SQL/DUMP

SQL

Module: `trafiPolluImp_SQL[.py]`

Libraies (python) utilisées

librairie principale python utilisée: `psycopg2`

Site vers la librairie python (dépot python lib): [Site vers la librairie python (dépot python lib)]

Description:

```
Psycopg is the most popular PostgreSQL database adapter for the Python programming language.
Its main features are the complete implementation of the Python DB API 2.0 specification and the thread safety (several threads can share the same connection).
It was designed for heavily multi-threaded applications that create and destroy lots of cursors and make a large number of concurrent “INSERT”s or “UPDATE”s.
```

Init

Dans l'init, on règle le dictinnaire `self._dict_sql_methods`:

```
self._dict_sql_methods = {
        'update_def_zone_test': self._update_tables_from_qgis,
        'update_table_edges_from_qgis': self._update_tables_from_qgis,
        'update_tables_from_def_zone_test': self._update_tables_from_qgis,
        #
        'update_table_detecting_roundabouts_from_qgis': self._update_tables_from_qgis,
        #
        'dump_informations_from_edges': self._request_for_edges,
        'dump_sides_from_edges': self._request_for_lanes,
        'dump_informations_from_nodes': self._request_for_nodes,
        'dump_informations_from_lane_interconnexion': self._request_for_interconnexions,
        # TODO: travail sur les rond points [desactiver]
        # 'dump_roundabouts': self._request_for_roundabouts
    }
```

On code (en dur) les liens entre les scripts .sql et les méthodes/functions python pour les gérer/utiliser.

On a deux types de scripts SQL:

- [WRITE] ceux qui mettent à jour des tables/vues de la base (comme: `update_def_zone_test`)
- [READ] ceux qui sont utilisés pour le DUMP d'informations ((DB) PostGIS/SG -> Python (RAM))

Config INI

Ce module utilise le fichier: `config_interactive_map_tracking.trafipolluImp_SQL.ini` pour obtenir des valeurs de configs par rapport aux paramètres de connection au serveur SQL qu'on souhaite manipuler (local IGN, ou local VM, ...).

Contenu (par exemple) du fichier INI:

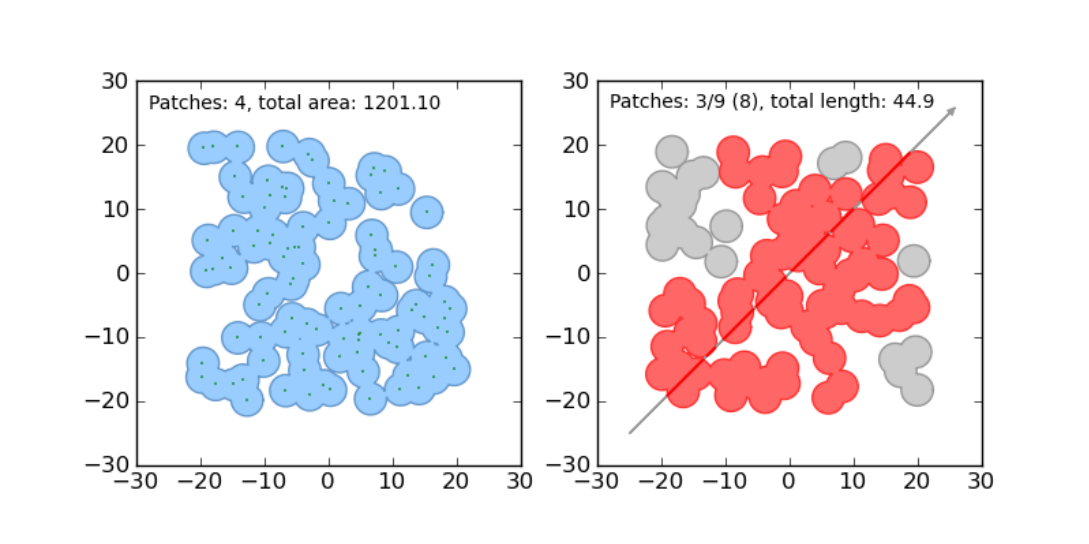
```
[SQL_LOCAL_SERVER]
host: localhost
port: 5433
user: postgres
password: postgres
connect_timeout: 2
#
dbname: street_gen_3
database: bdtopo_topological

[SQL_IGN_SERVER]
host: 172.16.3.50
port: 5432
user: streetgen
password: streetgen
connect_timeout: 2
#
dbname: street_gen_4
database: bdtopo_topological
```

DUMP

Module: `trafipolluImp_DUMP[.py]`

Libraies (python) utilisées

librairies (principales) utilisées pour ce module:



- Shapely: Manipulation and analysis of geometric objects in the Cartesian plane.
  - site
- NumPy: array processing for numbers, strings, records, and objects.
  - site

Interfaces principales du module

Interfaces pour le modules 'parent' SQL: `tpi_DUMP.dump_<entity>`:

- edges
- interconnexions
- lanes
- nodes
- (roundabouts)

But: Transférer et convertir les données StreetGEN/PostGIS vers un format exploitables:

- wkb: Shapely objects
- np_: NumPy arrays

Pour écrire ces sous DUMP modules (edges, interconnexions, ...), il faut regarder en même temps les scripts SQL correspondant.

Exemple: `dump_for_edges(...)` -> `dump_informations_from_edges.sql`

Les types de "base" (int, string, float, ...) sont automatiquement gérés par Python (par conversion implicite).

Il faut convertir les données géométriques 'binaires' (wkb) vers un format (zone mémoire) exploitable.

Processus en 2 étapes:

1. Load du buffer mémoire via Shapely: `load_geom_buffers_with_shapely(...)`
2. Transfert vers des tableaux NumPy: `load_arrays_with_numpely(...)`