GOALS

Information on how Giswater handles rasters, as well as how they can be loaded on both a LINUX server and a Windows server.

DESCRIPTION

1 - SYSTEM TABLES, VARIABLES AND FUNCTIONS

The two tables that act on the system are:
ext_raster_dem
ext_cat_raster

They are already created but the ext_raster_dem comes without *constraints*. In the case of existing **utils** schema, they are stored there, but have the different names:
raster_dem
cat_raster

The raster_dem table is filled from the outside according to the process explained in step 2.
In case of being in a utils corporate schema, the catalog table is filled automatically with an *AFTER INSERT* trigger in the raster taking:
file name (rastercat_id)
cur_user
tstamp

On the other hand, in order to use this functionality, there are two variables:

SYSTEM: admin_raster_dem (must be *TRUE*)
USER: edit_insert_elevation_from_dem (must be *TRUE*)
edit_update_elevation_from_dem (must be *TRUE*)

The *node / connec triggers functions* in both *Insert* and *Update* capture automatic dimension value. On the other hand, the *toolbox's* **gw_fct_update_elevation_from_dem** function automatically triggers the capture of all dimensions for the selected layer.

2 - LOAD RASTER IN DB

The clear concepts to take into account are:

File name:
It is recommended that the name should include as much information as possible about the raster since it will give information in the meta table of ext_cat_raster about the type that it is:
dg_dem_2019_u48 (provider, raster type, data year, map sheet)
In this way, when the raster is inserted, the raster catalog is also filled and it carries detailed information about it.

File typology:
If all DEM rasters are inserted into the same table, they must all be the same in terms of format so that the table column *constraints* does not break.

In this sense, when loading the first raster, the *constraints* must be created as defined in the point two of this document.

Raster stored inside or outside of ddbb

Since there are two O/S environments for the machine where PostgreSQL is hosted, this process is detailed for each of the two environments.

Very interesting option for not loading the database and *reload* files automatically (you just have to change the file)

WINDOWS ENVIRONMENT

1. Check that there is a raster2pgsql executable in the PostgreSQL *bin* folder.

2. Open command prompt (cmd), go to the PostgreSQL *bin* folder (cd C:\Program Files\PostgreSQL\ 11\ bin\).

3. Execute the process as in the example, putting the SRID, the path to the file, the size of the *tile*, the name of the table to which the raster is imported and the connection to the database:

```
raster2pgsql.exe -R -s 25831 -C -x raster.txt -t 1500x1500 -a utils.raster_dem -F -n rastercat_id  | psql -d giswater -U postgres -p 5432
```

LINUX ENVIRONMENT

It will depend on the distribution, but as a general rule it should be such that:

Since PostgreSQL normally installs on *path*, the command line can be totally straightforward:

```
raster2pgsql -s 25831 -C -x raster.txt -t 1500x1500 -a utils.raster_dem -F -n rastercat_id  | psql -d giswater -U postgres -p 5432
```

If for whatever reason the environment variables are disabled, they should be enabled:

OPTION A: *environment* file (with a *service postgresql reload*)

```
POSTGIS_ENABLE_OUTDB_RASTERS=1
POSTGIS_GDAL_ENABLED_DRIVERS=ENABLE_ALL
```

OPTION B: through console (much easier)

```
SET postgis.enable_outdb_rasters TO True;
SET postgis.enabled_drivers TO enable_all;
```

USER WARNING:
If it is done with a PostgreSQL user, this must have read permissions for the file.
If it is done with another user (*root* type) this must be registered in pg_hba.conf and in SGDB.

SENTENCE NOTATIONS:

```
[-R] -s 25831 -C -x raster.txt -t 1500x1500 -a utils.raster_dem -F -n rastercat_id  | psql -d giswater -U postgres -p 5432
```

[-R] (OPTIONAL) raster is stored outside the database. Failing that, it is stored inside.

WARNING: Problem is that it is not easy to work with. May be system user and postgres user must be the same and with permissions to read/write on files

Optional and essential depending on the chosen storage strategy.

NOTE: FOR THIS OPTION YOU MUST HAVE THE VARIABLES:

POSTGIS_ENABLE_OUTDB_RASTERS
POSTGIS_GDAL_ENABLED_DRIVERS=ENABLE_ALL

[-s 25831] SRID, **mandatory**

[-C] *add constraints*

Required only on loading the first raster. The *constraints* are:
raster height (number of rows in table), enforce_height_rast
raster width (number of columns in the raster): enforce_width_rast
value for no data: enforce_nodata_values_rast
number of bands (for dem 1) enforce_num_bands_rast
pixel type (1bit, 2bit, 4bit….) enforce_pixel_types_rast
set the scale for x: enforce_scalex_rast
set the scale for y: enforce_scaley_rast
set SRID: enforce_srid_rast
out_db (info maintenance outside database)
extension (enforce_max_extent_rast)

[-x] excludes the *constraint* from the spatial dimension. **Mandatory to use if the purpose is to put more than one raster in the same table (which will be usual)**
extension (enforce_max_extent_rast)

[raster.txt]
File name
Without spacing, but with metadata.

[-t 1500x1500]
Ddbb cell size.
Has limits - 5000x5000 - *Error out of memory. Failed on request of size*. Recommended size should not exceed 2000x2000 per row.

A new table will be created in the database (*updates* are not allowed) with the defined structure. The process divides the raster into parts (based on the defined size), each *row* in the table is a part of the raster.

The key point is that the size of the input 1500x1500 is a divisor of the raster size. Ideal divisor 1 to 1 but if the raster exceeds 2000x2000 it should be divided (always with dividers...). Examples:

for raster of 1000x1000 → t 1001x1001 (will be in 1 rows)
for raster of 1000x1000 → t 1000x1000 (will be in 4 rows)
for raster of 2000x2000 → t 2001x2001 (will be in 1 rows)
for raster of 2200x2200 → t 1100x1100 (will be in 4 rows)
for raster of 5555x5555 → t 1111x1111 (will be in 16 rows)

[-a utils.raster_dem] adds raster to the table, **mandatory** since otherwise it would create a new one with the conflict that this means.

[-F] adds file name, **mandatory** and important to know file name.

[-n rastercat_id] for the name of the column where to insert the file name. **Mandatory**

[| -d giswater -U postgres -p 5432]
Connection parameters: if done with a PostgreSQL user, it is direct. If it is done with another user, it will ask for the password which can also obviously be done...

FOR MASSIVE LOADING PROCESSES

NOTES

- Watch out for the constraints
  HINT: LOAD THE CONSTRAINT THEME FIRST AND THEN THE OTHERS

O/S

WINDOWS

```
for /r %%i in (*) do "%PG_PATH%/raster2pgsql.exe" -R -s 25830 -x %%i -t 1500x1500 -a utils.raster_dem -F -n rastercat_id | "%PG_PATH%/psql" -h 192.168.99.124 -d gis -U bgeoadmin
```

(Attach a bat file)

LINUX

Hint:
The user ($userSystem) must be the same as the one that is connected to the machine, otherwise the password issue is necessary and it becomes complicated (peer method, modify pg_hba...)

Watch out with the path (files & raster2pgsql)

Sentence
TEST:

```
for f in *.txt; do echo “Test $f”; done
```

EXECUTION:

```
for f in *.txt; do /raster2pgsql -R -s 25831 -C -x $f -t 1500x1500 -a utils.raster_dem -F -n rastercat_id  | psql -d giswater -U $userSystem -p 5432
```

REFERENCES

Documentation: https://postgis.net/docs/using_raster_dataman.html

REVIEWS

Action	User	Date
Created	Barbara Rzepka	21/11/2019
Modified	Xavier Torret	16/04/2020
Modified	Xavier Torret	29/07/2020
Modified	Xavier Torret	12/03/2021
Modified	Xavier Torret	28/06/2021