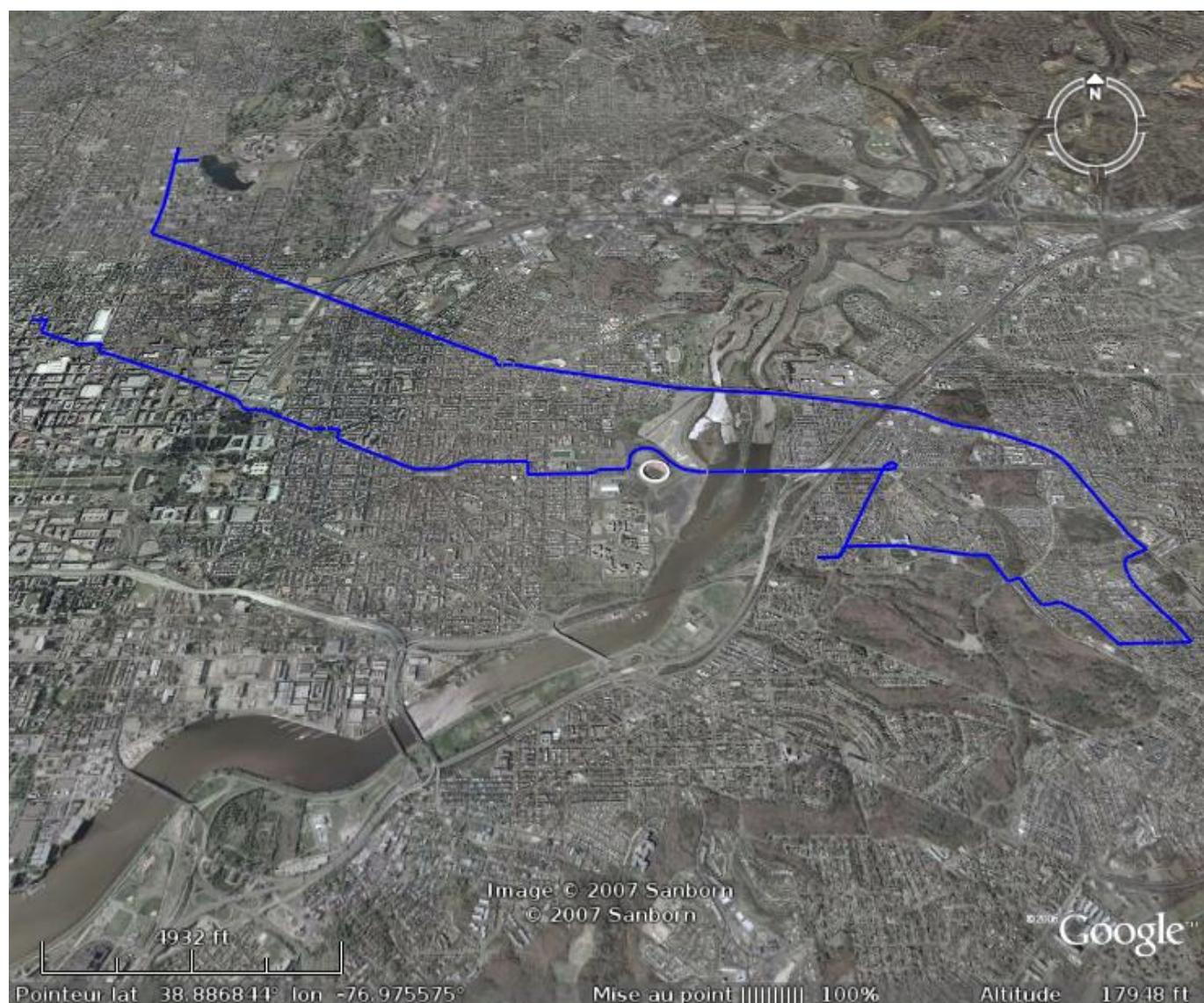


FIG. A.20 – Fonction `tsp_astar_as_geometry_internal_id_directed()` vue depuis GoogleEarth

Annexe B

Créer un modèle de base de données PostgreSQL contenant les fonctionnalités de PostGIS pour un utilisateur aux droits restreints sur une base

Ce genre de spécifié fait partie du travail qui pourrait attendre un administrateur PostGIS dans un cadre bien particulier. Exposons donc ce dernier. Supposons que vous êtes administrateur. A chaque utilisateur de la base correspond une base de données. Vous devez respecter les consigne suivantes que l'on vous demande

A chaque utilisateur correspond une base de données et réciproquement une base de données est propre à un unique utilisateur (un peu comme la philosophie à là "Oracle")

Chacune des bases de données doit contenir les fonctionnalités de PostGIS qu'il doit pouvoir utiliser sans souci

Un utilisateur est propriétaire de sa propre base de données à laquelle il peut se connecter, mais il ne peut n'y créer de base de données, ni de rôles supplémentaires, il a un mot de passe pour sa connexion

On pourra aussi substituer ci-dessus "groupe d'utilisateur spécifique" à "utilisateur" pour que par la suite un nouvel utilisateur puisse à son tour hériter des droits de son nouveau groupe sur la base

Voyons pour celà les diverses commandes à effectuer

B.1 Crédation du modèle contenant les fonctionnalités de PostGIS

Ici nous appelerons le modèle template_gis. Celà commence donc par la création d'une base de données PostGIS. Connectez-vous donc en tant qu'administrateur (que nous appellerons ici postgres)

```
su postgres
```

Une fois connecté, nous créons notre base template_gis

```
createdb template_gis
createlang plpgsql template_gis
psql -d template_gis -f [chemin_d_acces_vers_lwpostgis.sql]
psql -d template_gis -f [chemin_d_acces_vers_spatial_ref_sys.sql]
```

Je fais exprès ici d'employer [chemin_d_acces] afin d'éviter à distinguer Windows de GNU/Linux. On se connecte maintenant à la base en question pour prévenir PostgreSQL que la base que nous venons de créer est et sera à l'avenir un modèle de base de données (comme template1 par défaut)

```
psql template_gis
```

La requête est

```
UPDATE pg_database SET datistemplate='true' WHERE datname='template_gis';
```

B.2 Création de l'utilisateur et de la base de données

Nous allons ici appelé l'utilisateur jean et sa base meriim

B.2.1 Création de l'utilisateur

Pour créer l'utilisateur jean selon le cahier des charges attendu (droit de se connecter à une base de données, aura un mot de passe, simple utilisateur avec impossibilité de créer d'autre bases de données,impossibilité de créer d'autre rôles), nous ferons

```
#createuser -PReLUjean
Entrez le mot de passe pour le nouvel rôle :
Entrez-le de nouveau :
CREATE ROLE jean ENCRYPTED PASSWORD 'jean' NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT ←
    LOGIN;
CREATE ROLE
```

B.2.2 Création de la base de données

Pour créer la base de données, il suffira de faire

```
#createdb -O jean -T template_gis meriim
```

Pour avoir un exposé des commandes passés respectivement à createdb ou createuser, vous pouvez saisir `man createdb`, `man createuser`, `createdb --help`, `createuser --help`

B.2.3 Restriction des droits sur les tables `geometry_columns` et `spatial_ref_sys`

Pour restreindre les droits sur ces tables à l'utilisateur jean, nous ferons depuis psql, les requêtes SQL suivantes

```
GRANT ALL ON geometry_columns TO jean;
GRANT SELECT ON spatial_ref_sys TO jean;
```

B.2.4 Vérification

Connectons-nous en tant que jean sur la base meriim

```
postgres@bremko:/root$ psql -U jean meriim
Bienvenue dans psql 8.1.5, l'interface interactive de PostgreSQL.
```

```
Tapez: \copyright pour les termes de distribution
      \h pour l'aide-mémoire sur les commandes SQL
      \? pour l'aide-mémoire sur les commandes psql
      \g ou terminez avec un point-virgule pour exécuter une requête
      \q pour quitter
```

```
meriim=> select * from geometry_columns ;
 f_table_catalog | f_table_schema | f_table_name | f_geometry_column | coord_dimension | ←
 srid | type
```

```
-----+-----+-----+-----+-----+-----+
(0 lignes)

meriim=> \dt
          Liste des relations
 Schéma |      Nom       | Type   | Propriétaire
-----+-----+-----+-----+
 public | geometry_columns | table  | postgres
 public | spatial_ref_sys | table  | postgres
 public | test            | table  | jean
(3 lignes)
```

Ok si j'essais maintenant sur une autre base existante igo

```
meriim=> \c igo
Vous êtes maintenant connecté à la base de données «igo».
igo=> \dt
          Liste des relations
 Schéma |      Nom       | Type   | Propriétaire
-----+-----+-----+-----+
 public | 31_com_plus_hors_agglo | table  | postgres
 public | com_plus_hors_agglo  | table  | postgres
 public | geometry_columns      | table  | postgres
 public | ligne_tram_3           | table  | postgres
 public | spatial_ref_sys        | table  | postgres
(5 lignes)

igo=>
```

Je peux donc me connecter à cette base (option -l de createuser que nous avons utilisée). Maintenant si j'essais de listé le contenu de la table

```
igo=> SELECT * FROM "31_com_plus_hors_agglo";
ERREUR:  droit refusé pour la relation 31_com_plus_hors_agglo
igo=>
```

Mais puis-je créer au moins des tables. Je vais essayer en important des Shapefiles

```
postgres@bremko:~/igo$ shp2pgsql -DI 13.shp ligne | psql -U jean meriim
Shapefile type: Arc
Postgis type: MULTILINESTRING[2]
BEGIN
INFO: CREATE TABLE créera des séquences implicites «ligne_gid_seq» pour la colonne ←
      «serial» «ligne.gid»
INFO: CREATE TABLE / PRIMARY KEY créera un index implicite «ligne_pkey» pour la table ←
      «ligne»
CREATE TABLE
               addgeometrycolumn
-----
public.ligne.the_geom SRID:-1 TYPE:MULTILINESTRING DIMS:2
(1 ligne)

CREATE INDEX
COMMIT
```

Ok ! Mais puis-je au moins comme attendu utiliser les fonctionnalités de PostGIS en tant qu'utilisateur jean ?

```
meriim=> select geometrytype(the_geom),length(the_geom) from ligne;
 geometrytype |      length
-----+-----+
```

```
MULTILINESTRING | 19578.4949845822
(1 ligne)

meriim=> select postgis_full_version();
               postgis_full_version
-----
 POSTGIS="1.1.6" GEOS="2.2.3-CAPI-1.1.1" PROJ="Rel. 4.5.0, 22 Oct 2006" USE_STATS
(1 ligne)
```

Ok donc tout est Ok !

Si par le suite l'utilisateur fait usage de sa base sur un réseau, il incombera à l'administrateur d'adapter le fichier pg_hba.conf de PostgreSQL à cet utilisateur. (cf chapitre 4 sous-section 4.8.2)

Annexe C

Dblink : interroger plusieurs serveurs PostgreSQL distants

Dblink est un module de PostgreSQL qui permet d'interroger des serveurs distants ou des instances du même serveur sur la même machine. Il est fourni dans les sources de PostgreSQL.

C.1 Matériel requis pour la simulation

Nous prendrons pour cet article le cas de deux machines ayant chacune un serveur PostgreSQL. Ceci est bien sûr le cas le plus simple d'utilisation. On peut facilement l'étendre à 3,4 etc...

1. bremko : IP 192.168.0.4, base : testgis, table :communes_lr ;
2. jenna : IP 192.168.0.5, base : testgis2, table :departements_lr.

Le suffixe "_lr" désigne la région Languedoc-Roussillon. Il s'agit donc des communes et des départements du Languedoc-Roussillon. Mon but ici est de connaître l'ensemble des communes de la région limitrophes au département de l'Hérault (code insee=34). Les données et les tables ici présentées ont déjà été mentionnées dans le chapitre "Etudes de cas". C'est sur bremko que je décide d'installer le module dblink. Pour gagner du temps, je vais supposer ici que mes deux tables sont convenablement indexées sur leur champ géométrique respectif. Je vais aussi supposer que les deux tables sont dans le schéma public de leur base respective.

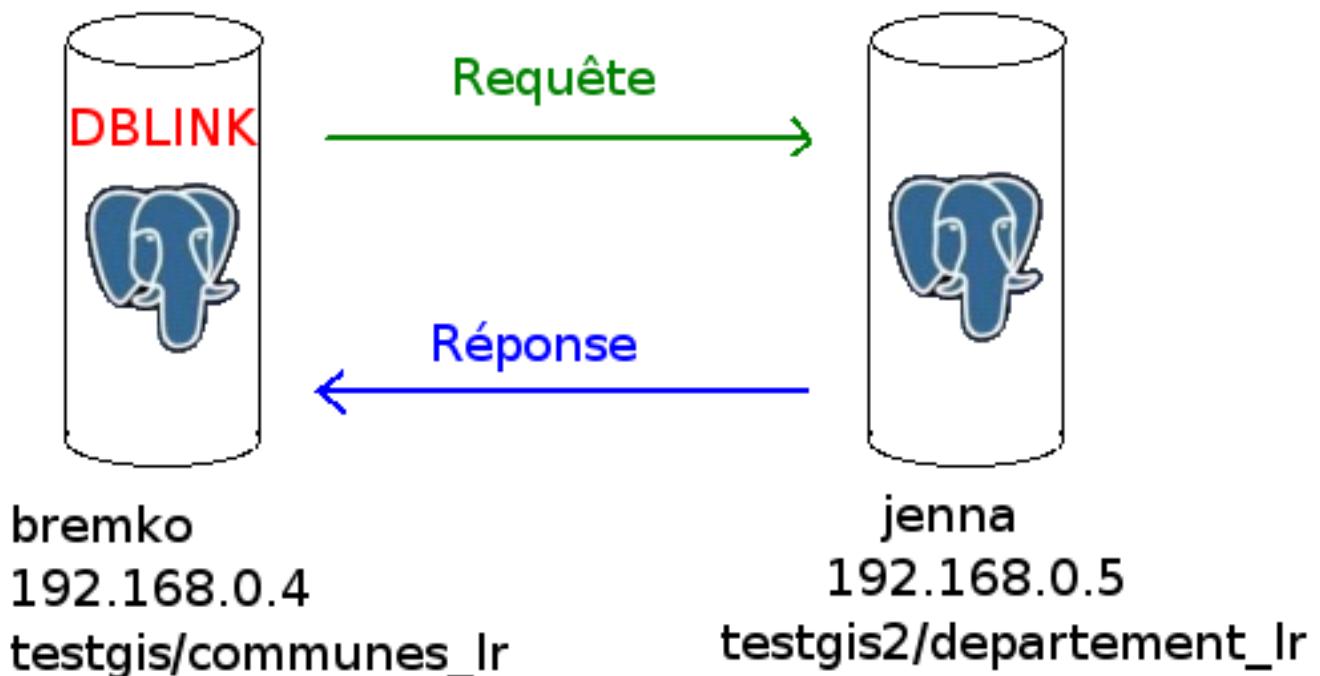


FIG. C.1 – Mes deux bases distantes : testgis et testgis2

C.2 Compilation et installation

L'installation de Dblink est une installation classique. Pour sa compilation en tant que root, je fais

```
cd /mnt/sources/postgresql-8.2.3/contrib/dblink
make
make install
```

Maintenant, je deviens propriétaire de la base testgis - ici postgres - sur bremko et je dote ma base des fonctionnalités de dblink

```
su postgres
psql -d testgis -f /opt/pgsql/share/contrib/dblink.sql
```

C.3 Mise en oeuvre

Je vais commencer par quelques test rudimentaires puis essayer la requête SQL attendue

Pour avoir l'inventaire des fonctionnalités de dblink, on peut consulter les divers fichiers (connexion, cursor, execute, misc, query) contenus dans le sous-répertoire doc des sources de dblink. Je ne présenterais ici qu'un échantillon des fonctionnalités.

C.3.1 Premiers tests

Depuis bremko, je me connecte à ma base testgis en faisant psql testgis. Grâce à dblink, je vais essayer de me connecter à la base testgis2 sur jenna. Sur jenna, l'utilisateur est damien ayant le mot de passe morphine. Ma connexion, je vais l'appeler c1. Je commence par ouvrir la connexion en utilisant la fonction dblink_connect() :

```
testgis=# SELECT * from dblink_connect('c1','hostaddr=192.168.0.5 port=5432 dbname=testgis2 ←
          user=damien password=morphine');
dblink_connect
-----
OK
(1 ligne)
```

J'effectue une requête simple qui me renvoie les données attributaires de la table departements_lr grâce à dblink()

```
testgis=# SELECT * FROM dblink('c1','SELECT gid,id,nom,numero FROM departements_lr',true)
AS foo(gid int,id int,nom text,numero text) ORDER BY nom;
      gid | id | nom           | numero
-----+---+-----+-----+
      4  |  4 | Aude          |  11
      2  |  2 | Gard           |  30
      1  |  1 | Hérault        |  34
      3  |  3 | Lozère         |  48
      5  |  5 | Pyrénées Orientales |  66
(5 lignes)
```

Je ferme ensuite ma connexion c1 grâce à dblink_disconnect()

```
testgis=# SELECT * from dblink_disconnect('c1');
dblink_disconnect
-----
OK
(1 ligne)
```

C.3.2 Test attendu

Pour se faire, je vais ici supposer qu'un connexion c1 est ouverte. Il me suffira de faire

```
testgis=# SELECT c.nom,c.insee FROM communes_lr c,
dblink('c1','SELECT nom,the_geom FROM departements_lr',true) AS foo(nom text,the_geom ←
          geometry)
WHERE foo.nom='Hérault'
AND c.the_geom && foo.the_geom
AND touches(c.the_geom,foo.the_geom)
AND c.nom NOT LIKE '34%';
```

On notera ici que

WHERE foo.nom='Hérault' est utilisé pour ne retourner que le contour départemental de l'Hérault de la table departements_lr;
c.the_geom && foo.the_geom est utilisé pour tirer profit des index spatiaux ;
AND c.nom NOT LIKE '34%' pour ne pas retourner les communes internes de l'Hérault.

Au niveau de , pour la confirmation de l'utilisation des index spatiaux, un simple EXPLAIN ANALYZE au début de la requête confirmara leur emploi.

Annexe D

Pouvoir faire des sauvegardes des bases locales ou distantes d'un serveur PostgreSQL du réseau intranet vers une machine-cliente

Mon but ici n'est pas de développer ici le sujet sur la restauration mais de profiter du fait de disposer de MinGW - donc des utilitaires de base de GNU/Linux - pour proposer dans un premier temps d'exploiter les utilitaires de PostgreSQL dans le cas d'un backup/restauration. Puis dans un second temps, de proposer par un shell-script bash, un exemple de script qui permette de sauvegarder vos bases (schémas+tables) aux divers formats : .sql, zippé et en archive compressé .tar.gz . Le deuxième cas exploitera uniquement l'utilitaire pg_dump sans avoir recours à pg_restore.

ATTENTION : La sauvegarde ici proposée ne prend en compte que les tables contenues dans les bases de données : leurs schémas et leurs contenus. Les fonctionnalités de PostGIS (sauf pour les tables geometry_columns et spatial_ref_sys) et autres fonctionnalités faites par l'utilisateur ne sont pas prises en compte. Nous partons ici du principe que l'utilisateur n'effectue pas ses sauvegardes pour une mise à jour de PostGIS éventuelle ! Mais bien dans le souci de continuer à travailler avec la même version de PostGIS. Le cas contraire - celle d'une mise à jour de PostGIS - , veuillez consultez la documentation de PostGIS ou l'annexe Foire Aux Questions.

D.1 Proposition 1 : Sauvegarde et restauration rapide par une base

Nous allons ici proposer grâce aux utilitaires effectuer un exemple de sauvegarde/restauration rapide

D.1.1 Sauvegarde au format tar du schéma public de la base de données - pg_dump -

Les données que nous importons dans notre base de données sont stockées dans le schéma public. PostgreSQL propose par défaut 3 formats : sql (texte), .tar (archive), * (compression). Nous allons ici proposer le format de sauvegarde archive. cela se fait en utilisant l'utilitaire en ligne de commande **pg_dump**. On obtient un descriptif de ce dernier en faisant depuis DOS :

```
pg_dump --help
```

Dans mon cas, j'utilise souvent la ligne de commande suivante¹

```
pg_dump -n public -vFt -f [base].tar [base]
```

où les options

-n public permet de préciser qu'il ne faut sauvegarder que le schéma public de la base qui contient les données que nous avons importées ;

¹ où le diminutif de l'option -vFt correspond aussi à -v -Ft

On pourra dire aussi que si l'on veut avoir une sau-

- v** permet (mode verbeux) permet lors de l'exécution de cette commande d'afficher tous les étapes de la sauvegarde ;
- Ft** permet de préciser que le format de sortie est du .tar ;
- f** précise le nom du fichier de sauvegarde.

Par exemple pour sauvegarder la base de données madatabase (cf. "Chapitre PostGIS"), en tapant la ligne de commande ci-dessus on obtient en guise d'output

```
>pg_dump -vFt -f madatabase.tar madatabase
pg_dump: lecture des schémas
pg_dump: lecture des fonctions définies par l'utilisateur
pg_dump: lecture des types définis par l'utilisateur
pg_dump: lecture des langages procéduraux
pg_dump: lecture des fonctions d'aggrégats définies par l'utilisateur
pg_dump: lecture des opérateurs définis par l'utilisateur
pg_dump: lecture des classes d'opérateurs définies par l'utilisateur
pg_dump: lecture des conversions définies par l'utilisateur
pg_dump: lecture des tables définies par l'utilisateur
pg_dump: lecture des informations d'héritage des tables
pg_dump: lecture des règles de ré-écriture
pg_dump: lecture des conversions de type
pg_dump: recherche des relations d'héritage
pg_dump: lecture des informations sur les colonnes pour les tables intéressantes
pg_dump: recherche des colonnes et types de la table «geometry_columns»
pg_dump: recherche des colonnes et types de la table «great_roads»
pg_dump: recherche des expressions par défaut de la table «great_roads»
pg_dump: recherche des contraintes de vérification pour la table «great_roads»
pg_dump: recherche des colonnes et types de la table «mapserver_desc»
pg_dump: recherche des expressions par défaut de la table «mapserver_desc»
pg_dump: recherche des colonnes et types de la table «parcs»
pg_dump: recherche des expressions par défaut de la table «parcs»
pg_dump: recherche des contraintes de vérification pour la table «parcs»
pg_dump: recherche des colonnes et types de la table «personnes»
pg_dump: recherche des expressions par défaut de la table «personnes»
pg_dump: recherche des contraintes de vérification pour la table «personnes»
pg_dump: recherche des colonnes et types de la table «pq_logdir_ls»
pg_dump: recherche des colonnes et types de la table «rivers»
pg_dump: recherche des expressions par défaut de la table «rivers»
pg_dump: recherche des contraintes de vérification pour la table «rivers»
pg_dump: recherche des colonnes et types de la table «small_roads»
pg_dump: recherche des expressions par défaut de la table «small_roads»
pg_dump: recherche des contraintes de vérification pour la table «small_roads»
pg_dump: recherche des colonnes et types de la table «spatial_ref_sys»
pg_dump: recherche des colonnes et types de la table «buildings»
pg_dump: recherche des expressions par défaut de la table «buildings»
pg_dump: recherche des contraintes de vérification pour la table «buildings»
pg_dump: ajout de drapeaux pour les colonnes héritées des sous-tables
pg_dump: lecture des index
pg_dump: lecture des index de la table «geometry_columns»
pg_dump: lecture des index de la table «great_roads»
pg_dump: lecture des index de la table «mapserver_desc»
pg_dump: lecture des index de la table «parcs»
pg_dump: lecture des index de la table «personnes»
pg_dump: lecture des index de la table «rivers»
pg_dump: lecture des index de la table «small_roads»
pg_dump: lecture des index de la table «spatial_ref_sys»
pg_dump: lecture des contraintes
pg_dump: lecture des déclencheurs (triggers)
pg_dump: récupération des données de dépendance
pg_dump: sauvegarde du codage
pg_dump: sauvegarde de la définition de la base de données
pg_dump: sauvegarde du contenu de la table buildings
pg_dump: sauvegarde du contenu de la table geometry_columns
pg_dump: sauvegarde du contenu de la table great_roads
pg_dump: sauvegarde du contenu de la table mapserver_desc
pg_dump: sauvegarde du contenu de la table parcs
```

FIG. D.1 – Sauvegarde du schéma public de la base madatabase - pg_dump -

D.1.2 Restauration - pg_restore -

Dans le cas pratique d'une restauration ,nous avons recours à l'utilitaire **pg_restore**. On obtient un descriptif en faisant

```
pg_restore --help
```

Dans mon cas, j'utilise souvent la ligne de commande suivante pour une sauvegarde complète

```
pg_restore -d [base] -civ0 [base].tar
```

où les options

- c** précise d'effacer la table en cours avant de la créer et de la peupler ;
- i** demande de passer outre si la version du serveur PostgreSQL qui charge le contenu de l'archive n'est pas le même que celui du serveur qui a généré l'archive ;
- O** demande de ne pas prendre en compte le nom du super-utilisateur qui a généré l'archive.
- v** a été abordé dans la sous-section précédente ;

Dans le cas particulier, où je ne veux que restaurer par exemple une table, il faut utiliser l'option -t. Ce qui donne la commande suivante :

```
pg_restore -d [base] -civ0t [table] [base].tar
```

Par exemple dans le cas de la base madatabase pour restaurer la table buildings, on aura l'output suivant

```
c:\ Invite de commandes

>pg_restore -d madatabase -civ0t buildings madatabase.tar
pg_restore: connexion à la base de données pour la restauration
pg_restore: suppression de TABLE buildings
pg_restore: création de TABLE buildings
pg_restore: restauration des données de la table «buildings»
pg_restore: initialisation du propriétaire et des privilèges pour TABLE buildings
>
```

FIG. D.2 – Restauration de la table buildings dans la base madatabase - pg_restore -



AVERTISSEMENT

A la suite d'une restauration même partielle, si vos données importées sont indexées, il est souvent conseiller de faire un **VACUUM ANALYZE**.

D.2 Proposition 2 : Sauvegarde pour toutes les bases et toutes les tables sans les définitions des fonctions de PostGIS ou autre- Script général (Côté-client)

Le script suivant demande que vous ayez créé un répertoire où se trouvera vos sauvegardes. L'exemple de répertoire choisit ici est **/home/\$USERNAME** qui doit exister avant toute utilisation du script. Le sous-répertoires **\$USERNAME** correspond à celui que MinGW vous aura créé. Au passage merci à Gérald FENOY pour son aide pour ce script... :-). J'utilise ici l'utilitaire **pg_dump** dont vous pouvez obtenir un descriptif succinct d'utilisation en faisant

```
pg_dump --help
```

**AVERTISSEMENT**

Dans l'intérieur du script, les paramètres de connexion à la machine distante supposent que vous connaissiez les paramètres de connexion à la machine-serveur et que vous soyez logué en tant que super-utilisateur sur toutes les bases distantes.

Le format choisi des sauvegardes dans les trois formats est suffixé avec la date et l'heure de création de la sauvegarde.

```
path="/home/${USERNAME}/dump"
path_sql="${path}/sql/"
path_zip="${path}/zip/"
path_archive="${path}/archive/"

#
# Les paramètres de la machine-serveur --- LIGNES A MODIFIER !!! ---
#
hote="jenna"
port="5432"
user="davidgis"
password="empr888"
echo ${password} > password.file
echo >> password.file
for i in ${path} ${path_sql} ${path_zip} ${path_archive}
do
if [ ! -d "$i" ]; then
    mkdir -p "$i"
fi
done

liste_Bases=$(psql -h ${hote} -p ${port} -U ${user} -d template1 -c \
    "SELECT pg_database.datname \
     FROM pg_database, pg_user \
     WHERE pg_database.datdba = pg_user.usename \
     and pg_database.datname not \
     like 'template%' order by datname" < password.file \
     | grep -v "\-\-\-\-" | grep -v "datname" | grep -v "(")

for r in ${liste_Bases}
do
base=$r
for s in ${path_sql} ${path_zip} ${path_archive}
do
if [ ! -d "${s}${base}" ]; then
    mkdir -p "${s}${base}"
fi
done

Heure=$(date +%Y-%m-%d-%I-%M-%S)
echo ${password} > password.file
echo >> password.file
filename_sql="${base}-${Heure}.sql"
filename_zip="${base}-${Heure}.zip"
filename_archive="${base}-${Heure}.tar.gz"
echo =====
echo Debut extraction de la base ${base}
echo =====
echo "a) Debut Ecriture dans fichier sql:"
liste_Tables=$(psql -h ${hote} -p ${port} -U ${user} -d ${base} -c "select \
tablename from pg_tables where (tablename not like 'pg_%') and \
(tablename not like 'spatial_ref_sys') and (tablename not like 'sql%') \
order by tablename" < password.file \
| grep -v "\-\-\-\-" | grep -v "tablename" | grep -v "(")
```

```
for j in ${liste_Tables}
do
    echo -n "Exportation de la table $j de la base ${base} en cours....";
    pg_dump -h ${hote} -p ${port} -U ${user} -i -c -O -t $j ${base} < password.file >> \
    ${path_sql}${base}/${filename_sql};
    echo -e "terminee.";
done

echo Fin ecriture dans fichier sql

cd ${path_sql}${base}
echo -n "b) Compression au format .zip de la base ${base} en cours..."
echo "." | zip -c -z9 ${filename_zip} ${filename_sql} > /dev/null
mv ${filename_zip} ${path_zip}${base}/${filename_zip}
echo -e "..termine"

echo -n "c) Compression au format .tar.gz de la base ${base} en cours..."
echo "." | tar czvf ${filename_archive} ${filename_sql} > /dev/null
mv ${filename_archive} ${path_archive}${base}/${filename_archive}
echo -e "..termine"
echo ""
echo "    RESUME"
echo "    Consultez les fichiers:"
echo "        - SQL : ${path_sql}${base}/${filename_sql}"
echo "        - ZIP : ${path_zip}${base}/${filename_zip}"
echo "        - TAR.GZ : ${path_archive}${base}/${filename_archive}"
echo ""
echo =====
echo Fin extraction de la base ${base}
echo =====
done
```

En exemple d'output par exemple, on obtient

```
=====
Debut extraction de la base Tlieux
=====
a) Debut Ecriture dans fichier sql:
Exportation de la table communes_lr de la base Tlieux en cours....terminee.
Exportation de la table dep30 de la base Tlieux en cours....terminee.
Exportation de la table departements_lr de la base Tlieux en cours....terminee.
Exportation de la table geometry_columns de la base Tlieux en cours....terminee.
Exportation de la table tlieuxclic de la base Tlieux en cours....terminee.
Exportation de la table tlieuxclictemp de la base Tlieux en cours....terminee.
Exportation de la table tlieuxpersagees de la base Tlieux en cours....terminee.
Exportation de la table tlieuxpersageestemp de la base Tlieux en cours....terminee.
Fin ecriture dans fichier sql
b) Compression au format .zip de la base Tlieux en cours.....termine
c) Compression au format .tar.gz de la base Tlieux en cours.....termine

RESUME
Consultez les fichiers:
- SQL : /home/david/dump/sql/Tlieux/Tlieux-2005-07-31-09-59-43.sql
- ZIP : /home/david/dump/zip/Tlieux/Tlieux-2005-07-31-09-59-43.zip
- TAR.GZ : /home/david/dump/archive/Tlieux/Tlieux-2005-07-31-09-59-43.tar.gz

=====
Fin extraction de la base Tlieux
=====
=====
Debut extraction de la base madatabase
```

```
=====
a) Debut Ecriture dans fichier sql:  
Exportation de la table buildings de la base madatabase en cours.....terminee.  
Exportation de la table geometry_columns de la base madatabase en cours.....terminee.  
Exportation de la table great_roads de la base madatabase en cours.....terminee.  
Exportation de la table mapserver_desc de la base madatabase en cours.....terminee.  
Exportation de la table parcs de la base madatabase en cours.....terminee.  
Exportation de la table personnes de la base madatabase en cours.....terminee.  
Exportation de la table rivers de la base madatabase en cours.....terminee.  
Exportation de la table small_roads de la base madatabase en cours.....terminee.  
Fin ecriture dans fichier sql  
b) Compression au format .zip de la base madatabase en cours.....termine  
c) Compression au format .tar.gz de la base madatabase en cours.....termine  
  
RESUME  
Consultez les fichiers:  
- SQL : /home/david/dump/sql/madatabase/madatabase-2005-07-31-10-00-17.sql  
- ZIP : /home/david/dump/zip/madatabase/madatabase-2005-07-31-10-00-17.zip  
- TAR.GZ : /home/david/dump/archive/madatabase/madatabase-2005-07-31-10-00-17.tar.gz  
=====
```

```
Fin extraction de la base madatabase  
=====
```

N'hésitez pas à consulter la documentation de PostgreSQL pour obtenir de meilleures informations sur l'utilitaire pg_dump. Pour le support du format zip qui n'est pas livré avec MinGW, vous pouvez le trouver sur le site <http://gnuwin32.sourceforge.net>.

D.3 Proposition 2 : Restauration

La restauration est une intervention à utiliser par exemple dans le cas de corruption de données etc...Nous allons ici envisager la restauration pour les deux formats. Avant tout ,il faut effacer la base, la recréer avec PostGIS et ensuite choisir le format voulu. L'effacement d'une base a lieu en utilisant l'utilitaire dropdb :

```
dropdb votre_base
```

Pour la recréer, reportez-vous à la première section du chapitre 4..

D.3.1 Format sql

La restauration a lieu ici en utilisant l'utilitaire psql. Par exemple pour restaurer la base 'madatabase', nous ferons depuis MinGW :

```
psql -d madatabase -f madatabase-.....sql
```

où désigne le suffixe de date du fichier. A titre d'exemple concret, nous pourrions écrire en tenant compte de la sortie obtenue dans la section précédente :

```
psql -d madatabase -f /home/david/dump/sql/madatabase/madatabase-2005-07-31-10-00-17.sql
```

D.3.2 Format tar.gz

La restauration a lieu ici en utilisant l'utilitaire psql et tar. Depuis MinGW,- par exemple pour restaurer la base 'madatabase' - il suffit de faire :

```
psql -d madatabase -f 'tar xvzf madatabase-....tar.gz'
```

où désigne le suffixe de date du fichier. En guise d'illustration, nous obtiendrons l'affichage suivant

```
#  
# On se rend dans le répertoire des archives  
#  
$ cd /home/david/dump/archive/madatabase/  
#  
# On lance la décompression à la volée l'archive  
# qui nous restitue le fichier au format sql  
#  
$psql -d madatabase -f 'tar xvzf madatabase-2005-07-31-10-00-17.tar.gz'  
SET  
SET  
SET  
SET  
psql:madatabase-2005-07-31-10-00-17.sql:11: ERROR: index "buildings_index_spatial" does ←  
not exist  
psql:madatabase-2005-07-31-10-00-17.sql:12: ERROR: relation "public.buildings" does not ←  
exist  
psql:madatabase-2005-07-31-10-00-17.sql:13: ERROR: table "buildings" does not exist  
SET  
SET  
CREATE TABLE  
ALTER TABLE  
CREATE INDEX  
. . .  
SET  
SET  
SET  
SET  
psql:madatabase-2005-07-31-10-00-17.sql:442: ERROR: index "small_roads_index_spatial" does ←  
not exist  
psql:madatabase-2005-07-31-10-00-17.sql:443: ERROR: relation "public.small_roads" does not ←  
exist  
psql:madatabase-2005-07-31-10-00-17.sql:444: ERROR: table "small_roads" does not exist  
SET  
SET  
CREATE TABLE  
ALTER TABLE  
CREATE INDEX  
#  
# Pour des raisons d'encombrements sur le disque on efface le  
# fichier sql décompressé  
#  
$rm -f madatabase-2005-07-31-10-00-17.sql
```

Les notes signalées - les tables ERROR : table "...." does not exist - n'importent pas puisque les tables n'existent plus avant l'import. Si vous relancez les commandes pour une seconde fois -facultatif- !, celles-ci ne seront plus affichées.



AVERTISSEMENT

A la suite d'une restauration même partielle, si vos données importées sont indexées, il est souvent conseillé de faire un **VACUUM ANALYZE**.

Annexe E

PostgreSQL et les index

Dans cet article, nous allons voir quelques exemples d'utilisation de PostgreSQL avec les index.

Ici je ne discute pas de l'intérêt des index sur des tables ou quand on doit y recourir pour optimiser des plans de requêtes. Mon but avant tout est de rappeler des exemples d'utilisation des index avec PostgreSQL. En premier lieu, je ne saurais vous conseiller de vous accorder dans votre temps libre un moment pour consulter le chapitre 11 de la documentation française <http://docs.postgresqlfr.org/8.2/indexes.html>

E.1 Importation d'un jeu de données

Nous allons ici commencer par importer le jeu de données provenant de l'URL suivante <http://www.galichon.com/codesgeo-data/insee.zip> ensuite on le décomprime

```
$ wget http://www.galichon.com/codesgeo/data/insee.zip  
$ unzip insee.zip
```

Ce zip contient un fichier CSV réalisé par Jérôme GALICHON

E.2 Importer des données au format CSV dans PostgreSQL

En faisant un petit less insee.csv, on voit que les données doivent être importées dans une table dont les champs sont

```
# less insee.csv  
Commune;Codepos;Departement;INSEE
```

Pour importer les données, il faut donc commencer par créer une table que nous appellerons insee en respectant la casse

```
igo=# CREATE TABLE insee ("Commune" text , "Codepos" varchar(5) , "Departement" text , "INSEE" →  
int4);
```

Comme sur la première ligne du fichier, il y a les noms des colonnes, il nous faut utiliser la commande COPY FROM en spécifiant l'option HEADER :

```
igo#COPY insee FROM '/home/david/download/insee.csv' DELIMITERS ';' CSV HEADER;
```

Vérifions que nous avons une table pleine

```
igo=# SELECT * FROM insee LIMIT 5;
          Commune      | Codepos | Departement | INSEE
-----+-----+-----+-----+
L ABERGEMENT CLEMENCIAIT | 01400   | AIN         | 1001
L ABERGEMENT DE VAREY   | 01640   | AIN         | 1002
AMAREINS                 | 01090   | AIN         | 1003
AMBERIEU EN BUGEY       | 01500   | AIN         | 1004
AMBERIEUX EN DOMBES     | 01330   | AIN         | 1005
(5 lignes)
```

E.3 Index B-tree, opérateur =

Les index B-tree sont les index classiques avec PostgreSQL. Supposons pour commencer ce que celà donne sans utiliser d'index si on veut connaître la commune dont le code postal est 34170

```
igo=# SELECT "Commune" FROM insee WHERE "Codepos"='34170';
          Commune
-----
 CASTELNAU LE LEZ
(1 ligne)
```

En utilisant EXPLAIN ANALYZE, j'ai une premier plan de requête

```
igo=# EXPLAIN ANALYZE SELECT "Commune" FROM insee WHERE "Codepos"='34170';
          QUERY PLAN
-----
Seq Scan on insee  (cost=0.00..680.40 rows=121 width=32) (actual time=6.821..25.082 rows=3
=1 loops=1)
  Filter: ((("Codepos")::text = '34170'::text)
Total runtime: 25.135 ms
(3 lignes)
```

Seq Scan on insee m'indique que la requête a exigé un parcours séquentiel sur les lignes du tableau. Je n'ai que 36000 en tous. Voyons si un index peut nous aider à améliorer tout ça

Ici bien sûr, je vous renvoie à la documentation de PostgreSQL citée en début d'article.

```
igo=# CREATE INDEX codepos_idx ON insee("Codepos");
CREATE INDEX
igo=# VACUUM FULL ANALYZE ;
```

Refaisons la requête à nouveau

```
igo=# EXPLAIN ANALYZE SELECT "Commune" FROM insee WHERE "Codepos"='34170';
          QUERY PLAN
-----
Index Scan using codepos_idx on insee  (cost=0.00..8.55 rows=11 width=15) (actual time=0.033..0.039 rows=1 loops=1)
  Index Cond: ((("Codepos")::text = '34170'::text)
Total runtime: 0.097 ms
(3 lignes)
```

Déjà je note un temps d'exécution significatif en ayant recours à un index !

E.4 Index B-tree, fonctions

Dans ma table insee, les noms des communes sont capitalisées. Or lower() est une fonction de PostgreSQL qui permet sur une chaînes de caractères de la renvoyer en minuscule.

```
igo=# explain analyze select * FROM insee WHERE lower("Commune")='castelnau de levis';
                                         QUERY PLAN
-----
Seq Scan on insee  (cost=0.00..962.24 rows=195 width=40) (actual time=114.664..121.935 ←
    rows=1 loops=1)
  Filter: (lower("Commune") = 'castelnau de levis'::text)
Total runtime: 122.141 ms
(3 lignes)
```

On peut aussi créer des index sur des fonctions

```
igo=# CREATE INDEX communes_idx on insee(lower("Commune"));
CREATE INDEX
igo=# VACUUM FULL ANALYZE ;
VACUUM
igo=# explain analyze select * FROM insee WHERE lower("Commune")='castelnau de levis';
                                         QUERY PLAN
-----
Index Scan using communes_idx on insee  (cost=0.00..8.28 rows=1 width=40) (actual time ←
    =0.076..0.082 rows=1 loops=1)
  Index Cond: (lower("Commune") = 'castelnau de levis'::text)
Total runtime: 0.136 ms
(3 lignes)

igo=
```

E.5 Index B-tree, recherche sur motif, like "chaîne%"

Ici par rapport à la section précédente, nous allons essayer d'inverser la tendance.

Avec PostgreSQL, il n'est pas possible d'utiliser d'index pour la recherche de motif pour **LIKE '%chaîne%'**. En revanche, il est tout à fait possible de la faire pour un motif de type **LIKE 'chaîne%'**. A condition bien sûr, de prendre en compte la local. Consultez donc <http://docs.postgresqlfr.org/pgsql-8.2.1-fr/charset.html>

Voyons cela en détail. Chez moi, ma LOCALE n'est pas en 'C'

```
$locale
LANG=fr_FR.UTF-8
LANGUAGE=fr_FR:fr:en_GB:en
LC_CTYPE="fr_FR.UTF-8"
LC_NUMERIC="fr_FR.UTF-8"
LC_TIME="fr_FR.UTF-8"
LC_COLLATE="fr_FR.UTF-8"
LC_MONETARY="fr_FR.UTF-8"
LC_MESSAGES="fr_FR.UTF-8"
LC_PAPER="fr_FR.UTF-8"
LC_NAME="fr_FR.UTF-8"
LC_ADDRESS="fr_FR.UTF-8"
LC_TELEPHONE="fr_FR.UTF-8"
LC_MEASUREMENT="fr_FR.UTF-8"
LC_IDENTIFICATION="fr_FR.UTF-8"
```

```
LC_ALL=
```

Donc si je veux utiliser la recherche de motif "LIKE 'chaine%', sur le champ Commune de ma table insee, il faut que je fasse

```
igo=# explain analyze select * FROM insee WHERE "Commune" LIKE 'CASTELNAU%';
          QUERY PLAN
```

```
-----
```

```
Seq Scan on insee  (cost=0.00..864.86 rows=1 width=40) (actual time=1.331..24.396 rows=27 ←
    loops=1)
  Filter: ("Commune" ~~ 'CASTELNAU%':text)
Total runtime: 24.568 ms
(3 lignes)
```

Comme mon champs Communes est de type text, il me suffit de faire

```
igo=# CREATE INDEX communes_idxx ON insee("Commune" text_pattern_ops);
CREATE INDEX
igo=# VACUUM FULL ANALYZE ;
VACUUM
igo=# explain analyze select * FROM insee WHERE "Commune" LIKE 'CASTELNAU%';
          QUERY PLAN
```

```
-----
```

```
Index Scan using communes_idxx on insee  (cost=0.00..8.27 rows=1 width=40) (actual time ←
    =0.174..0.352 rows=27 loops=1)
  Index Cond: ((("Commune" ~>=~ 'CASTELNAU':text) AND ("Commune" ~<~ 'CASTELNAU':text))
  Filter: ("Commune" ~~ 'CASTELNAU%':text)
Total runtime: 0.525 ms
(4 lignes)
```

E.6 Index B-tree, recherche sur motif, like "%chaine"

L'idée ici est de se baser de ce que nous avons appris dans la sous-section précédente. Ici par rapport à la section précédente, il nous faut une petite subtilité. Cette dernière va consister par commencer par la création d'une fonction

E.6.1 Fonction inversant une chaîne en PostgreSQL

On pourra par exemple utiliser la fonction suivante

```
CREATE OR REPLACE FUNCTION "public"."inverserchaine" (chaine text) RETURNS text AS
$body$
/*
   Inverse la chaine

   Exemple : SELECT inverserchaine('postgresql');
              lqsergtsop
*/
DECLARE
  i integer;
  resultat text = '';
BEGIN
  FOR i IN REVERSE length(chaine)..1 LOOP
    resultat:=resultat||substring(chaine from i for 1);
  END LOOP;
  return resultat;
END;
$body$
LANGUAGE 'plpgsql' RETURNS NULL ON NULL INPUT SECURITY INVOKER IMMUTABLE;
```

Cette fonction a été écrite par Damien Griessinger et elle est disponible à <http://dgiessinger.developpez.com/postgresql/udf/?page=chaines#inverserchaine>. En vertu de la documentation de PostgreSQL à <http://traduc.postgresqlfr.org/pgsql-8.2.1-fr/sql-createindex.html>

Toutes les fonctions et opérateurs utilisés dans la définition d'index doivent être « immutable » (NDT : immuable), c'est-à-dire que leur résultat ne doit dépendre que de leurs arguments et jamais d'une influence externe (telle que le contenu d'une autre table ou l'heure). Cette restriction permet de s'assurer que le comportement de l'index est strictement défini. Pour utiliser une fonction utilisateur dans une expression d'index ou dans une clause WHERE, cette fonction doit être marquée immutable lors de sa création.

J'ai donc supprimé la spécificité VOLATILE dans la fonction.

E.6.2 Crédation de l'index sur le champs "Commune" de la table insee

L'index sera créé en faisant

```
CREATE INDEX communes_idxxx ON insee(inverserchaine("Commune") text_pattern_ops);
```

Puis on fera tout simplement un

```
VACUUM FULL ANALYZE;
```

E.6.3 Requêtes

La requête par exemple pour trouver toutes les communes qui se terminent par "lez" par exemple sera

```
igo=# SELECT "Commune" FROM insee WHERE inverserchaine("Commune") LIKE inverserchaine('LEZ ↔
      ') || '%';
      Commune
-----
LES BORDES SUR LEZ
CAMLEZ
VIEUX VILLEZ
TREFLEZ
LEZ
CASTELNAU LE LEZ
MONTFERRIER SUR LEZ
PRADES LE LEZ
GALEZ
LIMETZ VILLEZ
PORT VILLEZ
(11 lignes)
```

Maintenant si on fait des comparaison avec EXPLAIN ANALYZE

```
igo=# EXPLAIN ANALYZE SELECT "Commune" FROM insee WHERE inverserchaine("Commune") LIKE ↔
      inverserchaine('LEZ') || '%';
                                         QUERY PLAN
-----
Bitmap Heap Scan on insee  (cost=4.28..11.88 rows=2 width=15)  (actual time=0.146..0.579 ←
  rows=11 loops=1)
  Filter: (inverserchaine("Commune") ~~ 'ZEL%'::text)
  -> Bitmap Index Scan on communes_idxxx  (cost=0.00..4.28 rows=2 width=0)  (actual time ←
    =0.052..0.052 rows=11 loops=1)
      Index Cond: ((inverserchaine("Commune") ~>=~ 'ZEL%'::text) AND (inverserchaine("←
        Commune") ~<~ 'ZEM%'::text))
Total runtime: 0.697 ms
```

```
(5 lignes)

igo=# EXPLAIN ANALYZE SELECT "Commune" FROM insee WHERE "Commune" LIKE '%LEZ';
                                         QUERY PLAN
-----
Seq Scan on insee  (cost=0.00..864.86 rows=309 width=15)  (actual time=4.559..68.744 rows <-
 =11 loops=1)
  Filter: ("Commune" ~~ '%LEZ'::text)
Total runtime: 68.857 ms
(3 lignes)
```

Annexe F

Connaître l'espace disque occupé par les données (dbsize)



AVERTISSEMENT

Depuis la version 8.1.0, le sous-répertoire contrib/dbsize ne fait plus partie de la distribution des sources de PostgreSQL - pour une raison que j'ignore d'ailleurs -. Mais il était présent pour les versions 8.0.X. Pour voir les modifications apportées, merci de consulter le lien suivant

<http://traduc.postgresqlfr.org/pgsql-8.1.1-fr/release-8-1.html>

La gestion des données dont doit s'acquitter un administrateur de PostgreSQL est une tâche parfois bien ardu. Cette dernière concerne notamment la gestion du volume physique occupée par les données. Il existe à ce jour des solutions logiciels pour pouvoir gérer ce genre de problématique. Mais ici, on travaille à l'ancienne = "Je n'ai foi qu'en ce que je vois d'écrit dans mon terminal Shell" dixit Gerald.... "Ouais ! Mais c'est moi qui me tape la doc !", dixit David... Avant toute chose, la solution ici exposée n'est pas nécessairement la plus belle.

Il est tout à fait possible de connaître la taille qu'occupent les bases ou les tables en ayant recours à des requêtes, ou en utilisant VACUUM etc... - voir pour cela C :\PostgreSQL\8.2.1\doc\postgresql\html\diskusage.html -...Celle que nous avons retenue est dbsize qui est fournie avec les sources de PostgreSQL.

F.1 Dbsize - directement dans le backend de PostgreSQL -

Dbsize faisait partie des outils de contribution de PostgreSQL. Il a été directement inclus maintenant dans le backend de PostgreSQL : les fonctions de ce module sont maintenant "incluses" dans PostgreSQL. Pour avoir une liste des fonctions disponibles avec ce module, on pourra essayer par exemple depuis MinGW la commande

```
psql -d template1 -AtF " " -c "\df pg_*_size" | awk '{print $2}'
```

qui nous renverra

```
pg_column_size
pg_database_size
pg_database_size
pg_relation_size
pg_relation_size
pg_tablespace_size
pg_tablespace_size
pg_total_relation_size
pg_total_relation_size
```

F.2 Script Shell

Avec MinGW, il est possible par exemple d'utiliser le script suivant

```
hote="localhost"
base="madatabase"
port="5432"
user="david"

liste_Tables=$(psql -h ${hote} -p ${port} -U ${user} -d ${base} -c "select \
tablename from pg_tables where (tablename not like 'pg_%)' and \
(tablename not like 'sql%' ) order by tablename" \
| grep -v "\-\-\-\-" | grep -v "tablename" | grep -v "(")"

for j in ${liste_Tables}
do
echo -n $j.....
Taille_Table="select \
pg_size.pretty(tablesize+indexsize+toastsize+toastindexsize) \
FROM \
(SELECT pg_relation_size(cl.oid) AS tablesize, \
COALESCE((SELECT SUM(pg_relation_size(indexrelid))::bigint \
FROM pg_index WHERE cl.oid=indrelid), 0) AS indexsize, \
CASE WHEN \
    reltoastrelid=0 THEN 0 \
    ELSE pg_relation_size(reltoastrelid) \
END AS toastsize, \
CASE WHEN reltoastrelid=0 THEN 0 \
    ELSE pg_relation_size((SELECT reltoastid FROM pg_class ct \
                           WHERE ct.oid = cl.reltoastrelid)) \
END AS toastindexsize \
FROM pg_class cl \
      WHERE relname = '${j}' ) ss"

psql -h ${hote} -p ${port} -U ${user} -d ${base} -c "${Taille_Table}" | \
grep -v "\-\-\-\-" | grep -v "pg_size.pretty" | grep -v "("
done
```

qui nous renverra en sortie par exemple

```
buildings..... 40 kB
geometry_columns..... 24 kB
great_roads..... 40 kB
mapserver_desc..... 32 kB
parcs..... 40 kB
personnes..... 40 kB
rivers..... 40 kB
small_roads..... 40 kB
spatial_ref_sys..... 2104 kB
```

F.3 Script PHP

Est proposé ci-dessous un petit script php permettant de connaître la taille de chacune des tables d'une base. Est pris en compte le toasting et l'indexage éventuel (voir la doc officielle de PostgreSQL pour de plus amples informations)

```
1 <?php
2 //
3 // Ce script permet de connaitre la taille des diverses tables contenues sur la
4 // base ...
5 //
6 // Paramètres à adapter pour la connexion à PostgreSQL
7 //
8 $hote="localhost";
9 $base_de_donnees="ma_base_de_donnees";
10 $utilisateur="utilisateur";
11 $mot_de_passe="mot_de_passe";
12 //
13 // Chargement de la librairie PostgreSQL pour PHP
14 //
15 switch (PHP_OS)
16 {
17
18 case "WINNT": if (!extension_loaded('pgsql')) dl("pgsql.dll");
19         break;
20
21 default: if (!extension_loaded('pgsql')) dl("php_pgsql.so");
22         break;
23
24 }
25 //
26 // Connexion au serveur PostgreSQL
27 //
28 $db_handle = pg_connect("host=".$hote." dbname=".$base_de_donnees."
29                         user=".$utilisateur." password=".$mot_de_passe."");
30
31 $Requete_Listing_Table = "select tablename as nom_table from pg_tables
32 where (tablename not like 'pg_%') and
33 (tablename not like 'spatial_ref_sys' ) and (tablename not like 'sql%' )
34 and (tablename not like 'geom%' ) order by tablename";
35
36 $Resultat_Listing_Table = pg_exec( $db_handle, $Requete_Listing_Table);
37
38 echo "<table border='1'>";
39
40 for($Iter_Table=0;$Iter_Table<pg_numrows($Resultat_Listing_Table);$Iter_Table++)
41 {
42     $Nom_Table = pg_result( $Resultat_Listing_Table,$Iter_Table,0);
43
44
45     $Requete_Table_Size = "SELECT
46         pg_size.pretty(tablesize+indexsize+toastsize+toastindexsize) AS totalsize
47     FROM
48     (SELECT pg_relation_size(cl.oid) AS tablesize,
49         COALESCE((SELECT SUM(pg_relation_size(indexrelid))::bigint
50                     FROM pg_index WHERE cl.oid=indrelid), 0) AS indexsize,
51         CASE WHEN reltoastrelid=0 THEN 0
52             ELSE pg_relation_size(reltoastrelid)
53         END AS toastsize,
54         CASE WHEN reltoastrelid=0 THEN 0
55             ELSE pg_relation_size((SELECT reltoastid FROM pg_class ct
56                                     WHERE ct.oid = cl.reltoastrelid))
57     )
```

```
57     END AS toastindexsize
58 FROM pg_class cl
59 WHERE relname = '". $Nom_Table ."' ) ss";
60
61 $Resultat_Table_Size = pg_result( pg_exec( $db_handle, $Requete_Table_Size ), 0, 0 );
62
63 echo "<tr><td>". $Nom_Table ."</td><td>". $Resultat_Table_Size ."</td></tr>\n";
64
65 }
66
67 echo "</table>";
68 pg_close($db_handle);
69
70 ?>
```

Annexe G

PostgreSQL et Stunnel

G.1 Introduction

Pour protéger ses données lors d'une connexion entre un serveur et un client avec PostgreSQL, plusieurs solutions existent : soit en utilisant PostgreSQL compilé avec OpenSSL en natif, utiliser un tunnel SSH avec redirection de Port ect... Un des moyens que j'aime bien est d'utiliser Stunnel/OpenSSL.

Stunnel est une enveloppe SSL, permettant donc d'étendre les fonctionnalités de SSL à un démon qui à l'origine n'est pas prévu pour être une couche de sécurité. On peut donc par exemple créer une connexion sécurisée entre vers une base de données, consolidant ainsi la connexion du système.

Un autre intérêt de l'installation avec Stunnel est qu'il peut-être installé en tant que service (automatiquement relancé au démarrage de la machine). Ce que ne propose pas une redirection par SSH. Je ne propose pas ici l'installation de stunnel avec xinetd, trop contrariant à mon sens.

Les tests présentés ici ont été effectués sur une Ubuntu Dapper et une Ubuntu Edgy Eft (environnement Debian). Ce chapitre ne couvre pas l'utilisation de Stunnel et de PostgreSQL sous Windows bien que cela soit possible.

G.2 Pré-requis

Je pars du principe ici que les configurations réseaux avec PostgreSQL sont acquises par le lecteur. Pour les test ici, nous aurons besoin de deux machines. Sur mon réseau domestique, j'ai deux machines dont les noms sont respectivement jenna et bremko :

1. jenna dont l'IP est 192.168.0.5 fera office de serveur. Je lui ai installé un serveur PostgreSQL 8.1.5 ; On s'assure aussi que le paramètre **listen_addresses = '*'** est activé dans le fichier **postgresql.conf** de configuration de PostgreSQL.
2. bremko dont l'IP est 192.168.0.4 fera office de client. On s'assurera d'avoir un logiciel client de PostgreSQL comme pgadmin3 ou psql sur la machine cliente. Ici c'est psql que j'utiliserais.

Pour vérifier que les données sont bien chiffrées où non, nous avons besoin d'installer un sniffer comme nast ou etherreal (...) entre jenna et bremko ! J'opte ici pour NAST (Network Analyze Sniffer Tool/http://nast.berlios.de/) que j'installe en faisant - en tant que root - sur bremko

Exemple G.1 Installation de nast

```
apt-get install nast
```

Pour les besoins de mes tests, je crée un super-utilisateur damien sur jenna dont le mot de passe sera 'morphine'

Exemple G.2 PostgreSQL sur la machine-serveur : création d'un super-utilisateur damien ayant pour mot de passe 'morphine'

```
root@jenna:/root$ su postgres
postgres@jenna:/root$ createuser -sEPe damien
Entrez le mot de passe pour le nouvel rôle :
Entrez-le de nouveau :
CREATE ROLE damien ENCRYPTED PASSWORD 'morphine' SUPERUSER CREATEDB CREATEROLE INHERIT ←
    LOGIN;
CREATE ROLE
```

G.3 Motivations : sniffer une connexion non sécurisée avec NAST, limites d'une connexion par mot de passe en md5 !

Dans cette section, nous allons mettre en évidence l'intérêt de sécuriser une connexion avec PostgreSQL.

G.3.1 Test sans mot de passe

Commençons donc par vérifier que si je laisse le mot de passe par défaut sur à 'password' sur le réseau on arrive quand même à le récupérer. Pour celà, dans le pg_hba.conf de jenna, je fais la modification suivante

Exemple G.3 Modification du fichier de configuration (pg_hba.conf) sur jenna pour une connexion par mot de passe classique

```
host 192.168.0.4 255.255.255.255 password
```

et je redémarre ensuite mon serveur

Exemple G.4 Redémarrage du serveur

```
/etc/init.d/postgresql restart
```

Sur bremko, dans un terminal pour sniffer les connexion je fais

Exemple G.5 Nast sur bremko : sniffage des paquets envoyés à jenna sur le port 5432

```
nast -i eth0 -pd -f "dst 192.168.0.5" -f "port 5432"
```

Dans un autre terminal (toujours depuis bremko), j'ouvre ma connexion à jenna en faisant

Exemple G.6 Connexion à jenna depuis bremko

```
psql -h jenna -U damien template1
```

Dans le terminal que contient le processus actif de nast, je vois a un moment passé le message

Exemple G.7 Portion de paquets interceptionnés

```
---[ TCP Data ]-----  
      ( user damien database template1  
---[ TCP ]-----  
192.168.0.5:5432(postgresql) -> 192.168.0.4:50884(unknown)  
TTL: 64          Window: 1448      Version: 4      Length: 61  
FLAGS: ---PA--  SEQ: 1495197601 - ACK: 923679297  
Packet Number: 26  
  
---[ TCP Data ]-----  
  
R  
---[ TCP ]-----  
192.168.0.4:50884(unknown) -> 192.168.0.5:5432(postgresql)  
TTL: 64          Window: 1460      Version: 4      Length: 66  
FLAGS: ---PA--  SEQ: 923679297 - ACK: 1495197610  
Packet Number: 27  
  
---[ TCP Data ]-----  
  
morphine  
---[ TCP ]-----
```

preuve que ça ne suffit pas !

G.3.2 Test avec MD5

Allons donc ! Changeons donc le mot-clé '**password**' par '**md5**' dans le fichier pg_hba.conf de jenna

Exemple G.8 Modification du fichier de configuration de jenna (pg_hba.conf) : connexion par md5

```
host 192.168.0.4 255.255.255.255 md5
```

et redémarrons le serveur (**/etc/init.d/postgresql restart**). Relançons donc une connexion au serveur depuis bremko. Maintenant depuis nast, j'obtiens

Exemple G.9 Réception de paquets par Nast : mise en évidence de l'intérêt de md5

```
---[ TCP Data ]-----  
      ( user damien database template1  
---[ TCP ]-----  
192.168.0.5:5432(postgresql) -> 192.168.0.4:45585(unknown)  
TTL: 64          Window: 1448      Version: 4      Length: 65  
FLAGS: ---PA--  SEQ: 1852264915 - ACK: 1258303094  
Packet Number: 85  
  
---[ TCP Data ]-----  
  
R          +F  
---[ TCP ]-----  
192.168.0.4:45585(unknown) -> 192.168.0.5:5432(postgresql)  
TTL: 64          Window: 1460      Version: 4      Length: 93  
FLAGS: ---PA--  SEQ: 1258303094 - ACK: 1852264928  
Packet Number: 86  
  
---[ TCP Data ]-----  
  
p  (md5063971165646bb85c855953e66c01196  
---[ TCP ]-----
```

NOTE

Extrait de la documentation française de PostgreSQL concernant la méthode MD5
La méthode d'authentification MD5 crypte deux fois le mot de passe sur le client avant de l'envoyer au serveur. Il le crypte tout d'abord à partir du nom de l'utilisateur puis il le crypte à partir d'un élément du hasard envoyé par le serveur au moment de la connexion. Cette valeur, deux fois cryptée, est envoyée sur le réseau au serveur. Le double cryptage empêche non seulement la découverte du mot de passe, il empêche aussi une autre connexion en rejouant la même valeur de double cryptage dans une connexion future.

Gagné ! Enfin ne nous réjouissons pas trop vite car si je tape une requête depuis le client je vois apparaître par exemple

Exemple G.10 Réception de paquets par Nast : insuffisance pour la sécurisation de la méthode

```
--[ TCP Data ]-----  
Q %select * from geometry_columns ;ima  
---[ TCP ]-----  
192.168.0.5:5432(postgresql) -> 192.168.0.4:48508(unknown)  
TTL: 64 Window: 6091 Version: 4 Lenght: 52  
FLAGS: ----A-- SEQ: 2056534420 - ACK: 1484467451  
Packet Number: 161  
  
---[ TCP ]-----  
192.168.0.5:5432(postgresql) -> 192.168.0.4:48508(unknown)  
TTL: 64 Window: 6091 Version: 4 Lenght: 1500  
FLAGS: ----A-- SEQ: 2056534420 - ACK: 1484467451  
Packet Number: 162  
  
---[ TCP Data ]-----  
  
T f_table_catalog f_table_schema f_table_name ←  
    f_geometry_column  
coord_dimension srid type " D E ←  
    public    apb_lr the_geom 2  
-1   MULTIPOLYGOND U public a_cours_d_eau_n2_v3 the_geom 2 -1 ←  
    MULTILINESTRING
```

Oups !

En effet, depuis bremko j'ai envoyé la requête

```
select * from geometry_columns
```

à jenna. Les résultats me sont envoyés en claire comme me le confirme la ligne **192.168.0.5 :5432(postgresql) 192.168.0.4 :48508(unknown)** ainsi que le reste des résultats.

Première conclusion : Bon md5 protège bien mon mot de passe mais pas mes requêtes ainsi que les résultats renvoyés par le serveur ! Et c'est là justement qu'intervient Stunnel !

G.4 Stunnel : sécurisation de la connexion

Nous devons commencer par installer OpenSSL pour utiliser Stunnel par la suite

G.4.1 Pré-requis : OpenSSL

Stunnel a besoin de OpenSSL pour pouvoir fonctionner

Exemple G.11 Pré-requis pour l'installation de Stunnel

```
apt-get install openssl libssl-dev
```

G.4.2 Installation de Stunnel

Le site de stunnel est <http://www.stunnel.org>. Nous aurons besoin ici de l'installer à la fois sur le serveur et sur le client. Je fournis ici les commandes que j'ai utilisé pour l'installer sans plus de détails

Exemple G.12 Installation de Stunnel

```
wget http://www.stunnel.org/download/stunnel/src/stunnel-4.20.tar.gz
tar xvzf stunnel-4.20.tar.gz
cd stunnel-4.20
./configure --with-ssl=/usr --prefix=/opt/stunnel
make
make install
```

L'installation aura donc lieu ici dans le répertoire /opt/stunnel mais vous pouvez l'installer où bon vous semble.

Lors de l'installation, un certificat auto-signé **stunnel.pem** sera généré dont voici une copie d'écran de ce que j'ai renseigné pour jenna

Exemple G.13 Génération du certificat auto-signé

```
Generating a 1024 bit RSA private key
+++++
writing new private key to 'stunnel.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [PL]:FR
State or Province Name (full name) [Some-State]:Hérault
Locality Name (eg, city) []:Castelnau-Le-Lez
Organization Name (eg, company) [Stunnel Developers Ltd]:01MAP
Organizational Unit Name (eg, section) []:01MAP
Common Name (FQDN of your server) [localhost]:jenna
```

Je renouvelle l'instalalction sur bremko, en changeant la ligne **Common Name (FQDN of your server) [localhost] :bremko**

Le certificat autu-signé **stunnel.pem** est utilisé ici uniquement sur la base de test et pour les besoins de l'article. Il pourra être remplacé plus tard par un certificat signé par une autorité de certification (CA).

G.4.3 Mise en oeuvre

J'ouvre maintenant un terminal sur jenna et j'y tape

Exemple G.14 Stunnel en tant que serveur sur jenna

```
root@jenna:/opt/stunnel/sbin/stunnel3 -D 7 -f -P /opt/stunnel/stunnel.pid -o /opt/stunnel/ ←
stunnel.log -p /opt/stunnel/etc/stunnel/stunnel.pem -d 9000 -r localhost:5432
```

avec les options suivantes :

- D 7 : pour spécifier le niveau de déboggage afin de collecter le maximum d'informations ;
- f : me permet de faire tourner le processus en arrière-plan dans le terminal. Un simple [CTRL]+[C] me suffira donc pour arrêter le processus si besoin est ;

- P /opt/stunnel/stunnel.pid pour le PID du processus ;
- o /opt/stunnel/stunnel.log pour spécifier un fichier de log ;
- p /opt/stunnel/etc/stunnel/stunnel.pem pour le certificat auto-signé

Un processus démon de Stunnel par cette commande est donc lancée. Le paramètre **-d 9000** est le port d'écoute du démon et lui demande d'attendre de données chiffrées sur ce port. Le paramètre **-r localhost :5432** indique au démon que les données reçues sur son port d'écoute (9000), il devra les déchiffrer et les envoyer sur le port 5432 de la machine locale. Or ce dernier port n'est autre que le port d'écoute du serveur PostgreSQL sur jenna.

Sur bremko, je lance

Exemple G.15 Stunnel en tant que client sur bremko

```
root@bremko:/opt/stunnel# /opt/stunnel/sbin/stunnel3 -D 7 -f -P/opt/stunnel/stunnel.pid -o ↵
    /opt/stunnel/log/stunnel.log -c -d localhost:5433 -c -d localhost:5433 -r ↵
    192.168.0.5:9000
```

Dans la commande précédente c'est la portion **-c -d localhost :5433 -r 192.168.0.5 :9000** qui nous intéresse. Une instance de Stunnel est ainsi lancée en mode client grâce au paramètre **-c** et lui demande d'écouter sur le port 5433. Le paramètre **-r 192.168.0.5 :9000** indique à cette instance que la machine-serveur (jenna) a pour adresse 192.168.0.5 et que son port d'écoute est le 9000.

Pour m'assurer que les connexions seront chiffrées dans un troisième terminal sur bremko, je fais

```
nast -pd -i eth0 -f "dst 192.168.0.5" -f "port 9000"
```

Dans un nouveau terminal toujours - sur bremko - j'ouvre maintenant une connexion PostgreSQL en faisant

```
psql -h localhost -p 5433 -U damien direnlr
```

Je saisirai quelques requêtes et j'apprécie le travail en regardant les lignes retournées par nast depuis le troisième terminal en question. Par exemple pour la requête

```
direnlr=# select * from geometry_columns limit 1;
-[ RECORD 1 ]-----+
f_table_catalog | 
f_table_schema | public
f_table_name | apb_lr
f_geometry_column | the_geom
coord_dimension | 2
srid | -1
type | MULTIPOLYGON
```

j'obtiens avec nast

Exemple G.16 Paquets réceptionnés : Sécurisation de la connexion

```
---[ TCP ]-----
192.168.0.4:45480(unknown) -> 192.168.0.5:9000(unknown)
TTL: 64      Window: 2546      Version: 4      Length: 126
FLAGS: ---PA-- SEQ: 2814905551 - ACK: 2149157549
Packet Number: 1

---[ TCP Data ]-----

' jp;o , 9      e nH} .

---[ TCP ]-----
192.168.0.5:9000(unknown) -> 192.168.0.4:45480(unknown)
TTL: 64      Window: 1984      Version: 4      Length: 126
FLAGS: ---PA-- SEQ: 2149157549 - ACK: 2814905625
Packet Number: 2

---[ TCP Data ]-----

D n~ % N 4; r M { g.X= > q      v 1      -N{W      u 0 A 9 *41 _K
---[ TCP ]-----
192.168.0.4:45480(unknown) -> 192.168.0.5:9000(unknown)
TTL: 64      Window: 2546      Version: 4      Length: 52
FLAGS: ----A-- SEQ: 2814905625 - ACK: 2149157623
Packet Number: 3
```

G.5 Installation en service de Stunnel

Pour l'installation en service sur les deux machines, il suffit de modifier le script de démarrage de Ubuntu à savoir **/etc/init.d/rc.local**. J'y ai par exemple effectuer les modifications suivantes sur jenna

Exemple G.17 Script de démarrage pour stunnel en tant que service sur la machine-serveur

```
#!/bin/sh
# Modications du PATH pour accéder à stunnel
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/opt/stunnel/sbin
[ -f /etc/default/rcS ] && . /etc/default/rcS
. /lib/lsb/init-functions

do_start() {
    if [ -x /etc/rc.local ]; then
        log_begin_msg "Running local boot scripts (/etc/rc.local)"
        /etc/rc.local
        log_end_msg $?
    fi
}
# Ma ligne pour lancer stunnel sur jenna (Serveur PostgreSQL)
# à adapter en conséquence sur bremko (voir ci-dessus dans la doc)
/opt/stunnel/sbin/stunnel3 -D 7 -f -P /opt/stunnel/stunnel.pid -p /opt/stunnel/etc/stunnel/ ←
    stunnel.pem -d 9000 -r localhost:5432 -o /opt/stunnel/log/stunnel.log

case "$1" in
    start)
        do_start
        ;;
    restart|reload|force-reload)
        echo "Error: argument '$1' not supported" >&2
        exit 3
        ;;
    stop)
        ;;
    *)
        echo "Usage: $0 start|stop" >&2
        exit 3
        ;;
esac
```

Ne pas oublier d'adapter votre script de démarrage sur la machine cliente

ATTENTION

 Ne pas oublier d'adapter votre script de démarrage (**/etc/init.d/rc.local** (pour Ubuntu) ou **/etc/rc.d/rc.local...**) sur la machine-cliente pour la commande

```
/opt/stunnel/sbin/stunnel3 -D 7 -f -P /opt/stunnel/stunnel.pid -o /opt/stunnel/log/ ←
    stunnel.log -c -d localhost:5433 -o /opt/stunnel/log/stunnel.log -c -d localhost ←
    :5433 -r 192.168.0.5:9000
```

G.6 Pour aller plus loin

Il est tout à fait possible avec Stunnel d'utiliser le fichier de configuration de stunnel dans `/opt/stunnel/etc/stunnel` ou lui proposer un fichier personnel de configuration. Je ne me suis pas attarder à le proposer dans cette article afin de ne pas trop le surcharger.

Annexe H

MapServer : faire une image avec zones réactives

MapServer permet de générer des images au format raster (png,gif,jpg). Mais il est aussi possible de pouvoir générer des images avec des zones réactives.



Avertissement

Avant de consulter ce chapitre merci de vous avoir assurer que vous avez consulté le chapitre 4, notamment la section "Cas pratiques avec MapServer"

H.1 Démo en ligne et ressource en ligne

La ressource qui a permis la rédaction de ce chapitre peut être consulté à <http://mapserver.gis.umn.edu/cgi-bin/wiki.pl?ImageMap>
La démo est visible [ici](#)

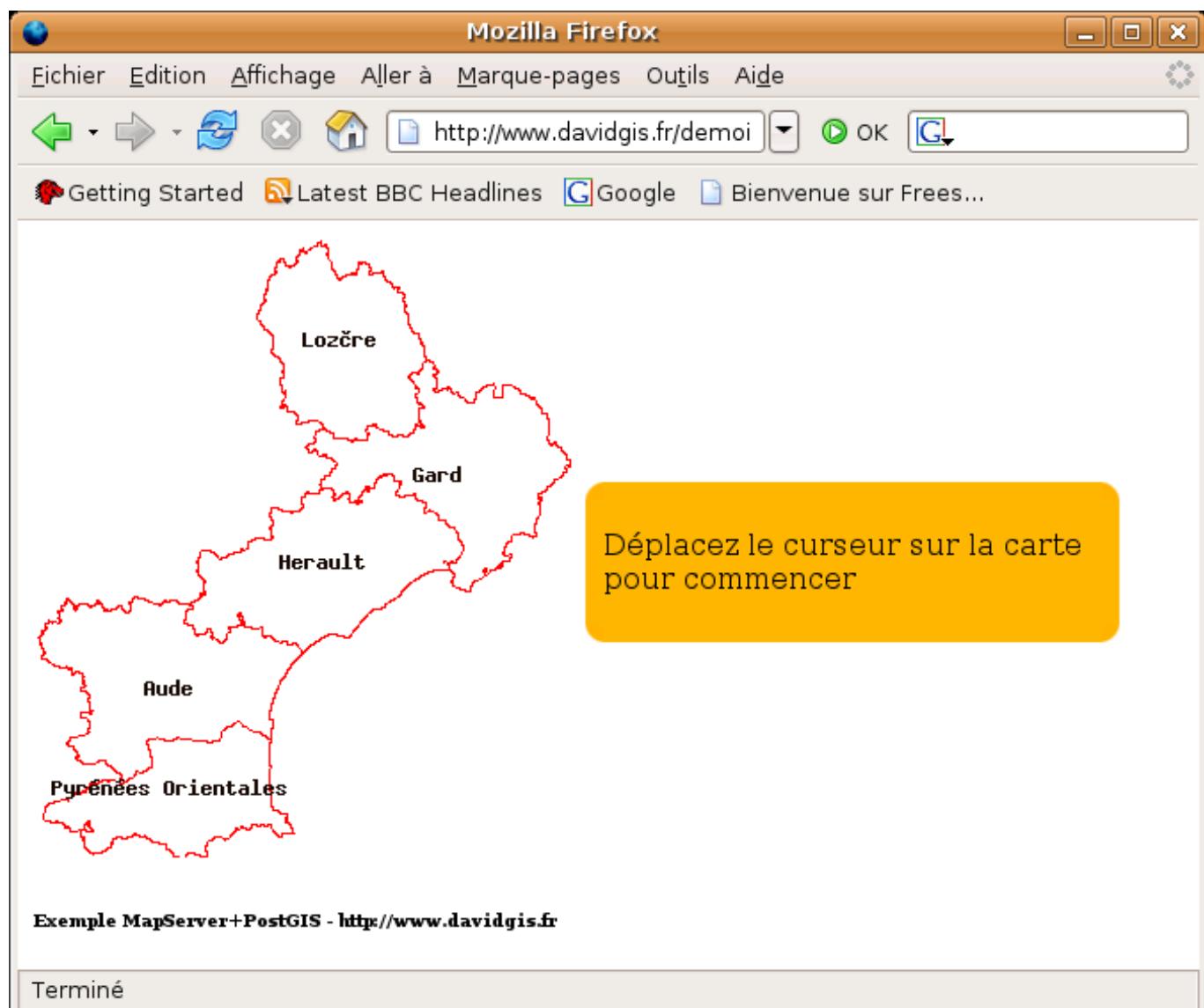


FIG. H.1 – ImageMap à l'initialisation

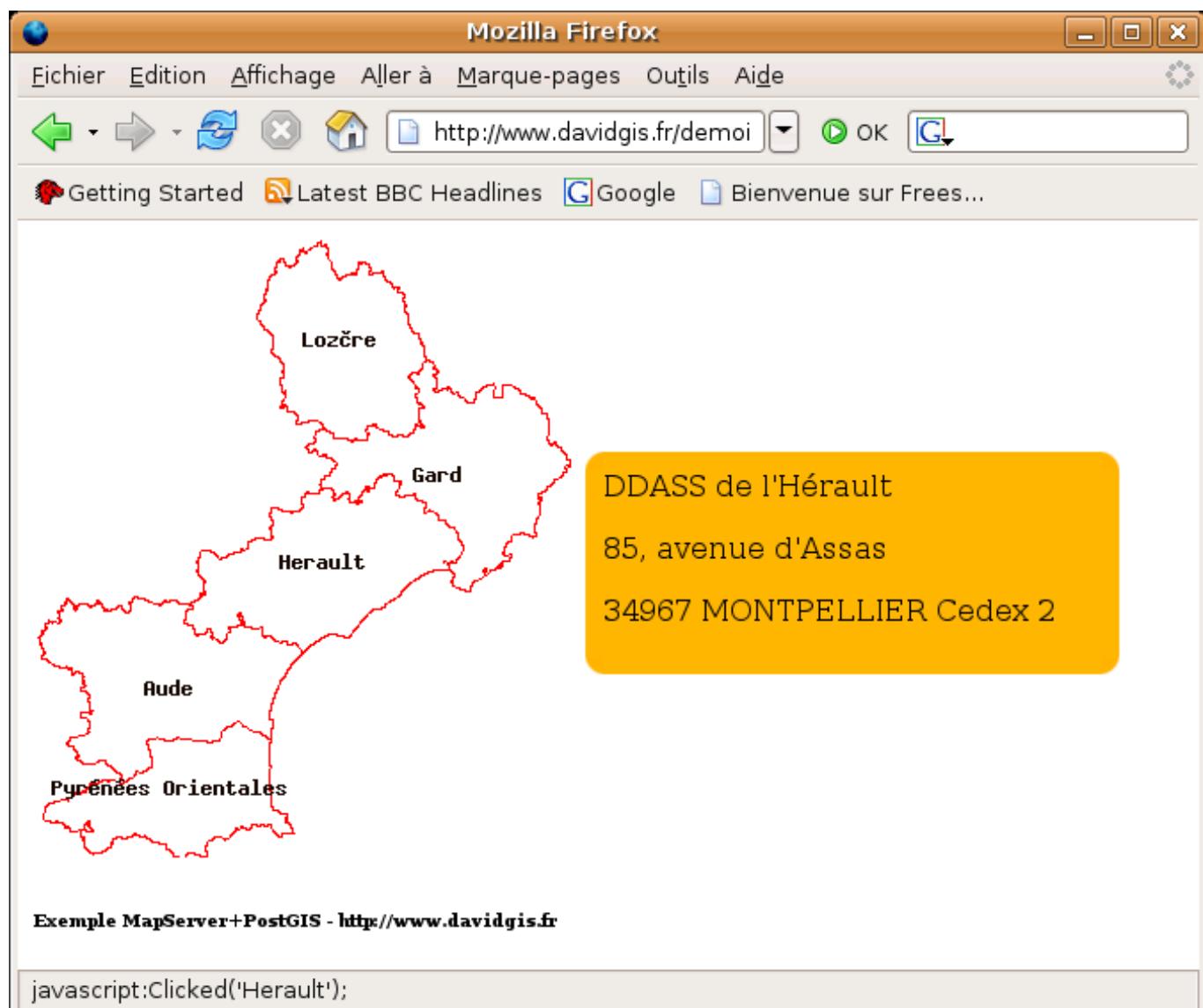


FIG. H.2 – ImageMap lors du passage du pointeur de la souris sur l'Hérault

H.2 Création de la base de données

La création de la base de données aura lieu en faisant

Exemple H.1 Création de la base de données

```
createdb madatabase
createlang plpgsql madatabase
psql -d madatabase -f C:\PostgreSQL\8.2.1\share\contrib\lwpostgis.sql
psql -d madatabase -f C:\PostgreSQL\8.2.1\share\contrib\spatial_ref_sys.sql
```

H.3 Importation des données dans une base PostGIS

Nous allons importer nos données en faisant tout simplement

Exemple H.2 Importation des données issues du shapefile dans la base

```
shp2pgsql -DI -s 27582 departements_lr.shp departements_lr | psql madatabase
```

Les données peuvent être téléchargées [ici](#).

H.4 La mapfile

La mapfile se présente ainsi

Exemple H.3 Contenu de la mapfile

```
1 MAP
2   EXTENT 547293 1703209.875 801423 1997767.125
3   IMAGECOLOR 125 125 125
4   SIZE 500 579.540491087239
5   FONTSET "../etc/fonts.txt"
6   SYMBOLSET "../etc/symbols.sym"
7   =====
8   # Les divers outputs possibles
9   =====
10  #
11  # Pour le raster en png
12  #
13  OUTPUTFORMAT
14    NAME png
15    MIMETYPE image/png
16    DRIVER GD/PNG
17    EXTENSION png
18    IMAGEMODE PC256
19    TRANSPARENT FALSE
20 END
21 #
22 # Pour imagemap: image avec zones réactives
23 #
24 OUTPUTFORMAT
25   NAME imagemap
26   MIMETYPE imagemap
27   DRIVER imagemap
28   EXTENSION html
29   IMAGEMODE PC256
30   TRANSPARENT FALSE
31   # Ce qui permet d'émuler le MouseOver en JavaScript
32   FORMATOPTION "POLYMOUSEOVER=javascript:getdata('%s')"
33 END
34 =====
35 # Web pour MapServer
36 =====
37 WEB
38   IMAGEPATH "/var/www/demoimagemap/tmp/"
39   IMAGEURL "/demoimagemap/tmp/"
40 END
```

Exemple H.4 Contenu de la mapfile (suite)

```
#=====
# Couche des départements
#=====

LAYER
    NAME "departements_lr"
    CONNECTION "user=david dbname=madatabase host=localhost"
    CONNECTIONTYPE POSTGIS
    DATA "the_geom from departements_lr"
    STATUS DEFAULT
    TYPE POLYGON
    LABELITEM "nom"
    CLASS
        LABEL
            SIZE MEDIUM
            TYPE BITMAP
            BUFFER 0
            COLOR 22 8 3
            FORCE FALSE
            MINDISTANCE -1
            MINFEATURESIZE -1
            OFFSET 0 0
            PARTIALS TRUE
            POSITION CC
        END
        STYLE
            OUTLINECOLOR 0 0 255
            SIZE 3
            SYMBOL "circle"
        END
    END
#=====
# FIN DE LA MAPFILE
#=====

END
```

Voyons quelques points importants concernant cette mapfile. Déjà concernant EXTENSION, nous avons vu ce paramètre au chapitre précédent. Le plus important ici réside dans l'ajout des deux sections OUTPUTFORMAT appelé respectivement png et imagemap. En effet ces deux sections sont ici nécessaires :

png : permettra de générer le fond de l'image figée ;

imagemap : permettra de créer les images réactives

Ces deux OUTPUT sont en fait superposés ! Le fond de l'image accueille les zones réactives par-dessus. Maintenant que nous avons les deux fonds attendus, il faut pouvoir les générer à partir de PHP

H.5 Script php

Le script php utilisé est le suivant

Exemple H.5 Script php utilisé pour générer l'image à zones réactives

```
<script type="text/javascript" src="getdata.js"></script>
<?php
/*
    Vérifiacation des chargements des extensions pour
    PostgreSQL et PhpMapScript

    ATTENTION: Prévu pour Windows et GNU/Linux
                Pas encore pour Mac OS X!
*/
switch (PHP_OS)
{
    case "WINNT": $Ext_Suffix = "dll";// suffixe pour Win32
                    break;

    default:   $Ext_Suffix = "so";// Suffixe pour GNU/Linux
                    break;
}

if ( extension_loaded('MapScript')+extension_loaded('pgsql') != 2)
{
    if (!extension_loaded('MapScript'))
    {
        if(!dl("php_mapscript".$Ext_Suffix ))
            {echo "<b>ATTENTION: Impossible de trouver la librairie pour PHPMAPSCRIPT</b>";
             exit("");
            }
        }
    if (!extension_loaded('pgsql'))
    {
        if(!dl("php_pgsql".$Ext_Suffix ))
            {echo "<b>ATTENTION: Impossible de trouver la librairie pour POSTGRESQL</b>";
             exit("");
            }
        }
}

$sw_MapFile = "/var/www/demoimagemap/mapfiles/lr.map";
/* Chargement de la mapfile */
$map = ms_newMapObj( $sw_MapFile );
/* Commençons par charger le fond de couleur en blanc*/
$map->imagecolor->setRGB(255,255,255);
// On modifie les dimensions de l'image
$map->set("width",300);
$map->set("height",347.72429);
/* On agit sur le layer "departements_lr"*/
$layer=$map->getLayerByName("departements_lr");
$layer->set("status",MS_ON);

$class = $layer->getClass(0);
$class->deleteStyle(0);
$style = ms_newStyleObj($class);
$style->set("size",3);
$style->color->setRGB(255,255,255);
$style->outlinecolor->setRGB(255,0,0);

$map->selectOutputFormat("png");
$image = $map->draw();
$image_url = $image->saveWebImage();
```

Exemple H.6 Script php utilisé pour générer l'image à zones réactives (suite)

```
$map->selectOutputFormat("imagemap");
$imagemap = $map->draw();
$imagemap_url = $imagemap->saveWebImage();

printf("<TABLE BORDER='0' BORDERSPACE=0>");
printf("<tr>");
printf("<td>");

echo "<FORM NAME='mymap' METHOD=POST ACTION='".$self."'>";
/*
En fonction, du nvavigateur FireFox ou IE, on adapte s'il faut
ajoutet les balises <object></object> ou pas!
*/
if (substr_count($_ENV["HTTP_USER_AGENT"],"FireFox") > 0)
{
    echo "<OBJECT TYPE='image/png' NAME='mapa' usemap ='#map1' BORDER=0 DATA='http://".$_ENV["SERVER_NAME"].":".$_ENV["SERVER_PORT"].".".$image_url."'>";
    include "http://".$_ENV["SERVER_NAME"].":".$_ENV["SERVER_PORT"].".".$.imagemap_url;
    echo "</OBJECT><BR>";
}
else
{
    echo "<IMG NAME='IEmapa' usemap ='#map1' BORDER=0 SRC='http://".$_ENV["SERVER_NAME"] .
        ".$_ENV["SERVER_PORT"].".".$image_url."/>'><BR>";
    include "http://".$_ENV["SERVER_NAME"].":".$_ENV["SERVER_PORT"].".".$.imagemap_url;
}

echo "</FORM>";
?>
<?php
printf("</td>\n");
printf("<td>");
printf("</td>\n");
?>
<td>
<!--
    Un tableau sur lequel on affiche les données attributaires attendus
-->


<table width="300" cellpadding="0" cellspacing="0" border="0">
        <tr>
            <td width=10></td >
            <td width=280 bgcolor="#FFB400"></td>
            <td width=10></td >
        </tr>
        <tr>
            <td width="10" bgcolor="#FFB400"></td>
            <td width="280" bgcolor="#FFB400">
                <p id="AffichageAdresse1"> </p>
                <p id="AffichageAdresse2">Déplacez le curseur sur la carte pour commencer</p>
                <p id="AffichageAdresse3"> </p>
                <p id="AffichageAdresse4"> </p>
            </td>
        </tr>
    </table>


```

Exemple H.7 Script php utilisé pour générer l'image à zones réactives (suite)

```
<td width="10" bgcolor="#FFB400"></td>
</tr>
<tr>
    <td width="10"></td> ←
    >
    <td width="280" bgcolor="#FFB400"></td>
    <td width="10"></td> ←
    >
</tr>
</table>
</div>
</td>
</tr>
</table>


Exemple MapServer+PostGIS - http://www.davidgis.fr


</b></p>
```

La mapfile interagit avec la fonction `getdata()` (voir le paramètre `FORMATOPTION "POLYMOUSEOVER=javascript :getdata('%s')"` de la mapfile) qui est écrit ici en JavaScript. On mettra ici le code de cette fonction dans un fichier supplémentaire `getdata.js` dont le contenu est

Exemple H.8 Script JavaScript utilisé pour un peu d'interactivité avec l'image à zones réactives

```
function getdata(departement)
{
dept = new Array(
"DDASS de l'Aude","14 rue rue du 4 septembre - BP 48","CARCASSONNE Cedex",
"DDASS du Gard","6, rue du Mail","30906 NIMES Cedex",
"DDASS de l'Hérault","85, avenue d'Assas","34967 MONTPELLIER Cedex 2",
"DDASS de la Lozère","Avenue du 11 novembre 1918","Immeuble le Saint-Clair - BP 136","48005 ←
MENDE Cedex",
"DDASS des Pyrénées-Orientales","12, boulevard Mercader - BP 928","66020 PERPIGNAN Cedex");
switch(departement)
{
    case 'Aude':
        document.getElementById("AffichageAdresse1").firstChild.nodeValue = ←
            dept[0];
        document.getElementById("AffichageAdresse2").firstChild.nodeValue = ←
            dept[1];
        document.getElementById("AffichageAdresse3").firstChild.nodeValue = ←
            dept[2];
        document.getElementById("AffichageAdresse4").firstChild.nodeValue = ←
            '';
        break
    case 'Gard':
        document.getElementById("AffichageAdresse1").firstChild.nodeValue = ←
            dept[3];
        document.getElementById("AffichageAdresse2").firstChild.nodeValue = ←
            dept[4];
        document.getElementById("AffichageAdresse3").firstChild.nodeValue = ←
            dept[5];
        document.getElementById("AffichageAdresse4").firstChild.nodeValue = ←
            '';
        break
    case 'Herault':
        document.getElementById("AffichageAdresse1").firstChild.nodeValue = ←
            dept[6];
        document.getElementById("AffichageAdresse2").firstChild.nodeValue = ←
            dept[7];
        document.getElementById("AffichageAdresse3").firstChild.nodeValue = ←
            dept[8];
        document.getElementById("AffichageAdresse4").firstChild.nodeValue = ←
            '';
        break
    case 'Lozère':
        document.getElementById("AffichageAdresse1").firstChild.nodeValue = ←
            dept[9];
        document.getElementById("AffichageAdresse2").firstChild.nodeValue = ←
            dept[10];
        document.getElementById("AffichageAdresse3").firstChild.nodeValue = ←
            dept[11];
        document.getElementById("AffichageAdresse4").firstChild.nodeValue = ←
            dept[12];
        break
    default:
        document.getElementById("AffichageAdresse1").firstChild.nodeValue = ←
            dept[13];
        document.getElementById("AffichageAdresse2").firstChild.nodeValue = ←
            dept[14];
        document.getElementById("AffichageAdresse3").firstChild.nodeValue = ←
            dept[15];
        document.getElementById("AffichageAdresse4").firstChild.nodeValue = ←
            '';
        break
}
}
```

Annexe I

Foire Aux Questions

Cet annexe est un recueil de questions/problématiques auxquels nous avons été moi et Gérald souvent été confrontés aussi bien sous GNU/Linux que sous Win32. Nous tentons ici de répondre aux questions les plus pertinentes. Les questions abordées ici sont données en vrac et servent en fait de garde-fou/grenier !

I.1 Existe-il des installateurs pour PostgreSQL et PostGIS qui évitent d'avoir à les compiler soi-même ?

La réponse est oui. La première possibilité est <http://www.postgis.fr>. La deuxième possibilité est de se rendre à pour PostgreSQL : <http://www.postgresql.org/ftp/win32/> . Vous trouverez un exemple d'installation à cet adresse <http://fadace.developpez.com/postgresql/> IMPORTANT : LORS DE L'INSTALLATION NE PRENEZ PAS LE SUPPORT 'EXTENSION SPATIAL POSTGIS' SINON IL VOUS INSTALLERA UNE ANCIENNE VERSION DE POSTGIS !!! PgAdmin est livré avec cette installeur.

pour PostGIS (incluant Geos et Proj) : <http://postgis.refractions.net/download/windows/> tenu par Marc CAVE AYLAND

I.2 Quel logiciel utilisé pour gérer/administrer un/des serveurs PostgreSQL sous Windows ?

Le mieux à mon sens est de recourir à pgadmin qui est totalement libre d'utilisation. Vous avez aussi le portage de ce projet sous php = phppgadmin , disponible à <http://www.phppgadmin.org/> . Vous pouvez télécharger pgadmin sur le site <http://www.pgadmin.org>

I.3 Quel logiciel utilisé pour visualiser ses données de PostGIS ?

Il y a plusieurs projets pour cela. A part MapServer, citons ici comme projets déjà prometteurs en Java UDIG : <http://udig.refractions.net> . Réalisé par le Gourou de PostGIS : Sir Paul RAMSEY !!! en Qt QGIS : <http://qgis.org> dont la version 0.7 fonctionne pour PostGIS >=1.0.X pour Windows. La version 0.6 est utilisable pour Windows mais elle ne fonctionnait que pour PostGIS 0.9.X.

Nous ne faisons ici que citer ceux qui nous paraissent pertinents. D'autres existent aussi : JUMP, FME etc...

I.4 Comment migrer des données de PostGIS à travers un réseau intranet/extranet ?

C'est une question qui revient souvent surtout lorsqu'on doit alimenter par exemple un serveur cartographique. Dans ce cas d'étude, nous sommes donc en présence de deux machines. L'une fait office de client dans le sens où elle doit accueillir des données, l'autre fait office de serveur dans le sens où elle doit "offrir" les données. Ici nous allons supposer que l'utilisateur se trouve sur la machine-serveur.

Sur les deux machines, on s'assurera que celle-ci autorise les connexions TCP/IP pour PostgreSQL - voir pour cela le chapitre "Paramétriser PostgreSQL" -"section Autoriser les machines du réseau intranet/extranet à se connecter au serveur". Donc les connexions doivent être ouvertes sur la machine de l'utilisateur et sur l'autre.

Dans le meilleur des mondes, le deuxième impératif serait que les deux machines est la même version de PostgreSQL. Cette information peut-être obtenu en faisant

1. pour le côté client :

```
psql -h [ip_hote_client] -U [user_client] -d [base_client] -c "SELECT Version()"
```

2. pour le côté serveur

```
psql -d [base_serveur] -c "SELECT Version()"
```

Si les deux versions ne correspondent pas, dans le meilleur des cas, il est préférable d'avoir Version_Serveur < Version_Client. On est au moins sûr dans ce cas par exemple si on doit migrer des données provenant d'une 7.4.X version un 8.0.X d'assurer la migration surtout s'il s'agit de toutes les tables d'une base.

La commande générale à utiliser est alors pour migrer une table, si l'utilisateur est côté serveur :

```
pg_dump -cio -t [table_serveur] [base_serveur] | psql -h [ip_hote_client] -U [user_client] ←  
-d [base_client]
```

N'hésitez pas pour obtenir une meilleure information sur l'emploi de pg_dump d'utiliser

```
pg_dump --help
```

I.5 Comment passer du format MapInfo à PostGIS ?



AVERTISSEMENT

Je présente ici deux façons d'utiliser ogr2ogr. Ce dernier est livré avec Gdal <http://www.remotesensing.org/gdal/>. Les options que j'ai utilisé ici ne marchent que pour la version >= 1.3.2 de Gdal.

Il existe dans la distribution de l'outil Gdal, un utilitaire **ogr2ogr** - disponible aussi sous win32 - qui permet de passer de MapInfo à PostGIS. Par exemple pour importer le fichier Directive_Habitats_30.MIF (format mif/mid de MapInfo), dans la base diren, je peux faire

```
ogr2ogr -f PostgreSQL PG:'host=localhost dbname=ingeresin user=postgres' -lco GEOMETRY_NAME ←  
=the_geom Directive_Habitats_30.MIF
```

-lco GEOMETRY_NAME=the_geom est important sinon votre colonne spatiale s'appellera **wkb_geometry** par défaut
PG :'host=XXX dbname=XXX user=XXX' permet la connexion à la base

Personnellement, je préfère couper la poire en deux : utiliser ogr2ogr pour une première conversion avec ogr2ogr MapInfo ---> Shapefile puis une deuxième conversion avec shp2pgsql Shapefile ---> PostGIS. En effet, si vous compilez les sources de Gdal sous GNU/Linux, la conversion directe n'importera pas les données avec les structures/schémas usuels de PostGIS.

NOTE

Ogr2ogr faisant parti de la distribution de gdal est téléchargeable sur le site <http://www.maptools.org>.

Sous Win32, pour effectuer les deux conversions pour par exemple un fichier communes.tab, on effectuera les commandes suivantes :

Pour la méthode exposée ici, nous prendrons en considération que si les champs des colonnes de vos données attributaires sont encodées sur plus de 10 caractères, ce sont ces 10 premiers caractères qui seront conservées, notamment lors de la conversion de MapInfo vers Shapefile. Il en résulte que les autres caractères seront donc perdus. Par exemple si votre fichier de MapInfo contient une colonne nommée "communes_de_Lille", le shapefile contiendra la colonne "communes_de_". Celà est dû notamment pour des raisons historiques dans l'implémentation des Shapefiles.

1. de MapInfo vers Shapefile

```
ogr2ogr -f "ESRI Shapefile" communes.shp communes.tab
```

2. de Shapefile vers PostGIS

```
shp2pgsql -s [srid] -I -D communes.shp | psql [base]
```

I.6 PostGIS : Comment passer de PostGIS à ESRI Shapefile ?

Il faut pour cela utiliser l'utilitaire **pgsql2shp** qui est disponible avec PostGIS. La commande la plus courante est

```
pgsql2shp [base] [table]
```

qui produit directement le fichier shp ayant le même nom que la table. Il est également possible de créer un fichier shp résultant d'une requête :

```
pgsql2shp -f [nom_shapefile].shp [base] "[REQUETE]"
```

I.7 PostGIS : Est-il possible de calculer l'extent sur des objets de nature géométrique différente ?

La réponse est oui ! Prenons ici l'exemple des données de la table test du chapitre 4, "Exemples de requêtes spatiales I" où les données correspondent au cas envisagé ici.

```
# select geometrytype(geom) from test;
geometrytype
-----
POINT
POINT
POLYGON
POLYGON
POLYGON
POINT
POINT
LINESTRING
LINESTRING
MULTILINESTRING
(10 lignes)
```

Et on obtiendra

```
# select extent(geom) from test;
   extent
-----
 BOX(1 4,50 135)
(1 ligne)
```

I.8 PostgreSQL : Comment exporter des données au format CSV ?

Il y a bien plusieurs façons de faire. Je donnerais en premier lieu la manière de le faire avec psql , ensuite avec ogr2ogr. Je vais ici prendre l'exemple des données du chapitre 4,"Exemples de requêtes spatiales II" où sur les champs de la table buildings, je veux les données attributaires en renommant :

le champs "id" en "ID" ;
le champs "data" en "Nom du bâtiment".

Je veux pouvoir donc générer un fichier CSV, que je pourrais par la suite ouvrir avec OOpenOffice Calc ou Excel dont le contenu ressemblera à

```
ID,Noms du bâtiments
1,Collège Arthur Rimbaud
2,Résidence des Mousquetaires
3,Hotel des Impots
4,E.D.F
5,Bibliothèque Victor Hugo
6,Mairie
7,Office du Tourisme
```

I.8.1 A partir de psql

Un des moyens intéressants est par exemple de faire

```
psql -d testgis -AF "," -c "select id as \"ID\",data as \"Noms du bâtiments\" from ←
buildings" | grep ","
```

I.8.2 Avec ogr2ogr

Je manipule ogr2ogr en faisant

```
ogr2ogr -f "CSV" test.csv -sql "select id as \"ID\",data as \"Nom du bâtiment\" from ←
buildings" "PG:dbname=testgis user=postgres host=localhost password=..."
```

ce qui me génère un répertoire out qui contiendra le fichier nommé sql_statement.csv attendu.



ATTENTION

Avec psql, il est possible de pouvoir exporter aussi les données géométriques. Malheureusement, cela ne semble pas être le cas avec ogr2ogr. Je parle ici sous la garantie de n'avoir testé que la version fournie avec gdal 1.3.1.

I.9 Comment connaître les objets d'une table qui intersectionnent une fenêtre ?

Considérons par exemple la situation suivante exposé sur l'image ci-dessus

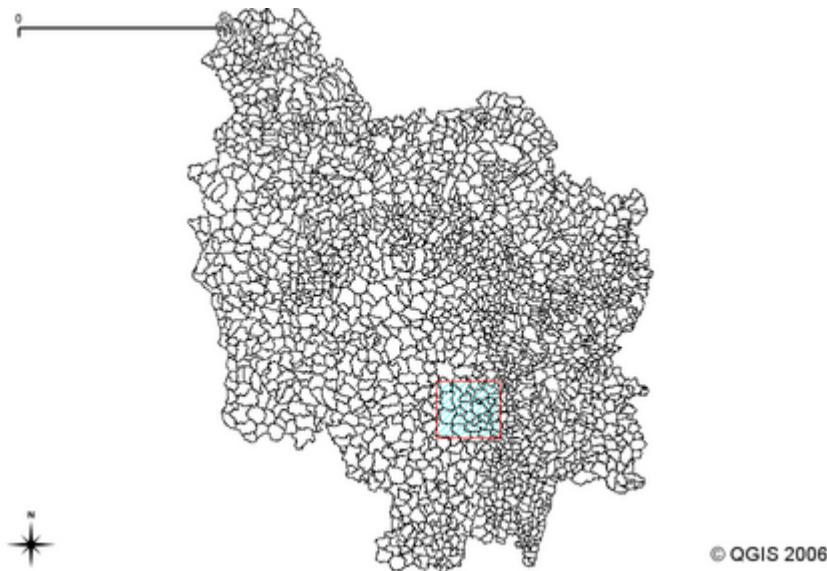


FIG. I.1 – Une fenêtre intersectionnant des objets (ici les communes en région bourgogne)

Pour cela, nous ferons tout simplement

```
# SELECT count(*) FROM communes_bourgogne WHERE the_geom && SetSRID('BOX3D(746387 2189036,773809 2213846)'::box3d,27582);
count
-----
 53
(1 ligne)
```

Nous comprenons ici que la colonne the_geom est ici indexée spatialement.

I.10 Comment convertir une BOX3d en Polygon ?

Prenons comme exemple la BOX3d de la sous-section précédente. BOX3d(746387 2189036,773809 2213846) Pour la convertir, il suffit tout simplement d'utiliser la fonction BuildArea() qui ici fonctionnera car une BOX3d est un objet fermé

```
bourgogne=# SELECT astext(buildarea(SetSRID('BOX3D(746387 2189036,773809 2213846)'::box3d,27582)));
astext
-----
 POLYGON((746387 2189036,746387 2213846,773809 2213846,773809 2189036,746387 2189036))
(1 ligne)
```

I.11 PostGIS : Comment mettre à jour et passer de PostGIS 0.X. à 1.0.X pour PostgreSQL 8.0.X ?

Je détaille ici les notes pour une mise à jour d'une version de PostGIS 0.9.X vers PostGIS 1.0.X. Pour une meilleure information, consultez le chapitre 2 de la documentation de PostGIS. Je vais commencer par la partie HARD UPGRADE.

Pour la version 1.0.X, on s'assure pour l'instant qu'on a fait un make MAIS PAS ENCORE DE make install. Le premier des impératifs est de s'assurer d'avoir un DUMP de ses données. Depuis MinGW :

```
cd ~/sources/PostGIS/postgis-1.0.X
# On configure PostGIS avec les options habituelles
configure [...]
# On compile mais pas de make install pour le moment
make
# On dump la ou les bases voulues
pg_dump -Fc base_de_donnees > base_de_donnees.dump
# Maintenant on fait le make install depuis MinGW
make install
```

On va sous DOS dans le répertoire de postgis-1.0.X et on exécute le script postgis_restore.pl prévu à cet effet. ATTENTION :Pour la version 1.0.3, le script en question ne fonctionne que sous DOS.

```
cd C:\msys\1.0\home\david\sources\PostGIS\postgis-1.0.X
utils\postgis_restore.pl lwpostgis.sql nouvelle_base_de_donnees base_de_donnees.dump > ↪
restore.log
```

Pour m'assurer que les fonctionnalités éventuelles autre que celles de PostGIS que j'aurais ajouté à la base base_de_donnees ont été conserver, je fais depuis MINGW :

```
grep ^KEEPING restore.log
```

Si cette commande ne renvoit rien c'est que je n'ai pas ajouté de fonctionnalités supplémentaires à mon ancienne base.

Ceci me créer une nouvelle base de données nouvelle_base_de_donnees qui contient les mises à jours attendues. Il ne me est plus qu'à mettre à jour ma table spatial_ref_sys en faisant depuis MinGW ou DOS - puisque je suis dans la répertoire postgis 1.0.X qui contient le nouveau fichier spatial_ref_sys.sql :

```
psql -d nouvelle_base_de_donnees -c "delete from spatial_ref_sys"
psql -d nouvelle_base_de_donnees -f spatial_ref_sys.sql
```

Il ne me reste plus qu'à faire la partie de SOFT UPGRADE en faisant depuis DOS :

```
utils\postgis_proc_upgrade.pl lwpostgis.sql | psql nouvelle_base_de_donnees
```

Je peux maintenant effacer mon ancienne base :

```
dropdb base_de_donnees
```

I.12 Notes pour PostGIS 1.1.2 : Tests de régression

Avec l'option de compilation pour PostgreSQL --enable-nls (voir chapitre 2) qui permet d'avoir notamment les messages d'erreur et de notification en français, le mieux à mon sens pour réussir l'intégralité des 12 tests attendus est de compiler PostgreSQL sans cette option.

NOTE

Attention, les tests de régression que j'ai testé ne marche pas si j'use de cette option. Ceci est une remarque que j'émets uniquement sur Win32. En effet, pour GNU/Linux si PostgreSQL est compilé avec cette option cela marche. Celà vient à mon sens du fait que sous Windows on ne dispose pas tout l'outillage nécessaire pour la locale en C. Ce qui est le cas sous GNU/Linux et pour lequel les tests de régression de PostGIS fonctionnent

Si quelqu'un a une autre piste plus précise que la mienne merci de m'en faire part.

Voici la marche à suivre. Désinstallez et recompilez PostgreSQL sans l'option --enable-nls et réinstallez

```
cd ~/sources/PostgreSQL/postgresql-8.1.3
make uninstall clean distclean
configure --prefix=/c/PostgreSQL/8.1.3
make && make install
```

Allez ensuite dans le répertoire des sources de PostGIS. Créez ensuite la variable d'environnement PGUSER=postgres. Copiez le fichier lwpostgis.sql dans le répertoire regress

```
cd ~/sources/PostGIS/postgis-1.1.2
export PGUSER=postgres
cp lwpostgis.sql regress/
make test
```

Vous devriez obtenir les résultats suivants

```
david@BREMKO ~/sources/PostGIS/postgis-1.1.2
$ make test
make -C regress test
make[1]: Entering directory '/home/david/sources/PostGIS/postgis-1.1.2/regress'
make -C ..../lwgeom all-shared-lib
make[2]: Entering directory '/home/david/sources/PostGIS/postgis-1.1.2/lwgeom'
make[2]: Nothing to be done for 'all-shared-lib'.
make[2]: Leaving directory '/home/david/sources/PostGIS/postgis-1.1.2/lwgeom'
Creating spatial db postgis_reg

PostgreSQL 8.1.3 on i686-pc-mingw32, compiled by GCC gcc.exe (GCC) 3.4.2 (mingw-special)

Postgis 1.1.2
- 2006-04-22 14:08:24

GEOS: 2.2.1-CAPI-1.0.1

JTS:

PROJ: Rel. 4.4.9, 29 Oct 2004

Running tests

    regress: Ok.
    regress_index: Ok.
    lwgeom_regress: Ok.
    regress_lrs: Ok.
    removepoint: Ok.
    setpoint: Ok.
    simplify: Ok.
    snaptogrid: Ok.
    affine: Ok.
    regress_ogc: Ok.
    regress_bdpoly: Ok.
    regress_proj: Ok.

Run tests: 12
Successful: 12
Failed: 0
make[1]: Leaving directory '/home/david/sources/PostGIS/postgis-1.1.2/regress'
```

Et oh merveille, les tests de régression de PostGIS passent sans souci !

NOTE

D'autres erreurs pour cette version de PostGIS ont été signalées par Alexandra sur la mailing-list de PostGIS. Merci de consulter ce lien auquel a répondu Marc CAVE AYLAND <http://postgis.refractions.net/pipermail/postgis-users/2006-April/011884.html>

I.13 Sous GNU/Linux : depuis un terminal comment se connecter à une machine distante sans avoir à taper à chaque fois le mot de passe ?

Supposons qu'une machine distante du réseau ait l'IP 192.168.0.14, que nous soyons autorisés à nous connecter à cette machine sur une base ma_base_de_donnees, et que pour nous connecter au serveur PostgreSQL actuellement en service sur la machine, nous soyons obligés de fournir notre login postgres et notre mot de passe "minotore" depuis un shell GNU/Linux. Pour éviter de le faire, il suffit de faire

```
su postgres
```

puis

```
cd  
touch .pgpass
```

Il suffit ensuite de mettre la chaîne suivante dans le fichier

```
192.168.0.14:5432:ma_base_de_donnees:postgres:minotore
```

On restreint ensuite les droits sur ce fichier en faisant

```
chmod 0600 . pgpass
```

Ainsi la prochaine fois que nous ferons une connexion au serveur

```
psql -h 192.168.0.14 -U postgres -d ma_base_de_donnees
```

nous ne serons plus obligés de fournir le mot de passe.

NOTE

Pour de plus amples informations, merci de consulter le lien <http://docs.postgresqlfr.org/8.1/libpq-pgpass.html>

I.14 Sous GNU/Linux : comment connaître l'ensemble des connexions client-serveur actives ?

Depuis le client, il suffit de saisir

```
ps axv | grep postgres | egrep -v "(logger|writer|stats|su|grep)"
```

qui renverra par exemple

```
16788 ? ... 0.2 postgres: david ddass26 128.179.68.7(55324) idle  
16816 ? ... 0.3 postgres: jean ddass30 [local] idle
```

La première ligne indiquera par exemple qu'il y a une connexion cliente depuis la machine dont l'IP est 128.179.68.7. Sur cette machine, l'utilisateur david est connecté à la base ddass26. Quant à la seconde, l'utilisateur jean est connecté en local à la base ddass30. Le terme idle désigne une connexion en attente d'activité.

NOTE

Il est aussi possible de suivre les connexion clientes depuis pgadmin.

NOTE

Cette section est inspirée du travail de traduction de Guillaume LELARGE que vous pouvez trouver à cette adresse <http://docs.postgresqlfr.org/8.1/monitoring.html>

On peut aussi surveiller les connexions en essayant la vue pg_stat_activity suivante :

```
SELECT * FROM pg_stat_activity ;
```

AVERTISSEMENT

Pour la vue pg_stat_activity, il est nécessaire de modifier certaines valeurs de votre fichier de configuration de PostgreSQL. Merci de consulter ce lien pour de plus amples informations <http://docs.postgresqlfr.org/8.1/monitoring-stats.html>

Par exemple pour avoir de meilleures informations sur les statistiques du serveur chez moi, je fais dans mon fichier de configuration

```
#-----  
# RUNTIME STATISTICS  
#-----  
  
# - Statistics Monitoring -  
  
log_parser_stats = true  
log_planner_stats = true  
log_executor_stats = true  
log_statement_stats = off  
  
# - Query/Index Statistics Collector -  
  
stats_start_collector = true  
stats_command_string = true  
stats_block_level = true  
stats_row_level = true  
stats_reset_on_server_start = true
```

par exemple pour connaître la requête envoyé par un client sur le serveur (colonne current_query du tableau renvoyé par pg_stats_activity).

I.15 PostgreSQL : Connaître l'OID d'une table ?

On pourra par exemple essayer la requête suivante

```
SELECT c.oid  
      FROM pg_catalog.pg_class c  
            LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace  
      WHERE c.relname ~ '??????????'$'  
            AND pg_catalog.pg_table_is_visible(c.oid)  
      ORDER BY 1
```

en remplaçant ici ??????? par le nom de la table

I.16 PostgreSQL : Connaitre les champs d'un table, ainsi que leurs type ?

On utilisera par exemple

```
SELECT a.attname as champs,
       pg_catalog.format_type(a.atttypid, a.atttypmod) as type,
       (SELECT substring(pg_catalog.pg_get_expr(d.adbin, d.adrelid) for 128)
        FROM pg_catalog.pg_attrdef d
        WHERE d.adrelid = a.attrelid AND d.adnum = a.attnum AND a.attasdef),
       a.attnotnull, a.attnum
    FROM pg_catalog.pg_attribute a
   WHERE a.attrelid in (SELECT c.oid
    FROM pg_catalog.pg_class c
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
   WHERE c.relname ~ '^troncon_route$'
         AND pg_catalog.pg_table_is_visible(c.oid)
  ORDER BY 1) AND a.attnum > 0 AND NOT a.attisdropped AND a.attname<>'the_geom'
  ORDER BY a.attnum
```

en remplaçant ici ?????????? par le nom de la table

I.17 Sous GNU/Linux : Comment migrer une base encodé en LATIN9 ou un shapefile encodé en LATIN9 vers une base encodée en UTF-8 ?

Pour celà, j'utilise l'utilitaire iconv de la manière suivante

```
..... |iconv -f LATIN1 -t UTF-8 |psql -d testgis -h [hôte] -U [utilisateur]
```

ici désigne une commande du style **shp2pgsql** ou **pg_dump**

NOTE

A la place de de iconv, on peut aussi utiliser recode en faisant

```
..... |recode 19..u8 |psql -d testgis -h [hôte] -U [utilisateur]
```

I.18 Sous GNU/Linux : Comment créer un espace logique avec PostgreSQL ?

Je fournirais ici la réponse pour GNU/Linux. Commençons d'abord par un petit rappel en consultant le lien suivant <http://docs.postgresqlf/8.1/manage-ag-tablespaces.html>. Par exemple pour créer la base de données ma_base_de_donnees dans le répertoire /mnt/disk1/data, je commencerais par créer l'espace logique en tant que root

```
mkdir /mnt/disk1/data
```

Je rend ensuite postgres propriétaire de ce répertoire

```
chown -R postgres /mnt/disk1/data
```

Je me connecte ensuite au modèle de base de données template1 (par exemple)

```
su postgres
psql template1
```

Je fournis la requête SQL suivante pour créer mon espace logique data

```
CREATE TABLESPACE data LOCATION '/mnt/disk1/data';
```

que je remplis de la base de données ma_base_de_donnees :

```
CREATE DATABASE ma_base_donnees WITH TABLESPACE data;
```

Pour connaître l'ensemble des espaces logiques disponibles, on peut essayer

```
psql -d template1 -c "\db"
```

qui renverra par exemple

```
Liste des espaces logiques
Nom | Propriétaire | Emplacement
-----+-----+-----
data | postgres     | /mnt/disk1/data
pg_default | postgres     |
pg_global | postgres     |
(3 lignes)
```

On peut aussi essayer la requête SQL suivante

```
SELECT spcname AS "Nom",
       pg_catalog.pg_get_userbyid(spcowner) AS "Propriétaire",
       spclocation AS "Emplacement"
  FROM pg_catalog.pg_tablespace
 ORDER BY 1;
```

Pour finir, si l'on souhaite connaître dans quel espace logique se trouve chacune de nos bases de données, on aura recours à la requête suivante :

```
SELECT datname AS "Base de données",
       spcname AS "Espace logique"
  FROM pg_database d JOIN pg_tablespace t ON ( dattablespace = t.oid ) ORDER BY 2;
```

qui renverra par exemple

```
Base de données | Espace logique
-----+-----
bourgogne | pg_default
ddass26 | tablespace
ddass30 | tablespace
gps | tablespace
ignportal | tablespace
postgres | pg_default
template0 | pg_default
template1 | pg_default
```

I.19 PostgreSQL : Peut-on copier un schéma dans un autre schéma et renommer l'ancien schéma ?

La réponse est oui ! Pour se faire nous utiliserons par exemple l'utilitaire pg_dump de PostgreSQL. Pour des rappels de base sur les schémas avec PostgreSQL, on pourra par exemple consulter le lien suivant <http://docs.postgresqlfr.org/8.1/sql-createschema.html>. Supposons ici que nous disposons d'un schéma appelé cadastre, que nous souhaiterions copier dans un nouveau schéma identique appelé cadastre_2006. Pour se faire, commençons par faire une sauvegarde temporaire de notre schéma de notre base test dans un fichier sql (cadastre.sql)

```
pg_dump -n cadastre -vFp -f cadastre.sql test
```

A l'aide de psql, il est ensuite possible de renommer le schéma actuel en faisant

```
ALTER SCHEMA cadastre RENAME TO cadastre_2006
```

On réimportera alors la sauvegarde en faisant

```
psql -d test -f cadastre.sql
```

Et si on souhaite renommer le schéma, on pourra toujours effectuer la même requête que celle ci-dessus.

I.20 Comment ajouter le support PL/Perl pour PostgreSQL sous Windows ?

Perl sera nécessaire pour la compilation avec PostgreSQL pour avec le support PL/Perl . Il suffit d'installer au moins la version Active Perl 5.6.X ou la 5.8.X. Téléchargez le fichier

Perl <http://www.activestate.com/Products/Download/Download.plex?id=ActivePerl>

Prendre l'installateur en MSI. C'est une installation standard, indiquez comme chemin d'instalaltion **c:\Perl**.

Retournez dans votre répertoire des sources de PostgreSQL. Il faut modifier temporairement la variable PATH de votre ordinateur pour que MinGW adapte votre variable d'environnement à Perl. Celà servira surtout pour la commande configure de PostgreSQL :

```
export PATH=/c/Perl/bin:/c/Perl/lib:$PATH
```

Il faudra ensuite recompiler PostgreSQL en ajoutant l'option --with-perl et ensuite faire les commandes habituelles de compilation et d'installation make et make install

```
configure ... --with-perl
make
make install
```

Pendant la compilation des messages de notification ou d'erreur peuvent apparaître. N'y prenez pas garde !

I.21 Comment charger l'extension de PostgreSQL sous PHP ?

Nous partons ici du principe que votre version de PHP n'a pas été compilée avec PostgreSQL. (`./configure ... --with-pgsql...`). Vérifiez que dans votre répertoire des extensions de php vous ayez le fichier **php_pgsql.dll**. Pour le charger, il suffit de remplacer depuis votre fichier **php.ini** la ligne `; extension = php_pgsql.dll` par `extension = php_pgsql.dll - enlever le ; -`. Sinon si vous ne voulez pas ouvrir votre fichier et à condition bien sûr que le fichier `php_pgsql.dll` existe depuis votre script php :

```
<?php
...
if ( extension_loaded('pgsql') != 1)
{
    switch (PHP_OS)
    {
        case "WINNT": $php_suffix = ".dll";
                        break;

        default:   $php_suffix = ".so";
                        break;
    }
    dl("php_pgsql".$php_suffix);
}
...
?>
```

Annexe J

Mémento sur les commandes SQL de PostgreSQL

Ici est fournit un mémento sur les commandes SQL de PostgreSQL

Commande : ABORT

Description : abandonner la transaction en cours

Syntaxe :

ABORT [WORK | TRANSACTION]

Commande : ALTER AGGREGATE

Description : modifier la définition d'une fonction d'agrégation

Syntaxe :

ALTER AGGREGATE nom (type) RENAME TO nouveau_nom

ALTER AGGREGATE nom (type) OWNER TO nouveauPropriétaire

ALTER AGGREGATE nom (type) SET SCHEMA nouveau_schéma

Commande : ALTER CONVERSION

Description : modifier la définition d'une conversion

Syntaxe :

ALTER CONVERSION nom RENAME TO nouveau_nom

ALTER CONVERSION nom OWNER TO nouveauPropriétaire

Commande : ALTER DATABASE

Description : modifier une base de données

Syntaxe :

ALTER DATABASE nom [[WITH] option [...]]

où option fait partie de :

CONNECTION LIMIT limite_connexion

ALTER DATABASE nom SET paramètre { TO | = } { valeur | DEFAULT }

ALTER DATABASE nom RESET paramètre

ALTER DATABASE nom RENAME TO nouveau_nom

ALTER DATABASE nom OWNER TO nouveauPropriétaire

Commande : ALTER DOMAIN

Description : modifier la définition d'un domaine

Syntaxe :

ALTER DOMAIN nom

{ SET DEFAULT expression | DROP DEFAULT }

ALTER DOMAIN nom

{ SET | DROP } NOT NULL

ALTER DOMAIN nom

```
ADD contrainte_domaine
ALTER DOMAIN nom
    DROP CONSTRAINT nom_contrainte [ RESTRICT | CASCADE ]
ALTER DOMAIN nom
    OWNER TO nouveau_propriétaire
ALTER DOMAIN nom
    SET SCHEMA nouveau_schéma
```

Commande : ALTER FUNCTION

Description : modifier la définition d'une fonction

Syntaxe :

```
ALTER FUNCTION nom ( [ [ mode_arg ] [ nom_arg ] type_arg [, ...] ] )
    action [, ...] [ RESTRICT ]
ALTER FUNCTION nom ( [ [ mode_arg ] [ nom_arg ] type_arg [, ...] ] )
    RENAME TO nouveau_nom
ALTER FUNCTION nom ( [ [ mode_arg ] [ nom_arg ] type_arg [, ...] ] )
    OWNER TO nouveau_propriétaire
ALTER FUNCTION nom ( [ [ mode_arg ] [ nom_arg ] type_arg [, ...] ] )
    SET SCHEMA nouveau_schéma
```

où action fait partie de :

```
CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
IMMUTABLE | STABLE | VOLATILE
[ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
```

Commande : ALTER GROUP

Description : modifie le nom du rôle ou son appartenance

Syntaxe :

```
ALTER GROUP nom_groupe ADD USER nom_utilisateur [, ...]
ALTER GROUP nom_groupe DROP USER nom_utilisateur [, ...]
```

```
ALTER GROUP nom_groupe RENAME TO nouveau_nom
```

Commande : ALTER INDEX

Description : modifier la définition d'un index

Syntaxe :

```
ALTER INDEX nom RENAME TO nouveau_nom
ALTER INDEX nom SET TABLESPACE nom_espace_logique
```

Commande : ALTER LANGUAGE

Description : modifier la définition d'un langage procédural

Syntaxe :

```
ALTER LANGUAGE nom RENAME TO nouveau_nom
```

Commande : ALTER OPERATOR CLASS

Description : modifier la définition d'une classe d'opérateur

Syntaxe :

```
ALTER OPERATOR CLASS nom USING méthode_indexation RENAME TO nouveau_nom
ALTER OPERATOR CLASS nom USING méthode_indexation OWNER TO nouveau_propriétaire
```

Commande : ALTER OPERATOR

Description : modifier la définition d'un opérateur

Syntaxe :

```
ALTER OPERATOR nom ( { lefttype | NONE } , { righttype | NONE } ) OWNER TO ←
    nouveau_propriétaire
```

Commande : ALTER ROLE

Description : modifie un rôle de la base de données

Syntaxe :

```
ALTER ROLE nom [ [ WITH ] option [ ... ] ]
```

où option fait partie de :

```
SUPERUSER | NOSUPERUSER
| CREATEDB | NOCREATEDB
| CREATEROLE | NOCREATEROLE
| CREATEUSER | NOCREATEUSER
| INHERIT | NOINHERIT
| LOGIN | NOLOGIN
| CONNECTION LIMIT connlimit
| [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
| VALID UNTIL 'timestamp'
```

```
ALTER ROLE nom RENAME TO nouveau_nom
```

```
ALTER ROLE nom SET paramètre { TO | = } { valeur | DEFAULT }
ALTER ROLE nom RESET paramètre
```

Commande : ALTER SCHEMA

Description : modifier la définition d'un schéma

Syntaxe :

```
ALTER SCHEMA nom RENAME TO nouveau_nom
ALTER SCHEMA nom OWNER TO nouveauPropriétaire
```

Commande : ALTER SEQUENCE

Description : modifier la définition d'un générateur de séquence

Syntaxe :

```
ALTER SEQUENCE nom [ INCREMENT [ BY ] incrément ]
[ MINVALUE valeur_min | NO MINVALUE ] [ MAXVALUE valeur_max | NO MAXVALUE ]
[ RESTART [ WITH ] début ] [ CACHE cache ] [ [ NO ] CYCLE ]
ALTER SEQUENCE nom SET SCHEMA nouveau_schéma
```

Commande : ALTER TABLE

Description : modifier la définition d'une table

Syntaxe :

```
ALTER TABLE [ ONLY ] nom [ * ]
action [, ... ]
ALTER TABLE [ ONLY ] nom [ * ]
    RENAME [ COLUMN ] colonne TO nouvelle_colonne
ALTER TABLE nom
    RENAME TO nouveau_nom
ALTER TABLE nom
    SET SCHEMA nouveau_schéma
```

où action fait partie de :

```
ADD [ COLUMN ] colonne type [ contrainte_colonne [ ... ] ]
DROP [ COLUMN ] colonne [ RESTRICT | CASCADE ]
ALTER [ COLUMN ] colonne TYPE type [ USING expression ]
ALTER [ COLUMN ] colonne SET DEFAULT expression
ALTER [ COLUMN ] colonne DROP DEFAULT
ALTER [ COLUMN ] colonne { SET | DROP } NOT NULL
ALTER [ COLUMN ] colonne SET STATISTICS integer
ALTER [ COLUMN ] colonne SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
ADD contrainte_table
DROP CONSTRAINT nom_contrainte [ RESTRICT | CASCADE ]
DISABLE TRIGGER [ nom_déclencheur | ALL | USER ]
ENABLE TRIGGER [ nom_déclencheur | ALL | USER ]
CLUSTER ON nom_index
SET WITHOUT CLUSTER
SET WITHOUT OIDS
OWNER TO nouveauPropriétaire
SET TABLESPACE nouvel_espacelogique
```

Commande : ALTER TABLESPACE
Description : modifie la définition d'un espace logique
Syntaxe :
ALTER TABLESPACE nom RENAME TO nouveau_nom
ALTER TABLESPACE nom OWNER TO nouveauPropriétaire

Commande : ALTER TRIGGER
Description : modifier la définition d'un déclencheur
Syntaxe :
ALTER TRIGGER nom ON table RENAME TO nouveau_nom

Commande : ALTER TYPE
Description : modifie la définition d'un type
Syntaxe :
ALTER TYPE nom OWNER TO nouveauPropriétaire
ALTER TYPE nom SET SCHEMA nouveauSchéma

Commande : ALTER USER
Description : modifie un rôle de la base de données
Syntaxe :
ALTER USER nom [[WITH] option [...]]

où option fait partie de :

SUPERUSER | NOSUPERUSER
| CREATEDB | NOCREATEDB
| CREATEROLE | NOCREATEROLE
| CREATEUSER | NOCREATEUSER
| INHERIT | NOINHERIT
| LOGIN | NOLOGIN
| CONNECTION LIMIT limite_connexion
| [ENCRYPTED | UNENCRYPTED] PASSWORD 'motdepasse'
| VALID UNTIL 'timestamp'

ALTER USER nom RENAME TO nouveau_nom

ALTER USER nom SET paramètre { TO | = } { valeur | DEFAULT }
ALTER USER nom RESET paramètre

Commande : ANALYZE
Description : acquérir des statistiques concernant la base de données
Syntaxe :
ANALYZE [VERBOSE] [table [(colonne [, ...])]]

Commande : BEGIN
Description : démarrer un bloc de transaction
Syntaxe :
BEGIN [WORK | TRANSACTION] [transaction_mode [, ...]]

où transaction_mode fait partie de :

ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }
READ WRITE | READ ONLY

Commande : CHECKPOINT
Description : Forcer l'enregistrement immédiat des journaux de transaction
Syntaxe :
CHECKPOINT

Commande : CLOSE
Description : fermer un curseur

```
Syntaxe :
CLOSE nom

Commande :      CLUSTER
Description : clusteriser une table à partir d'un index
Syntaxe :
CLUSTER nom_index ON nom_table
CLUSTER nom_table
CLUSTER

Commande :      COMMENT
Description : définir ou modifier les commentaires d'un objet
Syntaxe :
COMMENT ON
{
    TABLE nom_objet |
    COLUMN nom_table.nmo_colonne |
    AGGREGATE nom_agg (type_agg) |
    CAST (type_source AS type_cible) |
    CONSTRAINT nom_contrainte ON nom_table |
    CONVERSION nom_objet |
    DATABASE nom_objet |
    DOMAIN nom_objet |
    FUNCTION nom_fonc ( [ [ mode_arg ] [ nom_arg ] type_arg [, ...] ] ) |
    INDEX nom_objet |
    LARGE OBJECT oid_large_objet |
    OPERATOR op (type_opérande_gauche, type_opérande_droit) |
    OPERATOR CLASS nom_objet USING méthode_indexation |
    [ PROCEDURAL ] LANGUAGE nom_objet |
    RULE nom_règle ON nom_table |
    SCHEMA nom_objet |
    SEQUENCE nom_objet |
    TRIGGER nom_déclencheur ON nom_table |
    TYPE nom_objet |
    VIEW nom_objet
} IS 'texte'

Commande :      COMMIT
Description : valider la transaction en cours
Syntaxe :
COMMIT [ WORK | TRANSACTION ]

Commande :      COMMIT PREPARED
Description : valide une transaction qui a été précédemment préparée pour une validation en ↔
              deux phases
Syntaxe :
COMMIT PREPARED id_transaction

Commande :      COPY
Description : copier des données entre un fichier et une table
Syntaxe :
COPY nom_table [ ( colonne [, ...] ) ]
  FROM { 'nom_fichier' | STDIN }
  [ [ WITH ]
    [ BINARY ]
    [ OIDS ]
    [ DELIMITER [ AS ] 'délimiteur' ]
    [ NULL [ AS ] 'chaîne null' ]
    [ CSV [ HEADER ]
      [ QUOTE [ AS ] 'guillemet' ]
      [ ESCAPE [ AS ] 'échappement' ]
      [ FORCE NOT NULL colonne [, ...] ]
```

```
COPY tablename [ ( colonne [, ...] ) ]
TO { 'nom_fichier' | STDOUT }
[ [ WITH ]
  [ BINARY ]
  [ HEADER ]
  [ OIDS ]
  [ DELIMITER [ AS ] 'délimiteur' ]
  [ NULL [ AS ] 'chaîne null' ]
  [ CSV [ HEADER ]
    [ QUOTE [ AS ] 'guillemet' ]
    [ ESCAPE [ AS ] 'échappement' ]
    [ FORCE QUOTE colonne [, ...] ]
```

Commande : CREATE AGGREGATE

Description : définir une nouvelle fonction d'agrégation

Syntaxe :

```
CREATE AGGREGATE nom (
  BASETYPE = type_données_en_entrée,
  SFUNC = fonction_s,
  STYPE = type_données_état
  [ , FINALFUNC = fonction_f ]
  [ , INITCOND = condition_initiale ]
  [ , SORTOP = opérateur_tri ]
)
```

Commande : CREATE CAST

Description : définir une nouvelle conversion explicite

Syntaxe :

```
CREATE CAST (type_source AS type_cible)
  WITH FUNCTION nom_fonction (type_argument)
  [ AS ASSIGNMENT | AS IMPLICIT ]
```

CREATE CAST (type_source AS type_cible)

WITHOUT FUNCTION

[AS ASSIGNMENT | AS IMPLICIT]

Commande : CREATE CONSTRAINT TRIGGER

Description : définir une nouvelle contrainte de déclenchement

Syntaxe :

```
CREATE CONSTRAINT TRIGGER nom
  AFTER événements ON
  nom_table attributs_contrainte
  FOR EACH ROW EXECUTE PROCEDURE nom_fonction ( args )
```

Commande : CREATE CONVERSION

Description : définir une nouvelle conversion de codage

Syntaxe :

```
CREATE [DEFAULT] CONVERSION nom
  FOR encodage_source TO encodage_cible FROM nom_fonction
```

Commande : CREATE DATABASE

Description : créer une nouvelle base de données

Syntaxe :

```
CREATE DATABASE nom
  [ [ WITH ] [ OWNER [=] nomPropriétaire ]
    [ TEMPLATE [=] modèle ]
    [ ENCODING [=] codage ]
    [ TABLESPACE [=] espaceLogique ]
    [ CONNECTION LIMIT [=] limite_connexion ] ]
```

Commande : CREATE DOMAIN

Description : définir un nouveau domaine

Syntaxe :

```
CREATE DOMAIN nom [AS] type_données
    [ DEFAULT expression ]
    [ contrainte [ ... ] ]
```

avec comme contrainte :

```
[ CONSTRAINT nom_contrainte ]
{ NOT NULL | NULL | CHECK (expression) }
```

Commande : CREATE FUNCTION

Description : définir une nouvelle fonction

Syntaxe :

```
CREATE [ OR REPLACE ] FUNCTION
    nom ( [ [ mode_arg ] [ nom_arg ] type_arg [, ...] ] )
    [ RETURNS type_retour ]
{ LANGUAGE nom_langage
| IMMUTABLE | STABLE | VOLATILE
| CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
| [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
| AS 'définition'
| AS 'fichier_objet', 'symbole_lien'
} ...
[ WITH ( attribut [, ...] ) ]
```

Commande : CREATE GROUP

Description : définir un nouveau rôle dans la base de données

Syntaxe :

```
CREATE GROUP nom [ [ WITH ] option [ ... ] ]
```

où option fait partie de :

```
SUPERUSER | NOSUPERUSER
| CREATEDB | NOCREATEDB
| CREATEROLE | NOCREATEROLE
| CREATEUSER | NOCREATEUSER
| INHERIT | NOINHERIT
| LOGIN | NOLOGIN
| [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'motdepasse'
| VALID UNTIL 'timestamp'
| IN ROLE nom_rôle [, ...]
| IN GROUP nom_rôle [, ...]
| ROLE nom_rôle [, ...]
| ADMIN nom_rôle [, ...]
| USER nom_rôle [, ...]
| SYSID uid
```

Commande : CREATE INDEX

Description : définir un nouvel index

Syntaxe :

```
CREATE [ UNIQUE ] INDEX nom ON table [ USING méthode ]
( { colonne | ( expression ) } [ classe_opérateur ] [, ...] )
[ TABLESPACE espace_logique ]
[ WHERE prédicat ]
```

Commande : CREATE LANGUAGE

Description : définir un nouveau langage de programmation de procédures stockées

Syntaxe :

```
CREATE [ PROCEDURAL ] LANGUAGE nom
CREATE [ TRUSTED ] [ PROCEDURAL ] LANGUAGE nom
    HANDLER gestionnaire_appels [ VALIDATOR fonction_val ]
```

```
Commande : CREATE OPERATOR CLASS
Description : définir une nouvelle classe opérateur
Syntaxe :
CREATE OPERATOR CLASS nom [ DEFAULT ] FOR TYPE type_données USING méthode_indexation AS
{ OPERATOR numéro_stratégie nom_opérateur [ ( type_op, type_op ) ] [ RECHECK ]
| FUNCTION numéro_support nom_fonction ( type_argument [, ...] )
| STORAGE type_stockage
} [, ... ]

Commande : CREATE OPERATOR
Description : définir un nouvel opérateur
Syntaxe :
CREATE OPERATOR nom (
    PROCEDURE = nom_fonction
    [, LEFTARG = type_gauche ] [, RIGHTARG = type_droit ]
    [, COMMUTATOR = op_commutation ] [, NEGATOR = op_négation ]
    [, RESTRICT = proc_restriction ] [, JOIN = procJointure ]
    [, HASHES ] [, MERGES ]
    [, SORT1 = op_tri_droit ] [, SORT2 = op_tri_gauche ]
    [, LTCMP = op_plus_petit_que ] [, GTCMP = op_plus_grand_que ]
)

Commande : CREATE ROLE
Description : définir un nouveau rôle dans la base de données
Syntaxe :
CREATE ROLE nom [ [ WITH ] option [ ... ] ]

où option fait partie de :

    SUPERUSER | NOSUPERUSER
    | CREATEDB | NOCREATEDB
    | CREATEROLE | NOCREATEROLE
    | CREATEUSER | NOCREATEUSER
    | INHERIT | NOINHERIT
    | LOGIN | NOLOGIN
    | CONNECTION LIMIT limite_connexion
    | [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'motdepasse'
    | VALID UNTIL 'timestamp'
    | IN ROLE nom_rôle [, ...]
    | IN GROUP nom_rôle [, ...]
    | ROLE nom_rôle [, ...]
    | ADMIN nom_rôle [, ...]
    | USER nom_rôle [, ...]
    | SYSID uid

Commande : CREATE RULE
Description : définir une nouvelle règle de réécriture
Syntaxe :
CREATE [ OR REPLACE ] RULE nom AS ON événement
    TO table [ WHERE condition ]
    DO [ ALSO | INSTEAD ] { NOTHING | commande | ( commande ; commande ... ) }

Commande : CREATE SCHEMA
Description : définir un nouveau schéma
Syntaxe :
CREATE SCHEMA nom_schema [ AUTHORIZATION nom_utilisateur ] [ element_schema [ ... ] ]
CREATE SCHEMA AUTHORIZATION nom_utilisateur [ element_schema [ ... ] ]

Commande : CREATE SEQUENCE
Description : définir un nouveau générateur de séquence
Syntaxe :
```

```
CREATE [ TEMPORARY | TEMP ] SEQUENCE nom [ INCREMENT [ BY ] incrémentation ]
[ MINVALUE valeur_mini | NO MINVALUE ] [ MAXVALUE valeur_maxi | NO MAXVALUE ]
[ START [ WITH ] valeur_départ ] [ CACHE en_cache ] [ [ NO ] CYCLE ]
```

Commande : CREATE TABLE

Description : définir une nouvelle table

Syntaxe :

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE nom_table ( [
{ nom_colonne type_données [ DEFAULT expr_par_défaut ] [ contrainte_colonne [ ... ] ]
| contrainte_table
| LIKE table_parent [ { INCLUDING | EXCLUDING } DEFAULTS ] }
[, ... ]
] )
[ INHERITS ( table_parent [, ... ] ) ]
[ WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE espace_logique ]
```

où colonne_contrainte fait partie de :

```
[ CONSTRAINT nom_contrainte ]
{ NOT NULL |
NULL |
UNIQUE [ USING INDEX TABLESPACE espace_logique ] |
PRIMARY KEY [ USING INDEX TABLESPACE espace_logique ] |
CHECK (expression) |
REFERENCES table_référente [ ( colonne_referrante ) ] [ MATCH FULL | MATCH PARTIAL | ←
MATCH SIMPLE ]
[ ON DELETE action ] [ ON UPDATE action ] }
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

et contrainte_table fait partie de :

```
[ CONSTRAINT nmo_contrainte ]
{ UNIQUE ( nom_colonne [, ... ] ) [ USING INDEX TABLESPACE espace_logique ] |
PRIMARY KEY ( nom_colonne [, ... ] ) [ USING INDEX TABLESPACE espace_logique ] |
CHECK ( expression ) |
FOREIGN KEY ( nom_colonne [, ... ] ) REFERENCES table_référente [ ( colonne_référente [, ←
... ] ) ]
[ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE action ] ←
}
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

Commande : CREATE TABLE AS

Description : définir une nouvelle table à partir des résultats d'une requête

Syntaxe :

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE nom_table
[ (nom_colonne [, ...] ) ] [ [ WITH | WITHOUT ] OIDS ]
AS requête
```

Commande : CREATE TABLESPACE

Description : définir un nouvel espace logique

Syntaxe :

```
CREATE TABLESPACE nom_espace_logique [ OWNER nom_utilisateur ] LOCATION 'répertoire'
```

Commande : CREATE TRIGGER

Description : définir un nouveau déclencheur

Syntaxe :

```
CREATE TRIGGER nom { BEFORE | AFTER } { événement [ OR ... ] }
ON table [ FOR [ EACH ] { ROW | STATEMENT } ]
EXECUTE PROCEDURE nom_fonction ( arguments )
```

Commande : CREATE TYPE
Description : définir un nouveau type de données
Syntaxe :
CREATE TYPE nom AS
(nom_attribut type_données [, ...])

CREATE TYPE nom (
 INPUT = fonction_en_entrée,
 OUTPUT = fonction_en_sortie
 [, RECEIVE = fonction_recevant_type]
 [, SEND = fonction_renvoiant_type]
 [, ANALYZE = fonction_analyze]
 [, INTERNALLENGTH = { longueur_intervalle | VARIABLE }]
 [, PASSEDBYVALUE]
 [, ALIGNMENT = alignement]
 [, STORAGE = stockage]
 [, DEFAULT = valeur_par_défaut]
 [, ELEMENT = élément]
 [, DELIMITER = séparateur]
)

Commande : CREATE USER
Description : définir un nouveau rôle dans la base de données
Syntaxe :
CREATE USER nom [[WITH] option [...]]

où option fait partie de :

SUPERUSER | NOSUPERUSER
| CREATEDB | NOCREATEDB
| CREATEROLE | NOCREATEROLE
| CREATEUSER | NOCREATEUSER
| INHERIT | NOINHERIT
| LOGIN | NOLOGIN
| CONNECTION LIMIT limite_connexion
| [ENCRYPTED | UNENCRYPTED] PASSWORD 'motdepasse'
| VALID UNTIL 'timestamp'
| IN ROLE nom_rôle [, ...]
| IN GROUP nom_rôle [, ...]
| ROLE nom_rôle [, ...]
| ADMIN nom_rôle [, ...]
| USER nom_rôle [, ...]
| SYSID uid

Commande : CREATE VIEW
Description : définir une nouvelle vue
Syntaxe :
CREATE [OR REPLACE] [TEMP | TEMPORARY] VIEW nom [(nom_colonne [, ...])]
AS requête

Commande : DEALLOCATE
Description : désactiver une expression préparée à l'avance
Syntaxe :
DEALLOCATE [PREPARE] nom_plan

Commande : DECLARE
Description : définir un curseur
Syntaxe :
DECLARE nom [BINARY] [INSENSITIVE] [[NO] SCROLL]
CURSOR [{ WITH | WITHOUT } HOLD] FOR requête
[FOR { READ ONLY | UPDATE [OF colonne [, ...]] }]

```
Commande :      DELETE
Description : supprimer des colonnes dans une table
Syntaxe :
DELETE FROM [ ONLY ] table
[ USING liste_using ]
[ WHERE condition ]

Commande :      DROP AGGREGATE
Description : supprimer une fonction d'agrégation
Syntaxe :
DROP AGGREGATE nom ( type ) [ CASCADE | RESTRICT ]

Commande :      DROP CAST
Description : supprimer une conversion explicite
Syntaxe :
DROP CAST (type_source AS type_cible) [ CASCADE | RESTRICT ]

Commande :      DROP CONVERSION
Description : supprimer une conversion
Syntaxe :
DROP CONVERSION name [ CASCADE | RESTRICT ]

Commande :      DROP DATABASE
Description : supprimer une base de données
Syntaxe :
DROP DATABASE name

Commande :      DROP DOMAIN
Description : supprimer un domaine
Syntaxe :
DROP DOMAIN name [, ...] [ CASCADE | RESTRICT ]

Commande :      DROP FUNCTION
Description : supprimer une fonction
Syntaxe :
DROP FUNCTION nom ( [ [ mode_arg ] [ nom_arg ] type_arg [, ...] ] )
[ CASCADE | RESTRICT ]

Commande :      DROP GROUP
Description : supprimer un rôle de la base de données
Syntaxe :
DROP GROUP nom [, ...]

Commande :      DROP INDEX
Description : supprimer un index
Syntaxe :
DROP INDEX name [, ...] [ CASCADE | RESTRICT ]

Commande :      DROP LANGUAGE
Description : supprimer un langage procédural
Syntaxe :
DROP [ PROCEDURAL ] LANGUAGE nom [ CASCADE | RESTRICT ]

Commande :      DROP OPERATOR CLASS
Description : supprimer une classe d'opérateur
Syntaxe :
DROP OPERATOR CLASS nom USING méthode_indexation [ CASCADE | RESTRICT ]

Commande :      DROP OPERATOR
Description : supprimer un opérateur
Syntaxe :
DROP OPERATOR nom ( { type_gauche | NONE } , { type_droit | NONE } ) [ CASCADE | RESTRICT ]
```

```
Commande :      DROP ROLE
Description : supprimer un rôle de la base de données
Syntaxe :
DROP ROLE nom [, ...]

Commande :      DROP RULE
Description : supprimer une règle de réécriture
Syntaxe :
DROP RULE nom ON relation [ CASCADE | RESTRICT ]

Commande :      DROP SCHEMA
Description : supprimer un schéma
Syntaxe :
DROP SCHEMA name [, ...] [ CASCADE | RESTRICT ]

Commande :      DROP SEQUENCE
Description : supprimer une séquence
Syntaxe :
DROP SEQUENCE name [, ...] [ CASCADE | RESTRICT ]

Commande :      DROP TABLE
Description : supprimer une table
Syntaxe :
DROP TABLE name [, ...] [ CASCADE | RESTRICT ]

Commande :      DROP TABLESPACE
Description : supprimer un espace logique
Syntaxe :
DROP TABLESPACE espace logique

Commande :      DROP TRIGGER
Description : supprimer un déclencheur
Syntaxe :
DROP TRIGGER nom ON table [ CASCADE | RESTRICT ]

Commande :      DROP TYPE
Description : supprimer un type de données
Syntaxe :
DROP TYPE name [, ...] [ CASCADE | RESTRICT ]

Commande :      DROP USER
Description : supprimer un rôle de la base de données
Syntaxe :
DROP USER nom [, ...]

Commande :      DROP VIEW
Description : supprimer une vue
Syntaxe :
DROP VIEW name [, ...] [ CASCADE | RESTRICT ]

Commande :      END
Description : valider la transaction en cours
Syntaxe :
END [ WORK | TRANSACTION ]

Commande :      EXECUTE
Description : exécuter une expression préparée à l'avance
Syntaxe :
EXECUTE nom_plan [ (paramètre [, ...] ) ]
```

Commande : EXPLAIN

Description : afficher le plan d'exécution d'une expression

Syntaxe :

```
EXPLAIN [ ANALYZE ] [ VERBOSE ] expression
```

Commande : FETCH

Description : extraire des lignes d'une requête en utilisant un curseur

Syntaxe :

```
FETCH [ direction { FROM | IN } ] nom curseur
```

sans préciser de direction ou en choisissant une des directions suivantes:

```
NEXT  
PRIOR  
FIRST  
LAST  
ABSOLUTE nombre  
RELATIVE nombre  
count  
ALL  
FORWARD  
FORWARD nombre  
FORWARD ALL  
BACKWARD  
BACKWARD nombre  
BACKWARD ALL
```

Commande : GRANT

Description : définir des priviléges d'accès

Syntaxe :

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER }  
[,...] | ALL [ PRIVILEGES ] }  
ON [ TABLE ] nom_table [, ...]  
TO { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { { CREATE | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }  
ON DATABASE nom_base [, ...]  
TO { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { EXECUTE | ALL [ PRIVILEGES ] }  
ON FUNCTION nom_fonction ( [ [ mode_arg ] [ nom_arg ] type_arg [, ...] ] ) [, ...]  
TO { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { USAGE | ALL [ PRIVILEGES ] }  
ON LANGUAGE nom_langage [, ...]  
TO { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { { CREATE | USAGE } [,...] | ALL [ PRIVILEGES ] }  
ON SCHEMA nom_schéma [, ...]  
TO { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { CREATE | ALL [ PRIVILEGES ] }  
ON TABLESPACE nom_espacelogique [, ...]  
TO { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT role [, ...]  
TO { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...] [ WITH ADMIN OPTION ]
```

Commande : INSERT

Description : créer de nouvelles lignes dans une table

Syntaxe :

```
INSERT INTO table [ ( colonne [, ...] ) ]  
{ DEFAULT VALUES | VALUES ( { expression | DEFAULT } [, ...] ) | requête }
```

```
Commande : LISTEN
Description : être à l'écoute d'une notification
Syntaxe :
LISTEN nom

Commande : LOAD
Description : charger ou recharger un fichier de librairie partagée
Syntaxe :
LOAD 'nom_de_fichier'

Commande : LOCK
Description : verrouiller une table
Syntaxe :
LOCK [ TABLE ] nom [, ...] [ IN mode_verrouillage MODE ] [ NOWAIT ]
avec un mode_verrouillage parmi les valeurs suivantes :

ACCESS SHARE | ROW SHARE | ROW EXCLUSIVE | SHARE UPDATE EXCLUSIVE
| SHARE | SHARE ROW EXCLUSIVE | EXCLUSIVE | ACCESS EXCLUSIVE

Commande : MOVE
Description : positionner un curseur
Syntaxe :
MOVE [ direction { FROM | IN } ] nom_de curseur

Commande : NOTIFY
Description : générer une notification
Syntaxe :
NOTIFY nom

Commande : PREPARE
Description : préparer à l'avance l'exécution d'une expression
Syntaxe :
PREPARE nom_plan [ (type_données [, ...] ) ] AS expression

Commande : PREPARE TRANSACTION
Description : prépare la transaction en cours pour une validation en deux phases
Syntaxe :
PREPARE TRANSACTION id_transaction

Commande : REINDEX
Description : reconstruire des indexes
Syntaxe :
REINDEX { INDEX | TABLE | DATABASE | SYSTEM } nom [ FORCE ]

Commande : RELEASE SAVEPOINT
Description : détruire un point de sauvegarde précédemment défini
Syntaxe :
RELEASE [ SAVEPOINT ] nom_point_de_sauvegarde

Commande : RESET
Description : réinitialiser un paramètre run-time à sa valeur par défaut
Syntaxe :
RESET nom
RESET ALL

Commande : REVOKE
Description : supprimer des priviléges d'accès
Syntaxe :
REVOKE [ GRANT OPTION FOR ]
{ { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER }
```

```
[,...] | ALL [ PRIVILEGES ] }
ON [ TABLE ] nom_table [, ...]
FROM { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ { CREATE | TEMPORARY | TEMP } [,,...] | ALL [ PRIVILEGES ] }
ON DATABASE nom_base [, ...]
FROM { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ EXECUTE | ALL [ PRIVILEGES ] }
ON FUNCTION nom_fonction ( [ [ mode_arg ] [ nom_arg ] type_arg [, ...] ] ) [, ...]
FROM { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ USAGE | ALL [ PRIVILEGES ] }
ON LANGUAGE nom_langage [, ...]
FROM { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ { CREATE | USAGE } [,,...] | ALL [ PRIVILEGES ] }
ON SCHEMA nom_schéma [, ...]
FROM { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
{ CREATE | ALL [ PRIVILEGES ] }
ON TABLESPACE nom_espacelogique [, ...]
FROM { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

REVOKE [ ADMIN OPTION FOR ]
rôle [, ...]
FROM { nom_utilisateur | GROUP nom_groupe | PUBLIC } [, ...]
[ CASCADE | RESTRICT ]

Commande :      ROLLBACK
Description : abandonner la transaction en cours
Syntaxe :
ROLLBACK [ WORK | TRANSACTION ]

Commande :      ROLLBACK PREPARED
Description : annule une transaction qui a été précédemment préparée pour une validation en ↵
              deux phases
Syntaxe :
ROLLBACK PREPARED id_transaction

Commande :      ROLLBACK TO SAVEPOINT
Description : retourner à un point de sauvegarde
Syntaxe :
ROLLBACK [ WORK | TRANSACTION ] TO [ SAVEPOINT ] nom_point_de_sauvegarde

Commande :      SAVEPOINT
Description : définir un nouveau point de sauvegarde pour la transaction en cours
Syntaxe :
SAVEPOINT nom_point_de_sauvegarde

Commande :      SELECT
```

Description : extraire des lignes d'une table ou d'une vue
Syntaxe :

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
       * | expression [ AS nom_affichage ] [, ...]
       [ FROM élément_from [, ...] ]
       [ WHERE condition ]
       [ GROUP BY expression [, ...] ]
       [ HAVING condition [, ...] ]
       [ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]
       [ ORDER BY expression [ ASC | DESC | USING opérateur ] [, ...] ]
       [ LIMIT { nombre | ALL } ]
       [ OFFSET début ]
       [ FOR { UPDATE | SHARE } [ OF nom_table [, ...] ] [ NOWAIT ] ]
```

où élément_from fait partie de :

```
[ ONLY ] nom_table [ * ] [ [ AS ] alias [ ( alias_colonne [, ...] ) ] ]
( select ) [ AS ] alias [ ( alias_colonne [, ...] ) ]
nom_fonction ( [ argument [, ...] ] ) [ AS ] alias [ ( alias_colonne [, ...] ) | ←
définition_colonne [, ...] ]
nom_fonction ( [ argument [, ...] ] ) AS ( définition_colonne [, ...] )
élément_from [ NATURAL ] type_jointure élément_from [ ON condition_jointure | USING ( ←
colonne_jointure [, ...] ) ]
```

Commande : SELECT INTO

Description : définir une nouvelle table à partir des résultats d'une requête

Syntaxe :

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
       * | expression [ AS nom_affichage ] [, ...]
       INTO [ TEMPORARY | TEMP ] [ TABLE ] nouvelle_table
       [ FROM élément_from [, ...] ]
       [ WHERE condition ]
       [ GROUP BY expression [, ...] ]
       [ HAVING condition [, ...] ]
       [ { UNION | INTERSECT | EXCEPT } [ ALL ] sélection ]
       [ ORDER BY expression [ ASC | DESC | USING opérateur ] [, ...] ]
       [ LIMIT { nombre | ALL } ]
       [ OFFSET début ]
       [ FOR { UPDATE | SHARE } [ OF nom_table [, ...] ] [ NOWAIT ] ]
```

Commande : SET

Description : modifier un paramètre run-time

Syntaxe :

```
SET [ SESSION | LOCAL ] nom { TO | = } { value | 'valeur' | DEFAULT }
SET [ SESSION | LOCAL ] TIME ZONE { zone_horaire | LOCAL | DEFAULT }
```

Commande : SET CONSTRAINTS

Description : définir les modes de vérification de contrainte pour la transaction en cours

Syntaxe :

```
SET CONSTRAINTS { ALL | nom [, ...] } { DEFERRED | IMMEDIATE }
```

Commande : SET ROLE

Description : définir l'identifiant de l'utilisateur actuel de la session en cours

Syntaxe :

```
SET [ SESSION | LOCAL ] ROLE nom_rôle
SET [ SESSION | LOCAL ] ROLE NONE
RESET ROLE
```

Commande : SET SESSION AUTHORIZATION

Description : définir l'identifiant de l'utilisateur de la session et l'identifiant de l'←
utilisateur actuel de la session en cours

Syntaxe :

```
SET [ SESSION | LOCAL ] SESSION AUTHORIZATION nom_utilisateur
SET [ SESSION | LOCAL ] SESSION AUTHORIZATION DEFAULT
RESET SESSION AUTHORIZATION
```

Commande : SET TRANSACTION

Description : définir les caractéristiques de la transaction en cours

Syntaxe :

```
SET TRANSACTION mode_transaction [, ...]
```

```
SET SESSION CHARACTERISTICS AS TRANSACTION mode_transaction [, ...]
```

où mode_transaction fait partie de :

```
ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }
READ WRITE | READ ONLY
```

Commande : SHOW

Description : afficher la valeur d'un paramètres run-time

Syntaxe :

```
SHOW nom
```

```
SHOW ALL
```

Commande : START TRANSACTION

Description : démarrer un bloc de transaction

Syntaxe :

```
START TRANSACTION [ mode_transaction [, ...] ]
```

où mode_transaction fait partie de :

```
ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }
READ WRITE | READ ONLY
```

Commande : TRUNCATE

Description : vider une table ou un ensemble de tables

Syntaxe :

```
TRUNCATE [ TABLE ] nom [, ...]
```

Commande : UNLISTEN

Description : ne plus être à l'écoute des notifications

Syntaxe :

```
UNLISTEN { nom | * }
```

Commande : UPDATE

Description : mettre à jour les lignes d'une table

Syntaxe :

```
UPDATE [ ONLY ] table SET colonne = { expression | DEFAULT } [, ...]
[ FROM depuis_liste ]
[ WHERE condition ]
```

Commande : VACUUM

Description : collecte des fragments avec en option la possibilité d'analyser une base de données

Syntaxe :

```
VACUUM [ FULL | FREEZE ] [ VERBOSE ] [ table ]
```

```
VACUUM [ FULL | FREEZE ] [ VERBOSE ] ANALYZE [ table [ (colonne [, ...] ) ] ]
```

Annexe K

Mémento sur les commandes internes de psql

Le mémento se présente sous cette forme

```
Général
\c[onnect] [NOM_BASE| - [NOM_UTILISATEUR]]
            se connecter à une autre base de données (actuellement «postgres»)
\cd [REPERTOIRE] changer de répertoire courant
\copyright    afficher les conditions d'utilisation et de distribution de PostgreSQL
\encoding [ENCODAGE]
            afficher ou initialiser l'encodage du client
\h [NOM]      aide-mémoire pour les commandes SQL, * pour toutes les commandes
\q            quitter psql
\set [NOM [VALEUR]]
            initialiser la variable interne ou les afficher toutes s'il n'y a aucun ↵
            paramètre
\timing       basculer l'activation du chronométrage des commandes
            (actuellement désactivé)
\unset NOM   désinitialiser (supprimer) la variable interne
\! [COMMANDÉ] exécuter la commande dans un shell ou lance un shell interactif

Tampon de requête
\e FICHIER    éditer le tampon de requête ou le fichier avec un éditeur externe
\g [FICHIER]   envoyer le tampon de requêtes au serveur (et les résultats au fichier
            ou |tube)
\p            afficher le contenu du tampon de requête
\q            effacer le tampon de requêtes
\q [FICHIER]   afficher l'historique ou le sauvegarder dans un fichier
\w [FICHIER]   écrire le contenu du tampon de requêtes dans un fichier

Entrée/Sortie
\echo [TEXTE]  écrire un texte sur la sortie standard
\i FICHIER    exécuter les commandes du fichier
\o [FICHIER]   envoyer les résultats de la requête vers un fichier ou un |tube
\qecho [TEXTE] écrire un texte sur la sortie pour les résultats des requêtes
            (voir \o)

Information
\dt [NOM]      afficher la description de la table, l'index, la séquence ou la vue
\dt[+][i|s|v|S] [MODELE] (ajoutez "+" pour plus de détails)
            liste les tables/index/séquences/vues/tables système
\da [MODELE]   afficher la liste des fonctions d'agrégation
\dn [MODELE]   affiche la liste des espaces logiques (ajoutez «+» pour plus de détails)
\dc [MODELE]   afficher la liste des conversions
\dc            afficher la liste des conversions explicites
\dd [MODELE]   afficher les commentaires pour un objet
```

```
\dD [MODELE]      afficher la liste des domaines
\df [MODELE]      afficher la liste des fonctions (ajoutez «+» pour plus de détails)
\dg [MODELE]      afficher la liste des groupes
\dn [MODELE]      afficher la liste des schémas (ajoutez «+» pour plus de détails)
\do [MODELE]      afficher la liste des opérateurs
\dl              afficher la liste des objets larges, identique à \lo_list
\dp [MODÈLE]      affiche la liste des privilèges d'accès aux tables, vues, séquences
\dT [MODELE]      afficher la liste des types de données (ajoutez «+»
                  pour plus de détails)
\du [MODELE]      afficher la liste des utilisateurs
\l              afficher toutes les bases de données (ajoutez «+» pour plus de détails)
\z [MODÈLE]       afficher la liste des privilèges d'accès aux tables, vues et
                  séquences (identique à \dp)
```

Formatage

```
\a              basculer entre les modes de sortie aligné et non aligné
\c [CHAINE]      initialiser le titre d'une table, ou initialise à rien si sans argument
\f [CHAINE]      afficher ou initialiser le séparateur de champ pour une sortie non
                  alignée des requêtes
\H              basculer le mode de sortie HTML (actuellement désactivé)
\pset NOM [VALEUR]
                  initialise l'option d'affichage de la table
                  (NOM := {format|border|expanded|fieldsep|footer|null|
                  numericlocale|recordsep|tuples_only|title|tableattr|pager})
\t              afficher seulement les lignes (actuellement désactivé)
\T [CHAINE]
                  argument
\tableattr
                  initialiser les attributs HTML de la balise <table>, ou l'annule si aucun ↔
\tableattr
\x              basculer l'affichage étendu (actuellement désactivé)
```

Copie, gros objets

```
\copy ...        exécuter SQL COPY avec le flux de données de l'hôte client
\lo_export OIDLOB FICHIER
\lo_import FICHIER [COMMENTAIRE]
\lo_list
\lo_unlink OIDLOB          opérations sur les gros objets
```

Annexe L

Suivi de PostGIS

Est présenté ici un descriptif des diverses contributions personnelles depuis notre travail sur PostGIS.

Date-Période	Travail
Décembre 2005	Tests de la C-API de Geos 2.2.1 avec Sandro Santilli avant la sortie de PostGIS 1.1.0
Aout 2005	<p>Ouverture du site PostGIS.fr avec l'accord de Paul RAMSEY. Site géré par Gérald FENOY et TECHER Jean David</p> <p>Proposition des sections téléchargement et de la documentation sous Windows. Proposition de téléchargement pour les versions actuelles Geos 2.1.3, Proj 4.4.9, PostGIS 1.0.3 et PostgreSQL 8.0.3</p> <p>E-builds pour Geos 2.1.4 sur le site de PostGIS.fr réalisé par Gerald FENOY</p>
Juin 2005 - Juillet 2005	Débogage avec Sandro Santilli pour des test de compilation de PostGIS 1.1.0 CVS sous Windows. Cette version se compilera directement depuis une distribution binaire de PostgreSQL. Ainsi, il ne sera plus nécessaire d'avoir les sources de PostgreSQL précompilées. Aide de Gerald FENOY pour l'obtention des divers patchs. Début de rédaction du document de compilation de PostGIS sous Windows pour le futur site PostGIS.fr
Janvier 2005 - Avril 2005	Réalisation de solutions avec Gerald FENOY pour solutions de serveurs cartographiques sous Gentoo et Fedora Core 3 et 4...
Septembre 2004 - Décembre 2004	<p>Demande du support pour l'incorporation de la fonction AsSVG() dans le développement de PostGIS (milite !) avec d'autres utilisateurs Andreas NEUMMAN, André WINTER et Klaus FOERSTER</p> <p>Travail avec Magnus HAGANDER - responsable du projet pginst de pgfoundry - pour le support de PostGIS 0.9.1 dans le développement de l'installateur msi 'pginstaller'</p> <p>- Collaboration avec Gilles BERETA</p> <p>Débogage de Geos avec Sandro Santilli sous Windows</p>
Mai 2004	<p>Réalisation d'une première distribution pgsql75windev sous WIndows pour PostGIS 0.8.2 et PostgreSQL 7.5 devel. Avec le support de la fonction AsSVG développée par Klaus FOERSTER</p> <p>Premières compilations de serveurs cartographiques sous Knoppix, Red Hat 9 : Proj, Geos, PostGIS, PostgreSQL, PHP, Gd, Gdal, Curl, MapServer...</p>
Mars 2004 - Avril 2004	Premiers test de compilation et d'installation de PostGIS 0.8.1 + CVS et PostgreSQL 7.5 devel sous Windows avec Marc CAVE HAYLAND et Romi HARDIYANTO
Novembrer 2003 - Mars 2004	Enrichissement du site contributif avec divers tutoriaux et projets personnels
Septembre 2003 - Octobre 2003	Ouverture du site contributif "Mes Travaux Sur PostGIS" à http://techer.pascal.free.fr/postgis/
Juin 2003-Juillet 2003	Premier test d'utilisation de PostGIS 0.7.3 et PostgreSQL 7.4.3 sous Windows avec Cygwin

Annexe M

PhpMapScript

Ici est donné à titre d'exemple les sources des fichiers qui ont permis de générer l'image servant pour les exemples dans le chapitre 4.

M.1 Table mapserver_desc

La table mapserver_desc est une table descriptive utilisée pour pouvoir gérer diverses options pour chaque table - correspondant à une layer de mapserver - comme la couleur des objets, la bordure, l'épaisseur etc... Voici son contenu

ms_gid	ms_table	ms_libelle	ms_name	ms_labelitem	ms_color	ms_outlinecolor	ms_symbol
0	small_roads	Petite routes	small_roads	data	80 80 80	80 80	↔
		80					
1	great_roads	Grandes routes	great_roads	data	125 125 125	125	↔
		125 125					
2	parcs	Parcs publiques	parcs	data	0 123 0	255	↔
		255 255					
3	rivers	Rivières	streams	data	0 12 189	0 12	↔
		189					
4	buildings	Bâtiments	buildings	data	234 156 78	234	↔
		156 78					
5	personnes	Piñons	personnes	data		255 0	↔
		0	circle				
(6 rows)							

M.2 Script PHP

Avant tout chose, nous avons besoin de partir d'une mapfile -base.map - dont le contenu est le suivant

```
MAP
EXTENT 0 0 1 1
SIZE 1 1
IMAGECOLOR 255 255 255
IMAGETYPE PNG
STATUS ON
OUTPUTFORMAT
  NAME png
  MIMETYPE image/png
```

```
DRIVER GD/PNG
EXTENSION png
IMAGEMODE PC256
TRANSPARENT FALSE
END
END
```

Le script php suivant générera la mapfile attendue :

```
<html>
<body>
<?php
/*
    Les paramètres à modifier

*/
$sw_MapFile = "./mapfiles/base.map";

/*
    Paramètres de connexion à PostgreSQL
*/

$pg_hote = "localhost";//"localhost";
$pg_base_de_donnees = "madatabase";
$pg_utilisateur = "david";
$pg_mot_de_passe = " ";
$pg_srid="-1";

/*
    On vérifie que les librairies de PHP sont chargées comme il faut
*/

if ( extension_loaded('pgsql') + extension_loaded('MapScript') != 2)
{
    switch (PHP_OS)
    {

        case "WINNT": if (!extension_loaded('pgsql'))  dl("php_pgsql.dll");
                        if (!extension_loaded('MapScript')) dl("php_mapscript_45.dll");
                        break;

        default:   if (!extension_loaded('pgsql'))  dl("php_pgsql.so");
                    if (!extension_loaded('MapScript')) dl("php_mapscript_dev.so");
                    break;
    }
}
/*
    OK.... On construit notre MapFile
*/
$map = ms_newMapObj( $sw_MapFile );
// Chargement du nom de la mapfile
$map->set( "name", "madatabase" );
// Déterminations des chemins d'accès vers les fonts et les symboles
switch (PHP_OS)
{

    case "WINNT": $mapserver_fontset = getcwd()."\\etc\\fonts.txt";
                    $mapserver_symbolset = getcwd()."\\etc\\symbols.sym";
                    $mapserver_imagepath = str_replace("\\\\","/",getcwd())."/tmp/";

}
```

```
        break;

default: $mapserver_fontset = getcwd()."etc/fonts.txt";
$mapserver_symbolset = getcwd()."etc/symbols.sym";
$mapserver_imagepath = getcwd()."tmp/";
        break;

}

$mapserver_imageurl = str_replace("MakeMap.php",
        "",
$_SERVER["REDIRECT_URL"]."tmp/";

$map->SetFontSet( $mapserver_fontset );
$map->setSymbolSet( $mapserver_symbolset );

$map->web->set( "imagepath" , $mapserver_imagepath );
$map->web->set( "imageurl" , $mapserver_imageurl );

// Connexion au serveur PostgreSQL
//



$db_handle = pg_connect("host=".$pg_hote."
                           dbname=".$pg_base_de_donnees."
                           user=".$pg_utilisateur."
                           password=".$pg_mot_de_passe."");

$Requete_MS_Tables = "SELECT * from mapserver_desc";
/*
if (!($Resultat_MS_Tables = pg_exec( $db_handle, $Requete_MS_Tables ) )){
{
    print( "Impossible d'interroger le serveur PostgreSQL");
    print( pg_errormessage( $db_handle) );
    exit("");
}
*/
$Resultat_MS_Tables = pg_exec( $db_handle, $Requete_MS_Tables );
// tableaux nécessaire pour calculer l'extent
$Xmin = array(); $Xmax = array();
$Ymin = array(); $Ymax = array();
while( $MS_Ligne = pg_fetch_object( $Resultat_MS_Tables ) )
{
    $layer = ms_newLayerObj( $map );
    // Nom du layer
    $layer->set( "name", $MS_Ligne->ms_name );
    // Affichage : oui par défaut
    $layer->set("status",MS_DEFAULT);
    // Type de connexion := PostGIS
    $layer->set("connectiontype",MS_POSTGIS);
    $layer->set("connection",
    "user=".$pg_utilisateur." dbname=".$pg_base_de_donnees." host=".$pg_hote);
    $layer->set("labelitem",$MS_Ligne->ms_labelitem);

/*
Pour effectuer la requete d'affichage, nous savons
que la colonne s'appelle 'the_geom' mais jouons le jeu
que nous l'ignorons. Or nous savons que le nom de la colonne géométrique
est stockée dans la colonne 'f_geometry_column' de la table
geometry_column et que le nom de la table associé est dans la colonne
f_table_name
```

```
/*
$MS_GEOM = pg_exec($db_handle,
    "SELECT f_geometry_column,type FROM
     geometry_columns WHERE f_table_name
      LIKE '". $MS_Ligne->ms_table."'"
    );

$colonne_geometrique = pg_result( $MS_GEOM,0,0 );
$type_geometrique = pg_result( $MS_GEOM,0,1 );

// La requete spatiale
$layer->set( "data",
              $colonne_geometrique." from ".$MS_Ligne->ms_table);
$MS_Code_Bordure_Couleur = array();
$MS_Code_Bordure_Couleur = explode( " ", $MS_Ligne->ms_outlinecolor);

$MS_Code_Couleur = array();
$MS_Code_Couleur = explode( " ", $MS_Ligne->ms_color);

switch ($type_geometrique)
{
    case "POINT":
        $layer->set("type",MS_LAYER_POINT);
        //
        // Ajout de la class
        //
        $class = ms_newClassObj($layer);
        //
        // Ajout du Label
        //
        //
        $label = $class->label;
        $label->set("size",MS_MEDIUM);
        $label->color->setRGB("22","8","3");
        //
        // Ajout du style
        //
        $style = ms_newStyleObj($class);
        //
        // Précision sur la taille de l'objet
        //
        $style->set("size",10);
        $style->set("symbolname", $MS_Ligne->ms_symbol );
        $style->color->setRGB($MS_Code_Bordure_Couleur[0],
                               $MS_Code_Bordure_Couleur[1],
                               $MS_Code_Bordure_Couleur[2]);
        break;

    case "POLYGON":
        $layer->set("type",MS_LAYER_POLYGON);
        // $layer->set("status",MS_OFF);
        $class = ms_newClassObj($layer);
        //
        // Ajout du Label
        //
        $label = $class->label;
        // Précision sur l'affichage de la données de type ↫
        // attributaire
        // la couleur, la taille de la donnée ...
        $label->set("size",MS_MEDIUM);
        $label->backgroundcolor->setRGB(255,255,255);
        $label->color->setRGB("22","8","3");
}
```

```
$label->outlinecolor->setRGB ("255", "255", "255");

$style = ms_newStyleObj ($class);
$style->set ("size", 5);
// $style->set ("symbolname", "circle");
$style->color->setRGB ($MS_Code_Couleur[0],
                           $MS_Code_Couleur[1],
                           $MS_Code_Couleur[2]);
$style->outlinecolor->setRGB (
                           $MS_Code_Bordure_Couleur[0],
                           $MS_Code_Bordure_Couleur[1],
                           $MS_Code_Bordure_Couleur[2]);
break;

default:   $layer->set ("type", MS_LAYER_LINE );
$class = ms_newClassObj ($layer);
//
// Ajout du Label
//
$label = $class->label;
//
//
// Ajout du style
//
$style = ms_newStyleObj ($class);
// Précision sur l'affichage des données géométriques
// couleur de bordure en code 128 0 0 pour RGB
$style->set ("size", 10);
$style->set ("symbolname", "circle");
$style->outlinecolor->setRGB (
                           $MS_Code_Bordure_Couleur[0],
                           $MS_Code_Bordure_Couleur[1],
                           $MS_Code_Bordure_Couleur[2]);
break;

}

/*
    calcul de l'Extent pour chaque table
*/
$Requete_Extent = "select
                    xmin(extent("$.colonne_geometrique.")),
                    ymin(extent("$.colonne_geometrique.")),
                    xmax(extent("$.colonne_geometrique.")),
                    ymax(extent("$.colonne_geometrique.")) from
                    ".$MS_Ligne->ms_table;

$Resultat_Extent = pg_exec( $db_handle, $Requete_Extent );

while ( $Row_Extent = pg_fetch_object( $Resultat_Extent ) )
{
    $Xmin[] = $Row_Extent->xmin;
    $Ymin[] = $Row_Extent->ymin;
    $Xmax[] = $Row_Extent->xmax;
    $Ymax[] = $Row_Extent->ymax;
}
```

```
}

/*
Calcul de l'Extent générale et des dimensions de l'image

*/
$xmin = min($Xmin);
$xmax = max( $Xmax );
$ymin = min($Ymin );
$ymax = max ($Ymax);
    $mapserver_longueur_image_mapfile = 600;
    $longueur_extent = abs($xmax-$xmin);
    $hauteur_extent = abs($ymax-$ymin);
    $rapport_extent = $longueur_extent / $hauteur_extent;
$mapserver_hauteur_image_mapfile = $mapserver_longueur_image_mapfile /
    $rapport_extent;

$map->setextent($xmin,$ymin,$xmax,$ymax);
$map->set("width", $mapserver_longueur_image_mapfile);
$map->set("height", $mapserver_hauteur_image_mapfile);

switch (PHP_OS)
{
    case "WINNT": $map->save(getcwd()."\\mapfiles\\madatabase.map");
                    break;

    default:
        $map->save(getcwd()." /mapfiles/madatabase.map");
                    break;
}

pg_freeresult( $Resultat_MS_Tables );
pg_close( $db_handle );
/*

Il faudrait commencer

*/
$image = $map->draw();
$image_url = $image->saveWebImage(MS_PNG,1,1,0);

echo "<IMG NAME='IEmapa'
BORDER=0
SRC='http://".$_ENV["SERVER_NAME"]."::$_ENV["SERVER_PORT"]." ".$image_url."/>
<BR>";

?>
</body>
</html>
```

M.3 Sortie : Mapfile

La mapfile obtenue est la suivante

```
MAP
  EXTENT 1 -10.900001525879 158.800003051758 146.900001525879
  FONTSET "c:\wamp\www\tutorial\etc\fonts.txt"
  IMAGECOLOR 255 255 255
  IMAGETYPE png
  SYMBOLSET "c:\wamp\www\tutorial\etc\symbols.sym"
  SIZE 700 594
  STATUS ON
  UNITS METERS
  NAME "madatabase"

  OUTPUTFORMAT
    NAME png
    MIMETYPE image/png
    DRIVER GD/PNG
    EXTENSION png
    IMAGEMODE PC256
    TRANSPARENT FALSE
  END

  LEGEND
    IMAGECOLOR 255 255 255
    KEYSIZE 20 10
    KEYSPACING 5 5
    LABEL
      SIZE MEDIUM
      TYPE BITMAP
      BUFFER 0
      COLOR 0 0 0
      FORCE FALSE
      MINDISTANCE -1
      MINFEATURESIZE -1
      OFFSET 0 0
      PARTIALS TRUE
      POSITION CC
    END
    POSITION LL
    STATUS OFF
  END

  QUERYMAP
    COLOR 255 255 0
    SIZE -1 -1
    STATUS OFF
    STYLE HILITE
  END

  SCALEBAR
    COLOR 0 0 0
    IMAGECOLOR 255 255 255
    INTERVALS 4
    LABEL
      SIZE MEDIUM
      TYPE BITMAP
      BUFFER 0
      COLOR 0 0 0
      FORCE FALSE
      MINDISTANCE -1
```

```
MINFEATURESIZE -1
OFFSET 0 0
PARTIALS TRUE
END
POSITION LL
SIZE 200 3
STATUS OFF
STYLE 0
UNITS MILES
END

WEB
IMAGEPATH "c:/wamp/www/tutorial/tmp/"
IMAGEURL "/tutorial/tmp/"
METADATA
END
QUERYFORMAT text/html
END

LAYER
CONNECTION "user=david dbname=madatabase host=localhost"
CONNECTIONTYPE POSTGIS
DATA "the_geom from small_roads"
LABELITEM "data"
METADATA
END
NAME "small_roads"
SIZEUNITS PIXELS
STATUS DEFAULT
TOLERANCE 0
TOLERANCEUNITS PIXELS
TYPE LINE
UNITS METERS
CLASS
METADATA
END
STYLE
ANGLE 360
OUTLINECOLOR 80 80 80
SIZE 10
SYMBOL "circle"
END
END
END

LAYER
CONNECTION "user=david dbname=madatabase host=localhost"
CONNECTIONTYPE POSTGIS
DATA "the_geom from great_roads"
LABELITEM "data"
METADATA
END
NAME "great_roads"
SIZEUNITS PIXELS
STATUS DEFAULT
TOLERANCE 0
TOLERANCEUNITS PIXELS
TYPE LINE
UNITS METERS
CLASS
METADATA
END
```

```
STYLE
  ANGLE 360
    OUTLINECOLOR 125 125 125
  SIZE 10
  SYMBOL "circle"
END
END
END

LAYER
  CONNECTION "user=david dbname=madatabase host=localhost"
  CONNECTIONTYPE POSTGIS
  DATA "the_geom from parcs"
  LABELITEM "data"
  METADATA
    END
  NAME "parcs"
  SIZEUNITS PIXELS
  STATUS DEFAULT
  TOLERANCE 0
  TOLERANCEUNITS PIXELS
  TYPE POLYGON
  UNITS METERS
  CLASS
    LABEL
      SIZE MEDIUM
      TYPE BITMAP
      BACKGROUNDCOLOR 255 255 255
      BUFFER 0
      COLOR 22 8 3
      FORCE FALSE
      MINDISTANCE -1
      MINFEATURESIZE -1
      OFFSET 0 0
      OUTLINECOLOR 255 255 255
      PARTIALS TRUE
      POSITION CC
    END
    METADATA
    END
  STYLE
    ANGLE 360
    COLOR 0 123 0
    OUTLINECOLOR 255 255 255
    SIZE 5
    SYMBOL 0
  END
END
END

LAYER
  CONNECTION "user=david dbname=madatabase host=localhost"
  CONNECTIONTYPE POSTGIS
  DATA "the_geom from rivers"
  LABELITEM "data"
  METADATA
    END
  NAME "rivers"
  SIZEUNITS PIXELS
  STATUS DEFAULT
  TOLERANCE 0
  TOLERANCEUNITS PIXELS
```

```
TYPE POLYGON
UNITS METERS
CLASS
LABEL
SIZE MEDIUM
TYPE BITMAP
BACKGROUNDCOLOR 255 255 255
BUFFER 0
COLOR 22 8 3
FORCE FALSE
MINDISTANCE -1
MINFEATURESIZE -1
OFFSET 0 0
OUTLINECOLOR 255 255 255
PARTIALS TRUE
POSITION CC
END
METADATA
END
STYLE
ANGLE 360
COLOR 0 12 189
OUTLINECOLOR 0 12 189
SIZE 5
SYMBOL 0
END
END
END

LAYER
CONNECTION "user=david dbname=madatabase host=localhost"
CONNECTIONTYPE POSTGIS
DATA "the_geom from buildings"
LABELITEM "data"
METADATA
END
NAME "buildings"
SIZEUNITS PIXELS
STATUS DEFAULT
TOLERANCE 0
TOLERANCEUNITS PIXELS
TYPE POLYGON
UNITS METERS
CLASS
LABEL
SIZE MEDIUM
TYPE BITMAP
BACKGROUNDCOLOR 255 255 255
BUFFER 0
COLOR 22 8 3
FORCE FALSE
MINDISTANCE -1
MINFEATURESIZE -1
OFFSET 0 0
OUTLINECOLOR 255 255 255
PARTIALS TRUE
POSITION CC
END
METADATA
END
STYLE
ANGLE 360
```

```
        COLOR 234 156 78
        OUTLINECOLOR 234 156 78
        SIZE 5
        SYMBOL 0
    END
END
END

LAYER
    CONNECTION "user=david dbname=madatabase host=localhost"
    CONNECTIONTYPE POSTGIS
    DATA "the_geom from personnes"
    LABELITEM "data"
    METADATA
    END
    NAME "personnes"
    SIZEUNITS PIXELS
    STATUS DEFAULT
    TOLERANCE 0
    TOLERANCEUNITS PIXELS
    TYPE POINT
    UNITS METERS
    CLASS
        LABEL
            SIZE MEDIUM
            TYPE BITMAP
            BUFFER 0
            COLOR 22 8 3
            FORCE FALSE
            MINDISTANCE -1
            MINFEATURESIZE -1
            OFFSET 0 0
            PARTIALS TRUE
            POSITION CC
        END
        METADATA
        END
    STYLE
        ANGLE 360
        COLOR 255 0 0
        SIZE 8
        SYMBOL "circle"
    END
END
END

END
```

Annexe N

Auteurs et contributeurs de PostGIS

Est mis ici les divers contributeurs pour PostGIS en France

Gérald FENOY : Titulaire d'un DEA d'Informatique, Gérald - développeur sous Gentoo - est le fondateur du site postgis.fr.

Il maintient et teste les versions de PostGIS pour les ebuilds sous Gentoo et sur Mac OS X. Gérald possède une forte compétence dans le développement et le suivi de projet, recherche fondamentale et appliquée. Il possède aussi une forte plus-value concernant les systèmes de bases de données transactionnelles bien au delà de celle de PostgreSQL.

Jean David TECHER : Titulaire d'une DEA de mathématiques, David suit le développement de PostGIS depuis l'été 2003.

David est plus spécialisé dans l'installation de serveur cartographique de A à Z sur Debian, maintenance de serveur PostgreSQL/PostGIS. David est co-fondateur du site postgis.fr.

Annexe O

Bibliographie

O.1 [1]

[PostGIS Manual] Paul RAMSEY et Sandro Santilli, *PostGIS Manual* (<http://postgis.refractions.net>), Copyright © 2006 PostGIS Team.

O.2 [2]

[Documentation PostgreSQL 8.2.1] "Documentation PostgreSQL 8.2.1 / Traduction de Guillaume LELARGE <http://traduc.postgresqlfr.org/pgsql-8.2.1-fr>", Copyright © 1996-2006 The PostgreSQL Global Development Group.