

# 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/):

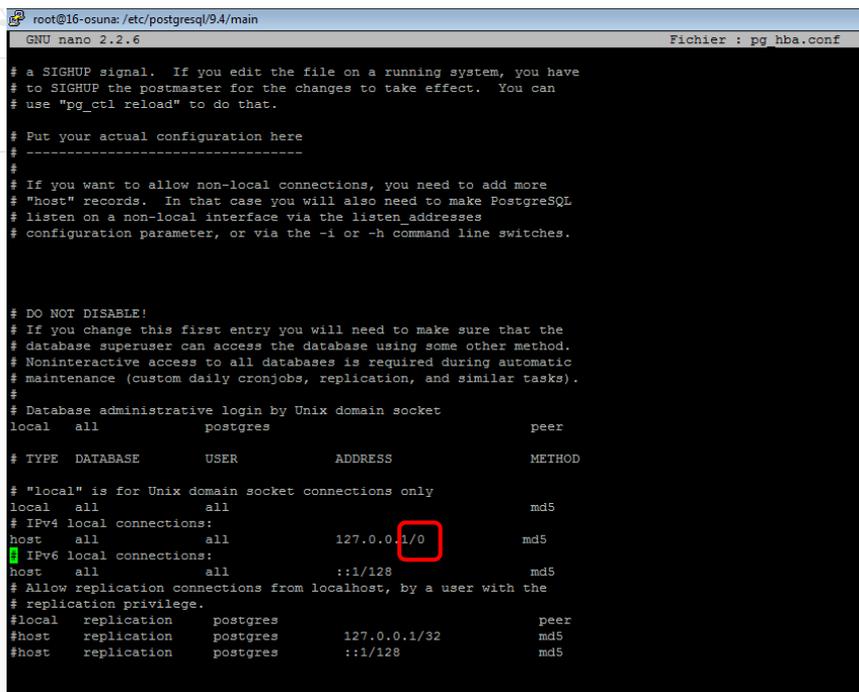
Postgis.sql

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.



```
root@16-osuna: /etc/postgresql/9.4/main
GNU nano 2.2.6 Fichier : pg_hba.conf
# a SIGHUP signal. If you edit the file on a running system, you have
# to SIGHUP the postmaster for the changes to take effect. You can
# use "pg_ctl reload" to do that.

# Put your actual configuration here
#-----
# If you want to allow non-local connections, you need to add more
# "host" records. In that case you will also need to make PostgreSQL
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -i or -h command line switches.

# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#-----
# Database administrative login by Unix domain socket
local all postgres peer

# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all md5
# IPv4 local connections:
host all all 127.0.0.1/0 md5
# IPv6 local connections:
host all all ::1/128 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication postgres peer
#host replication postgres 127.0.0.1/32 md5
#host replication postgres ::1/128 md5
```

**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.

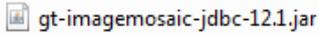
## 2. Préparation de géoserver

- 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.

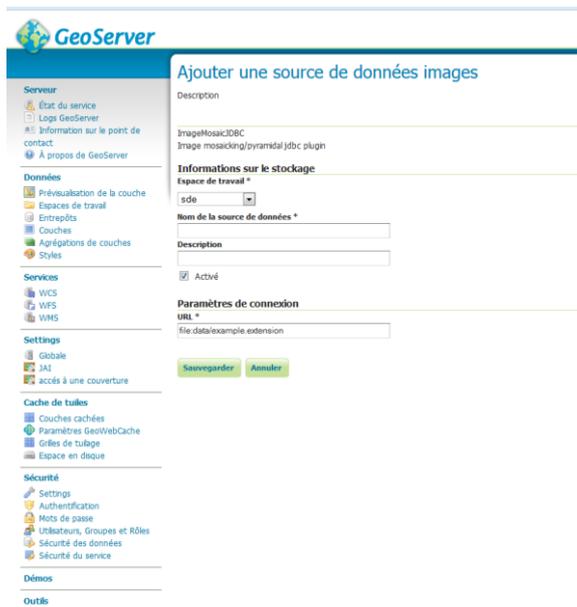
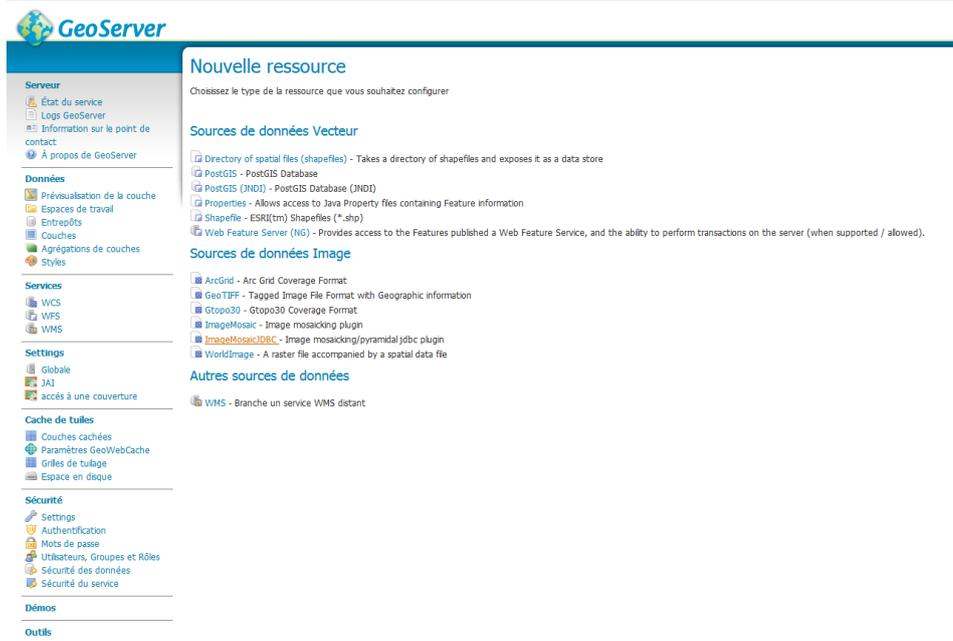
<http://docs.geoserver.org/stable/en/user/data/raster/imagemosaicjdbc.html>

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](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 postgres 8.4 (vérifier dans /WEB-INF/lib/).

```
i6-osuna:/var/lib/tomcat7/webapps/geoserver/WEB-INF/lib# ls
activation-1.1.jar commons-lang-2.1.jar gs-web-sec-core-2.6.1.jar gt-process-raster-12.1.jar gwc-ve-1.6.0.jar jts-1.13.jar spring-
activation-1.0.jar commons-logging-1.1.1.jar gs-web-sec-ldap-2.6.1.jar gt-property-12.1.jar gwc-wms-1.6.0.jar jst-ucils-1.3.1.jar spring-
batik-anim-1.7.jar commons-pool-1.6.9.jar gs-web-sec-ldap-2.6.1.jar gs-rendering-12.1.jar h2-1.1.119.jar jst-vectorize-1.3.1.jar spring-
batik-awt-util-1.7.jar com.noelios.restlet.ext.servlet-1.0.8.jar gs-web-wfs-2.6.1.jar gt-shapefile-12.1.jar hsqldb-2.2.8.jar jst-zonalstats-1.3.1.jar spring-
batik-bridge-1.7.jar com.noelios.restlet.ext.simple-1.0.8.jar gs-wfs-2.6.1.jar gt-svg-12.1.jar htmlvalidator-1.2.jar loq4j-1.2.14.jar spring-
batik-dom-1.7.jar ecore-2.6.1.jar gs-wfs-2.6.1.jar gt-transform-12.1.jar imageio-ext-arcogrid-1.1.10.jar mail-1.4.jar spring-
batik-ext-1.7.jar shochu-1.6.2.jar gs-wms-2.6.1.jar gt-wfs-mpg-12.1.jar imageio-ext-geocore-1.1.10.jar net.opengis.fes-12.1.jar spring-
batik-gvt-1.7.jar encoder-1.1.jar gt-api-12.1.jar gt-wms-12.1.jar imageio-ext-png-1.1.10.jar net.opengis.ows-12.1.jar spring-
batik-js-1.7.jar esmorph-1.0.6.jar gt-arcogrid-12.1.jar gt-xml-12.1.jar imageio-ext-streams-1.1.10.jar net.opengis.wcs-12.1.jar spring-
batik-parser-1.7.jar freemarker-2.3.18.jar gt-coverage-12.1.jar gt-xsd-core-12.1.jar imageio-ext-tiff-1.1.10.jar net.opengis.wfs-12.1.jar spring-
batik-script-1.7.jar gs-gwc-2.6.1.jar gt-ogc-12.1.jar gt-xsd-fes-12.1.jar imageio-ext-utilities-1.1.10.jar org.json-2.0.jar spring-
batik-svg-dom-1.7.jar gs-kml-2.6.1.jar gt-data-12.1.jar gt-xsd-filter-12.1.jar itext-2.1.5.jar org.restlet-1.0.8.jar stax-1
batik-svggen-1.7.jar gs-main-2.6.1.jar gt-spsq-hsql-12.1.jar gt-xsd-gml3-12.1.jar jai_codec-1.1.3.jar org.restlet.ext.freemarker-1.0.8.jar stak-af
batik-transcoder-1.7.jar gs-ows-2.6.1.jar gt-geotiff-12.1.jar gt-xsd-gml3-12.1.jar jai_core-1.1.3.jar org.restlet.ext.json-1.0.8.jar vecmath
batik-util-1.7.jar gs-platform-2.6.1.jar gt-geotiff-12.1.jar gt-xsd-ows-12.1.jar jai_imageio-1.1.jar org.restlet.ext.spring-1.0.8.jar wicket-
batik-xml-1.7.jar gs-rest-2.6.1.jar gt-graph-12.1.jar gt-xsd-sld-12.1.jar jasypt-1.8.jar org.simplerframework-3.1.3.jar wicket-
beprow-jdk14-1.46.jar gs-restconf-2.6.1.jar gt-grid-12.1.jar gt-xsd-wcs-12.1.jar javaxiffxml-2.2.0.jar org.w3.xml-12.1.jar wicket-
cqlib-nodet-2.2.jar gs-sec-jdbc-2.6.1.jar gt-rcopso-12.1.jar gt-xsd-wfs-12.1.jar jdom-1.1.3.jar piccontainer-1.2.jar wicket-
common-2.6.0.jar gs-sec-ldap-2.6.1.jar gt-image-12.1.jar gt-xsd-ows-12.1.jar quava-17.0.jar jettison-1.0.1.jar rcs-3.0.4.jar xml-api
commons-beanutils-1.7.0.jar gs-wcs1_0-2.6.1.jar gt-imagemoaic-12.1.jar gwc-core-1.6.0.jar jgridshift-1.0.jar spring-3.0.4.RELEASE.jar xml-api
commons-codec-1.9.jar gs-wcs1_1-2.6.1.jar gt-jdbc-12.1.jar gwc-core-1.6.0.jar jjson-lib-2.2.3-jdk15.jar s444-net-2.2.0.jar xml-cdm
commons-collections-3.1.jar gs-wcs2_0-2.6.1.jar gt-jdbc-postgis-12.1.jar gwc-diskquota-core-1.6.0.jar json-simple-1.1.jar sl4j-log4j2-1.4.2.jar xml-cdm
commons-collections-3.2.jar gs-wcs2_1.jar gt-main-12.1.jar gwc-diskquota-jdbc-1.6.0.jar jst-2.1.0-beta-2.jar spring-3.1.4.RELEASE.jar xpp3-1
commons-fileupload-1.2.1.jar gs-web-core-2.6.1.jar gt-metadata-12.1.jar gwc-gmapss-1.6.0.jar jst-305-2.0.3.jar spring-ams-3.1.4.RELEASE.jar xpp3_2
commons-httpclient-3.1.jar gs-web-demo-2.6.1.jar gt-opengis-12.1.jar gwc-kml-1.6.0.jar jst-attributeop-1.3.1.jar spring-beans-3.1.4.RELEASE.jar xsd-2.6
commons-io-2.1.jar gs-web-gwc-2.6.1.jar gt-opengis-12.1.jar gwc-rest-1.6.0.jar jst-contour-1.3.1.jar spring-context-3.1.4.RELEASE.jar xstream
commons-jxpath-1.3.jar gs-web-rest-2.6.1.jar gt-process-12.1.jar gwc-ua-1.6.0.jar jst-rangeLookup-1.3.1.jar spring-context-support-3.1.4.RELEASE.jar
```

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

### 3. Mise en forme des données

Il est conseillé de tuer 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.globe.org/gdal/gdal\\_retile.html](http://gdal.globe.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 Tiff -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](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>
```

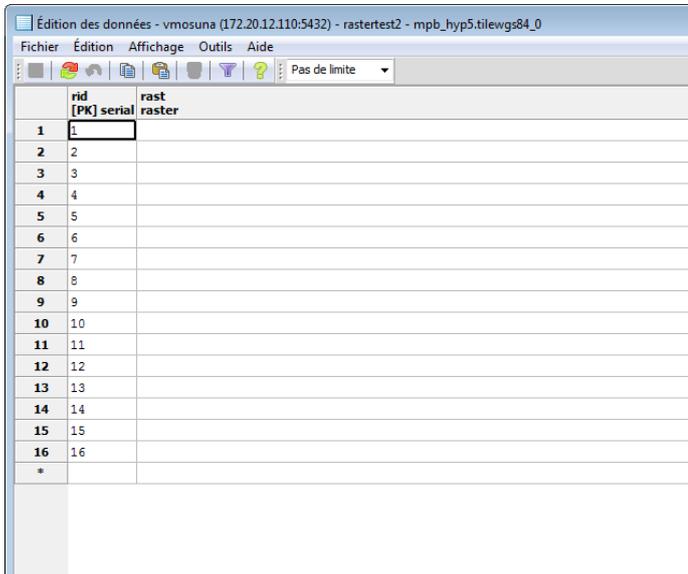
-I (option) : create overview of the raster. For more than one factor, separate with comma(.). Overview table name follows the pattern o\_ overview\_factor\_table, where overview\_factor is a placeholder for numerical overview factor and table is replaced with the base table name. Created overview is 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

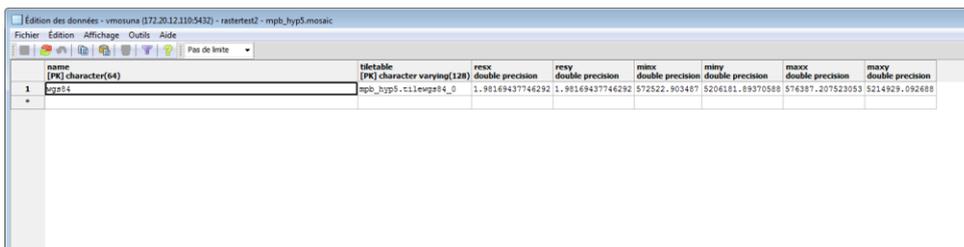


	rid [PK] serial	rast raster
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	
11	11	
12	12	
13	13	
14	14	
15	15	
16	16	
*		

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));
```



	name [PK] character(64)	tiletable [PK] character varying(128)	resx double precision	resy double precision	minx double precision	miny double precision	maxx double precision	maxy double precision
1	hyp54	mpb_hyp5_tilewgs84_0	1.98169437746292	1.98169437746292	572522.903487	5206181.89370588	574387.207523053	5214929.092888
*								

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"/>
    <tileTableNameAttribute name="tiletable" />
    <resXAttribute name="resx"/>
    <resYAttribute name="resy"/>
    <minXAttribute name="minx"/>
    <minYAttribute name="miny"/>
    <maxXAttribute name="maxx"/>
    <maxYAttribute name="maxy"/>
  </masterTable>
  <tileTable>
    <blobAttributeName name="rast" />
    <keyAttributeName name="rid" />
  </tileTable>
</mapping>
```

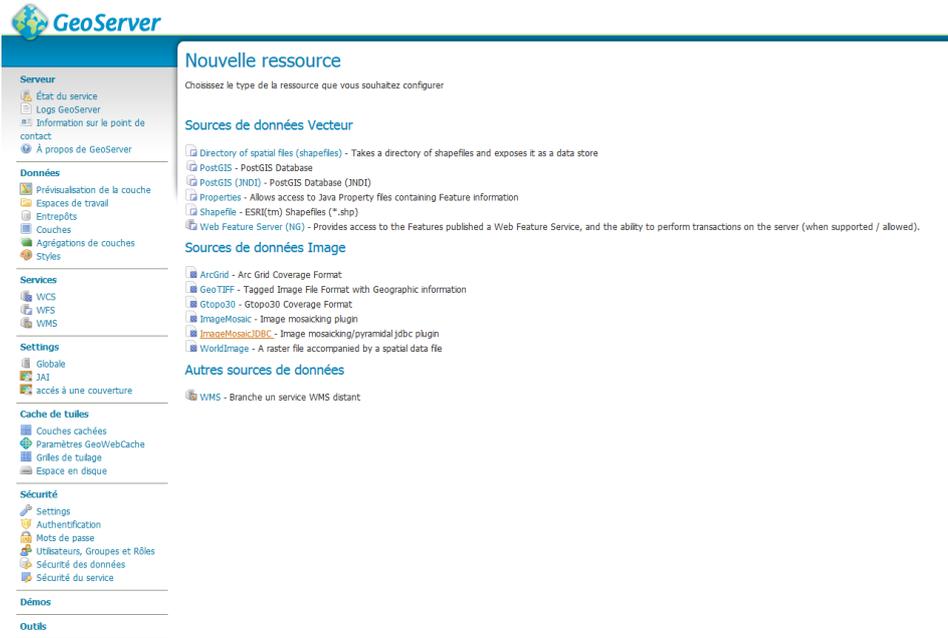
Le fichier wgs84.postgis.xml qui fait le lien avec les deux autres fichiers

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE ImageMosaicJDBConfig [
  <!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 -->
```

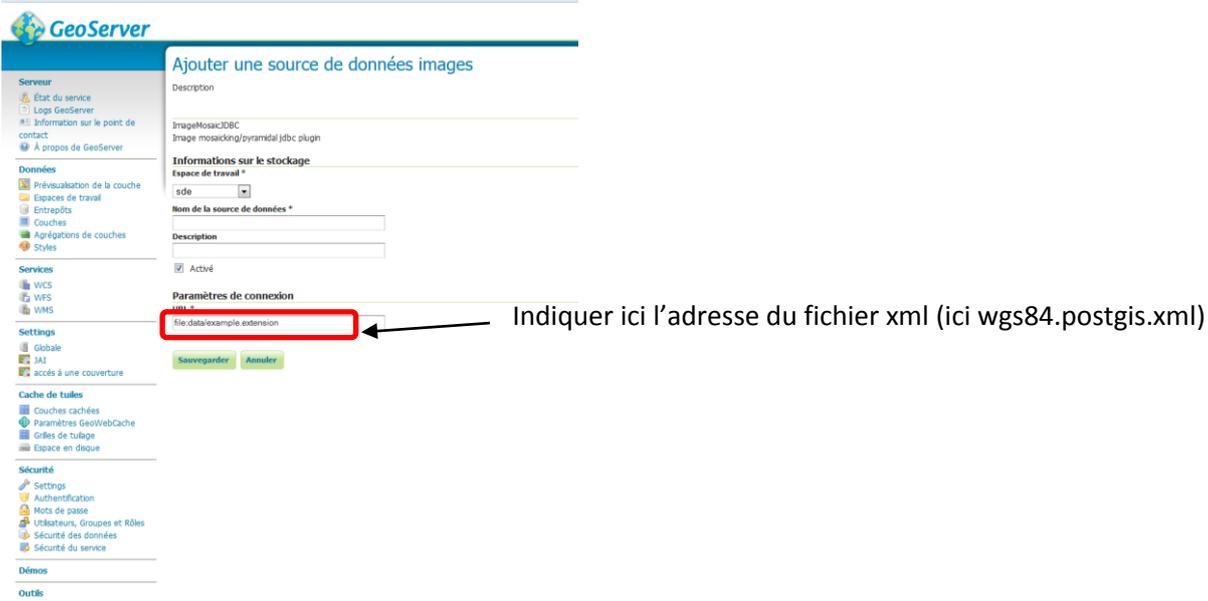
```
<scaleop interpolation="1"/>
<axisOrder ignore="false"/>
  &mapping;
  &connect;
</config>
```

## 5. Connexion à la base de données postgis raster

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



The screenshot shows the 'Nouvelle ressource' (New Resource) page in GeoServer. The left sidebar contains navigation menus for 'Serveur', 'Données', 'Services', 'Settings', 'Cache de tuiles', 'Sécurité', 'Demos', and 'Outils'. The main content area is titled 'Nouvelle ressource' and includes a description: 'Choisissez le type de la ressource que vous souhaitez configurer'. Below this, there are two main sections: 'Sources de données Vecteur' (Vector Data Sources) and 'Sources de données Image' (Image Data Sources). The 'ImageMosaicJDBC' option is selected in the 'Sources de données Image' section. Other options include 'ArcGrid', 'GeoTIFF', 'Gtopo30', 'ImageMosaic', and 'WorldImage'. There is also an 'Autres sources de données' (Other Data Sources) section with a 'WMS' option.



The screenshot shows the 'Ajouter une source de données images' (Add Image Data Source) page in GeoServer. The left sidebar is the same as in the previous screenshot. The main content area is titled 'Ajouter une source de données images' and includes a description: 'Description'. Below this, there are several sections: 'Informations sur le stockage' (Storage Information), 'Paramètres de connexion' (Connection Parameters), and 'Sauvegarder' (Save) and 'Annuler' (Cancel) buttons. The 'Paramètres de connexion' section has a red box around the 'file.data\exemple.extension' field, with an arrow pointing to it and the text 'Indiquer ici l'adresse du fichier xml (ici wgs84.postgis.xml)'. The 'file.data\exemple.extension' field is highlighted in red.

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 (à 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  $\leq 100 \times 100$  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  $\leq 100 \times 100$  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 postgresql dans QGIS ni dans geoserver, il est alors préférable de consulter directement les raster dans les fichiers.