Connexion depuis geoserver à des données raster stockées dans postgresql

(04/2015)

Version utilisées : Geoserver 2.7, Postgresql 9.4, Postgis 2.1.7

1. Prérequis dans postgresql

Pour stocker des données raster dans Postgresql, il faut installer le plugin postgis.

Puis à la création de la base de données qui servira à stocker des données raster, il faut lui appliquer les scripts sql suivants (qui sont stockées dans /usr/share/postgresql/9.4/contrib/postgis-2.1/):

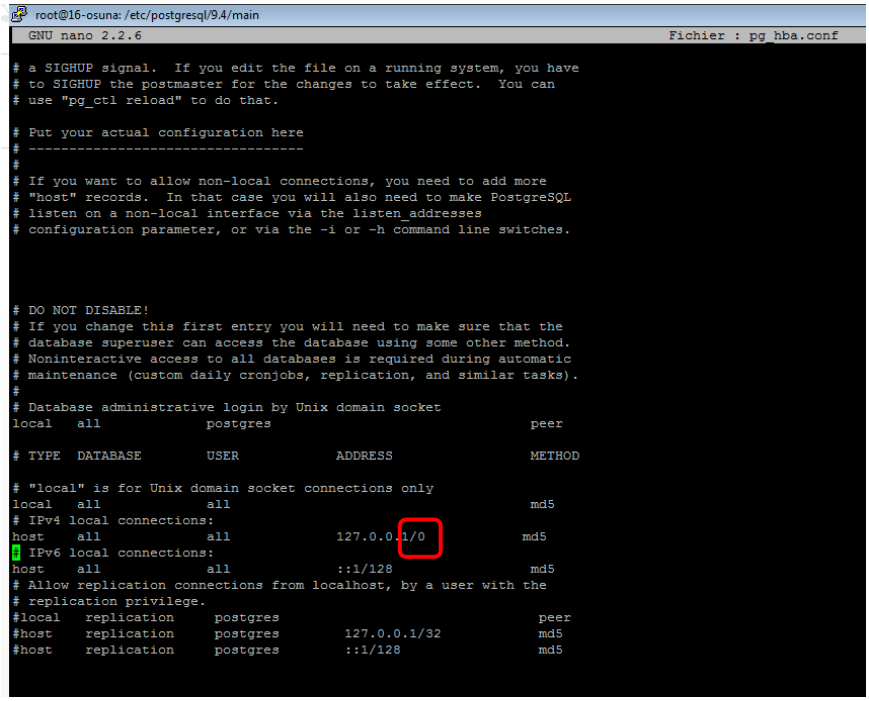
Postgis.sql

Spatial_ref_sys.sql

Rtpostgis.sql (ce dernier est nécessaire pour les données raster)

Penser également à bien vérifier les droits sur la base de données

Autoriser les connexions depuis l'extérieur, ici tout le monde est autorisé en modifiant dans un fichier de conf de postgresql : pg_hba.conf, les ports de connexion.



Attention : ici tester en restreignant, car pour le moment tout le monde pourrait s'y connecter

Ceci afin d'éviter les erreurs dans geoserver de type : *Connexion refusée. Vérifiez que le nom de machine et le port sont corrects et que postmaster accepte les connexions TCP/IP.*

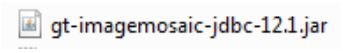
2. Préparation de géoserver

- **Installation du plugin image mosaic JDBC**

Pour pouvoir se connecter à une base de données postgresql stockant des données raster, nous allons utiliser le plugin Image Mosaic JDBC.

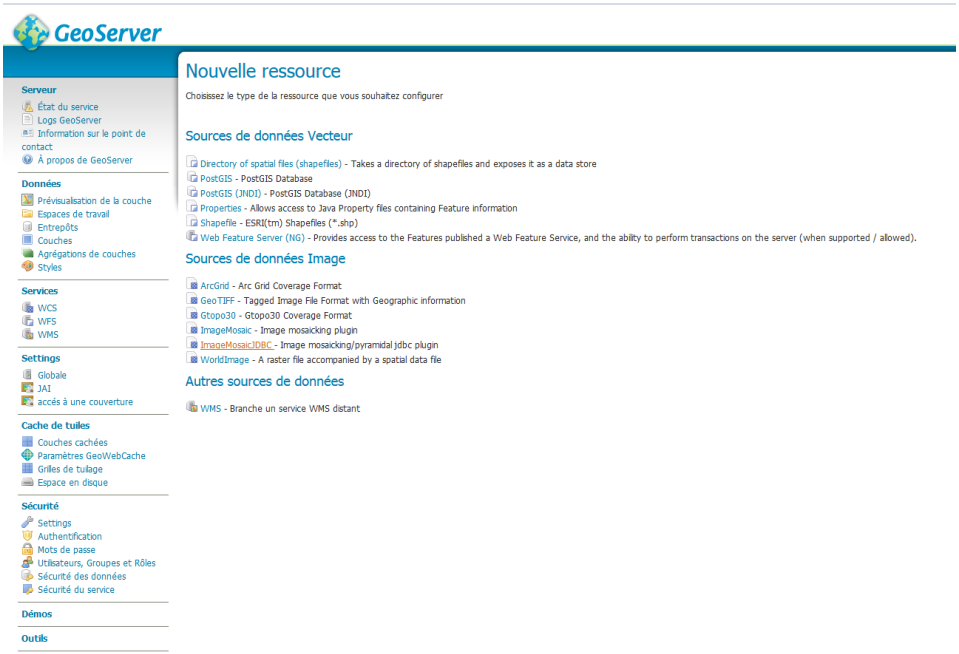
http://docs.geoserver.org/stable/en/user/data/raster/imagemosaicjdbc.html

Vérifier que celui-ci est bien installé dans les dossiers de geoserver (/WEB-INF/lib/)



Sinon il suffit de le télécharger et de l'installer

après un redémarrage de geoserver, il est visible lors de la création d'un entrepôt de données.



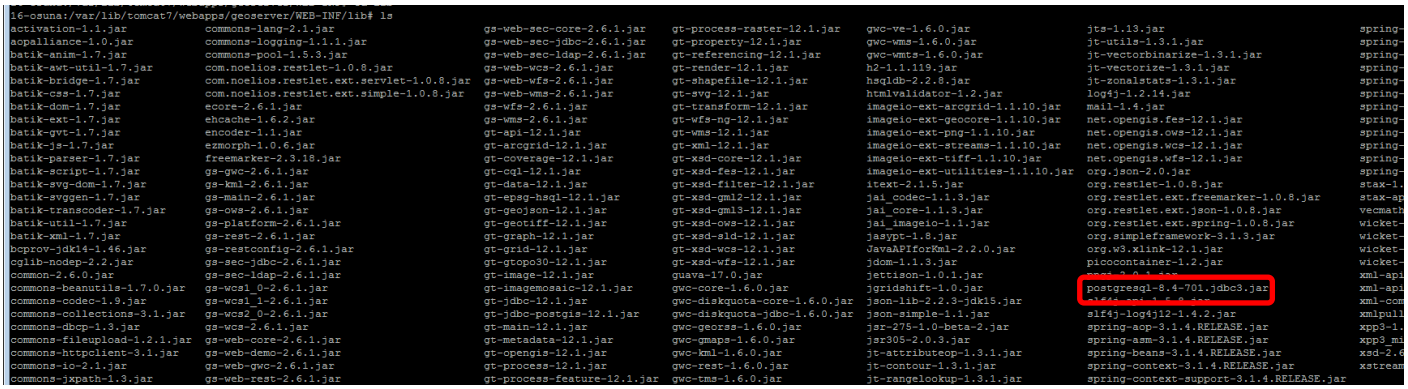
Voici le tutoriel pour utiliser ce plug-in dans geoserver

http://docs.geoserver.org/stable/en/user/tutorials/imagemosaic-jdbc/imagemosaic-jdbc_tutorial.html

pour une description plus complète du plug-in :
http://docs.geotools.org/latest/userguide/library/coverage/jdbc/index.html

- **Vérifier que geoserver dispose des fichiers de connexion avec postgresql à jour**

En effet geoserver ne dispose pas généralement du dernier driver de postgresql, c'est en effet la version de postgresl 8.4 (vérifier dans /WEB-INF/lib/).



Télécharger une version plus récente sur https://jdbc.postgresql.org/download.html

3. Mise en forme des données

Il est conseillé de tuiler les données raster avant de les intégrer dans une base de données postgresql

L'outil qui peut être utilisé est gdal_retile.py pour créer une mosaïque

http://gdal.gloobe.org/gdal/gdal_retile.html

gdal_retile.py -ps <taille du pixel à utiliser pour le fichier de sortie> -of <format de sortie> -levels <nombre de niveaux de pyramide à construire> –targetDir <le répertoire dans lequel les tuiles résultantes seront créées> < adresse de l'image de départ>

exemple d'une image de 1950 x4414 pixels

gdal_retile.py -ps 488 1104 -of Gtiff -levels 2 –targetDir /var/lib/tomcat7/webapps/geoserver/data/tiles2 /osuna-data/test/2002_08_bourgneuf_MPB_PicChloA.tif

Intégration des données raster dans postgresql avec l'exécutable raster2pgsql

http://postgis.net/docs/using_raster_dataman.html

qui s'utilise de cette manière en ligne de commande

raster2pgsql -I -C -r -s 32630 -t *< taille de la tuile ex.488x1104> <adresse du dossier où se trouve les images tuilées à intégrer> <nomschema.nomtable>* **|psql -U** *<nom du propriétaire>* **-d** *<nom de la table>* **–p** *<port>*

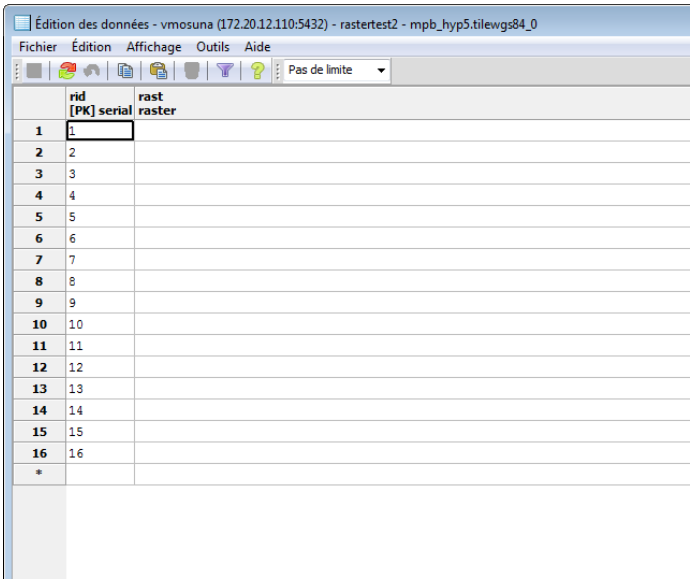
-I (option) : create overview of the raster. For more than one factor, separate with comma(,). Overview table name follows the pattern o_`overview factor_table`, where `overview factor` is a placeholder for numerical overview factor and `table` is replaced with the base table name. Created overview is stored in the database and is not affected by -R. Note that your generated sql file will contain both the main table and overview tables.

-C (option): Apply raster constraints -- srid, pixelsize etc. to ensure raster is properly registered in `raster_columns` view.

-r (option): Set the constraints (spatially unique and coverage tile) for regular blocking. Only applied if -C flag is also used.

…

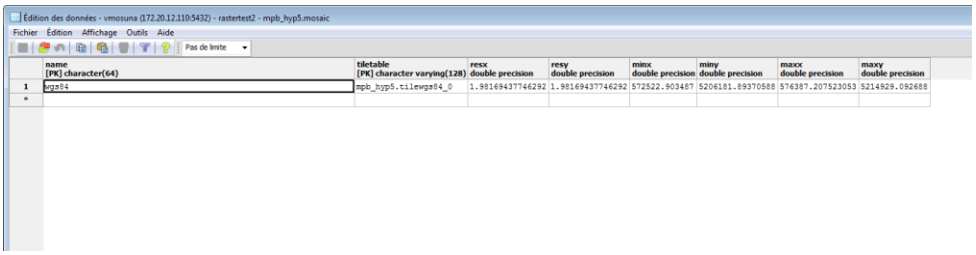
Exemple de structure de table de données obtenue : 1 tuile par enregistrement



Une table métadonnées est à créer, qui fait le lien entre toutes les images créées (ou pyramide) par l'outil GDAL_RETILE

Exemple de script pour la créer par exemple :

CREATE TABLE schema.mosaic(name CHARACTER (64) NOT NULL,tiletable VARCHAR (128) NOT NULL,resx FLOAT8,resy FLOAT8,minx FLOAT8,miny FLOAT8,maxx FLOAT8,maxy FLOAT8,CONSTRAINT MASTER_PK PRIMARY KEY (name,tiletable));



Puis il faudra remplir les deux premiers champs : name et tiletable.

Le champ name correspondra au nom que l'on donnera au jeu de données raster

Le champ tiletable correspond au nom que l'on a attribué à la table stockant les tuiles (il peut y avoir plusieurs tables avec des tuiles de différentes tailles pour un même jeu de données raster

4. Configuration dans geoserver

Créer 3 fichiers de configuration qui seront utilisés par geoserver avec le plugin Image mosaic jdbc (à placer dans le même dossier)

3 fichiers qui seront nommés dans l'exemple :

Connect.postgis.xml.inc

Mapping.xml.inc

Wgs84.postgis.xml

Le fichier **connect.postgis.xml.inc** sert à décrire la connexion à la bdd postgresql

```
<connect>
 <!-- value DBCP or JNDI -->
 <dstype value="DBCP"/>
 <!--  <jndiReferenceName value=""/>  -->
        <username value="nom de l'utilisateur de la bdd" />
        <password value="mot de passe" />
        <jdbcUrl value="jdbc:postgresql://localhost:5432/nom de la bdd" />
        <driverClassName value="org.postgresql.Driver"/>
        <maxActive value="10"/>
        <maxIdle value="0"/>
</connect>
```

Le fichier wgs84.postgis.xml

Le **fichier mapping.xml.inc** qui sert à décrire chaque champs de toutes les tables du jeu de données raster dans la bdd postgres

```
<!-- possible values: universal,postgis,db2,mysql,oracle -->
<spatialExtension name="pgraster"/>
<mapping>
  <masterTable name="nom de la table métadonnées" >
   <coverageNameAttribute name="name"/>
        <tileTableNameAtribute  name="tiletable" />
        <resXAttribute name="resx"/>
        <resYAttribute name="rresy"/>
        <minXAttribute name="minx"/>
        <minYAttribute name="miny"/>
         <maxXAttribute name="maxx"/>
         <maxYAttribute name="maxy"/>
  </masterTable>
  <tileTable>
        <blobAttributeName name="rast" />
         <keyAttributeName name="rid" />
  </tileTable>

</mapping>
```

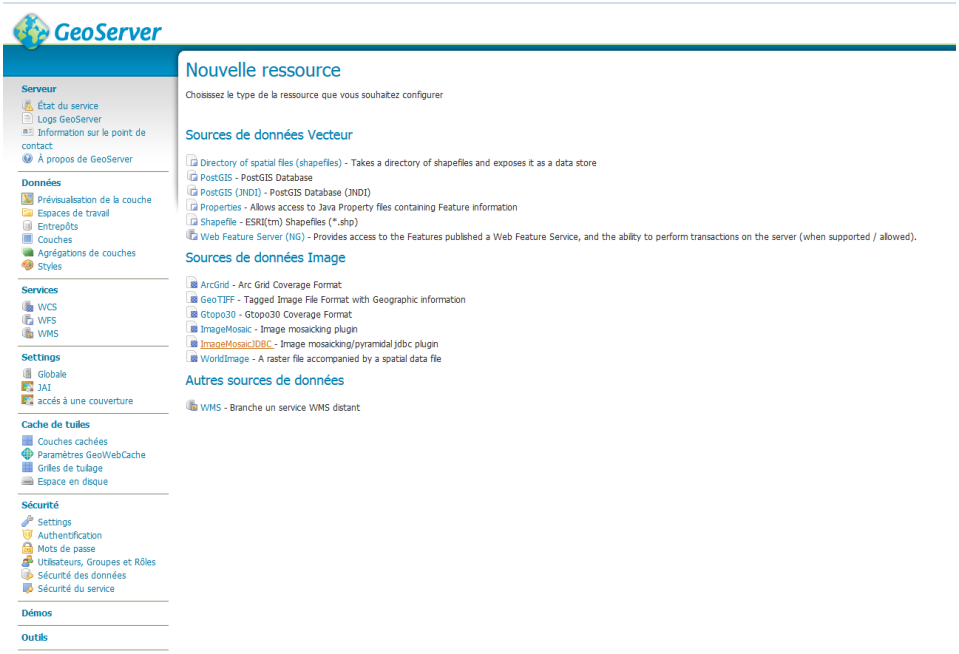
Le fichier wgs84.postgis.xml qui fait le lien avec les deux autres fichiers

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE ImageMosaicJDBCConfig [
 <!ENTITY mapping PUBLIC "mapping"  "mapping.postgis.xml.inc">
 <!ENTITY connect PUBLIC "connect"  "connect.postgis.xml.inc">]>
<config version="1.0">
 <coverageName name="nom du jeu de données que l'on a indiqué dans le champs name de la table métadonnées"/>
 <coordsys name="EPSG:32630"/>
 <!-- interpolation 1 = nearest neighbour, 2 = bilinear, 3 = bicubic -->
```

```
<scaleop  interpolation="1"/>
 <axisOrder ignore="false"/>
   &mapping;
   &connect;
        </config>
```

5. Connexion à la base de données postgis raster

A la création d'un entrepôt dans geoserver, utiliser ImageMosaicJDBC



Indiquer ici l'adresse du fichier xml (ici wgs84.postgis.xml)

Si un message d'erreur dans les logs de geoserver apparait de ce type :

'rt_raster_to_gdal: Could not load the output GDAL driver'

il semblerait que The GUC postgis.gdal_enabled_drivers is not available in 2.1. Instead you need to specify the equivalent environmental variable for PostgreSQL.

Il faut installer une variable environnement dans un fichier de postgresql

/etc/postgresql/9.4/main/environnement

Ajouter ceci **POSTGIS_GDAL_ENABLED_DRIVERS=ENABLE_ALL**

Recommandations :

Les données rasters étant des données assez lourdes, éviter de les stocker directement dans postgresql. En effet il n'est pas possible de stocker des raster de plus de 65535 x 65535 pixels (a vérifier).

Il faut les tuiler

The maximum width x height permitted for the PostGIS raster type is 65535 x 65535, regardless of whether or not the raster is in-db or out-db. The other reason to tile your raster (though this may not apply in your case) is that the maximum field size permitted by PostgreSQL is 1 GB [1].

As for optimal tile size, I can only suggest two things.

1. Tile sizes <= 100 x 100 are best. smaller is faster but consumes more storage space.
2. If possible, find a tile size that is cleanly divisible from the raster's dimensions. So for a raster of 42971 x 77138, no tile size <= 100 x 100 works cleanly. In these situations, I usually just go 50 x 50 or something in that neighborhood.

Il est conseillé de stocker les raster dans des fichiers et d'utiliser la fonction out-db raster avec l'option –R de raster2pgsql

-R, --register

Register the raster as a filesystem (out-db) raster.

Only the metadata of the raster and path location to the raster is stored in the database (not the pixels).

S

Inconvenient : les données ne sont pas visibles à partir de postgresl dans QGIS ni dans geoserver, il est alors preferable de consulter directement les raster dans les fichiers.