ISSN : 1620-4859

Géomatique Expert

N° 81 - Juillet-Août 2011 - 14 €

Modélisation 2D du bâti 3D

PostGIS 2.0 arrive

Télédétection des toitures végétalisées

PostGIS : vers la version 2.0

Une conférence dans un cadre inhabituel.

Résumé de la conférence organisée par Dalibo, société spécialisée dans PostGreSQL et Oslandia, experte en PostGIS, autour de l'utilisation de PostGIS et des perspectives d'avenir du produit. L'occasion, une fois n'est pas coutume, de rencontrer trois développeurs, dont un venu du Québec et l'autre d'Italie...

Dalibo, spécialiste de la base de données libre *PostGreSQL*, ainsi qu'*Oslandia*, son pendant pour la cartouche spatiale *PostGIS*, avaient décidé d'organiser de concert une journée de conférence autour des utilisations de la cartouche spatiale et de son avenir proche, c'est-à-dire les nouveautés de la version 2.0 (maintenant prévue pour l'automne). Vincent Picavet et Olivier Courtin, fondateurs d'*Oslandia* et développeurs *PostGIS*, avaient invité deux de leurs homologues, Pierre Racine, venu du Québec, et Sandro Santilli, venu des rives du lac Majeur pour parler de deux innovations principales de la version 2.0, à savoir respectivement l'intégration d'images (ou de données *raster*) et la gestion de la topologie.

Témoignages utilisateur

Utilisation de PostGIS au Conservatoire des espaces naturels du Languedoc-Roussillon – Matthieu Bossært

Matthieu Bossaert, responsable du SIG au Conservatoire des espaces naturels du Languedoc-Roussillon (une association responsable de la préservation de l'environnement dans la région), dresse l'historique de l'équipement géomatique dans sa structure.

Au tout début, comme souvent, l'association était équipée de licences individuelles *MapInfo* sur une arborescence fichier, ce qui posait les traditionnels problèmes d'accès aux référentiels (avec parfois découvertes de nombreuses copies !), de données divergentes (différentes versions), sans parler des frustrations et des difficultés nées du fait que, chacun possédant ses propres informations, les études et données produites se trouvaient éparpillées ça et là, voire se perdaient.

Dans ce contexte, on ne parlait même pas de diffusion de la donnée (publique) à l'extérieur, ni de métadonnées. Il fallait donc tout réorganiser, faciliter l'accès simultané aux informations, proposer des accès géographiques et thématiques, regrouper sur une plate-forme unique les données exogènes (référentiels…) et endogènes (études…) et saisir les métadonnées. Le Conservatoire s'est donc tourné vers la base de données *PostGIS* pour répondre à tous ces objectifs.

Sur ce noyau, les chargés d'études se sont progressivement dotés du logiciel de consultation *QGis*, afin de pouvoir visualiser, traiter et surtout imprimer les données produites ; pour la diffusion, c'est évidemment *MapServer* qui a été retenu, au sein d'un environnement plus large composé d'*Open Studio*, *GDAL/OGR*.

Avec l'organisation actuelle, l'ensemble des informations produites par le Conservatoire est directement intégré à la base de données (par exemple, saisie de traces GPS, inventaires floristiques ou faunistiques) ; *QGis* est utilisé pour intégrer ces données à la base (saisie), et pour les consulter (quoique *MapInfo* serve encore ponctuellement dans ce but, après export fichier au bon format). La diffusion externe passe par le couple *Carmen* & *MDweb*.

Parmi les projets qui ont pu être menés à bien grâce à la nouvelle architecture et la puissance de *PostGIS*, on peut citer :

• Une analyse des enjeux de conservation de la nature sur le patrimoine (5 à 6 000 parcelles) où il fallait hiérarchiser les parcelles les plus importantes ;

• Moderniser l'inventaire des *ZNIEFF* (sous la responsabilité Muséum). Projet démarré en 2005, comportant une collecte de l'information auprès de différents partenaires (réalisée à partir de 2007), suivie du dessin des nouvelles zones. Ce dessin a été effectué à la main, puis les périmètres soumis à la validation d'un conseil scientifique régional. Une fois le quitus acquis, des fiches précisant l'intérêt et les enjeux ont été rédigées, et des cartes dressées pour publication sous *MapServer*. Ces dernières donnent accès aux informations attributaires (raison du classement, renvoi aux pages correspondantes du site du Muséum) et redirigent sur le portail national lorsqu'il s'agit de zones *Natura 2000* ;

• Cartographie des enjeux de préservation des propriétés du Conservatoire : déterminer quel est l'enjeu et comment mettre en œuvre la politique de protection. Il s'agissait d'intégrer six grilles de notation, de pondérer les critères en fonctions de paramètres comme la surface, etc. et, enfin, de cartographier les mesures.

En conclusion, *PostGreSQL*, utilisé depuis 2006 est qualifié d'outil très gratifiant : ni très simple, ni très compliqué, même des stagiaires novices en informatique ont su le prendre correctement en main. La facilité d'administration est également mise en avant. À ce jour, tous les objectifs du Conservatoire ont été atteints, sans rencontrer ni bogues, ni limite fonctionnelle.

Parmi les perspectives : se débarrasser de *MapInfo* au profit de *QGis* ; incorporer les *rasters* dans la base avec *PostGIS* 2.0, développer un couplage avec la statistique et la détection de niche. En outre, en guise de participation au projet, le Conservatoire publie l'intégralité des requêtes et des fonctions qu'il développe sur un site *web*.

Préfecture de Police : outil de géocodage – Julien Moquet

La Préfecture de Police, particularité parisienne (dans toutes les autres villes de France, le maire

détient les pouvoirs de police) regroupe 40 000 agents sous tutelle du ministère de l'Intérieur. Il dispose de trois niveaux de SI, issus du ministère de l'Intérieur, de la direction technique et de la sous-direction responsable du développement des logiciels.

La principale préoccupation « géographique » de la Préfecture concerne le géocodage et la validation des adresses. Répondre à

PostGIS 2.0 sera-t-il compatible avec les ordinateurs du Crétacé ?

ces besoins signifie, en premier lieu, adopter un référentiel de qualité. À l'origine, la PdP utilisait un vieux produit fonctionnant sur .NET ; cet outil périmé a été modernisé et porté en *Java*, rebaptisé au passage *JDonRef* ; prochainement, il devrait intégrer le référentiel produit par la BSPP sur Paris et la Petite couronne. Ce logiciel, issu de développements propres à la PdP, mais utilisable sur tout le territoire national, a été publié sur la forge *Adullact*, pour que le plus grand nombre en profite ; en effet, il n'a pas d'équivalent : la plupart des solutions existantes exploitent la norme postale, produite en collaboration avec l'UPI (Union postale internationale), qui n'est pas assez flexible (pas de possibilité de géocoder dans des cours, par exemple), géocodent moins bien (*JDonRef* revendique plus de 80 % de taux de succès) et sont trop onéreuses. De fait, l'Intérieur, le ministère de l'Écologie, entre autres, ont adopté ce produit.

JDonRef repose sur une interface d'échange client/serveur (protocoles *SOAP* et *JSON*) ; une adresse est transmise au service qui retourne une adresse validée et des coordonnées (x, y). Ce schéma autorise l'intégration non seulement dans un SIG, mais également dans n'importe quel applicatif métier. Ce dernier, dans le cas de la Préfecture, gère un certain nombre de services auxiliaires : l'analyse des abréviations (MOTS), le géocodage, la partie DAO, l'interface messagerie pour remonter les informations rapatriées.

Au sein de ce modèle, *PostGIS*/*PostGreSQL* servent uniquement à trouver la donnée, toute la gestion métier étant prise en charge par la couche *Java* ; par exemple, la gestion des écart vis-à-vis de la voie n'est pas dévolue à une fonction *PostGIS*, qui positionne par défaut sur la voie, mais par la *JTS* au sein de *JDonRef*.

Pourquoi une base de données plutôt qu'un fichier à plat ? Parce que la gestion de la donnée géographique réclame un *SGBD*, qui permet manipulations et mises à jour en temps réel. De plus, un paramétrage fin de la base ouvre la porte à des optimisations significatives en temps d'accès et performances, sans compter l'utilisation d'index spatiaux. À l'heure actuelle, la performance atteint environ dix géocodages par seconde.

Une nouvelle fonction de géocodage inverse vient d'être ajoutée. Elle s'appuie sur la primitive *ST_Distance*, sur un index *GIST* et une optimisation du rectangle englobant pour déterminer quelle est l'adresse (ou les adresses) les plus proches d'une coordonnée. Le développement de cette fonctionnalité s'est effectué totalement en interne, en six mois à mi-temps (soit 70 jours•homme).

Parmi les évolutions prévues pour 2012 :
- L'intégration des POI ;
- La notion de zone : une adresse fait-elle partie d'une zone à risques (îlot de délinquance…) ;
- L'ajout des DOM/TOM et de la Corse.

En revanche, certains projets ont été différés voire abandonnés : l'interface *web* de géocodage en masse ; l'utilisation de l'*ETL* libre *Talend* pour la diffusion des données ; la gestion des Cedex (base payante de La Poste).

À plus long terme, l'équipe de développement envisage peut-être un outil de géocodage compatible avec/lié à OSM, et un service OGC de type WMS/*WFS* (ou des normes de géocodage, mais qui ne sont actuellement pas utilisées par l'industrie).

PostGIS 2.0, les nouveautés

PostGIS Raster – Pierre Racine

L'idée d'intégrer le type *raster* dans la base *PostGIS* est née des besoins d'analyse forestière à l'échelle du Canada (soit 17 fois la France), qui se heurtaient au problème de représentation des données à très petite échelle. Le développement de cette nouvelle capacité a été financé par l'état canadien, plus différentes sociétés privées, et l'*OSGéo* pour l'infrastructure logicielle.

La fonctionnalité *raster* sous *PostGIS* vaut aussi bien pour données de type MNT que des images (il n'y a pas de distinction). Il s'agit d'un nouveau type de données, qui ressemble au type géométrie en ce sens que chaque ligne contient une image, et qu'une table stocke une couverture matricielle (on peut donc considérer, dans le cas des mosaïques, qu'une ligne égale une tuile). L'API de gestion de ce nouveau type, qui sera livré à l'automne avec la version 2.0 de *PostGIS*, s'appuie entièrement sur *SQL*.

Le type *raster* est tuilé et géoréférencé dans un méta-élément de type *Coverage*, qui contient les coordonnées du point supérieur gauche, la résolution du pixel, la largeur et hauteur du *raster*, ainsi qu'un éventuel angle de rotation (*skew*) ; les *rasters* multi-bandes sont pris en charge, selon plusieurs représentation possibles, sans aucune limitation sur le nombre de bandes.

La valeur conventionnelle *NULL* ou n'importe quelle valeur numérique précisée permet de représenter des « trous » ou des données non-significatives. *PostGIS* n'impose pas de limite sur la taille de la couverture, mise à part celles de *PostGreSQL* (1 Gio par tuile (= ligne), 32 Tio par table, limites *PostGreSQL*) ; pour gagner de la place, les données sont compressées à la volée par *PostGreSQL*. Mieux encore, le support de la multi-résolution (*overview*), c'est-à-dire la possibilité d'avoir des aperçus à plus faible résolution dans des images détaillées. Le type *raster* stocke ainsi des couvertures irrégulières ou régulières, de forme quelconque, voire, à la limite, non connexes. Il nécessite, à l'instar du type *Geometry*, l'ajout d'une métatable *raster_columns* identique à *Geometry_column*.

Concrètement, que peut-on faire avec *PostGIS raster* ? Premièrement, du stockage. Un outil de gestion des images, pendant de *shp2pgsql*, *raster2pgsql*, écrit en *Python*, insère des fichiers dans la base. Comme cet outil utilise *GDAL*, tous les formats *raster* reconnus par *GDAL* sont importables ; l'outil offre en outre la possibilité de « retuiler » les clichés à la volée. Inversement, *PostGIS raster* sait convertir une ligne vers n'importe quel format, au travers des fonctions *ST_AsGDALraster, ST_AsTIFF, ST_AsJPEG, ST_AsPNG* (les taux de compression sont paramétrables).

Secondement, la recherche des métadonnées liées à l'objet *raster* grâce à la fonction *ST_Metadata* : où se positionne l'objet, quelles sont ses dimensions, combien a-t-il de bandes, quel est le type numérique de chaque bande (octet, entier, réel…), quelle est la valeur non significative, etc. Des fonctions spécifiques changent ces métadonnées : *ST_SetScale, ST_SetGeoReference, ST_SetBandNoDataValue*… La reprojection des *rasters* s'effectue avec la primitive *ST_Transform*, pour laquelle plusieurs algorithmes d'interpolation (disponibles dans *GDAL*) sont utilisables. Pour l'instant, l'extraction de la base vers un fichier *raster* n'est pas réalisable ; plus tard, un outil dédié proposera une exportation ligne par ligne (mosaïque) ou comme une seule image (cliché plan).

Afin d'optimiser l'accès aux images dans le cas de portails *web*, par exemple, le type *raster* peut recevoir, au lieu d'un fichier, un simple lien vers un chemin d'accès. Il n'y a donc plus d'échange de données entre la base et le frontal de publication.

Cette option rend également la création des catalogues d'images plus efficace et dispense des soucis de sauvegarde. En revanche, dans l'état actuel du code, les requêtes *SQL* n'ont accès qu'aux métadonnées. À terme, ces requêtes agiront sur les images qu'elles soient stockées ou non dans la base.

Parmi les autres fonctions, certaines concernent les statistiques d'image : *ST_SummaryStat* fournit un *tuple* contenant les valeurs minimale et maximale, la moyenne, la variance de l'ensemble des pixels, ainsi que le nombre de pixels significatifs ; à titre d'exemple, sur une tuile

Pierre Racine a entamé et coordonne le développement du support raster pour PostGIS 2.0.

SRTM de 70 Mio, cette fonction prend une dizaine de secondes. *ST_Histogramme*, *ST_Quantile* calculent les quantités correspondantes. *ST_ValueCount* dénombre les pixels d'une valeur donnée. Toutes ces fonctions peuvent travailler sur une tuile ou une mosaïque.

Le passage du *raster* au vecteur n'a pas été oublié. La primitive *ST_AsPolygon* regroupe les pixels par valeur, et forme des polygones qui sont ensuite retournés accompagnés de la valeur des pixels intérieurs. Cette fonction pourra servir à afficher des *raster* dans des logiciels qui ne sont pas directement compatibles avec *PostGIS*, par exemple *ArcGIS 10*. Cependant, il faut bien comprendre que les très grandes images ne pourront pas être vectorisées (cela prendrait trop de temps).

L'édition n'a pas été oubliée : *ST_SetValue* affecte la valeur para-

mètre à un ou plusieurs pixels ; *ST_Reclass* opère une « reclassification », c'est-à-dire un changement d'échelle, éventuellement selon des indications données dans une table. Enfin, une autre fonction modifie la valeur non significative conventionnelle.

Un embryon d'algèbre de carte est disponible avec la primitive *ST_MapAlgebra* qui prend comme paramètres une bande *raster*, une expression mathématique à appliquer, une autre expression pour les pixels marginaux et un type de sortie ; l'expression est évaluée par le parser de *PostGreSQL*, ce qui signifie que toutes les fonctions mathé-

Sandro Santilli, principal développeur de la couche topologique de PostGIS 2.0.

matiques connues de la base sont utilisables (*sqrt*, etc.). Toutefois, dans la première version du code, les expressions sont limitées au pixel courant (on ne peut donc pas s'en servir pour réaliser du filtrage par convolution, par exemple).

L'intérêt du type *raster* réside également dans son croisement avec sa contrepartie vecteur. Ainsi, avec *ST_Intersects*, on peut extraire des valeurs d'élévation de points *Lidar* ou encore déterminer l'intersection de couches linéaires (routes…) avec un MNT, intersection qui n'existe qu'aux endroits où la couche *raster* possède une valeur significative.

Outre les points déjà signalés, les améliorations à venir incluent : la conversion de géométries vectorielles en *raster*, le calcul d'intersections *raster/raster*, la possibilité de réaliser des calculs algébriques sur des pixels voisins, l'implémentation de calculs de pentes (gradients), l'agrégation des tuiles, des fonctions d'interpolation et de ré-échantillonnage, la réalisation de cartes de densité…

Support de la topologie - Sandro Santilli

L'idée d'enrichir *PostGIS* avec des objets topologiques n'est pas nouvelle. Elle avait été formalisée, mais n'avait pas abouti à un code concret. En 2010, le projet a donc été repris par Sandro Santilli, sur financement de la région de Toscane, pour la version 2.0 ; il se conformera à la norme ISO *SQL/MM*, qui définit déjà les objets géométriques « classiques » : points, lignes, polygones, arcs.

L'intérêt d'utiliser des objets topologiques est multiple : les contraintes imposées (bordures communes, non chevauchement…) contraignent les géométries, et les primitives testent si les géométries satisfont ou non ces contraintes (exemple : fermeture).

Ces contraintes entre objets topologiques se traduisent par des relations de voisinage, qui, au niveau conceptuel, sont modélisées par des arcs orientés, et où les intersections (sommets des formes) se transforment en nœuds.

Cette représentation abstraite sous forme de graphe ne nécessite qu'une place fort réduite, tout en fournissant des informations essentielles, par exemple identifier, pour chaque côté, quel est l'objet à gauche et à droite, ou bien distinguer l'intérieur d'un objet de son extérieur grâce au sens de parcours des arcs orientés. Le modèle correspondant s'appelle *topo-géométrie*, il regroupe des objets éventuellement multiples (multi-polygones, multi-géométries).

L'implémentation de cette représentation abstraite passe par la définition d'un modèle topologique au travers de la création d'un schema *PostGreSQL* particulier qui contient toutes les fonctions et les objets. Ce schéma incorpore la définition des objets et des types, des métatables similaires à la colonne *Geometry* classique, ainsi que quatre tables différentes pour stocker les éléments du graphe modèle (edge, node, face, composition), avec la contrainte que chaque objet n'appartient nécessairement qu'à une seule table.

Pour créer une topologie, il faut ajouter une métatable *topogeometry*, spécifier le nom du schema courant (pas nécessairement celui dans lequel se trouve le modèle), le type (point, ligne…) et son nom. Les objets de même dimension peuvent être groupés dans des métaobjets de types particuliers (*lineal* en dimension 1, *areal* en dimension 2…).

L'utilisation des objets topologiques se trouve facilitée par la présence d'un *cast* implicite du type *topogeometry* vers le type classique *Geometry*, ce qui signifie pratiquement que toutes les fonctions classiques de *PostGIS* s'appliquent au type *topogeometry*. La topologie se modifie grâce à des primitives comme *ST_AddEdge* qui permet d'ajouter des arcs, sachant que *PostGIS* ferme les faces si besoin est ou en crée de nouvelles ; *ST_NewEdgeHeal*, par exemple, supprime un nœud (point) et joint les côtés adjacents en conséquence.

D'autres fonctions, non normalisées (*Edge, Node, Polygonize*) modifient également la structure mais n'opèrent pas nécessairement les contrôles appropriés. Les fonctions comme *GetNodeByPoint*, *GetFaceByPoint*… permettent de retrouver les objets qui contiennent un point particulier, *GetRingEdges* énumère les côtés composant un polygone fermé.

D'autres fonctions très utiles, quoique non normées par *SQL/MM*, sont également implémentées : *ST_Split* divise une forme par une autre forme (y compris un objet par un hyperplan), *ST_Snap* (avec une tolérance optionnelle) contraint les bords à adopter une géométrie précise, *ST_UnaryUnion* définit une union qui fonctionne pour toutes les géométries, même mixtes. Enfin, *ST_MakeValid* corrige la topologie (par exemple en transformant en multi-polygones un polygone auto-intersectant).

Les futurs ajouts prévus incluent une fonction de création d'une topologie à partir d'une collection, la suppression de côtés entiers ou encore la vérification des topologies. À venir également, la possibilité de convertir le type *Geometry* en *topogeometry* (opération assez longue à conduire à la main), l'import/export au format *Tiger* (*Topologically Integrated Geographic Encoding and Referencing system*, développé par l'office du recensement américain), un pilote *OGR* et/ou un interfaçage direct avec le modèle de *Grass*, qui est déjà topologique.

Fonctions 3D dans PostGIS 2.0 – Olivier Courtin

Les premiers travaux concernant l'intégration de données 3D dans *PostGIS* remontent à 2007, à une époque où il n'existait pas encore de véritables données, pas de visualisateurs adéquats, donc pas ou presque pas de besoins clients (aucun financement).

Actuellement, les technologies ont mûri et l'offre devient beaucoup plus pléthorique. En outre, les standards ont été promulgués (2006 : norme stockage) et les besoins évoluent du stockage pur à l'analyse topologique, dont la formalisation ne fait que commencer.

La représentation des objets 3D passe par un type de base, le triangle, bien que *PostGIS* 3D définisse également des formes polygonales ou des surfaces polyédriques, qui permettent par exemple de définir un objet 3D (parallélépipède, cube...) comme une succession de faces. Les facettes triangu-

Olivier Courtin (au premier plan), l'un des co-fondateurs d'Oslandia, participe a développement des fonctions 3D dans PostGIS ; Jonathan Derrough, de la société Orbe (au second plan), présente un démonstrateur de ces nouvelles fonctions.

laires peuvent être agrégées dans un *TIN*, lequel peut servir à décrire une surface plane ou gauche.

Les fonctions 3D prévues dans *PostGIS* 2.0 tourneront autour de la manipulation des *TIN*, des facettes triangulaires, des polyèdres, plus 40 à 50 « méta-fonctions » (export, manipulation, accès à la géométrie…) ; cet ensemble couvrira le stockage, des fonctions de base en analyse spatiale, l'import/export et la création d'index spatiaux multidimensionnels. L'objectif est de consolider un socle suffisant pour générer des besoins supplémentaires et financer de nouveaux développements.

Du côté des formats d'exportation vers *Internet*, pour l'instant, seul le *GML3* sera supporté, le *CityGML* en *WFS* arrivera plus tard ; quant à *web3DS*, faute d'implémentation *OpenSource* connue, il ne sera pas non plus disponible.

Que reste-t-il à faire ? Il manque beaucoup de fonctions : des algorithmes de triangulation, de généralisation (simplifier un *TIN* le plus possible en perdant le moins d'informations), des tests de validité de géométrie 3D (complexe !), la spécification de *data* verticaux (mais quelle grille d'altitude utiliser ?), le support des *TIN* de grand volume (ceux-ci seront limités en taille, dans un premier temps), le stockage des MNT, la gestion de textures, et enfin le support des opérations topologiques volumiques (intersections, unions…), un projet d'une telle ampleur qu'il est envisagé de répartir la charge entre plusieurs applications libres (synergies avec des projets comme *Blender* ?)

Les premiers démonstrateurs compatibles avec les objets 3D devraient faire leur apparition à l'automne, alors que les prototypes réels ne seront sans doute pas disponibles avant la fin de l'année, pour une mise en production courant 2012. ■