Transformer les Open Data brutes en graphes enrichis en vue d'une intégration dans les systèmes OLAP

Alain Berro[1], Imen Megdiche [2], Olivier Teste [3]

(1) *Manufacture des Tabacs, Université Toulouse I Capitole*
2 rue du Doyen Gabriel Marty, 31042 Toulouse, France

(2) *IRIT, Université Toulouse III Paul Sabatier*
118 route de Narbonne, 31062 Toulouse, France

(3) *IUT de Blagnac, Université Toulouse II Le Mirail*
1 place Georges Brassens, 31703 Blagnac, France

Alain.Berro@irit.fr, Imen.Megdiche@irit.fr, Olivier.Teste@irit.fr

RÉSUMÉ. *L'intégration des Open Data dans les systèmes OLAP est difficile en raison de l'absence de schémas sources, l'aspect brut des données et l'hétérogénéité sémantique et structurelle. La plupart des travaux existants s'intéressent aux Open Data de format RDF qui restent actuellement minoritairement disponibles. En revanche, peu de travaux s'intéressent aux Open Data de format brut, par exemple Excel qui représentent pourtant plus que 90% des données ouvertes disponibles.*
Dans cet article, nous proposons un processus automatique de transformation des Open Data brutes en graphes enrichis exploitables pour l'intégration. Ce processus est validé par l'utilisateur et s'inscrit dans notre démarche d'intégration des Open Data dans les entrepôts de données multidimensionnelles.

ABSTRACT. *The Open Data integration in the decision systems is challenged by the absence of schema, the raw data and the semantic and structural heterogeneousness. In the literature, the most of authors studies the integration of RDF'Open Data in information systems besides the little percentage of available data in this format. On the other hand, few works are interested of Excel'Open Data despite they represent more than 90% of the available data.*

In this paper, we provide an automatic process that transforms raw Open Data in exploitable rich graphs. This process is validated by the users. This is part of our generic approach for integrating the Open Data into multidimensional data warehouse.

MOTS-CLÉS : *Open Data, Entrepôts de données, Classification Conceptuelle, Graphe.*

KEYWORDS: *Open Data, Data warehouse, Conceptual Classification, Graph.*

1. Introduction

Les Open Data sont des données disponibles sous licence libre mises à disposition par des organismes publics tels que les administrations. L'ouverture de ces données a comme principal objectif la réutilisation afin de développer de nouveaux usages et de nouvelles exploitations à ces données. Notre objectif est d'exploiter ces données en les intégrant dans les systèmes décisionnels

En considérant les caractéristiques des Open Data, nous constatons qu'elles sont très riches en informations brutes. Ces informations sont précieuses pour enrichir les processus d'analyse de données (Ravat, *et al*., 2001). Toutefois ces données ont plusieurs problèmes : hétérogénéité de formats, dispersion sur une multitude de sources, métadonnées non standardisées, absence de schémas, hétérogénéité structurelle avec différents niveaux d'agrégation et imperfection des données, hétérogénéité sémantique avec des vocabulaires non communs. Nous étudions en particulier la résolution d'une partie de ces problèmes sur les Open Data de format Excel.

Les Open Data de format Excel disponibles comportent tous les problèmes cités ci-dessus. Ce type de sources correspond à la majorité des Open Data disponibles actuellement ; à titre d'exemple le site data.gouv.fr détient 300 000 fichiers dont 98,72% sont des fichiers au format Excel. Ce format intéresse depuis quelques temps la communauté scientifique (Coletta *et al.*, 2012 ; Seligman *et al.*, 2010). Plusieurs outils industriels (GoogleRefine, 2013; FusionTables, 2013) sont également proposés. Ces propositions sont intéressantes mais aucune d'elles ne traite le problème d'intégration des sources de données dans les systèmes décisionnels.

Nos travaux s'inscrivent dans le cadre des systèmes décisionnels à base d'entrepôts de données chargés de collecter et conserver les données décisionnelles. La construction d'un entrepôt de données repose sur une organisation multidimensionnelle des données entreposées (Teste, 2009). Il s'agit d'une organisation des données en sujets d'analyse appelés **faits** en fonction d'axes d'analyses appelés **dimensions**. Les dimensions sont composées de **paramètres** (ou niveau de dimension) représentant les différents niveaux de granularité possibles des axes d'analyse. Les paramètres sont organisés au sein d'une **hiérarchie**.

Dans ce contexte, notre problématique consiste à rendre les Open Data brutes exploitables dans les systèmes décisionnels par des utilisateurs. L'approche qui répond à notre problématique doit être de type Open Business Intelligence (Open BI) (Schneider *et al.*, 2011 ; Mazon *et al.*,2012) et hybride (Schneider *et al.*,2011). Une approche Open BI exige des mécanismes permettant l'extraction et l'intégration de sources hétérogènes et non structurées par des utilisateurs non-experts en BI. Une approche hybride de conception d'entrepôts est un compromis entre les données des sources et les besoins des utilisateurs pour définir le schéma multidimensionnel de l'entrepôt.

L'approche que nous proposons s'adresse à des utilisateurs chargés de concevoir l'entrepôt de données multidimensionnelles à partir d'une phase ETL semi-

automatisée pour l'intégration des Open Data sources. La définition du schéma multidimensionnel est guidée par des graphes conceptuels. L'utilisation des graphes nous permet d'une part d'assurer une évolutivité de la phase ETL (Schneider *et al.*, 2011), d'autre part une plus grande flexibilité pour l'intégration des données hétérogènes (Schneider *et al.*,2011). La phase ETL de notre approche se distingue par rapport à un outil industriel comme (Talend, 2014) par des algorithmes permettant la détection semi-automatique de données structurelles complexes et hétérogènes. Avec Talend l'utilisateur doit spécifier où se situent les données structurelles dans les fichiers.

Cet article est organisé comme suit. Nous présentons, dans la section 2, notre processus d'intégration des Open Data dans les entrepôts de données multidimensionnelles. La section 3 est dédiée à la présentation de la phase de préparation des données et la phase de définition des schémas des sources en graphes enrichis. La section 4 conclut en dressant les perspectives de ce travail.

2. Un processus d'entreposage des Open Data

Nous proposons dans la Figure 1 notre processus d'intégration des Open Data dans les entrepôts de données multidimensionnelles. Ce processus est composé de quatre phases. Il prend en entrée des sources Open Data brutes et génère en sortie un schéma multidimensionnel.

– **phase 1 : préparation des sources de données.** L'objectif de cette phase est de transformer automatiquement les données brutes en données annotées par leur type et enrichies sémantiquement. Dans cette phase nous détectons les données de type *Valeurs* contenues dans le corps des tableaux et les données de type *Structures* représentant les entêtes lignes et colonnes des tableaux comme le montre la Figure 2. Nous automatisons également la détection des données spatio-temporelles qui sont couramment utilisées dans les systèmes décisionnels. Par la suite, nous réalisons des classifications conceptuelles des données de type *Structures* afin d'enrichir les données. Enfin, l'utilisateur vérifie les détections automatiques pour garantir la validité des annotations.

– **phase 2 : définition des schémas des sources en graphes.** L'objectif de cette phase est de fusionner dans un graphe les données issues de la classification conceptuelle et les détections structurelles et spatio-temporelles. Chaque graphe enrichi obtenu représente le schéma unitaire d'un Open Data source ;

– **phase 3 : intégration des schémas en graphes des sources.** Cette phase prend en entrée un ensemble de graphes enrichis et produit un graphe intégré. Ce dernier doit vérifier un certain nombre de contraintes permettant d'assurer lors de la construction du schéma multidimensionnel, des hiérarchies strictes et couvrantes (Malinowski et Zimanyi; 2006, Hachicha, 2012) ;

– **phase 4 : définition incrémentale et semi-automatique par l'utilisateur des composants multidimensionnels.** A partir du graphe intégré des sources, l'utilisateur définit incrémentalement un schéma multidimensionnel. Ce dernier comporte d'une part les dimensions, les niveaux de dimensions et les hiérarchies,

d'autre part, l'utilisateur définit les mesures d'analyse dans le fait. Enfin, nous alimentons l'entrepôt avec les données de type *Valeurs* indexées par les instances du schéma multidimensionnel.

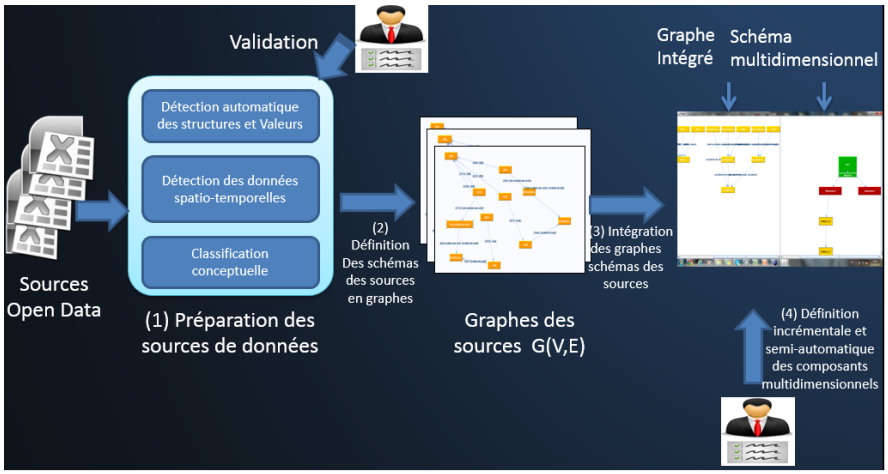


Figure 1. Processus d'entreposage des Open Data

Nous soulignons que la transformation des Open Data brutes en graphes (phase 1 et 2) peut éventuellement converger vers les solutions Linked Open Data (Böhm *et al.*, 2012) en appliquant les principes du LOD sur les graphes.

Dans ce papier, nous nous focalisons sur les phases 1 à 2 de notre processus. Nous présentons en détails dans la section 3 comment nous transformons les données Open Data brutes en graphes unitaires.

3. Transformation des Open Data brutes en graphes enrichis

Les Open Data brutes sont très riches en informations qui pourront faire l'objet d'analyses ou enrichir des analyses. Cependant elles sont difficilement exploitables dans leur format d'origine. Afin de rendre ces données exploitables nous proposons la transformation des Open Data brutes en graphe.

– un Open Data brutes contient des données de type *Valeurs* (cadre jaune) indexées par des données de type *Structures*. Ces dernières sont les concepts des entêtes lignes (cadre bleu vertical) et des entêtes colonnes (cadre bleu horizontal) (Wang, 1996) comme le montre la Figure 2 ;

– un graphe enrichi noté $G = (V, E)$ représente les relations entre les données de type *Structures* et les données de type *Valeurs*. Nous décrivons en détail les graphes enrichis dans la section 3.4.

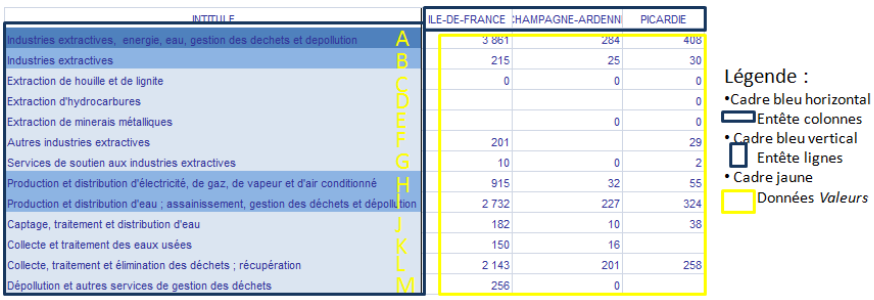


Figure 2. Un exemple d'Open Data annoté avec les types de données

3.1. Extraction des Structures et Valeurs des Open Data brutes

L'extraction des données de type *Structures* et *Valeurs* passe par cinq étapes. Dans la première étape, nous transformons les fichiers sources en matrices d'entiers en fonction du type des données dans les cellules des fichiers. Dans la deuxième étape, nous réalisons un pré-traitement sur les données brutes par exemple pour détecter les données calculées à partir d'autres données. Dans la troisième étape, nous appliquons un algorithme pour la recherche des blocs de données *Valeurs*. Dans la quatrième étape, nous faisons appel à des algorithmes de recherche des entêtes lignes et colonnes en raisonnant sur les blocs de données *Valeurs* trouvés dans l'étape 3. Enfin dans la cinquième étape, nous recherchons les blocs de données *Valeurs* similaires afin de préparer la classification conceptuelle voir section 3.3. Nous détaillons dans ce qui suit chacune des étapes.

3.1.1. Transformation des fichiers sources en matrices

Nous transformons les fichiers Open Data en matrice M d'entiers de taille $nbLigne \times nbColonne$. La matrice représente le type d'information (donnée, formule, etc.) contenu dans chaque cellule. Pour ce faire, le codage qui a été retenu est celui du Tableau 1. Par exemple une cellule qui contient une données de type *date* sera représentée dans la matrice M par l'entier 7. Les types des cellules sont détectés automatiquement. La matrice Mest définie comme suit :

$$M = (a_{i,j})_{1 \le i \le nbLigne, 1 \le j \le nbColonne} \; tel \; que \; a_{i,j} \in \{-1,0,1 \dots 9\}$$

Tableau 1. Les différents types de cellules

Type de cellules	Valeur de $a_{(i,j)}$
Vide	-1
Label	0
Numérique	1
Formule_numérique	2
Formule_label	3
Formule_erreur	4
Booléan	5
Formule_Booléan	6
Date	7
Formule_Date	8
Erreur	9

La transformation matricielle nous permet d'utiliser les types de cellules pour : découvrir automatiquement les blocs de données *Valeurs* (de type de cellules *numérique* codées par «1»), détecter les données *Structures* (de type de cellules *label* codées par «0»), détecter la présence de données temporelles, et détecter les données de type *formule*.

3.1.2. Vérification algorithmique des données de type Formule

La vérification des données de type *Formule* consiste à inspecter algorithmiquement la présence de données calculées à partir d'autres données existantes par des opérateurs tels que: avg, min, max,…. La transformation des Open Data en matrice nous permet de faire une première vérification automatique de la présence de ces dernières. En revanche, cela est insuffisant vu l'imperfection des Open Data. Ce qui nous amène à une deuxième phase de vérification qui consiste à analyser la relation entre les cellules impliquées dans le calcul d'une formule et les opérateurs qui en découlent (sum, avg, max,..).

3.1.3. Recherche des blocs de données de type Valeurs

L'objectif de cette phase est de rechercher dans la matrice *M* les blocs de données numériques, ces blocs représentent les données *Valeurs* des fichiers sources. Ces derniers sont désignés par **BlocNumUnit** (1). Ces blocs constituent des sous matrices de *M* où les indices de début et de fin de lignes et de colonnes de chaque bloc sont notés respectivement comme suit : Ligne Début (**LD**), Ligne Fin (**LF**), Colonne Début (**CD**), Colonne Fin (**CF**). L'algorithme 1 effectue la rechercher de tous les blocs de *Valeurs* contenus dans les sources de données. Le

principe de l'algorithme est de rechercher une succession de lignes qui contiennent des données numériques codées par «1» dans *M* ce qui correspond à la ligne 5 de l'algorithme. La fonction ***LignesDebEtFinBloc*** s'arrête à la première occurrence de données numériques trouvée ce qui garantit un cadrage rapide de la zone de recherche située entre LD et LF. Ainsi, dans ce cadre-là nous cherchons le(s) bloc(s) unitaire(s) numérique(s) qui se trouve(nt) dans cette zone en effectuant une vérification sur le contenu de toutes les colonnes, ce qui correspond aux lignes 8 et 12 de l'algorithme 1 en faisant appel à la fonction ***ColonnesDebEtFinBloc*.**

$$BlocNumUnit(k) = \left(a_{i,j}\right)_{LD_k \le i \le LF_k, CD_k \le j \le CF_k} \quad tel\ que \quad a_{i,j} = 1 \quad (1)$$
$$1 \le LD_k \le LF_k \le nbLigne$$
$$1 \le CD_k \le CF_k \le nbColonne$$

Algorithme 1. Recherche des Blocs de Valeurs

1: ***Recherche Blocs Valeurs***
2: {
3: *listeBlocsNumUnit* ← ∅
4: **Tant que** *cptLigne* < *nbLigne* **faire**
5: *blocNumUnit(i)* ← ***LignesDebEtFinBloc****(cptLigne)*
6: *cptColonne* ← *0*
7: *cptLigne* ← *blocNumUnit(i).LF+1*
8: **Tant que** *cptColonne* < *nbColonne* **faire**
9: *blocNumUnit(i)* ← ***ColonnesDebEtFinBloc****(cptColonne, blocNumUnit)*
10: *cptColonne* ← *blocNumUnit(i).CF +1*
11: *listeBlocsNumUnit* ← *listeBlocsNumUnit* ∪ { *blocNumUnit(i)*}
12: **Fin Tant que**
13: *i* ← *i +1*
14: **Fin Tant que**
15: }

LignesDebEtFinBloc*(cptLigne) est une fonction qui renvoie la ligne de début blocNumUnit.LD et la ligne de fin blocNumUnit.LF du premier bloc de valeurs numériques détecté dans la matrice M à partir de la ligne cptLigne.*

ColonnesDebEtFinBloc*(cptColonne, blocNumUnit) est une fonction qui cherche les sous blocs contenus dans le blocNumUnit entre les lignes blocNumUnit.LD et blocNumUnit.LF et à partir de la colonne cptColonne .Cette fonction renvoie la colonne début blocNumUnit.CD et colonne fin blocTemp.CF de chaque sous bloc trouvé dans le blocNumUnit*

3.1.4. Recherche des données de type Structures

Les données de type *Structures* sont les entêtes lignes et colonnes associées à chaque bloc de *Valeurs*. Nous considérons dans nos algorithmes de recherche le fait qu'un

bloc de *Valeurs* peut être décrit soit par une entête ligne, soit par une entête colonne, soit par les deux.

Nous désignons par LEC l'indice de la ligne de l'entête des colonnes et CEL l'indice de la colonne de l'entête des lignes.

− L'entête des colonnes d'un bloc de *Valeurs k* (noté blocNumUnit(k)) est défini comme suit :

$$EnteteColonnes(k) = (a_{LEC,j})_{blocNumUnit(k).CD \leq j \leq blocNumUnit(k).CF}$$

− L'entête des lignes d'un bloc de *Valeurs k* (noté blocNumUnit(k)) est défini comme suit :

$$EnteteLignes(k) = (a_{i,CEL})_{blocNumUnit(k).LD \leq i \leq blocNumUnit(k).LF}$$

Le principe de l'algorithme que nous utilisons pour identifier le vecteur EnteteColonnes(k) (respectivement EnteteLignes(k)) de chaque bloc de *Valeurs* BlocNumUnit(k) consiste à chercher dans la matrice *M* l'occurrence de la première ligne (respectivement colonne) de 0 (correspondant à des données de type Label) se trouvant au-dessus (respectivement à gauche) du BlocNumUnit(k).

3.1.5. Recherche des blocs similaires

Les blocs similaires sont des BlocNumUnit disjoints ayant soit la même entête des lignes notés BlocSimL comme le montre la définition (2), soit la même entête des colonnes notés BlocSimC comme le montre la définition (3).

$$\begin{gathered} BlocSimC = \bigcup_{1 \leq k \leq nbBlocSim} BlocNumUnit(k) \\ tel\ que \\ CD_l = \min_k\{CD_k\} \\ CF_l = \max_k\{CF_k\} \\ EnteteColonne(k) = EnteteColonne(l) \forall k \neq l, 1 \leq k, l \leq nbBlocSim \end{gathered} \quad (2)$$

$$\begin{gathered} BlocSimL = \bigcup_{1 \leq k \leq nbBlocSim} BlocNumUnit(k) \\ tel\ que \\ LD_l = \min_k\{LD_k\} \\ LF_l = \max_k\{LF_k\} \\ EnteteLignes(k) = EnteteLignes(l) \forall k \neq l, 1 \leq k, l \leq nbBlocSim \end{gathered} \quad (3)$$

3.2. Extraction des données spatio-temporelles

Plusieurs travaux dans la littérature traitent le problème d'extraction des données spatio-temporelles (Plumejeaud, 2011; Noel et Servigne, 2006). Dans la plupart de ces travaux, l'idée est de capturer des données spatio-temporelles évolutives dans le temps. Toutefois, les données spatio-temporelles dans les Open Data ne présentent pas cet aspect. Ce qui nous amène à choisir une solution plus simple qui consiste à définir deux graphes génériques pour les entités spatio-temporelles (qui s'apparentent de manière simplifiée au mécanisme d'ontologie).

Le premier graphe générique est celui des données temporelles, il s'agit d'un graphe orienté couvrant où les nœuds sont les entités temporelles (tel que Année, Mois, Trimestre…) et l'orientation des arcs est définie du niveau le moins détaillé vers le niveau le plus détaillé ; par exemple un mois est contenu dans un trimestre, on définit alors l'orientation Mois → Trimestre. Nous utilisons le graphe temporel défini par (Mansmann et Scholl, 2007, 37). Nous proposons de compléter ce graphe en ajoutant des arcs orientés de chaque niveau inférieur vers les niveaux les plus hauts. L'objectif est de trouver toujours des chemins entre les entités temporelles que nous découvrirons dans les Open Data.

Le deuxième graphe générique est celui des entités spatiales. Nous proposons que le graphe représente le découpage géographique concerné par les Open Data que nous analysons et qu'il soit complet. Le graphe que nous utilisons pour le découpage géographique de la France comporte les entités spatiales suivantes : commune → canton → arrondissement → département → région. Ce type de graphe peut être construit de manière automatique à partir de sources de données de référence décrivant l'espace géographique souhaité ; nous ne présentons pas ce processus de construction dans cet article.

Ces graphes génériques nous permettent d'annoter les données spatio-temporelles en fonction de l'entité à laquelle elles appartiennent. Pour les entités temporelles, nous utilisons les expressions régulières pour identifier les instances de chaque entité. Par exemple l'expression suivante permet de capturer les instances des années.

$$ExpRég(Année) = ([1..2][0..9][0..9][0..9] \cup \{-,/,[a-z]^*,[A-Z]^*\})^*$$

Pour les entités spatiales nous utilisons des listes[1] prédéfinies d'instances ou la base GeoNames[2].

3.3. Classification conceptuelle

Les données de type *Structures* sont souvent aplaties dans les entêtes lignes et colonnes présentes dans les Open Data. A titre d'exemple dans la Figure 2, nous pouvons remarquer différents intitulés de fonction d'intérim regroupés dans la même entête ligne mais représentant différents niveaux conceptuels (ce qui correspond à différentes nuances de couleur bleu). En analysant les données indexées par ces dernières, en particulier les totaux, nous obtenons la classification conceptuelle de la Figure 3. Dans cette section, nous présentons deux stratégies de classification conceptuelle exacte et approximative soumises toutes les deux à des contraintes particulières afin de transformer les données plates de type *Structures* en hiérarchies de concepts. Nous soulignons qu'à l'issue des deux classifications, la validation des utilisateurs est souhaitable vu que l'utilisation de la sémantique et l'imperfection des données des sources peuvent fausser les résultats escomptés.

---

[1] http://www.insee.fr/fr/methodes/nomenclatures/cog/telechargement.asp
[2] http://www.geonames.org/

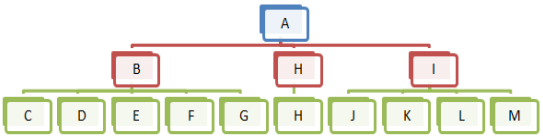


Figure 3. Classification conceptuelle des entêtes lignes

3.3.1. Les hiérarchies complexes dans les systèmes décisionnels

Un problème récurrent dans les systèmes décisionnels est la détection des hiérarchies complexes et leur impact sur les problèmes d'additivité (Mazón *et al.*, 2009). Dans la littérature, ce problème n'était pas considéré dans la phase de définition de schéma des sources (Maiz *et al.*, 2008). Toutefois, nous le trouvons bien étudié soit dans la phase d'intégration (Pedersen *et al.*, 1999) ou en temps réel dans la phase d'analyse (Hachicha, 2012). Vu les différentes difficultés que posent avec ce problème, nous avons choisi de traiter la présence de hiérarchies complexes au plus tôt dans notre démarche d'entreposage des Open Data afin d'éviter les problèmes d'additivité que nous pouvons rencontrer lors de la phase d'analyse.

Les hiérarchies sont complexes quand elles sont non-strictes, non-couvrantes ou non-strictes et non couvrantes. Nous rappelons qu'une hiérarchie est composée de paramètres.

- une hiérarchie est non-stricte (Malinowski et Zimanyi, 2006) si un paramètre fils a plus qu'un parent. Par exemple, un film A fait partie des deux catégories de films « science-fiction » et « tragédie »;
- une hiérarchie est non-couvrante (Malinowski et Zimanyi, 2006) si certains paramètres de la hiérarchie n'ont pas d'instances. Par exemple, dans la hiérarchie « magasin-ville-région-pays » un magasin peut être associé à une région sans être affecté à une ville;
- une hiérarchie non-ontologique ou non-équilibrée est un cas particulier de hiérarchie non-couvrante qui comporte des instances manquantes pour les paramètres de niveau feuille. Par exemple dans la hiérarchie « magasin-ville-région-pays », une ville peut ne pas héberger de magasin.

Dans la section suivante, nous expliquons comment nous interdisons ces types de hiérarchies sous la forme de contraintes imposées au processus de classification conceptuelle.

3.3.2. Les contraintes du processus de classification

Le processus de classification conceptuelle doit vérifier les contraintes C1, C2 et C3. Ce processus prend en entrée des données de type *Structure* (non spatio-temporelles) et produit en sortie des hiérarchies structurelles (englobant schéma et instance) sous format d'arbre. Les contraintes sont les suivantes:

- C1 : Pour chaque feuille *fi* de l'arbre(k), le chemin entre *fi* et la racine de l'arbre(k) est unique. Ce qui signifie que chaque nœud de l'arbre à l'exception

de la racine a exactement un seul parent. Cette condition permet de garantir des hiérarchies strictes à l'échelle du schéma de la source de données.

– C2 : Si un nœud *n* dans l'arbre à l'exception des racines n'a pas de parent ou un parent qui n'a pas de fils, nous dupliquons le nœud *n* dans le niveau manquant. Cette condition permet de garantir des hiérarchies couvrantes au niveau du schéma des sources.

– C3 : La hauteur de l'arbre doit être identique en partant de n'importe quelle feuille vers la racine de l'arbre. Cette condition permet de garantir des hiérarchies ontologiques au niveau du schéma de la source de données.

3.3.3. Classification conceptuelle exacte

Dans plusieurs fichiers Open Data, l'organisation des données pourrait indiquer la classification conceptuelle des données *Structures*. Par exemple, la disposition des blocs de données *Valeurs*, les données de type *Formule*, les cellules fusionnées sont des indicateurs de classification conceptuelle. Nous proposons dans cette section une première stratégie de classification conceptuelle exacte. Pour cette stratégie, nous avons en entrée les données de type *Structures* qui sont les concepts à classifier et en sortie nous générons des arbres (ou hiérarchies) de concepts vérifiant les contraintes C1, C2 et C3. Plusieurs algorithmes sont proposés :

(1) **ClassifConceptEntêtesLignes** est un algorithme qui raisonne sur la disposition des blocs de données *Valeurs* pour classifier les concepts de l'entête des lignes. La Figure 4 décrit à droite l'arbre résultant de l'application de cet algorithme. Ce dernier permet d'affecter les concepts entre les BlocNumUnit(k) au niveau 1. Par exemple pour le premier bloc de *Valeurs* (en jaune) à gauche dans la Figure 4 les deux concepts « Ecole maternelle » et « Ecole élémentaire » sont de niveau 1 ce qui correspond à la partie entre les lignes 6 à 11 de l'algorithme 2. Le concept « Enseignement public » est de niveau 2 et « Premier degré » est de niveau 3 ce qui correspond à la partie entre les lignes 12 à 18 dans l'algorithme 2.

(2) Un deuxième algorithme associé à la section 3.1.2 permet classifier les entêtes lignes ou colonnes qui indexent des lignes ou colonnes de données de type *Formules.* La relation entre les formules se traduit par un arbre entre les concepts de l'entête lignes ou colonnes. La Figure 3 illustre un exemple de cet algorithme, le concept H est dupliqué pour vérifier les contraintes C1, C2 et C3.

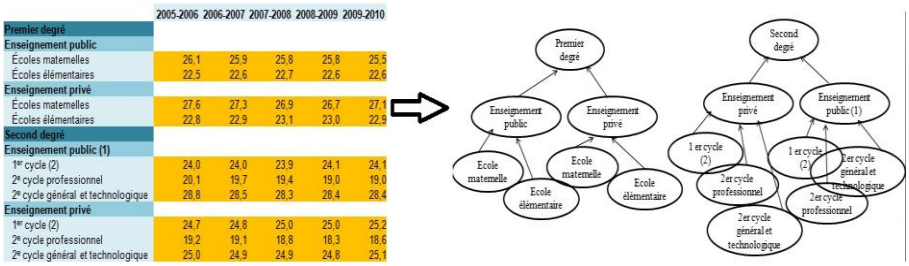
(3) Un algorithme pour la recherche des cellules fusionnées au-dessus des entêtes colonnes (respectivement à gauche des entêtes lignes) qui permet de construire un arbre tel que chaque cellule fusionnée est le parent des concepts fils au-dessous (respectivement à droite) de cette dernière.

Algorithme 2. Classification conceptuelle des concepts de l'entête des lignes

---

1: **ClassifConceptEntêtesLignes(BlocSimL)**
2: {
3: $k \leftarrow 1$

```
4:      sousBlocCrt ← BlocNumUnit(k)
5:      Tant que (i < BlocSimL.LF ET k < nbrBloc ) faire
6:       Si  (i =sousBlocCrt.LD ) alors
7:         Affecter tous les concepts entre  sousBlocCrt.LD et sousBlocCRt.LF à Niv1
8:         k ← k+1
9:         sousBlocCrt ← BlocNumUnit(k)
10:        i ← sousBlocCrt.LF + 1
11:      Fin Si
12:      Si ( i < sousBlocCrt.LD) alors
13:        Compter le nbreConcept  de type labels entre la ligne i et sousBlocCrt.LD
14:        Pour j de i à sousBlocCrt.LD faire
15:          Affecter à chaque concept label le niveau nbreConcept+1
16:          Décrémenter nbreConcept
17:        Fin Pour
18:       Fin Si
19:   Fin Tant que
20:}
```



	2005-2006	2006-2007	2007-2008	2008-2009	2009-2010
Premier degré					
Enseignement public					
Écoles maternelles	26,1	25,9	25,8	25,8	25,5
Écoles élémentaires	22,5	22,6	22,7	22,6	22,6
Enseignement privé					
Écoles maternelles	27,6	27,3	26,9	26,7	27,1
Écoles élémentaires	22,8	22,9	23,1	23,0	22,9
Second degré					
Enseignement public (1)					
1er cycle (2)	24,0	24,0	23,9	24,1	24,1
2e cycle professionnel	20,1	19,7	19,4	19,0	19,0
2e cycle général et technologique	26,8	26,5	26,3	26,4	26,4
Enseignement privé					
1er cycle (2)	24,7	24,8	25,0	25,0	25,2
2e cycle professionnel	19,2	19,1	18,8	18,3	18,6
2e cycle général et technologique	25,0	24,9	24,9	24,8	25,1

Figure 4. Exemple de classification exacte

3.3.4. Classification conceptuelle approximative

Dans la section *3.3.2*, nous avons présenté une approche de classification conceptuelle exacte qui exige la présence d'indicateurs d'organisation. Toutefois, vu l'imperfection des données présentes dans les Open Data, la première approche n'est pas toujours applicable ce qui nous amène à proposer une deuxième approche, plus souple, de classification conceptuelle approximative. Dans l'approche approximative, nous croisons les résultats de classification de la technique des treillis de gallois (Birkho, 1967) avec les résultats de l'approche RELEVANT (Bergamaschi *et al*., 2007) sous les contraintes C1, C2 et C3 pour pouvoir transformer un ensemble de données de type *Structures* qui sont les concepts à classifier en arbre à deux niveaux.

La classification conceptuelle avec les treillis de galois produit des contextes formels à plusieurs attributs. Cette technique ne prend pas en compte les aspects

sémantiques des concepts et les hiérarchies produites ne sont pas strictes. Pour les aspects sémantiques, nous avons fait appel à l'approche RELEVANT qui considère la similarité sémantique pour regrouper un ensemble de concepts en clusters représentés avec les attributs les plus pertinents selon la technique de clustering choisie. L'approche RELEVANT propose deux techniques de clustering : la première technique est hiérarchique, elle produit des clusters disjoints dont les noms sont composés de plusieurs attributs, la deuxième technique basée sur les clusters non-disjoints produits des clusters de nom mono-attribut. Nous avons choisi d'appliquer la technique de clustering hiérarchique afin d'obtenir des clusters disjoints plus proches de la contrainte C1. Nous proposons dans ce qui suit d'illustrer notre approche de classification sur les concepts de l'entête des lignes de la Figure 2 :

1. Préparation des données : Pour l'ensemble des *Concepts = {A ;B; C ; D; E; F; G; H; I; J; K; L; M}*, nous faisons une extraction des racines des mots distincts , nous obtenons l'ensemble *Attributs = {(A1) industr; (A2) extract; (A3) énerg; (A4) gestion; (A5) dépollu; (A6) houill; (A7) lignit; (A8) hydorcarbur; (A9) métalliques; (A10) autr; (T11) servic; (A12) product; (A13) distribu; (A14) électr; (A15) gaz; (A16) condition; (A17) déchet; (A18) captag; (A19) trait; (A20) collect; (A21) usé; (A21) élimin} ;*

2. Sur l'ensemble des *Concepts*, nous appliquons l'algorithme de classification des treillis de galois et nous faisons l'extraction des contextes formels mono-attribut (Berro *et al*., 2013), voir Figure 5 ;

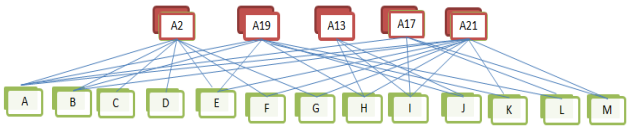


Figure 5. Contextes mono-attribut du treillis

3. Sur l'ensemble des *Concepts*, nous appliquons l'approche RELEVANT avec le clustering hiérarchique. Nous obtenons des clusters disjoints où chaque cluster est représenté par un ensemble d'attributs, voir Figure 6 ;

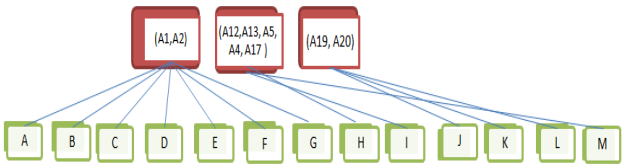


Figure 6. Clusters obtenus par RELEVANT

4. Nous sélectionnons les mono-attributs du treillis qui ont été sélectionnés comme pertinents par RELEVANT, dans notre exemple *AttriPerti = {A2, A13, A17, A19}*. Ensuite, chaque attribut sera lié à l'intersection des concepts entre les deux approches par exemple *A13* sera lié à $\{H, I\} = \{H, I, M\} \cap \{H, I, J\}$ et *A17* sera lié à $\{I, M\} = \{H, I, M\} \cap \{A, I, L, M\}$. Ceci produit des hiérarchies non-strictes, nous résolvons ce problème en

attachant le concept I à l'attribut le plus proche sémantiquement : I est plus proche sémantiquement de A13 que de A17 avec la mesure de Jaccard, voir Figure 7 pour le résultat de classification conceptuelle.

Figure 7. Exemple de classification approximative

En comparant les résultats de classification conceptuelle obtenus par l'approche exacte et l'approche approximative, nous remarquons que nous avons réussi à obtenir un sous ensemble des concepts correctement regroupés tel que les ensembles {J, K, L} ou {C, D, E, F, G}.

3.4. Règles de transformation des Open Data brutes en graphes enrichis

Un graphe enrichi, noté $G = (V, E)$, décrit les différents types de relations entre les données de type *Structures*, les concepts d'enrichissement et les données de type *Valeurs*.

L'ensemble V des sommets est partitionné en quatre sous-ensembles :

- $V_{EnteteLigne,BlocNumUnit(k)}$ l'ensemble des concepts de type *Structures* de l'entête des lignes associé au bloc de *Valeurs* (blocNumUnit (k)) ;
- $V_{EnteteColonne,BlocNumUnit(k)}$ l'ensemble des concepts de type *Structures* de l'entête des colonnes associés à des blocs de *Valeurs* (blocNumUnit(k)) ;
- $V_{StructEnrichi}$ l'ensemble des concepts associés au blocSim, issus de la classification conceptuelle externe ou des cellules fusionnées ;
- $V_{Val,BlocNumUnit(k)}$ l'ensemble des données de type *Valeurs* associées au BlocNumUnit(k).

L'ensemble E des arcs est partitionné en cinq sous-ensembles:

- $E_{StructEnrichi,StructEnrichi}$ l'ensemble des arcs (u, v) représentant les liens entre les concepts d'enrichissement où $u, v \in V_{StructEnrichi}$;
- $E_{structEnrichi,EnteteLignes}$ l'ensemble des arcs (u, v) représentant les liens entre les concepts d'enrichissement et les concepts de l'entêtes des lignes où $u \in V_{StructEnrichi}$ et $v \in V_{EnteteLignes,BlocNumUnit(k)}$;
- $E_{StructEnrichi,EnteteColonnes}$ l'ensemble des arcs (u, v) représentant les liens entre les concepts d'enrichissement et les concepts de l'entêtes de colonnes où $u \in V_{StructEnrichi}$ et $v \in V_{EnteteColonnes,BlocNumUnit(k)}$;
- $E_{EnteteLignes,Val,BlocNumUnit(k)}$l'ensemble des arcs (u, v) représentant les liens entre les concepts de l'entête des lignes et les données de type *Valeurs* où $u \in V_{EnteteLignes,BlocNumUnit(k)}$ et $v \in V_{Val,BlocNumUnit(k)}$;

− $E_{EnteteColonnes,Val,BlocNumUnit(k)}$l'ensemble des arcs (u, v) représentant les liens entre les concepts de l'entête des colonnes et les données de type *Valeurs* où $u \in V_{EnteteColonnes,BlocNumUnit(k)}\ et\ v \in V_{Val,BlocNumUnit(k)}$.

Nous présentons dans ce qui suit les règles de transformations des données extraites des Open Data brutes dans les graphes enrichis.

Règle 1 : Les données de type *Valeurs* extraites dans la section 3.1.3 sont transformées en sommets de type $V_{Val,BlocNumUnit(k)}$.
Règle 2 : Les données de type *Structures* extraites dans la section 3.1.4 sont transformées en sommets de type $V_{EnteteColonne,BlocNumUnit(k)}$ pour les concepts de l'entête des colonnes du BlocNumUnit(k) et des sommets de type $V_{EnteteLigne,BlocNumUnit(k)}$ pour les concepts de l'entête des lignes du BlocNumUnit(k).
Règle 3 : A partir des données spatio-temporelles, obtenues à la section 3.2, nous transformons les entités spatio-temporelles en sommets de type $V_{StructEnrichi}$ et les données spatio-temporelles extraites en $V_{EnteteLigne,BlocNumUnit(k)}$ ou $V_{EnteteColonne,BlocNumUnit(k)}$. Nous relions ces sommets avec des arcs de type $E_{structEnrichi,EnteteLignes}$ ou $E_{StructEnrichi,EnteteColonnes}$.
Règle 4 : A partir des arbres, obtenus à la section 3.3.3 et à la section 3.3.4, les feuilles sont les sommets obtenus à partir de la règle 2. Les autres nœuds des arbres sont transformés en sommets de type $V_{StructEnrichi}$. Nous relions ces concepts en respectant le sens des arcs dans les arbres d'origine avec des arcs de type $E_{structEnrichi,EnteteLignes}$, $E_{StructEnrichi,EnteteColonnes}$ et $E_{StructEnrichi,StructEnrichi}$

4. Conclusion

Les Open Data brutes sont actuellement difficilement exploitables dans les systèmes OLAP vu leur hétérogénéité structurelle et sémantique, leur aspect bruts (par exemple manque de hiérarchies), et les imperfections dans ces données. Nous avons proposé dans cet article un processus permettant de transformer les Open Data brutes en graphes. Les algorithmes de détection sont génériques (s'appliquent sur les fichiers contenant des données numériques et restent valables sur les autres types de fichiers). Toutefois les techniques de classification et les contraintes sous-jacentes ne peuvent pas être complétement automatisées vu l'imperfection des données traitées. Dans nos futurs travaux, nous présentons les résultats d'expérimentations de notre processus ETL sur une grande masse de fichiers Open Data brutes. Nous détaillons également la phase d'intégration des graphes enrichis dans les systèmes décisionnels à base d'entrepôts de données multidimensionnelles.

Bibliographie

Berro A., Megdiche I., Teste O. (2013). Vers l'intégration multidimensionnelle d'Open Data dans les entrepôts de données. *EDA'13*.

Birkho, G. (1967). *Lattice Theory*.

Böhm C, Freitag M, Heise A, et al. (2012). GovWILD: integrating open government data for transparency. *WWW (Companion Volume)*.

Coletta R., Castanier E., Valduriez P., Frisch C., Ngo DH., Bellahsene Z. (2012). Public Data Integration with WebSmatch, *CoRR'12*.

FusionTables. (2013). http://www.google.com/drive/apps.html#fusiontables

GoogleRefine. (2013). http://code.google.com/p/google-refine

Hachicha M. (2012). *Modélisation de hiérarchies complexes dans les entrepôts de données XML et traitement des problèmes d'additivité dans l'analyse en ligne XOLAP*. Thèse en Informatique, Université Lunière Lyon 2.

Maiz N., Boussaid O., Fadila Bentayeb F(2008). Fusion automatique des ontologies par classification hiérarchique pour la conception d'un entrepôt de données. *EGC'08*.

Malinowski E., Zimanyi E. (2006) Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data & Knowledge Engineering*, vol 59, n°2, p 348-377.

Malú C., Florian D., Garrigós I., Mazón J-N. (2013). Business Intelligence and the Web Guest editors' introduction. *Information Systems Frontiers*, vol.15, n° 3, p. 307-309.

Mansmann S., Scholl M.H. (2007). Empowering the OLAP Technology to Support Complex Dimension Hierarchies. *IJDWM*, vol. 3, n° 4. p 31-50.

Bergamaschi S., Sartori C., Guerra F. Mirko Orsini M. (2007). Extracting Relevant Attribute Values for Improved Search. *Internet Computing, IEEE*, vol. 11, n °5, p 26-35.

Mazón, J.-N., Lechtenbörger, J. and Trujillo, J. (2009). A survey on summarizability issues in multidimensional modeling. *Data & Knowledge Engineering*, vol 68, n° 12 , p 1452-1469

Mazon J.N., Zubcoff J.J., Garrigos I., Espinosa R., Rodriguez R. (2012). Open business intelligence: on the importance of data quality awareness in user-friendly data mining. *2nd International workshop on linked web data management, LWDM'12*.

Noel G., Servigne S. (2006). Structuration de données spatio-temporelles temps-réelles : vers la gestion de la saturation de base de données. INFORSID'06.

Pedersen T.B., Jensen C.S., Dyreson C.E (1999). Extending Practical Pre-Aggregation in On Line Analytical Processing. *In 25th International Conference on Very Large Data Bases, VLDB'99*.

Plumejeaud C. (2011) *Modèles et méthodes pour l'information spatio-temporelle évolutive*. Thèse de doctorat. Université de Grenoble.

Ravat F., Teste O., Zurfluh G. (2001). Modélisation multidimensionnelle des systèmes décisionnels. *Extraction et Gestion des Connaissances (EGC'01), Revue des Sciences et Technologies de l'Information, RIA-ECA, Hermès, Vol.1, N°1-2, p.201-212*,

Seligman L., Mork P., Halevy A., Smith K., Carey M.J., Chen K., Wolf C., Madhavan J.,Kannan A. (2010). OpenII: an open source information integration toolkit. *In Int, SIGMOd Conference*.

Schneider M., Vossen G., Zimanyi E. (2011). *Data Warehousing: from occasional OLAP to real-time business intelligence*. Dagstuhl Reports, 1(9), p.1-25.

Talend. (2014). http://fr.talend.com/products/data-integration.

Teste O. (2009). *Modélisation et manipulation des systèmes OLAP : de l'intégration des documents à l'usager*. Habilitation à Diriger des recherches de l'Université Paul Sabatier Toulouse 3.

Wang X. (1996). Tabular abstraction, Editing and formatting. Phd Thesis, University of Waretloo, Waterloo, Ontaria, Canada.