UNIVERSITE DE BRETAGNE OCCIDENTALE

Rapport de projet

Développement d'une interface d'intégration de données géo référencées sur une plateforme web d'analyse de la vulnérabilité systémique aux risques côtiers

Réalisé par :
Imene HIMEUR
Khireddine REZZIK

Clients :
Iwan LEBERRE
Mathias ROUEN
Olivier MARCEL

2017/2018

SOMMAIRE

I. Introduction :

OSIRISC consiste en une application permettant d'archiver les données spatio-temporelles d'indicateurs décrivant des composantes (aléa, enjeux, gestion et représentation), de les exploiter pour calculer des indices décrivant la vulnérabilité des territoires, puis de les représenter graphiquement. Adressé autant aux chercheurs qu'aux gestionnaires, le pilote de cette application en cours de développement doit servir d'interface d'analyse multicritère et d'aide à la gestion de la vulnérabilité.

Les données d'entrée se présentent selon différents niveaux de granularité et d'implantation spatiale. Le but de ce projet est de les transformées en donnée maillée pour ensuite les intégrées facilement dans une base de données puisqu'elle sera restituée sous forme tabulaire (fichier CSV).

Avant cette intégration il faudra une phase de validation où l'utilisateur pourra visionner la donnée maillée sous format image et valider par la suite le résultat qui sera intégré dans la base de données.

Au cours de ce projet plusieurs approches ont été explorées, la première était d'utiliser un ETL le choix de ce dernier n'étant pas ci évident une étude comparative a été réalisée. La deuxième approche était d'utiliser un script PyQgis.

Première Approche (ETL) :

II. Etude des ETL

1. ETL ? C'est quoi ?

Un ETL permet **l'Extraction**, la **Transformation** et le **Chargement** de données depuis des sources diverses (bases de données, fichiers) vers des cibles préalablement définies.

Dans notre cas les sources d'alimentation de la base de données peuvent être variées : des fichiers sémantiques (CSV, Excel, XML) et spatiales (SHP et MIF). Les ETL s'occupent de transformer ces sources, via de nombreux composants, en une ou plusieurs cibles qui peuvent être là aussi de n'importe quel format (notre cas c'est une base de données PostgreSQL).

2. Choix de l'outil ETL :

Il existe des douzaines de projets Open source qui proposent des fonctionnalités de type ETL (Apatar, Jdox, GeoKettle, Jaspersoft ETL, Talend Open Studio, Pentaho, CloverETL…). Mais seulement un nombre restreint de projets est capable de fournir l'ensemble des fonctions essentielles et indispensables d'un ETL. Le choix de l'ETL à utiliser est un petit peu fastidieux, vu la diversité des ETL.

Vu le contexte du projet nous nous somme focaliser que sur les outils open source et qui prennent en entrée des données cartographiques.

a. Talend ETL : [1]

Talend Open Studio est développé par Talend, une société française dynamique et relativement jeune. La première version de « Talend Open Studio » a vu le jour au 2ème semestre 2006.

C'est un ETL du type « générateur de code ». Pour chaque traitement d'intégration de données, un code spécifique est généré, ce dernier pouvant être en Java ou en Perl. Les données traitées et les traitements effectués sont donc liés.

Il utilise une interface graphique, le « Job Designer » (basée sur **Eclipse RCP**) qui permet la création des processus de manipulation de données.

De nombreux types d'étapes sont disponibles pour se connecter aux principaux SGBD (Oracle, DB2, MS SQL Server, PostgreSQL, MySQL,...) ainsi que pour traiter tous les types de fichiers plats (CSV, Excel, XML), aussi bien en lecture qu'en écriture.

Talend facilite la construction des requêtes dans les bases de données en détectant le schéma et les relations entre tables.

Un référentiel permet de stocker les métadonnées afin de pouvoir les exploiter dans différents jobs.

La totalité du code généré par Talend Open Studio, quelques soit le langage cible, est toujours visible et accessible depuis l'environnement de conception.

On peut bien sûr implémenter des spécificités « métiers » propres aux données traitées, ceci en ajoutant de nouvelles « routines ».

b. Pentaho : [2]

Cet ETL, « Kettle » à l'origine, est le fruit du travail de **Matt Casters**, un consultant BI qui l'a développé à l'origine pour ses propres besoins au début des années 2000. Courant 2006, il rejoint la société Pentaho, et « Kettle » devient « Pentaho Data Integration».

Contrairement à Talend Open Studio, Pentaho Data Integration est un «moteur de transformation » ETL: les données traitées et les traitements à effectuer sont parfaitement séparés. (On parle de « meta-data driven » ETL).il supporte de nombreux types de SGBD (une trentaine) ainsi que tous les types de fichiers plats (Csv, délimité, Excel, XML).

Pentaho Data Integration dispose d'une interface graphique « Spoon » basée sur SWT (Standard Widget Toolkit, bibliothèque graphique Java), depuis laquelle on peut créer deux types de traitements :

- des transformations : celles-ci constituent les traitements de base d'intégration de données avec toutes les étapes nécessaires à l'extraction, la transformation, et le chargement des données.

- des tâches (jobs) : ceux-ci permettent le séquencement de plusieurs transformations avec des fonctionnalités plus orientés « EAI » : gestion des erreurs, envoi de mails de notification, transferts FTP/SFTP, exécution de scripts Shell ou SQL, etc...

c. Clover ETL : [3]

Tout comme Kettle, Clover ETL est un moteur d'ETL en JAVA : les données et les traitements sont séparés et décrits dans des fichiers au format XML. Une interface graphique permet l'élaboration des traitements de données.

La partie motrice « clover.ETL » existe en licence commerciale et open source.

L'interface graphique de conception des traitements « clover.GUI » est utilisable gratuitement et uniquement pour des applications non commerciales.

Tout comme Talend, l'interface s'appuie sur Eclipse RCP.

On retrouve également la possibilité de décrire des métadonnées, la structure d'un fichier CSV par exemple, et de réutiliser cette description dans différents traitements.

CloverETL nécessite Eclipse (version 3.3 recommandée) avec le plugin GEF (Graphical Editing Framework for Eclipse).Son *téléchargement* se fait via le gestionnaire de mise à jour d'Eclipse, seulement après s'être enregistré sur le site.

3. Comparaison :

a. Architecture physique :

Talend et CloverETL utilisent le Framework Eclipse pour leurs éditeurs visuels de transformations ETL et utilisent la JRE pour exécuter ces transformations. La transformation de Talend peut également être exécutée en tant que script Perl.

Talend génère du code, puis le compile et l'exécute. Toute modification du code ne sont pas reflété dans un graphique visuel et doivent être conservés

manuellement. La transformation de Talend peut être exécutée séparément sans éditeur de transformation visuelle en tant que bibliothèque jar.

CloverETL est un outil basé sur les métadonnées, il ne nécessite aucune génération de code pour exécuter des travaux. Les modifications apportées aux transformations en dehors de l'éditeur visuel sont reflétées une fois chargé de nouveau dans le concepteur. Les transformations sont stockées sous forme de fichiers XML faciles à lire et à ajuster. Les transformations de CloverETL peuvent être exécutées en dehors de CloverETL Designer par la bibliothèque open source CloverETL Engine.

PDI est distribué en tant qu'application autonome. Il est écrit en Java et est portable sur d'autres plates-formes.
Tout comme CloverETL, PDI stockent les transformations dans des fichiers XML. Il est possible d'effectuer des transformations en dehors de l'éditeur visuel, en utilisant uniquement des bibliothèques Java (CloverETL Engine pour CloverETL et Les librairies Pentaho Kettle pour Pentaho).

b. Transformation :

Talend a un grand nombre de composants mais beaucoup d'entre eux fournissent les mêmes fonctions sous un nom différent. De plus, certains des composants que nous considérons comme essentiel - par ex. recherches persistantes - sont manquants.

CloverETL apporte une plus petite palette de composants mais leur fonctionnalité est plus complexe, ils ont battu les équivalents de Talend à bien des égards. C'est aussi plus facile de choisir composant approprié dans la palette de CloverETL que dans la palette de Talend.

Même pour un développeur ETL expérimenté, PDI est définitivement plus difficile à apprendre que CloverETL. Les composants ont souvent des noms inattendus et confus interface (par exemple un composant de calculatrice). De nombreux composants nécessitent une entrée triée, ce qui rend les graphiques plus encombrés. De nombreuses opérations ne sont pas intuitives et la documentation est très pauvre.

c. Pouvoir expressif :

Talend utilise Java comme langage de script et permet également l'utilisation de JavaScript et Groovy.

CloverETL lui utilise java mais il a un langage de script ETL spécial - CTL (Clover Transformation Language), facile à apprendre et permettant aux utilisateurs sans compétences en programmation de développer une transformation complexe en peu de temps.

Quant à PDI, lui utilise JavaScript ou Java pour les calculs / formules.

Voici un tableau qui montres les différences globales entres les 3 logiciels :

	Talend Open Studio	**Pentaho**	**CloverETL**
facile d'utilisation	✓	✓	
installation unique	✓		✓
partitionnement des données	✓		✓
Meta-driven		✓	✓
programmateur integé	✓		
connexion BDD Non-relationnelles	✓	✓	✓
client web services	✓	✓	
diversité des fichiers sources	✓	✓	✓
fractionner les flux de données	✓	✓	✓
Transformations complexes	✓	✓	✓
validations de données	✓	✓	✓
contrôle de version	✓		
possibilité de traitement par un langage de programmation	✓	✓	✓
ajout de nouvelles transformations et processus métier	✓	✓	✓
mapping graphique	✓		✓
Drap and drop (glisser/déposer)	✓	✓	✓
Lecture de table complète	✓	✓	✓
Lecture/écriture de types complexes de données	✓ (cartographique)		
Fichiers plats	✓	✓	✓
Open Source	✓	✓	✓

-Tableau récapitulatif des différences entres les outils-

4. Conclusion :

Talend Open Studio, même s'il est plus récent que Pentaho Data Integration, possède une plus grande communauté française. Talend est une société, notamment basée en France.

Le référentiel de métadonnées de Talend Open Studio est très complet, il permet de réutiliser des schémas de fichiers, de connections, de Web services et autres et de gagner ainsi beaucoup de temps. On peut aussi ajouter ou créer de nouvelles spécificités métiers (en Java ou Perl) en ajoutant de nouvelles routines.

De nombreux outils pour corriger des erreurs, de vérifier les statistiques, les logs et de commenter les développements ou d'ajouter de la documentation sont mis à disposition par Talend Open Studio.

Enfin, la cartouche spatiale SDI de Talend renferme une diversité de composantes spatiales (manipulation de données géographiques, gestion de plusieurs formats *.shp ou *.mif) par rapport au PDI.

Le choix de l'ETL, dépend donc de la nature du projet mais aussi, des préférences des développeurs tant ces trois ETL sont différents d'utilisation.

Après étude des trois ETL, Il nous semble alors, que Talend est plus adapté au contexte de notre projet, même si il est très difficile de les départager et qu’ils se complètent d'avantage qu'ils ne sont concurrents.

III. Solution proposée en Open Talend :

1. Prérequis :

Open Talend n’intègre pas directement l’extension spatiale, un ajout manuel est à prévoir. Il suffit de télécharger le zip(1) et d’ajouter son contenu dans le dossier plugins d’Open Talend.

2. Méthodologie:

Nous avons choisi de diviser le travail en deux sous jobs :

- Le job « **Traitement_shapefile** » contenant l'étape d'intersection, le calcul de la superficie, et l'extraction des données
- Le job « **Jointure** » pour l'étape de la jointure.

Nous avons choisi cette méthodologie à cause des traitements qui sont très longs, pour pourvoir simplifier la lisibilité et la compréhension des traitements et aussi pour pouvoir analyser les résultats intermédiaires.

3. Présentation des jobs :

Comme 1er exemple de traitement nous avons choisi de travailler avec le fichier ShapeFile qui représente les zones de submersion exceptionnelles de la commune de Guissény (telechargement).

a. Traitement_ShapeFile :

- ✓ **Préparation des métadonnées** : Cette section décrit comment configurer la métadonnée des fichiers sources (zones de submersion exceptionnelles et le carroyage). Les métadonnées stockées dans le référentiel peuvent être utilisées dans plusieurs Jobs, vous permettant ainsi de configurer rapidement vos Jobs sans avoir à définir chaque paramètre et schéma manuellement.
- ✓ **Déposer et relier les composants qui constituent le job** : Cette section est celle qui décrit notre job par ses différents composants. Pour ce premier job nous avons utilisé des composants spécifiques aux données spatiales, nous avons :
    - **sShapefileInput** : c'est le composant qui contient le fichier en entrée de notre traitement (format shapefile) et qui a un lien avec les métadonnées déjà définies.
    - **tMap** : c'est le composant qui décrit les traitements effectués sur nos fichiers en entrées (l'intersection entre les deux fichiers, le filtre pour récupérer les valeurs de chaque zone et le calcul de la superficie pour chaque zone).

- **sSimpleGeoMulti** : ce composant permet convertir nos données MultiPolygon.
- **sShapefileOutput :** c'est ce qui contiendra nos fichiers shapeFile résultants.

L'image suivante montre un récapitulatif du job et les résultats obtenus.

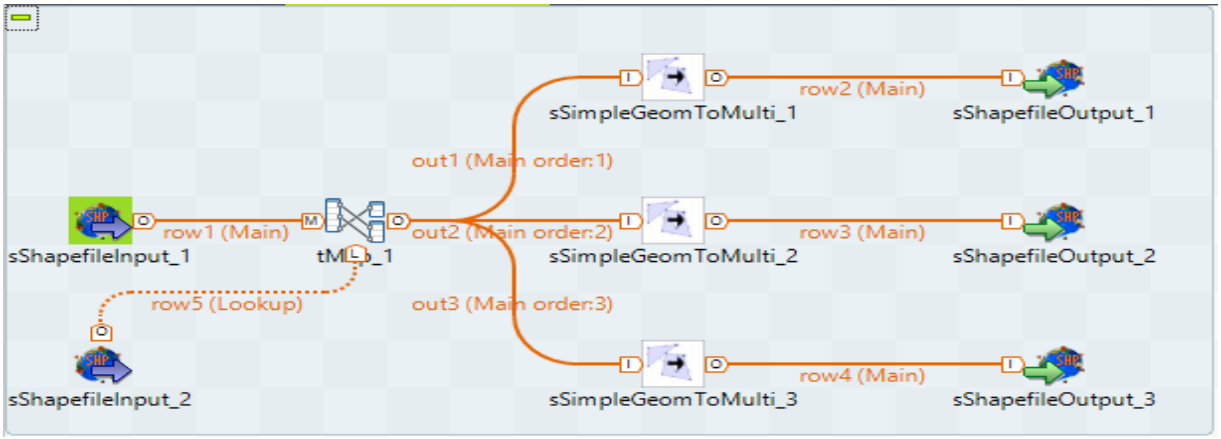


Figure1. Job «Traitement_ShapeFile»

b. <u>Jointure :</u>

Les fichiers de sortie du 1er job sont pris comme entrées en 2eme, la contrainte dans Open Talend est que le composant **tMap** ne prend que deux fichiers en entrées. La jointure se fait par rapport à l'id des carreaux donc en plus des résultats du 1er job nous auront aussi le fichier qui contient le carroyage.

L'image suivante montre la composition de notre job :

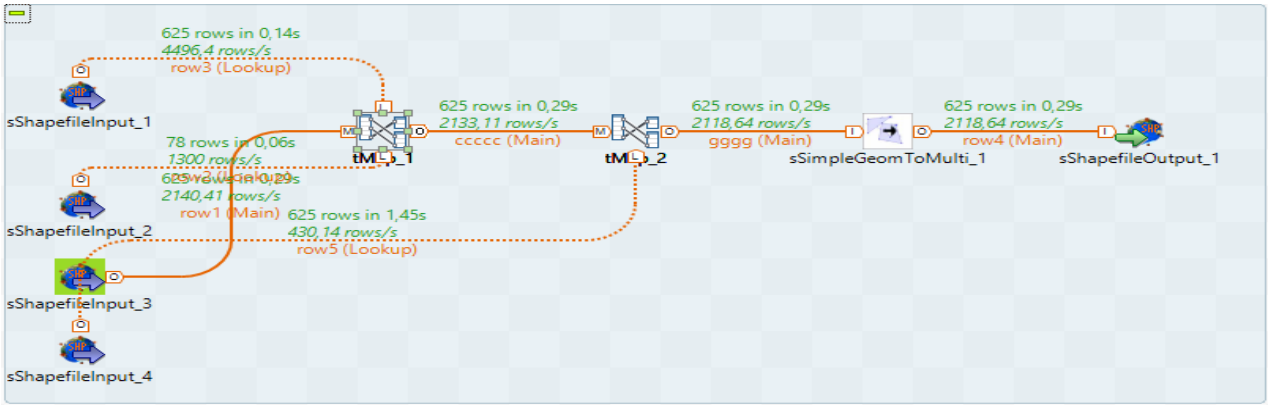


Figure2. Job « Jointure »

4. Résultat :

Le résultat de ce traitement est une donnée maillée où chaque carreau contient la valeur maximale de la superficie. L'image suivante montre un récapitulatif de notre traitement en Talend et le résultat obtenu.

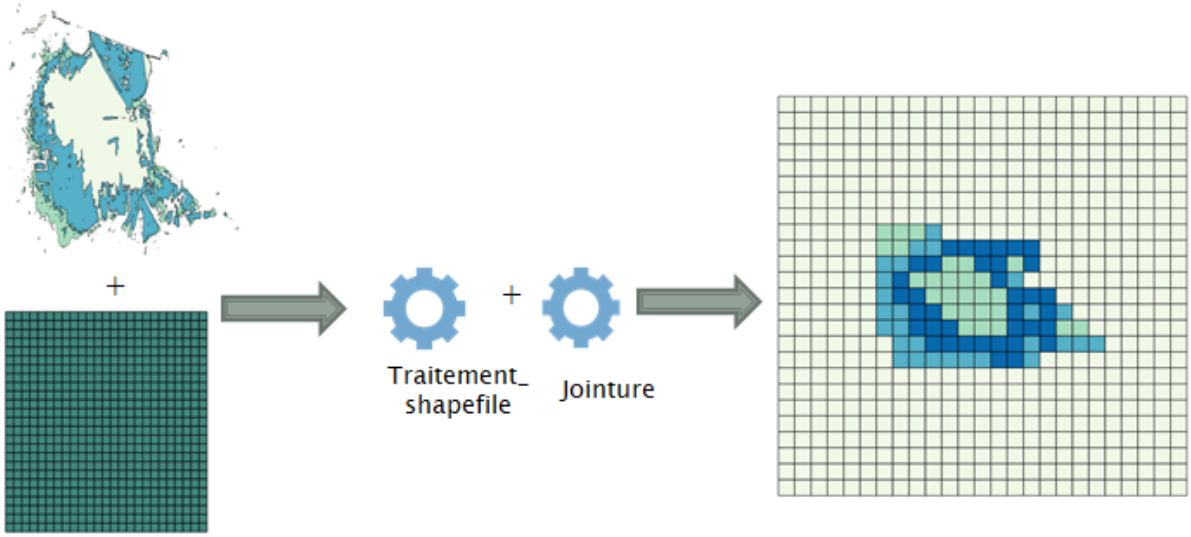


Figure3. Résumé du traitement en Open Talend.

IV. Comparaison entre Talend et PyQgis :

Après avoir exploré la solution en Open Talend, une autre solution nous a été proposée par Mr Olivier Marcel, cette fois c'est un script en PyQgis qui effectue le même traitement pour la même données en entrée (risque de submersion pour la commune de Guisseny).

Nous avons effectué une comparaison des deux solutions afin de choisir celle qui conviendra le mieux pour la suite du projet.

Voici un tableau qui contient les critères de comparaison pris en compte pour faire notre choix:

	PyQgis	Open Talend
Temps d'exécution	Rapide	Lent
Intégration à la base de données	Oui	Oui
Lisibilité du code	Bonne	Bonne
Modification du code	Facile	Difficile
Documentation	Faible	Bonne
Flexibilité	Bonne	Mauvaise
Tests	Facile	Difficile

Tableau2. Comparaison PyQgis et Open Talend

- **Temps d'exécution** : la solution en Open Talend prend beaucoup plus de temps que la solution en PyQgis.
- **Intégration à la base de données** : intégration assez facile pour les deux solutions.
- **Lisibilité du code** : Open Talend est un outil graphique, PyQgis c'est du code.
- **Modification du code** : Les scripts PyQgis sont plus faciles à modifier que les jobs d'Open Talend où une simple modification entrainera des modifications plus importantes.
- **Documentation** : Documentation manquante pour PyQgis sous Windows, c'est pour cela que nous avons choisi de travailler sous Ubuntu.

- **Flexibilité** : les scripts PyQgis sont plus flexibles que les jobs d'Open Talend.
- **Tests** : il est plus facile d'implémenter des tests de validation avec PyQgis qu'Open Talend.

Avec PyQgis on a la possibilité de généraliser le traitement de tous les types de donnée ce qui n'était pas possible avec Open Talend.

Pour la suite du projet nous avons donc choisi de continuer les Traitements en PyQgis.

Deuxième Approche (PyQgis) :

V. Solution proposée en PyQgis : [4]

1. Prérequis :

- ✓ Qgis 2.18 **[5]**
- ✓ Python 2.7 **[6]**
- ✓ Pygame **[7]**
- ✓ PostgresSQL **[8]**
- ✓ Psycopg2 **[9]**

2. Méthodologie :

L'approche en PyQgis a été traitée en plusieurs parties une première partie était de faire les traitements pour les autres types de données (Linéaire et ponctuel), une autre partie était d'intégrer la phase de validation par image et intégrer les résultats dans une base de données PostgreSQL et la dernière partie c'est la partie interaction avec l'utilisateur où le but était de présenter un script contenant un menu en ligne de commande et regroupant les autres parties.

3. Présentation des Traitements:

a. Données linéaire :

Cette fois l'exemple étudier est le réseau d'eau de la comme de Guissény. Pour ce faire un script PyQgis a été développé afin d'automatiser le carroyage pour ce type de données.

Le script est organisé comme suit:

- Lecture de données (lectures des couches) Guissény et le carroyage.
- Reprojection des données : si les données n'ont pas la même projection une reprojection sera faite afin de les rendre en 3035.
- L'exécution de l'algorithme: selon le type de données un algorithme bien spécifique sera exécuté.
- Enregistrer le résultat en format csv et Shapefile.

b. Données Ponctuelles :

Pour le type ponctuel nous avons comme entrée des données issues de la base de données SIRENE.

- ✓ Une fois les données téléchargées, on exécute une requête pour récupérer les données de la commune de Guissény en format csv.
- ✓ il faut géocoder les données pour pouvoir les exploiter
- ✓ ensuite il faut créer un fichier shapefile à partir des données téléchargé et géocoder.

Pour ce qui est du script la même organisation est faite avec les autres types.

4. Phase de validation :

Pour permettre aux utilisateurs de vérifier la validité des traitements une impression des résultats dans une image a été faite.

La solution proposée pour l'impression est une composition de deux solutions (rendu simple et composeur d'impression).

Pour le rendu simple c'est une impression de la couche résultante, le composeur d'impression est un plus qui nous permet d'afficher le titre, la légende et d'autres options en plus à l'image.

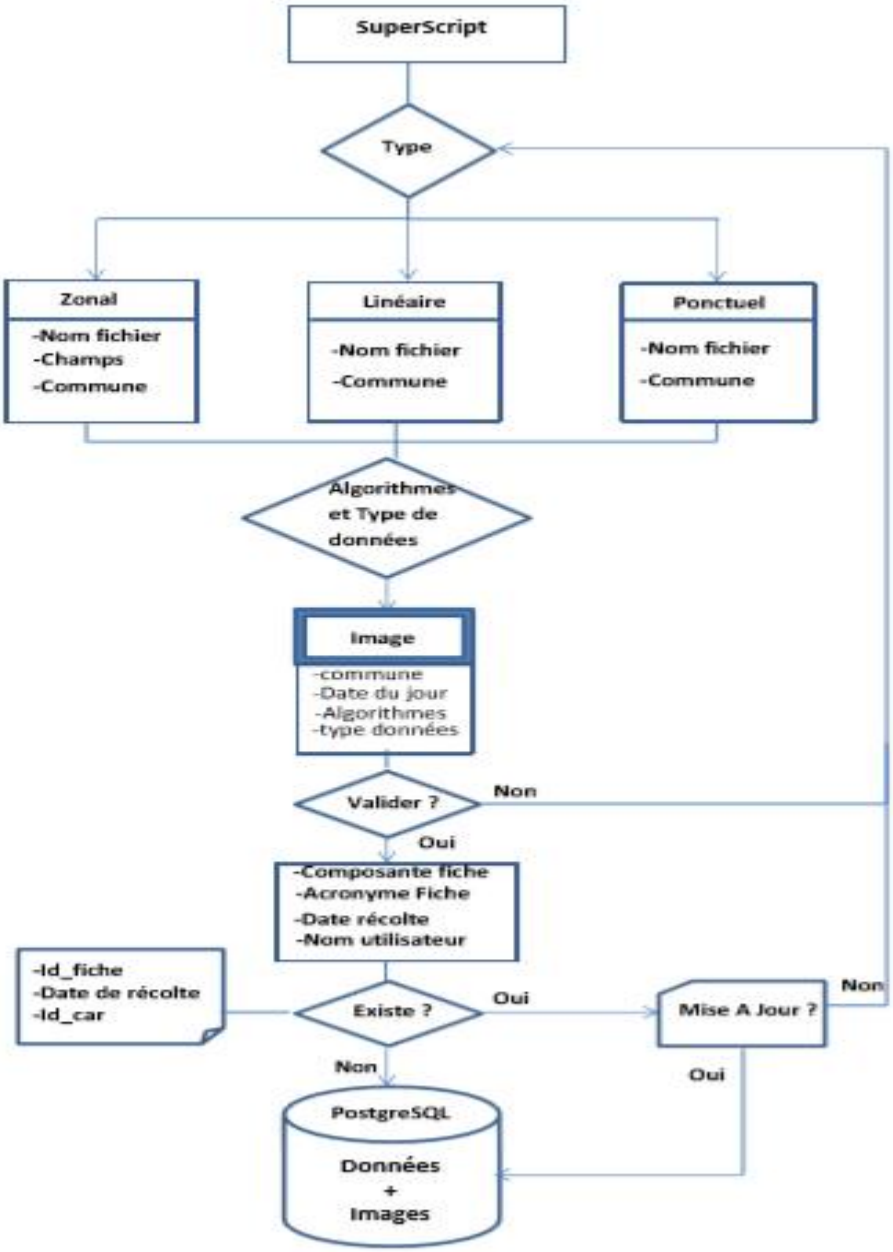
5. Phase d'intégration à la base de données :

Une fois le traitement validé par l'utilisateur une phase d'intégration a été faite. Cette dernière permet d'intégrer les données résultantes en format csv à la base de données PostgreSQL.

6. Menu interactif :

Pour permettre aux utilisateurs d'utiliser facilement ces traitements, nous avons proposé un menu interactif en ligne de commande regroupant tous les traitements effectués.

Nous présentant dans ce qui suit un schéma regroupant représentant l'architecture globale de notre travail.



-Architecture globale du programme-

- **Le super Script :** est le script principal, c'est là qu'est défini le menu interactif.
- **Le script Zonal** : est le script correspondant au traitement du type zonal et ses entrées.
- **Le script Linéaire** : est le script correspondant au traitement du type linéaire et ses entrées.
- **Le script ponctuel** : est le script correspondant au traitement du type ponctuel et ses entrées.
- **Image :** correspond à l'image de sortie après chaque traitement.
- **PostgreSQL :** correspond à la phase d'intégration à la base de données.

Dans ce qui suit nous présenterons un scénario afin de mieux comprendre le déroulement du programme.

1. L'utilisateur exécute le super Script
2. Un choix du type lui sera proposé (linaire, ponctuel et zonal)
3. Il choisit un type et valide
4. Il saisit le nom du fichier à traiter
5. Il saisit la commune à traiter.
6. Il choisit l'algorithme de traitement
7. Il saisit le type de données traitées (boolean, charater, integer…)
8. Une image contenant le résultat du traitement, la légende, le nom de la commune avec la date du jour, l'algorithme et le type de donnée lui sera affiché.
9. L'utilisateur à le choix soit de valider le traitement soit d'annuler et de refaire un autre traitement ou de quitter le programme.
10. S'il choisit de valider des données à saisir lui seront demandées afin de faire une intégration à la base de données.
11. Il choisit la composante de la fiche.
12. Une liste des acronymes lui sera affichée.
13. Il saisit l'année de récolte.
14. Il saisit le nom d'utilisateur.
15. Un traitement s'exécute afin de vérifier l'existence des données, la vérification se fait sur la clé primaire de la table indicateur. (id_fiche, date de récolte et id_car).

16. Si les données existent, le programme demande à l'utilisateur s'il veut les écrasées.
17. S'il valide une confirmation lui sera demandée et les données (csv+ images+ données saisies par l'utilisateur) seront intégrées.
18. Sinon si les données n'existent pas une intégration sera faite.

VI. Perspectives :

Dans les deux approches explorées nous avons comme entrées les données soit linéaires, ponctuelles ou zonales et aussi un fichier de carroyage. Nous avons voulus modifier l'une des entrées qui est le fichier de carroyage et mettre à la place un service WFS disponible sur le site indigeo afin de télécharger la couche de carroyage directement du site.

La solution a bien été exploitée mais quelques ajustements doivent être prévus pour mieux visualiser le résultat du traitement.

L'approche PyQgis a été développée en python 2.7 donc une mise à jour du code doit être faite afin de passé à python 3. Dans ce qui suit un aperçu des éventuels changements à faire dans le code source :

Il existe plusieurs différences entre les versions python 2.7 et python 3 :

- **Print :**

```
Python 2                                Python 3
print "Bonjour"                  →    print("Bonjour")
print "Bonjour", variable1       →    print("Bonjour", variable1)
print "\n".join([x, y])          →    print(x, y, sep="\n")
print >> sys.stderr, "erreur"    →    print("Erreur", file=sys.stderr)
print "une ligne ",              →    print("une ligne", end="")
```

- **Exceptions:**

```
Python 2                            Python 3
raise IOError, "file error"    →    raise IOError("file error")
raise "Erreur 404"             →    raise Exception("Erreur 404!")
raise TypeError, msg, tb       →    raise TypeError.with_traceback(tb)
```

D'autres différences sont à prévoir et les détails sont dans le lien suivant. **[10]**

Conclusion :

Ce projet a été pour nous une réelle préparation pour le monde professionnel puisque il s'est révélé très enrichissant sur plusieurs plans:

- Sur le plan méthodologique avec l'utilisation de la méthode agile concrétisé par des réunions hebdomadaire et une forte communication.
- Développement d'un esprit dynamique qui nous a permis de passer vers de nouvelles technologies durant un projet.