1. Contexte de l'analyse des traitements

– **Classe de préoccupations de toute analyse de S.I.**

» Au même titre que les données ou les flux de données, les traitements constituent une classe de préoccupations à étudier au moins aux 3 niveaux d'abstraction :

- Descriptif.
- Conceptuel.
- Logico-physique.

» Les traitements propres à tout S.I. relèvent de 6 catégories :

- Génération
- Acquisition.
- Communication.
- Maintenance
- Gestion.
- Intégration de l'information.

➔ Voir chapitre 1

2. Analyse des traitements au niveau descriptif

- **L'analyse préalable (existant – besoins) est menée au niveau descriptif**
  - » Outre les tâches ETL, les traitements les plus facilement identifiés (*ex. par enquête*) sont ceux qui relèvent du système opérant de l'organisation.
    - – *Il sont liés à la communication : visualisation, rapport, etc.*
    - – *Ils sont la source de problèmes : pannes, freins, conflits, etc.*
  - » Les traitements rapidement identifiés ne sont pas nécessairement les plus importants pour l'organisation (parfois les plus spectaculaires), c'est la raison pour laquelle tous les traitements doivent être analysés :
    - **Qualitativement** : niveau (opérationnel, tactique, stratégique), nature, données et flux entre services et avec l'extérieur.
    - **Quantitativement** : fréquences, volumes de données, coûts, durée…

3. Analyse des traitements au niveau conceptuel

– L'analyse des traitements au niveau conceptuel ne peut se faire sans l'analyse parallèle des jeux de **données**, de l'**organisation** et des **flux de données**.

– UML propose plusieurs vues statiques et dynamiques adaptées à ces analyses :

Exemples :

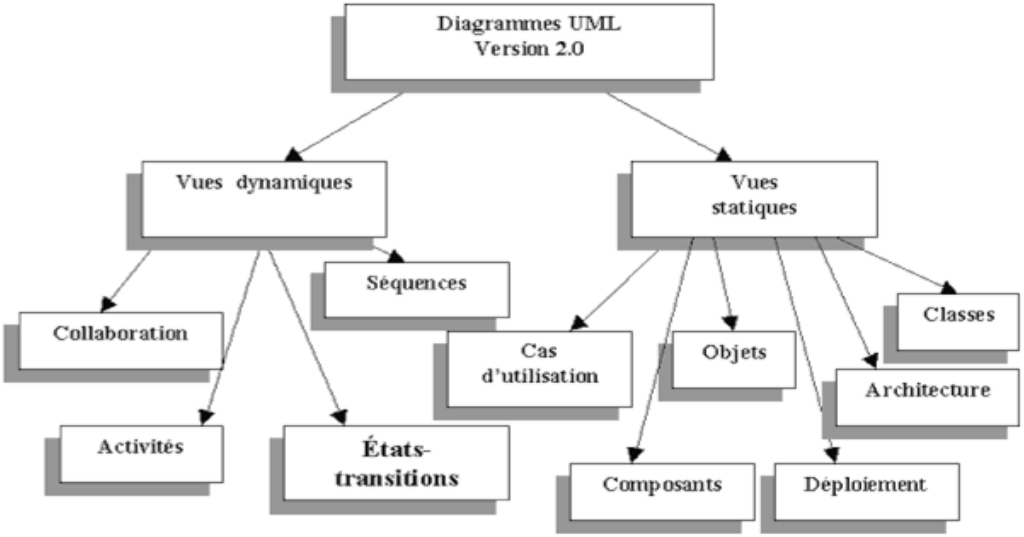
» Déploiement (statique) : l'utilisation de l'**infrastructure physique** par le système et la manière dont les composants du système sont répartis ainsi que leurs relations entre eux.

» Cas d'utilisation (statique) : représente les **fonctions** du système et les **acteurs** de l'organisation (ou de l'extérieur) qui y sont associés.

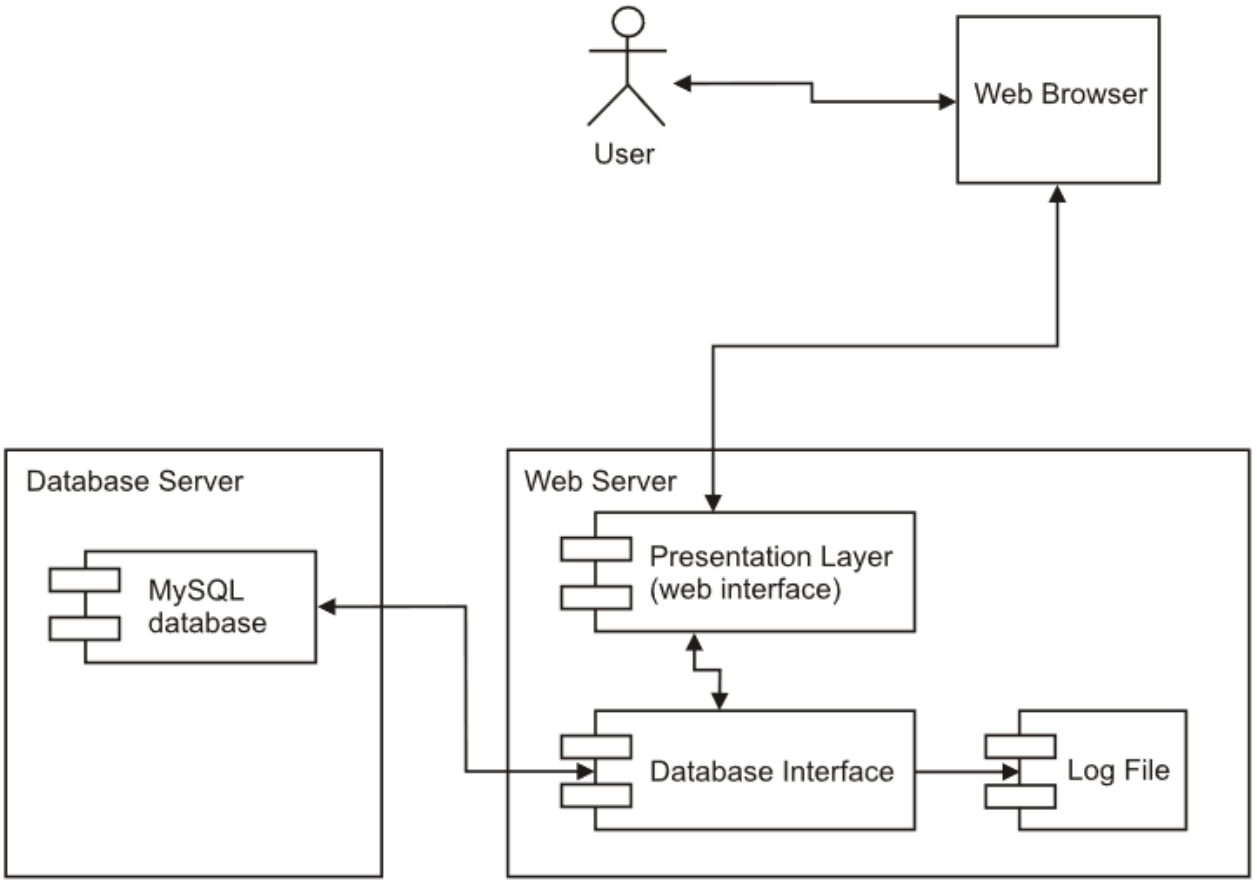
» De séquences (dynamique): représente les objets et leurs interactions **temporelles**.

» D'états-transitions (dynamique): modélise le **comportement** d'une **classe**.

» D'activités (dynamique) : exprime le **comportement** d'une **opération**.



– Diagramme de déploiement (exemple)



– Diagramme des cas d'utilisation

» Rechercher les acteurs qui agissent et identifier leurs **rôles** et leurs besoins.

» Rechercher les fonctionnalités du système par l'utilisation des « **cas d'utilisation** ».

» Rechercher les classes métiers et leurs associations.

Qui fait quoi ?

Acteurs	Rôles
Acteur 1	Rôle 1.1
	Actions *Description*
	Rôle 1.2
	Actions *Description*
Acteur 2	Rôle 2.1
	Actions *Description*
Acteur 3	Rôle 3.1
	Action *Description*
	Rôle 3.2
	Actions *Description*

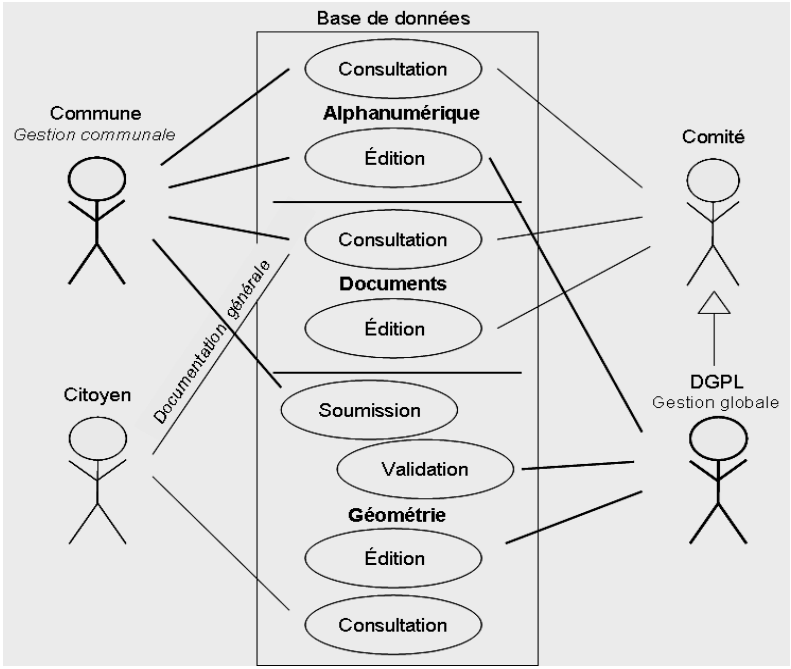


Diagramme des cas d'utilisation
(Projet PICVert de la DGPL)

– Diagramme de séquences

» L'objectif du diagramme de séquences est de montrer les **interactions** entre les objets du système et les acteurs sur une échelle de temps (**topologie temporelle**).

» Il est surtout utilisé pour étudier les problèmes de synchronisation, d'ordonnancement..., soit toutes les interactions qui dépendent du temps.

Quand qui fait quoi ?

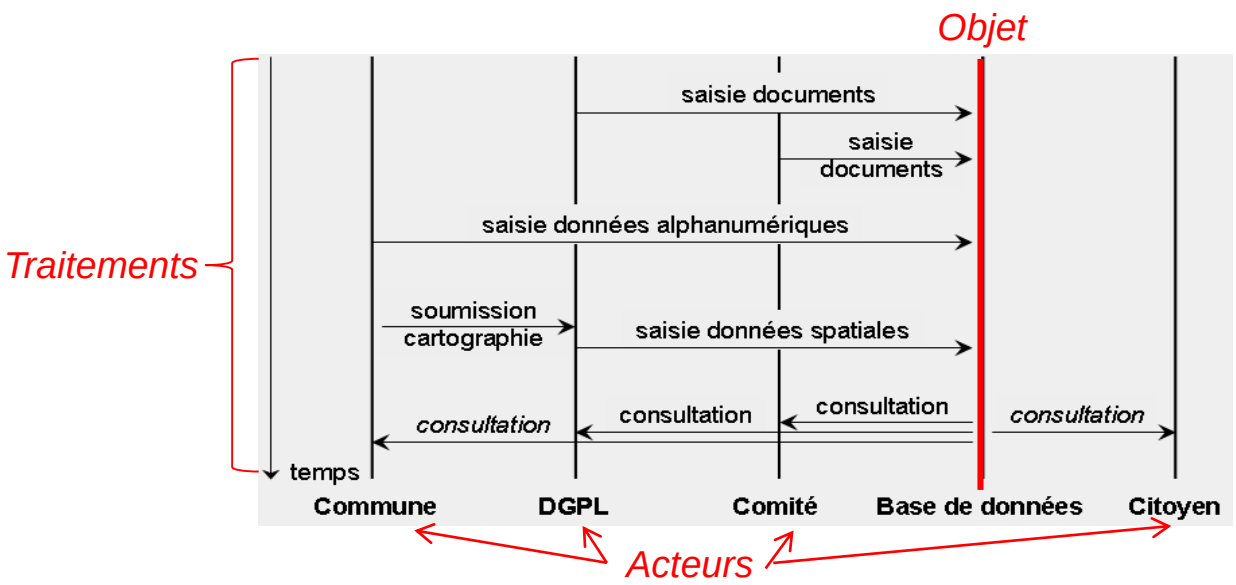
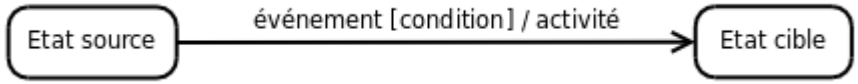


Diagramme de séquences des jeux de données (Projet PICVert de la DGPL)

– **Diagramme d'état-transition**

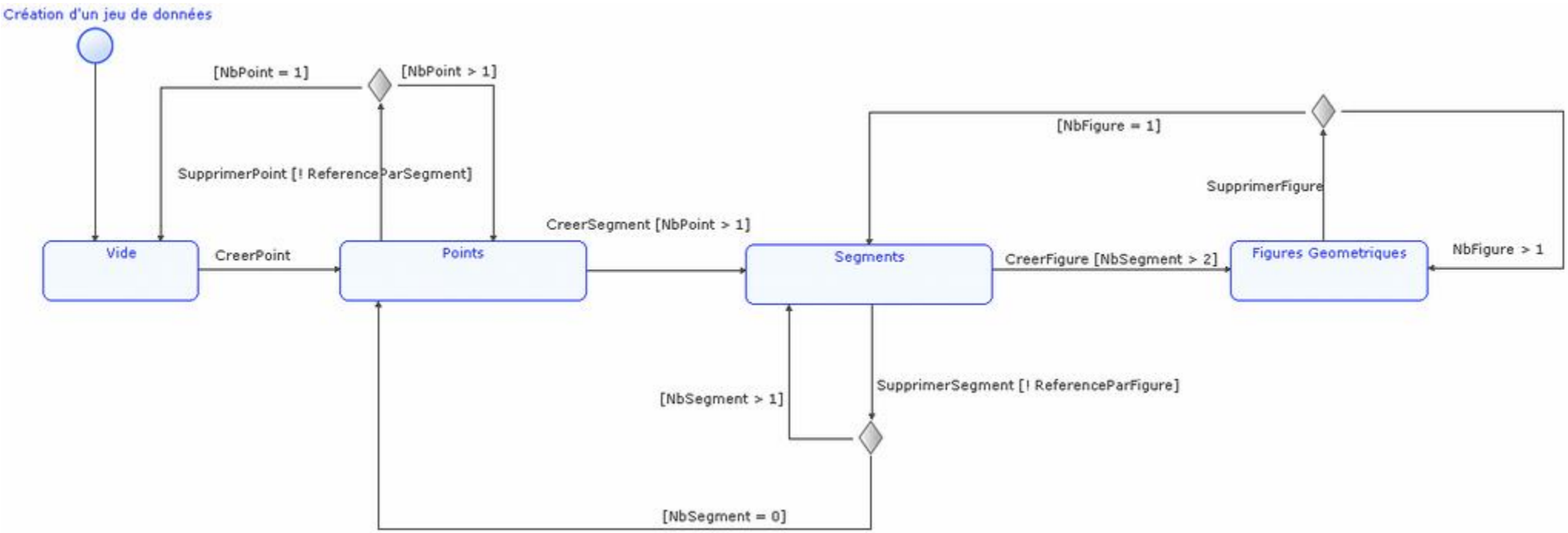
» Le diagramme **d'état-transition** offre une vision complète et non ambiguë de l'ensemble des comportements de **l'élément** auquel il est attaché. Il présente les séquences possibles d'états et d'actions qu'une **instance de classe** peut traiter au cours de son cycle de vie en réaction à des événements discrets.



» Un objet peut passer par une série d'états pendant sa durée de vie.

- Un **état** représente une période dans la vie d'un objet pendant laquelle ce dernier attend un événement ou accomplit une activité.

N.B. La vision globale du système n'apparaît pas sur ce type de diagrammes puisqu'ils ne s'intéressent qu'à un seul élément du système indépendamment de son environnement.

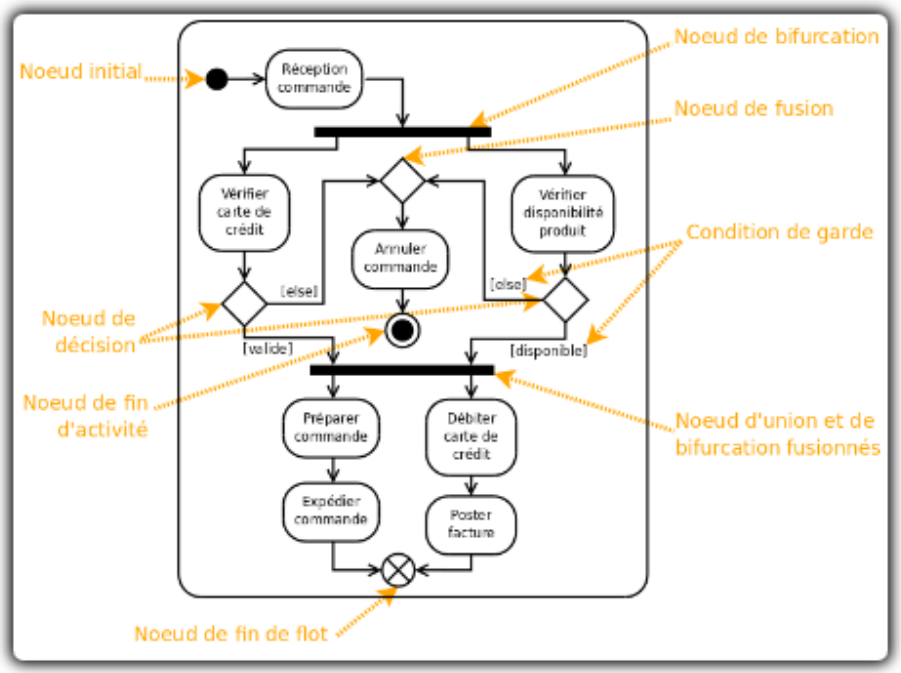


*Exemple de **diagramme d'état-transition** illustrant la création / suppression d'entités géométriques*

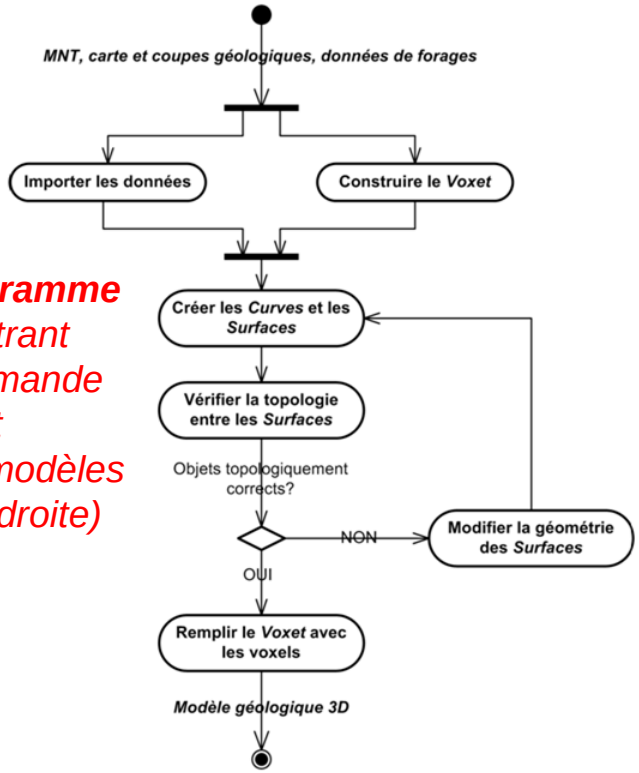
Comment faire quoi ?

– Diagramme d'activités

» Les diagrammes **d'activités** permettent de mettre l'accent sur les **traitements**. Ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle et de flots de données d'une activité à l'autre. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation (version « UML » des organigrammes classiques).



*Exemples de **diagramme d'activités** illustrant une prise de commande (gauche) et la construction de modèles géologiques 3D (droite)*



4. Analyse des traitements au niveau physique

– 4.1. Typologie des traitements

» Les analyses descriptives et conceptuelles ont dû mettre en évidence plusieurs caractéristiques de chaque traitement, permettant d'évaluer leur degré de difficulté de développement.

Exemples de caractéristiques		→ ***Degré de difficulté de développement*** →	
Données d'entrée	*Volume*	Faible	Élevé
	Diversité	Simple	Complexe
Opérations mises en œuvre	*Nombre*	Réduit	Élevé
	Complexité	Faible	Élevée
Fréquence d'utilisation		Rare	Fréquent
Sensibilité aux paramètres		Faible (peu ou pas de paramètres)	Élevée (essais & erreurs…)
Sortie	*Statique / dynamique*	Statique (tableaux, cartes, rapports…)	Dynamique (actions, ajout données…)
	État de la BD	Inchangé	Modifié
	État du schéma interne	Inchangé	Modifié
Utilisateurs	*Compétence*	Faible / Courante	Élevée / Spécialisée
	Nombre	Unique	Groupe
	Privilèges	Limités (consultation)	Larges (transaction)

» En fonction de ses caractéristiques, fixant son degré de difficulté, chaque traitement peut être développé selon une approche efficace.

Exemples de caractéristiques		Degré de difficulté de développement	
Données d'entrée	Volume	Faible	Élevé
	Diversité	Simple	Complexe
Opérations mises en œuvre	Nombre	Réduit	Élevé
	Complexité	Faible	Élevée
Fréquence d'utilisation		Rare	Fréquent
Sensibilité aux paramètres		Faible	Élevée
Sortie	Statique / dynamique	Statique	Dynamique
	État de la BD	Inchangé	Modifié
	État du schéma interne	Inchangé	Modifié
Utilisateurs	Compétence	Courante	Spécialité
	Nombre	Unique	Groupe
	Privilèges	Limités	Larges
Automatisation de l'application en **SQL procédural** à interface réduite			

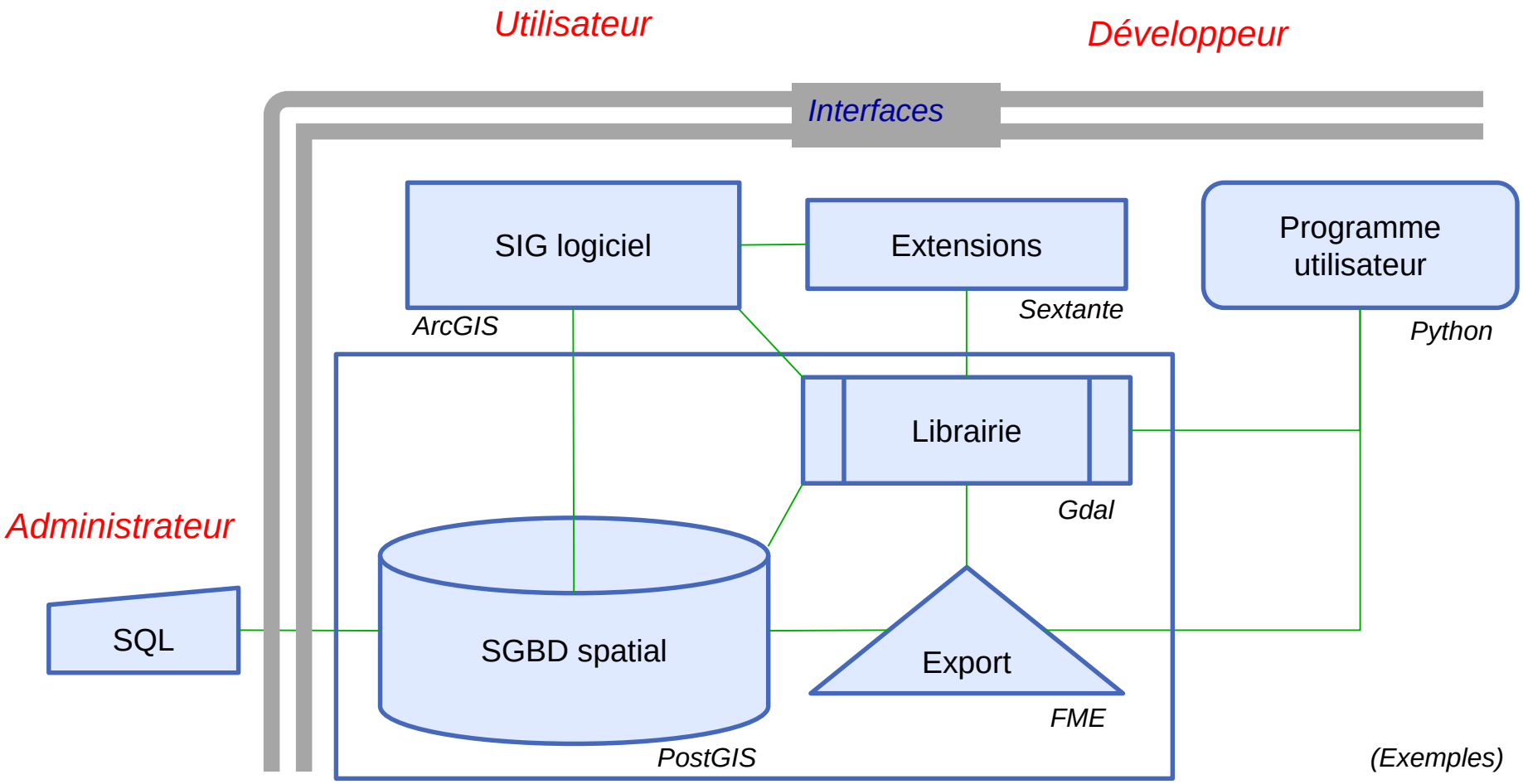
Exemples de caractéristiques		Degré de difficulté de développement	
Données d'entrée	Volume	Faible	Élevé
	Diversité	Simple	Complexe
Opérations mises en œuvre	Nombre	Réduit	Élevé
	Complexité	Faible	Élevée
Fréquence d'utilisation		Rare	Fréquent
Sensibilité aux paramètres		Faible	Élevée
Sortie	Statique / dynamique	Statique	Dynamique
	État de la BD	Inchangé	Modifié
	État du schéma interne	Inchangé	Modifié
Utilisateurs	Compétence	Courante	Spécialité
	Nombre	Unique	Groupe
	Privilèges	Limités	Larges
Développement de l'application dans un **script/programme** à interface dédiée avec contrôle d'intégrité en sortie			

4.2. Choix d'une stratégie de développement

» Le choix du type de développement peut se faire entre les options suivantes :

1. SQL procédural (et spatial) : programmation directe en SQL avec les fonctions d'administrateur sur le SGBD, avec interface dédiée.
2. Recours à des librairies d'extensions (écrites dans un langage de programmation) :
   - 2.1. Conception d'applications au moyen de modeleurs graphiques appelant les extensions, associés au logiciel **SIG** coiffant le SGBD, avec interfaces dédiées. Sauvegardées sous forme de **scripts**.
   - 2.2. Écriture de programmes d'applications « indépendants », dans un langage de programmation compatible avec les extensions, avec interface dédiées.
3. Développement laissé aux utilisateurs (analystes) dans l'environnement du logiciel SIG coiffant le SGBD (avec les extensions), sur une **copie** des données (version de la BD ou en mémoire), avec possibilité de sauvegarde sous forme de scripts avec interfaces minimales. Démarche « **exploratoire** » constituant l'exception.

Modélisation des traitements



Type d'environnement de développement d'applications sur les données spatiales du SGBD

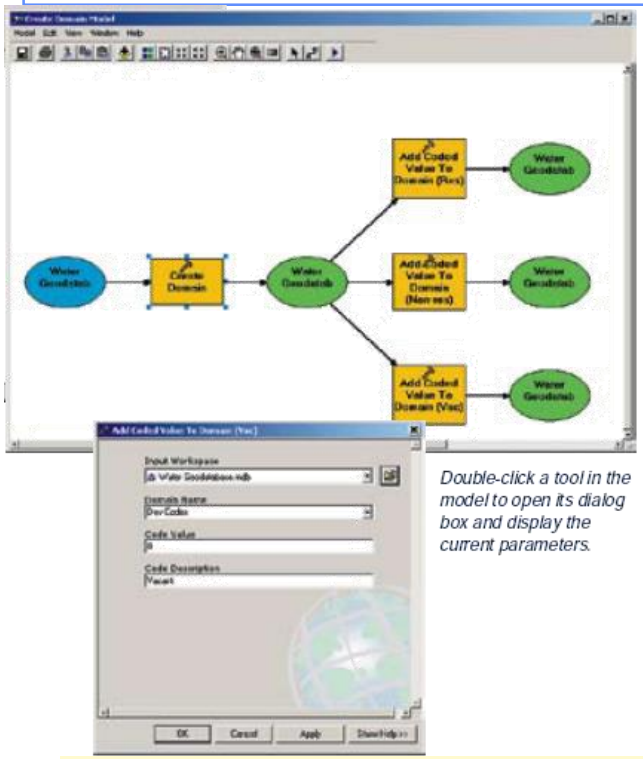
– 4.3. Les modeleurs graphiques de traitement

» Usage interactif d'un **formalisme graphique** symbolisant les « couches » de données et les « traitements » pour réaliser une application.

- Formalisme proche du diagramme d'activité UML
- L'invocation d'une couche de données dans le modeleur provoque la réalisation automatique d'une **interface d'entrée de données** qui sera proposée lors de l'exécution de l'application.
- L'invocation d'un traitement dans le modeleur permet d'ouvrir immédiatement une **fenêtre de paramétrage** du traitement invoqué.
- Une fois l'application modélisée graphiquement, sa forme et sa cohérence sont **validées** par le modeleur.
- Le modèle validé peut être exécuté et sauvegardé sous forme de **script** (*par exemple en Python*) pour un traitement différé ou répété.

» Utilise toutes les fonctions courantes du logiciel SIG hôte, ainsi que les **librairies** d'extensions accessibles depuis le logiciel SIG hôte, tant en mode vectoriel qu'en mode matriciel

Modélisation des traitements

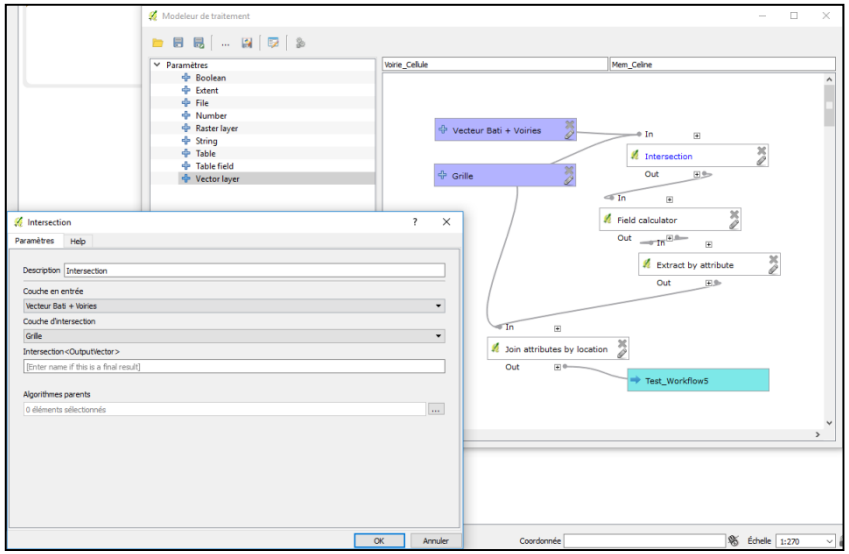


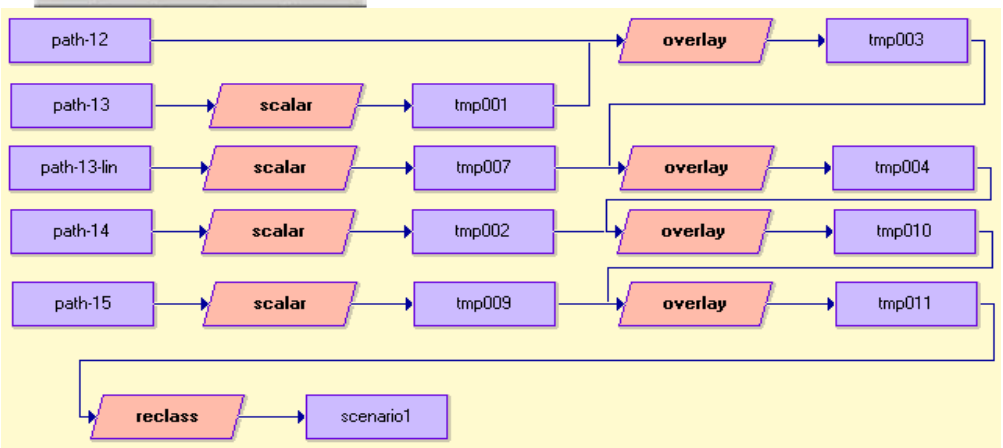
Double-click a tool in the model to open its dialog box and display the current parameters.

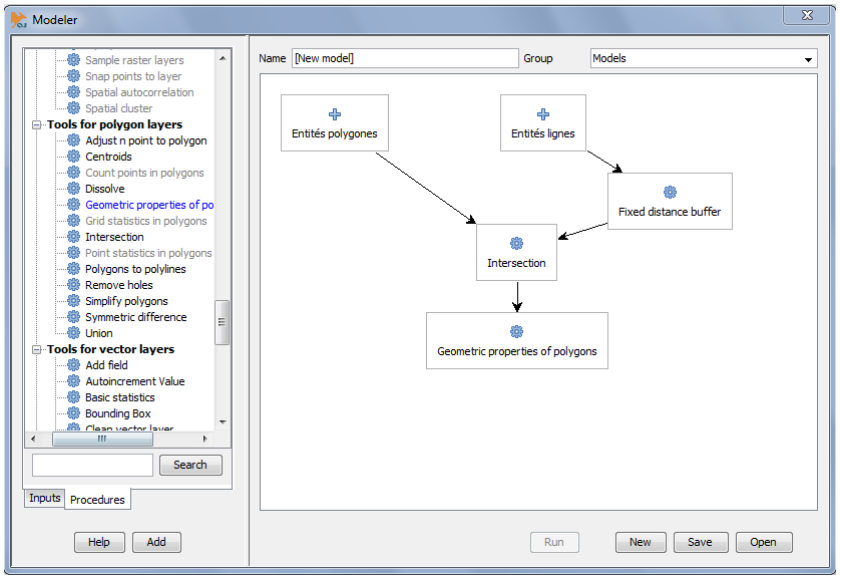
Modélisation au niveau physique d'applications :

En haut à gauche :
***Model Builder** de ArcGIS*
En haut à droite :
***Processing Modeler** de QGIS*

En bas à gauche :
***Macro Modeler** de Idrisi*
En bas à droite :
***Modeler Sextante** dans OpenJump*







– 4.4. Les librairies d'extensions

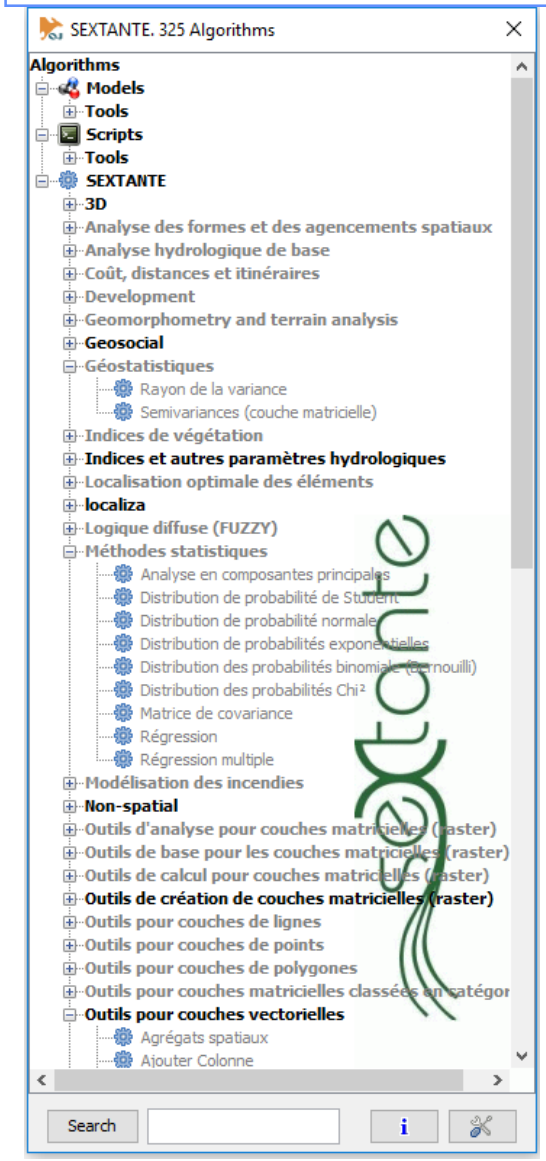
» Les applications spécialisées et les applications « métiers » utilisant les données du SGBD spatial, mais réclamant des procédures plus complexes que les seules fonctionnalités offertes par le SGBD (SQL) et le logiciel SIG hôte, peuvent être confiées à des extensions (algorithmes) conservés dans des librairies spécialisées.

- Les extensions sont liées au logiciel SIG hôte au travers d'une interface programmable d'application (**API**) dans un langage de programmation spécifique (Python, C++, Java…).
  - Une même librairie d'extension peut être liées à plusieurs logiciels SIG distincts grâce à des API polymorphes ou dédiées.

    *Exemples : librairie **Sextante**, accessible depuis ArcGIS ou OpenJump; librairie **GDAL** accessible depuis un très grand nombre de logiciels.*

» Selon le même mécanisme (API), un logiciel SIG peut invoquer les fonctionnalités offertes d'un **autre** logiciel SIG considéré comme une « librairie ».

- *Exemples : fonctionnalités de **GRASS** ou **SAGA** accessible depuis QGIS.*

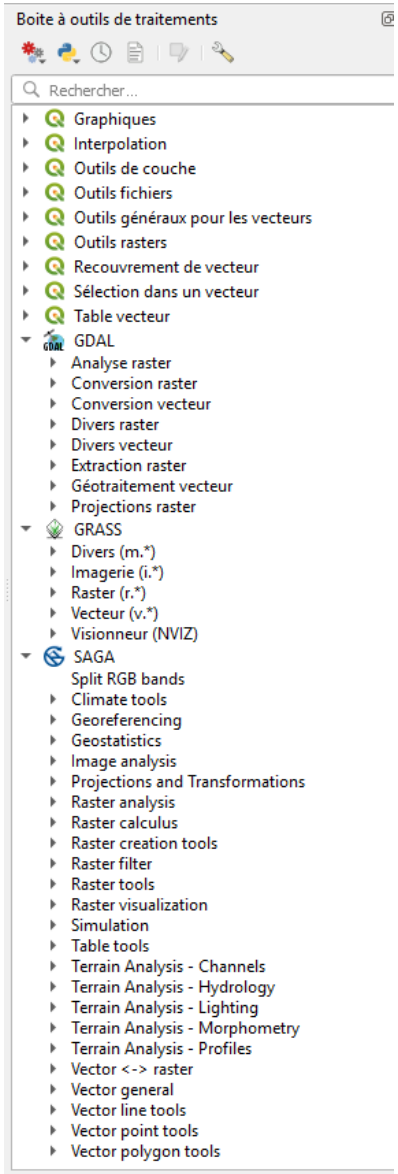


Les librairies d'extension

À gauche :
Arbre des extensions offertes par la librairie ***Sextante***
(325 algorithmes utilisés ici via OpenJump)

À droite :
Arbre des extensions offertes par les librairies ***GDAL, GRASS, SAGA****, en plus des extensions propres de QGIS (version 3.4)*

Tous ces algorithmes peuvent être évoqués dans les ***modeleurs*** *de traitement de ces 2 logiciels SIG O.S.*



– 4.5. Les programmes indépendants (programmes d'application)

» Des programmes indépendants peuvent être développés par l'utilisateur dans des langages de programmation courants (C, Java, Python…), utilisant les mêmes librairies d'algorithmes que celles invoquées par un logiciel SIG.

» *Exemple de librairie : GDAL* (*Geospatial Data Abstraction Library*) librairie *Open Source* présentant un modèle de données raster proche des spécifications *OpenGIS Grid Coverages,* ainsi que des fonctionnalités vectorielles.

- Les fonctions de la librairie sont importées dans le programme appelant et manipulables dans plusieurs langages de programmation (principe des **API** – *Application Progamming Interface*).

```
#####################################################
#
# Imports
#
#####################################################

import re
import os
import math
import uuid
import Polygon
import numpy
from osgeo import gdal
from osgeo import osr
from osgeo import gdalconst
from osgeo.gdalconst import *
from landsafe_const import *
from landsafe_config import *
from landsafe_logging import *
```

Exemple de liaison entre le programme appelant (Python) et une série de librairies (dont GDAL)

Utilisation d'une routine GDAL (warping) dans le corps du programme appelant (ici par exécution d'une ligne de commande avec tous ses paramètres)

```
        # Perform re-projection
        command_line = 'nohup /usr/local/bin/gdalwarp -dstnodata ' +
str(CST_NO_DATA) + ' -tr ' + str(pixel_width) + ' ' + str(pixel_height) + '
-t_srs ' + wkt_file + ' ' + orig_map_file + ' ' + dest_map_file
        log_dbg(CST_GDL_MOD, 'gdl_project_raster_map(): cmd: ' +
command_line)
        os.system(command_line)
        return True
```

• 5. Intégration des données

5.1. *Extract – Transform – Load (ETL)*

– La plus grande partie des données du SIG, tant vectorielles que raster, doivent être importées depuis des sources externes.

– **Identification : dès l'étape d'analyse descriptive détaillée.**

» Sources, coût, copyrights, formats, précisions, mise à jour.

– **Acquisition :**

» Accords commerciaux et légaux avec les fournisseurs de données.

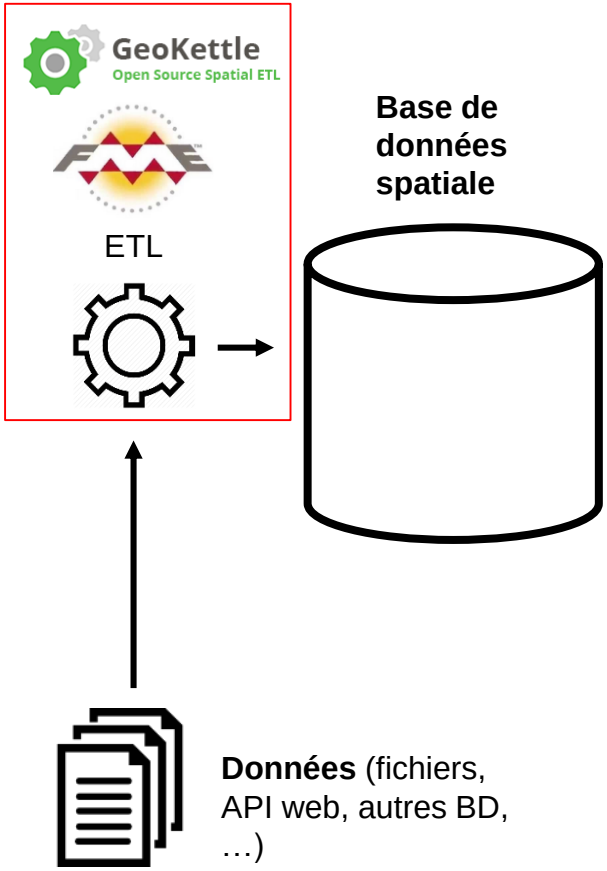
» Acquisition par téléchargement / services Web.

• Changement de formats, *parsing* (langage de description de données).

» Premier archivage sous forme brute (*RAW*) des données externes (hors SGBD).

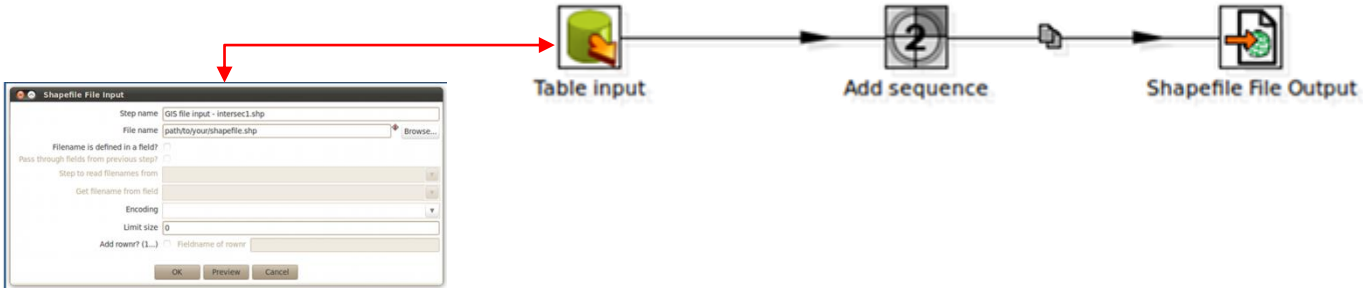
– **Stratégies d'implémentation des données :**

1. Via un logiciel ETL spatial : **FME** (*Feature Manipulation Engine*), **GeoKettle** (OS), **Talend** Spatial Extension (OS)...
   • Plusieurs outils CASE proposent de nombreuses fonctions ETL.
2. Via l'interface d'administration du **SGBD** (langage SQL) utilisant des tables temporaires.
3. Mixte.



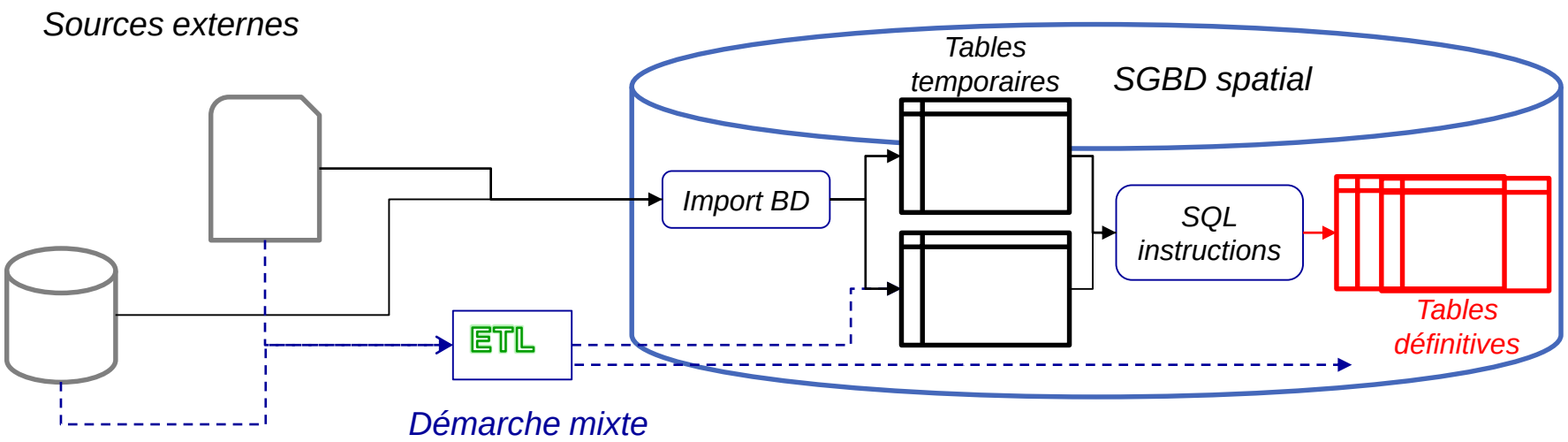
5.2. Les outils ETL spatiaux

- Logiciels permettant la synchronisation massive de données d'une source vers une autre, assurant des fonctions de **chargement**, **transformation** (y compris correction d'erreurs) et mise en conformité (**mappage**) des données, ainsi que leur **intégration** en vue des applications d'une entreprise.
  - » *N.B. La plupart des SGBD offrent des solutions avancées d'intégration.*
- L'objectif est de récupérer des données (géographiques) sous formes diverses (formats) et de les charger dans les tables (spatiales) du SGBD tout en assurant des transformations éventuelles à la volée.
- Le **modèle** des opérations est **formalisé** de manière interactive au moyen de pictogrammes, chaque opération étant paramétrée par l'utilisateur. Les modèles une fois validés peuvent être exécutés et sauvegardés.



5.3. *Data loading* via les fonctions d'administration du SGBD

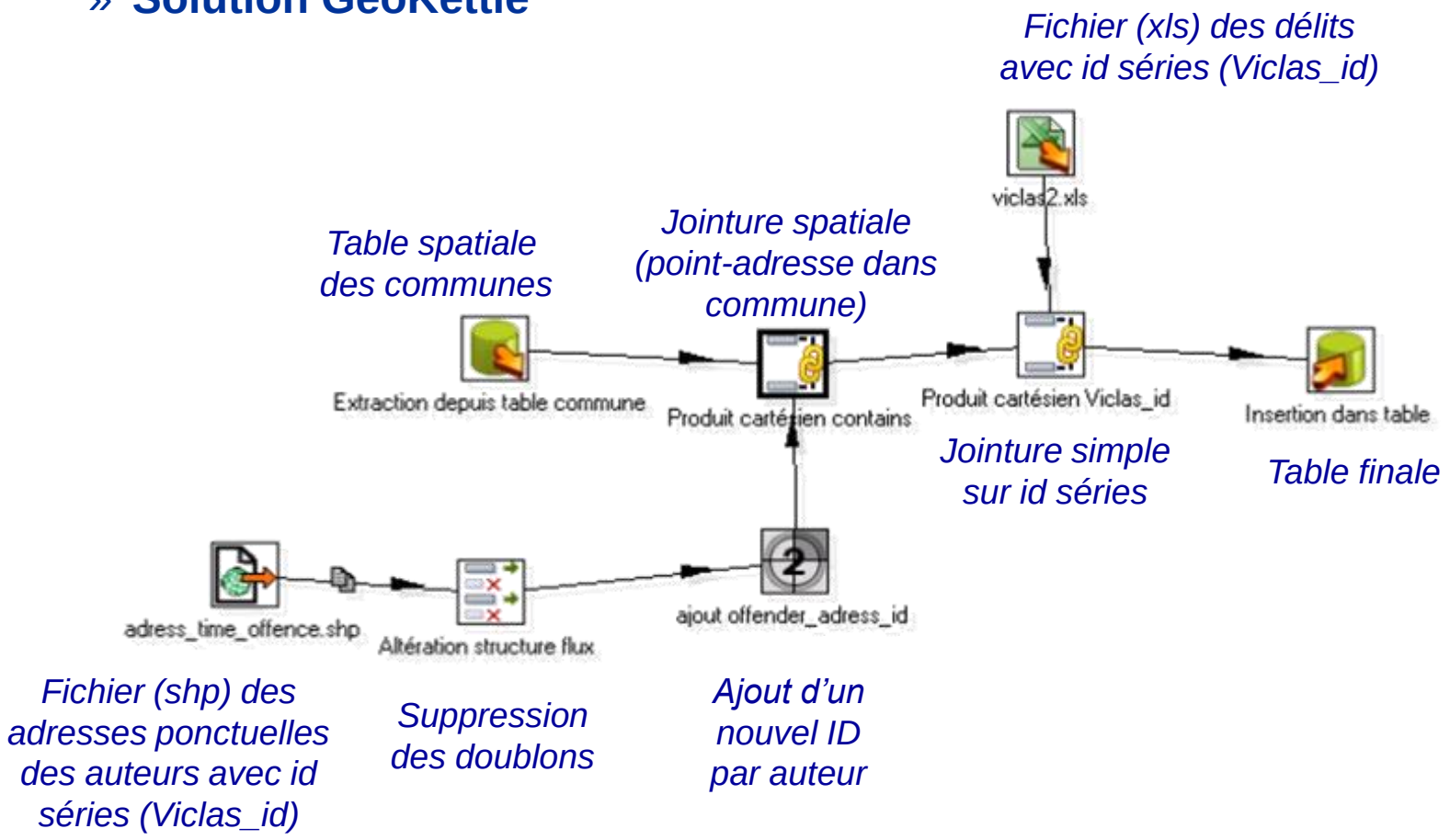
- Les SGBD disposent de fonctionnalités SQL d'importation et d'intégration de données qui peuvent être utilisées à la place ou en complément d'un logiciel ETL.
- Le principe consiste à charger « automatiquement » (le plus simplement possible) les données externes dans des **tables temporaires**, puis d'effectuer les traitements d'extraction et de transformation au moyen de commandes SQL adéquates, et enfin d'alimenter les tables définitives par les résultats de ces opérations.



5.4. *Exemple : extraction de données criminelles*

» Table finale des séries de délits combinant les informations des auteurs, leurs commune de résidence et les informations sur les délits

» **Solution GeoKettle**



- **6. Synthèse des différentes implémentations de traitements SIG**