sETL: Outils ETL pour la construction de graphes de connaissances en exploitant la sémantique implicite des schémas de données

S. Ouelhadj[1,2], P. Champin[1], J. Gaillard[2]

[1] Univ Lyon, UCBL, CNRS, INSA Lyon, Centrale Lyon, Univ Lyon 2, LIRIS, UMR5205, F-69622 Villeurbanne, France
[2] Métropole de Lyon, Lyon, France

{firstname.lastname}@liris.cnrs.fr; jegaillard@grandlyon.com

Résumé

Nous présentons sETL, une nouvelle approche pour la construction de graphes de connaissances (GCs) en utilisant les technologies du Web Sémantique (WS). Nous abordons les défis rencontrés par la Métropole de Lyon - une collectivité territoriale française - pour assurer l'interopérabilité des données ouvertes de sa plateforme `data.grandlyon.com`*. sETL exploite la sémantique implicite des schémas de données, et fournit une boîte à outils aux praticiens de données pour enrichir sémantiquement leurs données sans requérir des connaissances spécifiques en WS. Ce travail formalise un Modèle Sémantique, introduit le concept de Bundles pour la transformation des données, et présente une implémentation Python d'opérateurs ETL de haut niveau. L'approche est comparée à celles de l'état de l'art, mettant en évidence ses caractéristiques uniques, et sa capacité à répondre aux exigences spécifiques identifiées dans l'étude.*

Mots-clés

Enrichissement sémantique, ETL, Graphe de connaissances, Schéma de données.

Abstract

We introduces sETL, a novel approach to build knowledge graphs (KGs) using Semantic Web (SW) technologies. It addresses the challenges faced by the Metropolis of Lyon - a French local authority - in ensuring interoperability of open data from its `data.grandlyon.com` *platform. sETL leverages the implicit semantics of schemas, and provides a toolkit for data practitioners to semantically lift their data without requiring specific SW knowledge. The paper formalizes a Semantic Model, introduces the concept of Bundles for data transformation, and presents a Python implementation of high-level ETL operators. The approach is compared to related works, highlighting its unique features and ability to meet specific requirements identified in the study.*

Keywords

Semantic Lifting, ETL, Knowledge Graph, Data Schema.

1 Introduction

Les données sont continuellement produites et publiées sur le web conformément à des normes et standards, afin d'améliorer leur compréhensibilité, interopérabilité et intégration [25]. La Métropole de Lyon - une collectivité territoriale française - et ses partenaires ont embrassé le mouvement des données ouvertes, et ont mis en place un point d'accès central aux données locales qu'ils produisent, appelé `data.grandlyon.com`. Les producteurs de données de la Métropole de Lyon font des efforts considérables pour améliorer continuellement la qualité des données. Parmi ces efforts figure la participation à la production et à l'utilisation de schémas de données partagés afin de normaliser les données.

Les schémas de données permettent de décrire le modèle de données des jeux de données. Ils fournissent des descriptions précises et non ambiguës des différents champs qui composent un jeu de données : les valeurs possibles, les types, le caractère obligatoire ou non du remplissage de ces champs, etc. La production de données conformes à un schéma présente de nombreux avantages tels que la validation des données, l'amélioration de leur interopérabilité et croisement, la génération automatique de documentations, et la pérennité des modèles de données [1]. Pour cette raison, plusieurs communautés de producteurs de données, de réutilisateurs, d'experts métiers et techniques ont été constituées pour le développement de schémas de données partagés [2].

Toutefois, les défis liés à l'amélioration de la compréhensibilité, l'interopérabilité et l'intégration des données sont toujours d'actualité, car la sémantique de ces schémas de données est largement implicite. En effet, si les schémas de données capturent une sémantique partagée, notamment dans les descriptions textuelles de leurs champs qui peuvent

1. https://guides.data.gouv.fr/publier-des-donnees/guide-qualite/maitriser-les-schemas-de-donnees/comprendre-les-benefices-dutiliser-un-schema-de-donnees
2. par ex. https://schema.data.gouv.fr, https://smart-data-models.github.io/data-models, https://www.futurocite.be/standardiser-les-donnees-ouvertes/

être comprises par les producteurs et consommateurs de données, cette sémantique n'est pas accessible aux machines, et est donc moins prometteuse pour l'interopérabilité et l'intégration des données à grande échelle. Un moyen de relever ces défis est de construire des graphes de connaissances (GCs) en utilisant les technologies du Web Sémantique (WS) [2], et en exploitant la sémantique implicite des schémas de données déjà disponibles et produits par les producteurs de données. Les technologies du WS nécessitent l'utilisation de RDF [27] comme modèle de données basé sur des graphes, et d'ontologies [4] pour définir formellement la sémantique. Cela peut constituer un obstacle pour les praticiens de données qui ne sont pas familiers avec les technologies du WS, mais qui manipulent généralement des données dans des formats (semi-)structurés (ex. JSON, CSV), en utilisant des schémas de données [1, 17] au lieu d'ontologies. Ces schémas sont basés sur des spécifications techniques telles que JSON Schema [32] ou Table Schema [31].

Par conséquent, notre objectif dans le contexte de la Métropole de Lyon, et plus largement pour toute organisation productrice de données, est de permettre aux praticiens de données de construire des GCs à partir de données dans des formats (semi-)structurés. Pour ce faire, nous avons identifié 5 exigences (Ri) basées sur les conclusions d'un atelier mené avec les producteurs de données de la Métropole de Lyon [24]. Ces exigences sont les suivantes :

(R1) exploiter la sémantique implicite des schémas de données existants : les participants de notre atelier connaissent bien les schémas de données, certains s'appuient déjà sur des dictionnaires de données internes, et veulent développer de bonnes pratiques pour améliorer la qualité des données ;

(R2) impliquer des praticiens de données sans compétences approfondies en WS : aucun des participants de notre atelier n'était familier avec les concepts du WS ;

(R3) être applicable à des structures de données hétérogènes : les données qu'ils produisent sont dans des formats variés (ex. CSV, JSON, GeoJSON) ;

(R4) être en mesure de s'aligner avec les ontologies existantes : ils s'intéressent aux vocabulaires partagés développés par les instituts nationaux et les organisations gouvernementales tels que l'IGN [3], afin de fournir une compréhension commune de la signification des termes utilisés (harmonisation des termes), et de relier les données en interne et en externe ;

(R5) être en mesure de produire des ontologies manquantes lorsqu'aucune n'est disponible pour décrire les données en question : ils sont moins intéressés par les vocabulaires à usage général tels que schema.org [4] et DBpedia, dont les termes sont souvent définis vaguement. Une nouvelle ontologie légère, basée sur les éléments du schéma, est considérée comme plus souhaitable pour leurs objectifs.

A partir de ces exigences, nous proposons dans cet article la boîte à outils sETL, pour « semantic ETL » (*Extract Transform Load*). Elle est basée sur des concepts et technologies d'ingénierie des données bien connus (UML, ETL, Python, Pandas dataframes) intégrés en une nouvelle approche pour tenter d'abaisser la barrière des compétences en WS requises dans la construction de GCs, permettant ainsi une exploitation plus efficace des capacités des technologies du WS. Les contributions de ce travail sont les suivantes :

1. la formalisation d'un *Modèle Sémantique* (MS) qui exprime la sémantique implicite des données à partir des schémas fournis, et l'expose en vue d'un raffinement ultérieur ;
2. la définition du concept de *Bundle* qui permet de décomposer le jeu de données et d'en associer chaque partie avec l'élément du MS lui correspondant ;
3. la mise en œuvre d'un ensemble d'opérateurs ETL de haut niveau qui permettent d'affiner les données et leur sémantique correspondante au niveau du bundle, afin de s'adapter aux contraintes des ontologies cibles et de gagner en expressivité, avant de charger ces bundles en un GC.

Dans la suite de cet article, nous commençons par présenter un exemple pour illustrer les caractéristiques de sETL, basé sur un jeu de données ouvert existant. Dans la section 2, nous détaillons la boîte à outils ETL proposée (sETL) où nous définissons les concepts de Modèle Sémantique, Bundle, Graphes de Bundles, et comment nous appliquons dans sETL les trois phases du paradigme ETL. Ensuite, dans la section 3, nous décrivons les aspects de mise en œuvre de la boîte à outils. Puis, dans la section 4, nous passons en revue l'état de l'art des approches de construction de GCs, et les comparons à sETL à travers nos 5 critères ci-dessus. Enfin, dans la section 5, nous présentons nos conclusions et les travaux futurs possibles.

Exemple fil conducteur

Cet exemple est basé sur un jeu de données ouvert à partir du Point d'Accès National français aux données de transport. Le jeu de données décrit l'emplacement géographique et les caractéristiques techniques des Infrastructures de Recharge pour Véhicules Electriques (IRVE) [5]. Il est publié au format CSV, avec un schéma [6] exprimé selon la spécification Table Schema [31]. Ce jeu de données contient 40 colonnes. Par souci de concision, nous ne considérons que 7 colonnes : nom_operateur, contact_operateur, telephone_operateur, id_station_local, nom_station, adresse_station, implantation_station. Selon le schéma fourni, implantation_station a un ensemble défini de valeurs autorisées : "Voirie", "Parking public", "Parking privé à usage public", "Parking privé réservé à la clientèle",

3. http://data.ign.fr/data.html
4. https://schema.org/
5. https://transport.data.gouv.fr/datasets/fichier-irve-gireve?locale=fr
6. https://schema.data.gouv.fr/schemas/etalab/schema-irve-statique/2.2.0/schema-statique.json

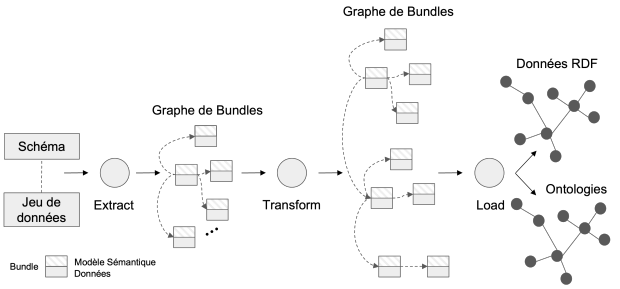

FIGURE 1 – Vue d'ensemble du processus de construction d'un graphe de connaissances avec la boîte à outils sETL.

"Station dédiée à la recharge rapide". Un échantillon du jeu de données est fourni dans le Tableau 1. Le jeu de données complet et son schéma sont disponibles dans le répertoire du projet [7].

2 sETL

La boîte à outils sETL vise à exploiter les schémas dans un processus de construction de GCs suivant le paradigme ETL. La Figure 1 présente une vue d'ensemble de ce processus. Au cours de ce processus, nous manipulons un nouveau type d'objets, appelés bundles, organisés au sein d'un graphe de bundles. Un bundle regroupe une partie des données avec son Modèle Sémantique. Ci-après, nous décrivons d'abord le Modèle Sémantique dans la section 2.1. Ensuite, nous définissons le Bundle et le Graphe de Bundles dans la section 2.2. Enfin, nous décrivons les trois phases du paradigme ETL dans le contexte de sETL. La phase *Extract* (Section 2.3) convertit un jeu de données et son schéma en un graphe de bundles. La phase *Transform* (Section 2.4) affine le graphe de bundles afin d'améliorer la sémantique des données. La phase *Load* (Section 2.5) exporte le graphe de bundles en une ontologie et des données RDF.

2.1 UML annoté : le Modèle Sémantique

Le modèle sémantique adopté dans sETL, également appelé UML annoté, est basé sur le diagramme de classes UML avec des ajouts détaillés ci-dessous. Les diagrammes de classes UML offrent un large éventail de types de composants. Pour cette première version de la boîte à outils, nous n'utilisons que les types de composants suivants : les classes (et leurs attributs), les énumérations (et leurs valeurs énumérées), et les associations. La Figure 2, partie C, illustre ces types de composants, dans un diagramme de classes UML capturant la sémantique implicite de notre exemple fil conducteur.

Ajouts de l'UML annoté

L'UML annoté étend les diagrammes de classes UML en permettant à chaque composant (classe, énumération, association, attribut et valeur énumérée) d'être annoté par un *IRI* et une *documentation textuelle* de ce composant. Les IRIs permettent de faire le lien entre l'UML annoté et les ontologies du WS. En outre, une énumération peut se voir attribuer un *type de données*, et chaque valeur énumérée peut être *alignée* avec des entités des GCs externes (ex. Wikidata, DBpedia).

Sur la base de ces considérations, nous définissons les notations suivantes pour les composants de l'UML annoté. Un Modèle Sémantique MS est un ensemble $\{MS_i \| \ i \in [0, n[\}$, où chaque MS_i est soit une $classe$, soit une $enumeration$, et où :

- chaque $classe$ a la forme $(nom,\ IRI,\ definition,\ attributs,\ associations)$;
- chaque $enumeration$ a la forme $(nom,\ IRI,\ definition,\ type,\ valeurs\_enumerees)$;
- chaque $attribut$ a la forme $(nom,\ IRI,\ definition,\ type,\ estIdentifiant)$;
- chaque $association$ a la forme $(nom,\ IRI,\ definition,\ destination)$, où $destination$ est une référence vers une $classe$ ou une $enumeration$ du MS ;
- chaque $valeur\_enumeree$ a la forme $(nom,\ IRI,\ definition,\ alignements)$;
- chaque $alignement$ a la forme $(autre\_entite,\ relation)$.

Comme tout diagramme de classes UML, un Modèle Sémantique (MS) peut être considéré comme un graphe orienté labellisé, dont les nœuds sont des classes et des énumérations, et dont les arêtes sont des associations.

2.2 Bundle et graphe de bundles

Dans sETL, les données ne sont jamais manipulées ou transformées de manière isolée : elles sont constamment liées à un Modèle Sémantique, dans des unités appelées *bundles*. En reliant chaque nœud (classe ou énumération) du Modèle Sémantique aux données correspondantes, nous obtenons une nouvelle structure : le *graphe de bundles*.

Considérons un bundle $b_i = (MS_i,\ D|MS_i)$ où MS_i est une classe ou une énumération du Modèle Sémantique, et $D|MS_i$ sont les données décrivant les instances de MS_i (notation inspirée de [3]). Un bundle peut être soit un bundle-classe ($b_i \in B_{class}$) si MS_i est une classe, soit un bundle-énumération ($b_i \in B_{enum}$) si MS_i est une énumération. Par conséquent, un graphe de bundles G_B est un graphe orienté labellisé où $B = \{(b_0, b_1, \cdots, bn);\ b_i = (MS_i,\ D|MS_i)\} = B_{class} \cup B_{enum}$.

Le Modèle Sémantique MS du graphe de bundles final représente l'ontologie sous-jacente du jeu de données en entrée. Ainsi, l'export de données RDF à partir du graphe de bundles final revient à peupler l'ontologie sous-jacente avec les données contenues dans chaque bundle du graphe de bundles. Par conséquent, nous remarquons que le graphe de bundles fournit une double vue, en faisant abstraction de la plupart des concepts du WS : une vue ontologique du jeu de données en entrée par le biais de l'UML annoté (MS), et une vue compacte et tabulaire des données RDF à produire par le biais des données contenues dans chaque bundle $D|MS_i$.

7. https://github.com/Sarra-Ouelhadj/SemanticLifting/tree/ic2024/Examples/Charging_Stations

TABLE 1 – Échantillon de données à partir de l'exemple fil conducteur

id_station_local	nom_station	adresse_station	nom_operateur	contact_operateur	telephone_operateur	implantation_station
756453	BornEco/ 63dcef1cde53 c3ec2928c1e	3 Place Maurice de Sully, Sully-sur-Loire 45600 France	Borneco | FR*BHM	technique.borneco @gmail.com	33123456789	Voirie
510419	WattzHub/ 625fc63fb907 c5cc90734800	40 Allée de la Mare Jodoin, Gif-sur-Yvette 91190 France	WattzHub | FR*SMI	contact @wattzhub.com	33185412867	Parking public

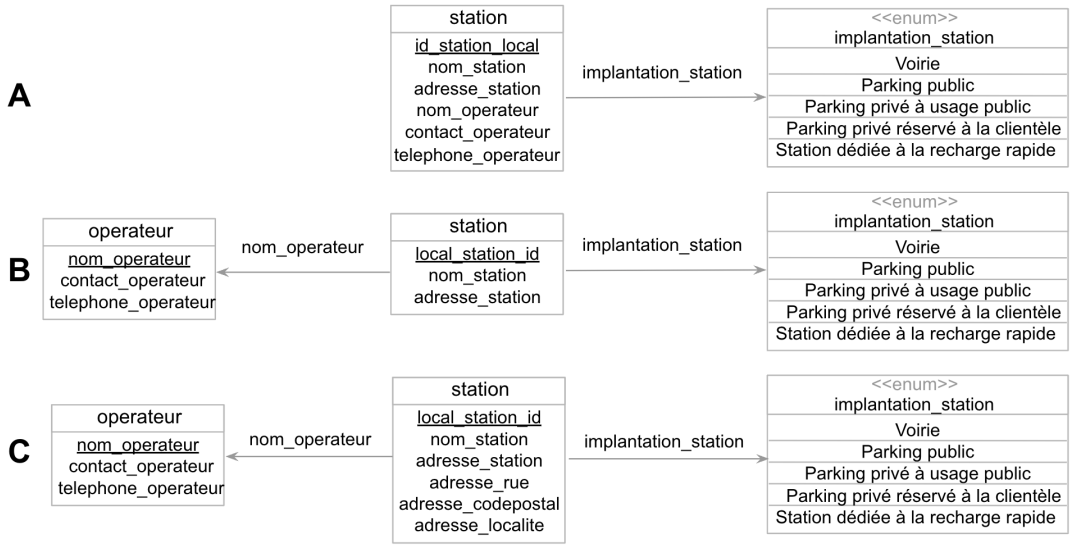

FIGURE 2 – Modèle Sémantique de notre exemple fil conducteur, à 3 phases du processus (A, B, et C).

2.3 Extracteurs

sETL extrait un graphe de bundles initial par le biais d'opérateurs appelées extracteurs à partir d'un schéma en entrée et de son jeu de données conforme. Les jeux de données dans différents formats de données sont conformes à des schémas définis à l'aide de différentes spécifications (ex. JSON Schema pour JSON, Table Schema pour CSV, etc.) Nous définissons donc un extracteur pour chaque paire de spécification de schéma et de format de données. Chaque extracteur met en correspondance chaque type d'élément du schéma source avec une construction correspondante dans notre Modèle Sémantique (MS).

Par exemple, notre extracteur Table Schema CSV appliqué sur l'exemple fil conducteur donne le résultat en Figure 2, partie A. Cet extracteur crée une classe unique pour l'ensemble de la table. Les colonnes déclarées par le schéma avec un type énuméré deviennent une association entre cette classe et une nouvelle énumération UML, également nommée d'après la colonne ("implantation_station" dans notre exemple). Toutes les autres colonnes deviennent des attributs de la classe. La $definition$ de chaque élément du MS est extraite à partir des descriptions textuelles fournies par le schéma (le cas échéant) ; il en va de même pour d'autres aspects du MS (par exemple $type$ ou $estIdentifiant$ pour les attributs). Les données liées à la classe sont la table entière. Les données liées à chaque énumération sont la colonne unique à partir de laquelle elle a été générée.

2.4 Transformeurs

Les transformeurs sont des opérateurs de raffinement disponibles pour l'utilisateur afin d'affiner le MS en un modèle plus complexe et riche sur le plan sémantique. Ces transformeurs sont appliqués sur les bundles, et sont listés dans le Tableau 2 où nous donnons de brèves descriptions de leurs effets. Par manque de place, nous ne pouvons pas donner une description détaillée de chacun d'entre eux, mais à titre d'exemple, nous donnons l'Algorithme 1 pour le transformeur *split*.

Dans l'exemple fil conducteur, nous appliquons 3 transformeurs : *split*, *apply*, et *annotate*. Premièrement, comme le bundle-classe "station" contient des données sur les opérateurs, qui sont sémantiquement différents des stations, nous divisons le bundle "station" afin d'ajouter un bundle-classe "opérateur" qui lui est lié. Les paramètres du transformeur *split* comprennent le nom (« opérateur » dans notre exemple) de la nouvelle classe, l'attribut servant d'identifiant à la nouvelle classe ("nom_operateur"), et les autres attributs et associations qui doivent être déplacés vers la nouvelle classe ("contact_operateur", et "telephone_operateur"). Le MS résultant est illustré dans la Figure 2, partie B. Les données associées à chaque bundle sont la projection correspondante du Tableau 1. Deuxièmement, nous affinons le contenu de l'attribut

TABLE 2 – Liste des transformeurs de sETL

Appliqué sur	Nom et Description
$B_{class} \cup B_{enum}$	**annotate** : assigner un IRI à un composant du MS_i. **document** : donner une définition textuelle à un composant du MS_i. **reconcile** : trouver les IRIs correspondant aux valeurs des colonnes spécifiées de chaque ligne de $D\|MS_i$ à partir des GCs externes (ex. Wikidata). **rename** : changer le nom d'un composant du MS_i, et mettre à jour les colonnes de $D\|MS_i$ en conséquence.
B_{class}	**apply** : appliquer une fonction aux valeurs des colonnes spécifiées de chaque ligne de $D\|MS_i$ et mettre à jour le MS_i en conséquence. Le(s) résultat(s) de la fonction est (sont) utilisé(s) pour remplir une (ou plusieurs) nouvelle(s) colonne(s) dans $D\|MS_i$, et le MS_i est enrichi des attributs correspondants. **mark_identifier** : marquer un attribut comme identifiant. **split** : diviser en 2 bundles-classes reliés entre eux par une association. **transform_attr_to_enum** : transformer un attribut en une énumération peuplée de toutes les valeurs de cet attribut dans $D\|MS_i$ en tant que B_{enum}.
B_{enum}	**add_value** : ajouter une valeur énumérée.

Algorithme 1 $b_1 = b_0.split(id,\ attributs,\ associations,\ nouv\_nom\_class)$

Entrées:

$b_0 = (MS_0,\ D_0|MS_0) \in B_{Class}$
$nouv\_nom\_class \in String$
$id \in MS_0.attributs$
$attributs \subset MS_0.attributs$
$associations \subseteq MS_0.associations$

soit MS_1 = class-vide
soit $D_1|MS_1$ = dataset-vide
$MS_1.nom \leftarrow nouv\_nom\_class$
$MS_1.definition \leftarrow id.definition$
$MS_0.attributs$.remove(id)
$id.estIdentifiant \leftarrow True$
$MS_1.attributs$.append(id)
$D_1|MS_1$.add_column($id.nom$)
pour $elem$ dans $attributs$ **faire**
    $MS_1.attributs$.append($elem$)
    $MS_0.attributs$.remove($elem$)
    $D_1|MS_1$.add_column($elem.nom$)
    $D_0|MS_0$.pop_column($elem.nom$)
pour $asso$ dans $associations$ **faire**
    $MS_1.associations$.append($asso$)
    $MS_0.associations$.remove($asso$)
    $D_1|MS_1$.add_column($asso.nom$)
    $D_0|MS_0$.pop_column($asso.nom$)
$b_1 \leftarrow$ BundleClass($MS_1,\ D_1|MS_1$)
$MS_0.associations$.append((nom = $id.nom$, definition = $id.definition$, destination = b_1))
retourne b_1

"adresse_station". Le schéma précise que les valeurs de cette colonne doivent contenir (1) le numéro et le nom de la rue, (2) le code postal et (3) la localité. Nous écrivons une fonction simple qui sépare ces trois composants et la transmettons au transformeur *apply*, avec les noms des trois colonnes à créer : "adresse_rue", "adresse_codepostal", "adresse_localite" (Figure 2, partie C).

Troisièmement, nous remarquons que plusieurs attributs de la classe "station" ont des termes correspondants dans l'ontologie schema.org. Nous utilisons l'opérateur *annotate* pour attacher l'IRI approprié à chacun de ces attributs. Il peut y avoir autant d'opérations de transformation que l'utilisateur le juge nécessaire. Dans notre exemple, l'utilisateur pourrait vouloir transformer les attributs de l'adresse postale en une classe distincte "adresse_postale" (en utilisant "adresse_station" comme clé primaire).

2.5 Loadeurs

Les loadeurs exportent le graphe de bundles final en un graphe de connaissances RDF. sETL fournit un loadeur d'ontologie et un loadeur de données RDF. Ces loadeurs attendent comme paramètres des espaces de noms, qui sont utilisés pour forger des IRIs pour les classes, les énumérations et les valeurs énumérées, et les instances.

Loadeur d'Ontologie. Lorsqu'un composant – classe, énumération, association, attribut ou valeur énumérée – n'est pas annoté explicitement par un IRI provenant d'une ontologie existante, un nouveau terme est créé à l'aide d'un template préconfiguré d'une ontologie (exemple dans Figure 3).

Loadeur de Données. Chaque ligne des données d'un bundle-classe représente une instance de cette classe. L'IRI de la classe est celui du MS s'il existe, sinon l'IRI forgé par le loadeur d'ontologie. Un triplet est généré pour chaque cellule du tableau des données du bundle-classe. Un exemple de triplets RDF est présenté dans le Listing 1. Le sujet de chaque ligne est l'IRI obtenu par la concaténation de l'espace de noms des instances avec la valeur de

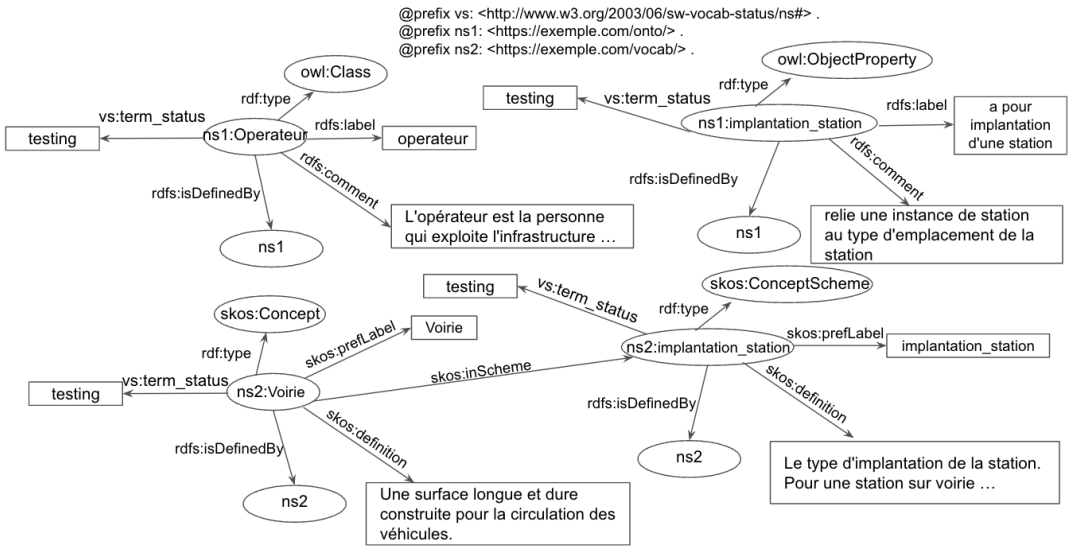

FIGURE 3 – Une partie de l'ontologie générée à partir du Modèle Sémantique du graphe de bundles illustré dans la Figure 2, partie C

```
@prefix ns1: <https://exemple.com/onto/> .
@prefix ns2: <https://exemple.com/vocab/> .
@prefix schema: <https://schema.org/> .

<https://exemple.com/id/station/756453> a ns1:Station ;
    schema:identifier "756453" ;
    schema:name "BornEco/63dcef1cde530c3ec2928c1e" ;
    schema:address "3  Place Maurice de Sully, Sully-sur-Loire  45600  France" ;
    schema:streetAddress "3  Place Maurice de Sully" ;
    schema:postalCode "45600" ;
    schema:addressLocality "Sully-sur-Loire" ;
    ns1:implantation_station ns2:Voirie ;
    ns1:nom_operateur <https://exemple.com/id/operateur/Borneco%20%7C%20FR%2ABHM> .

<https://exemple.com/id/operateur/Borneco%20%7C%20FR%2ABHM>
    a ns1:Operateur ;
    schema:name "Borneco | FR*BHM" ;
    schema:email "technique.borneco@gmail.com" ;
    schema:telephone  "33123456789" .
```

Listing 1 – Un exemple de triplets RDF générés

la colonne clé correctement échappée. Le prédicat est l'IRI de l'attribut ou de l'association correspondant à cette colonne, tel qu'annoté dans le MS ou forgé par le loadeur d'ontologie. Si la colonne correspond à un attribut, l'objet est un littéral, qui est la valeur de la colonne. Si la colonne correspond à une association, et que le bundle associé est un bundle-classe, l'objet est l'IRI identifiant l'instance correspondante au bundle-classe associé. Si la colonne correspond à une association, et que le bundle associé est un bundle-énumération, l'objet est l'IRI associé à la valeur énumérée dans la cellule actuelle.

3 Implémentation

La boîte à outils sETL est développée comme une bibliothèque Python. Les utilisateurs enrichissent leurs données en écrivant de simples scripts Python, en s'appuyant sur les classes (pour les bundles) et les fonctions (pour les opérateurs ETL) fournies par sETL. En interne, les données des bundles sont manipulées à l'aide de la bibliothèque Pandas [18] ; les loadeurs de données RDF et d'ontologies utilisent la bibliothèque RDFLib [8]. Un autre avantage de Python est la possibilité d'utiliser sETL dans une configuration interactive avec les Notebooks Jupyter [9], où les utilisateurs peuvent visualiser le graphe de bundles sous forme de diagrammes UML et inspecter les données de n'importe quel bundle, et à n'importe quelle étape de la transformation. Le code source est disponible en ligne [10], ainsi que des exemples de notebooks.

Nous avons développé 2 extracteurs jusqu'à présent : un pour les jeux de données GeoJSON conformes à JSON Schema, et un pour les jeux de données CSV conformes à Table Schema [31].

Nous avons testé avec succès sETL avec 3 jeux de données provenant de plateformes de données ouvertes françaises, chacun étant conforme à un schéma différent : les aménagements cyclables (GeoJSON/JSON Schema), lieux de stationnement (CSV/Table Schema), tous deux provenant de `schema.data.gouv.fr`, et les Infrastructures de Recharge pour Véhicules Électriques (CSV/Table Schema) provenant de `transport.data.gouv.fr`. Les résultats et pipelines correspondants sont disponibles dans le dépôt [11].

8. https://rdflib.readthedocs.io/
9. https://jupyter.org/
10. https://github.com/Sarra-Ouelhadj/SemanticLifting/
11. https://github.com/Sarra-Ouelhadj/SemanticLifting/tree/ic2024/Examples

4 Comparaison à l'état de l'art

De nombreuses approches d'enrichissement sémantique ont été proposées dans la littérature pour construire des graphes de connaissances (GCs). Ces approches peuvent être classées comme automatiques ou semi-automatiques. Les approches semi-automatiques impliquent une intervention humaine au cours d'étapes déterminées du processus de construction de GCS, tandis que les approches automatiques sont entièrement autonomes.

4.1 Approches semi-automatiques

Nous distinguons ici deux catégories : les approches déclaratives et les approches interactives. Dans la première, le mapping est fournie dans un langage déclaratif, tandis que dans la seconde, les utilisateurs construisent un mapping en interagissant avec le modèle de données.

Approches déclaratives

JSON-LD [12] et CSVW [29] permettent d'interpréter les jeux de données JSON et CSV, respectivement, en tant que RDF par le biais de fichiers de contexte. RML [7], qui étend la recommandation R2RML du W3C [6] pour prendre en charge des sources de données (semi-)structurées hétérogènes, expose les données en RDF par l'intermédiaire de mappings exprimés eux-mêmes en RDF. Le Linked data Modeling Language (LinkML) [21] est un framework de modélisation de données orienté objets basé sur une syntaxe de schéma YAML personnalisée inspirée de la structure de diagramme de classe UML. Il permet de schématiser une grande variété de formats de données en entrée, simplifiant ainsi la production de données RDF.

Au cours du processus d'enrichissement sémantique avec des approches déclaratives, des valeurs de données individuelles doivent parfois être transformées afin de s'adapter aux contraintes des ontologies cibles (ex. conversion d'unités, division ou fusion de plusieurs valeurs, etc.) Nous appelons cela une transformation fine. The Function Ontology (FnO) [20] est une description sémantique des fonctions. Elle décrit les paramètres d'une fonction, sa valeur de retour, et le problème qu'elle résout, ce qui permet de réutiliser la fonction. Les descriptions FnO s'intègrent parfaitement dans les représentations des mappings R2RML. La fonction FnO peut être utilisée comme une transformation fine des données ou comme une condition dans RML lors de l'exécution des mappings. SPARQL-Generate [16] étend le langage de requête SPARQL recommandé par le W3C [8] pour transformer des données (semi-)structurées hétérogènes en RDF à l'aide de modèles de graphes. Le mapping et les transformations fines sont exprimés grâce à l'expressivité de SPARQL. D-REPR [30] est un langage de mapping basé sur une syntaxe YAML personnalisée pour transformer des données (semi-)structurées hétérogènes en RDF. Il intègre des transformations fines de données pour le pré-traitement des données par le biais de fonctions personnalisables écrites en Python avec des paramètres codés en dur.

À l'exception de LinkML, aucune de ces approches n'exploite le schéma auquel les données en entrée peuvent se conformer (R1), ni n'est utilisable sans une bonne compréhension des technologies du WS (R2). Elles ne permettent pas non plus de générer des ontologies manquantes (R5). LinkML est différent en ce sens qu'il est basé sur un langage de schéma, qui pourrait en principe être généré à partir de schémas existants. Basé sur les concepts UML et la syntaxe YAML, il ne nécessite pas beaucoup de compétences en WS. Toutefois, les schémas LinkML sont étroitement liés à la structure des données d'origine (contrairement à nos graphes de bundles), ce qui rend difficile pour les praticiens de données de les adapter afin d'expliciter leur sémantique implicite. Par conséquent, LinkML ne satisfait que partiellement R1 et R2.

Approches interactives

OntoRefine [23] et Karma [14] permettent de faire des transformations fines des données de manière interactive à travers une représentation intermédiaire tabulaire des données en entrée. Bien que ce type de représentation intermédiaire soit familier aux praticiens de données qui n'ont pas d'expérience dans le domaine du WS, sa sémantique est pauvre par rapport aux représentations hiérarchiques ou en réseau qui mettent en évidence les concepts et leurs relations (R3), car souvent une seule ligne de structures tabulaires peut représenter plusieurs entités (voir notre exemple fil conducteur). Datalift [26], Csv2rdf4lod-automation [15], UnifiedViews [10], et LinkedPipes ETL [13] forment un autre groupe d'outils de manipulation de données qui convertissent systématiquement les données en entrée en RDF brut, puis appliquent d'autres processus d'enrichissement sémantique. Comme il faut manipuler des données RDF, l'interactivité de ces approches ne profite pas aux praticien des données qui n'ont pas de connaissances en WS (R2). En outre, aucune de ces approches n'exploite le schéma des données (R1), et ne permet de générer de nouvelles ontologies (R5).

4.2 Approches automatiques

Contrairement aux approches présentées ci-dessus, les approches automatiques visent à convertir les données sans aucune intervention humaine. Elles se répartissent en trois catégories : les mappings *hard-codés*, les mappings basés sur des schémas, et les mappings inférés.

Mappings hard-codés

Les outils dont les mappings sont hard-codés ciblent les données conformes à une spécification précise dont la sémantique a été codée en dur dans le système. gtfs-csv2rdf [12] prend en charge les données en entrée conformes à GTFS (General Transit Feed Specification) [13], et les convertit selon une ontologie prédéfinie (Linked GTFS vocabulary) [14] développée spécifiquement à cette fin. Guid-O-Matic [15] convertit les données en entrée conformes au stan-

12. https://github.com/OpenTransport/gtfs-csv2rdf
13. https://gtfs.org/
14. https://github.com/OpenTransport/linked-gtfs/blob/master/spec.md
15. https://github.com/baskaufs/guid-o-matic

dard Darwin Core [16] en RDF.

Mappings basées sur des schémas

Les outils avec mappings basés sur des schémas automatisent la génération de règles de mapping à partir de schémas. MIRROR[19], AutoMap4OBDA[28] et BOOTOX[11] sont des exemples de ces outils qui ciblent les bases de données relationnelles et, à notre connaissance, il n'existe aucune solution prenant en charge d'autres types de schémas largement utilisés.

Mappings inférés

Les solutions avec mappings inférés (également connues sous le nom d'annotation sémantique) automatisent la construction de GCs sans utiliser des règles de mapping. Au lieu de cela, elles infèrent automatiquement la correspondance des données avec un référentiel prédéfini. Par exemple, les travaux concourant au défi SemTab [17] [5, 22, 9] mettent en correspondance des éléments de données tabulaires (*i.e.*, cellules, colonnes, lignes) à des éléments sémantiques (*i.e.*, entités, classes, propriétés) provenant de GCs collaboratifs et généralistes (ex. Wikidata, DBpedia).

Aucune de ces approches automatiques n'implique d'intervention humaine (R2), ni n'est applicable à des structures de données hétérogènes (R3) puisqu'elles se concentrent sur une structure de données spécifique. Elles ne permettent pas non plus de générer de nouvelles ontologies (R5). De plus, les solutions avec des mappings inférés n'exploitent pas le schéma des données (R1). Cependant, les solutions basées sur des schémas satisfont partiellement R1 car elles sont basées sur les schémas sous-jacents des bases de données relationnelles.

4.3 Notre proposition

Nous montrons maintenant comment l'approche que nous proposons, sETL, répond aux 5 exigences présentées dans la Section 1. Premièrement, sETL tire son originalité de l'utilisation des schémas de données dans l'enrichissement sémantique des données (R1). Elle extrait le graphe de bundles initial - la représentation intermédiaire des données dans le système - à partir d'un jeu de données en entrée et de son schéma. Le Modèle Sémantique (MS) de ce graphe initial représente une transcription directe de la sémantique explicite du schéma. Deuxièmement, afin d'affiner la sémantique des données, sETL fournit un ensemble complet de transformeurs que les praticiens de données peuvent appliquer de manière interactive au graphe de bundles. Ces transformeurs garantissent la conformité continue entre le MS et les données de chaque bundle. Étant basée sur des concepts et des technologies d'ingénierie de données bien connus (UML, ETL, Python, Pandas), sETL a une faible barrière à l'entrée pour les praticiens n'ayant pas d'expérience en WS (R2). Troisièmement, le modèle de graphe de bundles est indépendant de tout format de jeux de données en entrée ou de toute spécification de schéma. Par conséquent, le système peut être étendu pour prendre en charge n'importe quelle structure de données en entrée (R3), ce qui convient à la nature hétérogène des données. Quatrièmement, sETL permet la réutilisation d'ontologies existantes en annotant le MS (R4), lorsque des vocabulaires avec la sémantique correspondante ont été identifiés, ou la génération de nouvelles ontologies légères (R5), dans les cas où aucun tel vocabulaire n'est disponible. Après l'initialisation du premier graphe de bundles au cours de la phase *Extract*, les données RDF et l'ontologie peuvent être matérialisées à tout moment. Combiné à l'utilisation interactive de sETL, cela permet un processus d'enrichissement sémantique incrémental, qui peut être utile pour des jeux de données complexes.

5 Conclusions et perspectives

Dans cet article, nous avons proposé la boîte à outils sETL qui relève le défi de l'enrichissement sémantique des données dans le contexte des organisations productrices de données n'ayant pas d'expertise étendue en WS. Cette approche a réussi à répondre aux 5 exigences que nous avons identifiées lors d'un travail antérieur réalisé avec un groupe de producteurs de données de la Métropole de Lyon. sETL permet la spécification d'un workflow complet d'enrichissement sémantique, et la production de données RDF qui en résultent. Elle comprend (1) un Modèle Sémantique étendant les diagrammes UML, pour capturer la sémantique implicite des schémas de données existants, (2) la notion de Bundle, qui groupe les éléments du MS avec leurs données correspondantes, et (3) des opérateurs ETL de haut niveau, qui font abstraction de la plupart des concepts du WS, pour le bénéfice des praticiens de données qui n'ont pas de connaissances approfondies en WS. Nous avons pu appliquer sETL à trois jeux de données complets provenant de plateformes de données ouvertes françaises.

Dans les travaux futurs, nous souhaitons étendre les fonctionnalités de la boîte à outils proposée à différents niveaux. Tout d'abord, nous souhaitons ajouter de nouveaux extracteurs pour d'autres spécifications de schémas et de formats de données (ex. les schémas de bases de données relationnelles), permettant aux utilisateurs de traiter des jeux de données plus variés et complexes. Nous souhaitons également enrichir les transformeurs. En particulier, nous souhaitons tirer parti, dans l'opérateur *reconcile*, des méthodes existantes de mapping inférés (annotation sémantique), pour faciliter la tâche d'alignement des valeurs dans les données tabulaires avec les concepts de GCs ouverts tels que Wikidata. Nous souhaitons également étudier l'ajout d'une interface graphique à sETL afin de faciliter son utilisation par les personnes qui ne maîtrisent pas l'écriture de scripts ou workflow de base sous forme de code. Enfin, nous souhaitons étudier l'intégration de sETL dans des outils existants (ex. des outils de nettoyage de données en amont, et des bases de données orientées graphe en aval) afin d'étendre les workflows existants, et de rendre sETL plus utilisable dans un environnement de production.

16. https://dwc.tdwg.org/
17. SW Challenge on Tabular Data to Knowledge Graph Matching

Références

[1] G. Aldebert and A. Augusti. schema.data.gouv.fr - An Open Data Schema Catalog for France, May 2020.

[2] S. Auer. Semantic integration and interoperability. In *Designing Data Spaces*, chapter 12, pages 195–210. Springer, Cham, 2022.

[3] F. Bariatti. *Mining Tractable Sets of Graph Patterns with the Minimum Description Length Principle*. Theses, Université de Rennes 1, November 2021.

[4] B. Chandrasekaran and et al. What are ontologies, and why do we need them ? *IEEE Intelligent Systems*, 14(01) :20–26, jan 1999.

[5] V. Cutrona and et al. Results of SemTab 2021. In *SemTab 2021*, volume 3103 of *CEUR-WS.org*, October 2021.

[6] S. Das and et al. R2RML : RDB to RDF Mapping Language. Technical report, W3C, 2012.

[7] A. Dimou and M. Vander Sande. RDF Mapping Language (RML). Technical report, RML.io, 2020.

[8] S. Harris and A. Seaborne. SPARQL 1.1 Query Language. Technical report, W3C, 2013.

[9] V-P. Huynh and et al. DAGOBAH : Table and Graph Contexts For Efficient Semantic Annotation Of Tabular Data. In *SemTab 2021*, volume 3103 of *CEUR-WS.org*, October 2021.

[10] K. Janowicz and et al. Unifiedviews : An etl tool for rdf data management. *Semantic Web*, 9(5) :661–676, jan 2018.

[11] E. Jiménez-Ruiz and et al. BootOX : Practical Mapping of RDBs to OWL 2. In *ISWC 2015*, pages 113–132. Springer, 2015.

[12] G. Kellogg and et al. JSON-LD 1.1. Technical report, W3C, July 2020.

[13] J. Klímek and et al. LinkedPipes ETL : Evolved Linked Data Preparation. In *Proc. ESWC 2016*. Springer, 2016.

[14] C. A. Knoblock and et al. Semi-automatically Mapping Structured Sources into the Semantic Web. In *Proc. ESWC 2012*, pages 375–390, Germany, 2012. Springer.

[15] T. Lebo and G.T. Williams. Converting governmental datasets into linked data. In *Proc. I-SEMANTICS 2010*, USA, 2010. ACM.

[16] M. Lefrançois and et al. A SPARQL extension for generating RDF from heterogeneous formats. In *Proc. ESWC 2017*, volume 10249, Slovenia, May 2017. Springer.

[17] Local Government Association. Open data | LG Inform Plus - Schemas, April 2023.

[18] W. McKinney. Data Structures for Statistical Computing in Python. In *Proc. SciPy*, 2010.

[19] L. F. de Medeiros and et al. MIRROR : Automatic R2RML Mapping Generation from Relational Databases. In *Proc. ICWE 2015*, pages 326–343. Springer, 2015.

[20] B. De Meester and et al. Implementation-independent function reuse. *Future Generation Computer Systems*, 110, 2020.

[21] S. Moxon and et al. The Linked Data Modeling Language (LinkML) : A General-Purpose Data Modeling Framework Grounded in Machine-Readable Semantics. In *Proc. ICBO 2021*, volume 3073 of *CEUR-WS.org*, 2021.

[22] P. Nguyen and et al. MTab4DBpedia : Semantic Annotation for Tabular Data with DBpedia. *Semantic Web Journal*, 2022.

[23] OntoText. Ontotext Refine, February 2023.

[24] S. Ouelhadj and et al. Méthode pour enrichir sémantiquement les données en utilisant l'UML annoté. EGC 2023, January 2023. Poster.

[25] M. Page and et al. *Open data maturity report 2023*. Publications Office of the European Union, LU, 2023.

[26] F. Scharffe and et al. Enabling linked data publication with the Datalift platform. In *AAAI Workshop on Semantic Cities*, Canada, July 2012. AAAI.

[27] G. Schreiber and Y. Raimond. RDF 1.1 Primer. W3C Working Group Note, W3C, June 2014.

[28] Á. Sicilia and G. Nemirovski. AutoMap4OBDA : Automated Generation of R2RML Mappings for OBDA. In *EKAW 2016*, page 577–592. Springer, 2016.

[29] J. Tennison. CSV on the Web : A Primer. Technical report, W3C, 2016.

[30] B. Vu and et al. D-repr : A language for describing and mapping diversely-structured data sources to rdf. In *Proc. K-CAP 2019*, USA, 2019. ACM.

[31] P. Walsh and R. Pollock. Table Schema. Technical report, Frictionless Standards, October 2021.

[32] A. Wright and et al. JSON Schema : A Media Type for Describing JSON Documents. Internet Draft 01, IETF, June 2022.