Génération SPARQL interactive de tableaux de données : Exécution guidée par ontologie appliquée aux bases de connaissances à grande échelle

Maheen Bakhtyar[*,**] Fabrice Jouanot[*]
Marie-Christine Rousset[*], Adam Sanchez-Ayte[**]

[*]LIG - Université Grenoble Alpes
maheen.bakhtyar@univ-grenoble-alpes.fr,
fabrice.jouanot@univ-grenoble-alpes.fr
https://www.liglab.fr/
[**]UNESS - Université Numérique en Santé et Sport
adam.sanchez@uness.fr
https://uness.fr

Résumé. Cet article présente une nouvelle approche de construction de requêtes SPARQL qui combine l'exploration d'une base de connaissance sous forme tabulaire avec une ontologie de domaine servant à guider l'utilisateur. Cette approche répond aux défis rencontrés par les scientifiques, non expert en base de connaissance, lors de l'exploration de bases de connaissances complexes et volumineuses. Bien que SPARQL permette des requêtes expressives sur les graphes RDF, leur formulation reste difficile pour un utilisateur. Notre solution préserve la représentation tabulaire, familière en science des données, mais en exploitant une ontologie pour guider la définition des colonnes composant la représentation tabulaire. Le système génère des échantillons de données pertitents, permettant aux utilisateurs d'affiner itérativement leurs requêtes, tout en maintenant la réactivité du système. Nous avons appliqué notre approche sur la base de connaissances d'enseignement en santé OntoSIDES.

1 Introduction

Les technologies du Web sémantique, en particulier les bases de connaissances RDF et leur langage de requête SPARQL, offrent de puissantes capacités pour explorer des données complexes et interconnectées. Bien que SPARQL permette des requêtes expressives sur les graphes RDF, la formulation de telles requêtes reste un défi pour les utilisateurs non experts du Web sémantique. Plusieurs approches ont tenté de répondre à ce défi par le biais de constructeurs de requêtes visuels (Haag et al., 2015), d'interfaces en langage naturel (Garwal et Swathi, 2022), et de construction guidée par les requêtes (Ferré, 2016). Cependant, ces solutions introduisent souvent leur propre complexité à travers des visualisations basées sur les graphes ou exigent des utilisateurs qu'ils comprennent les concepts du Web sémantique.

Ce travail présente une approche différente de la construction de requêtes SPARQL, centrée sur la préservation de la représentation tabulaire des données, familière aux experts du domaine, tout en exploitant les ontologies pour guider la formulation des requêtes. Notre approche s'appuie sur les recherches antérieures en matière d'interfaces de requêtes guidée par ontologie (Francart, 2023) et les étend en abordant les défis clés de la construction de requêtes SPARQL, tels que l'exploration des relations dans les grands graphes et la formulation de requêtes complexes. La solution proposée vise à prendre en charge les fonctionnalités de requête à travers une interface tabulaire intuitive qui révèle progressivement les possibilités de requête pertinentes basées sur l'ontologie sous-jacente.

Notre approche répond au défi du passage à l'échelle en générant dynamiquement des jeux de données échantillons appropriés basés sur le contexte actuel de la requête, permettant aux utilisateurs d'affiner itérativement leurs requêtes tout en maintenant la réactivité du système.

Nous appliquons notre approche à une base de connaissances d'enseignement en santé OntoSIDES (Palombi et al., 2019). Les interfaces SPARQL traditionnelles se sont révélées difficiles pour les chercheurs en éducation, en particulier la combinaison de données à travers plusieurs entités (par exemple, la performance des étudiants à travers différents examens), expression de motifs temporels (par exemple, la progression sur les périodes académiques) Notre approche vise à simplifier l'accès à des modèles de requêtes courants tout en préservant l'expressivité.

Les principales contributions de ce travail comprennent : 1. Une interface tabulaire pour la construction de requêtes SPARQL qui maintient l'expressivité des requêtes tout en offrant une expérience utilisateur familière, 2. Une interaction guidée par une ontologie qui aident les utilisateurs à découvrir les relations pertinentes entre les données, 3. Des méthodes de gestion de la complexité des requêtes dans les bases de connaissances à grande échelle grâce à l'échantillonnage contextuel, 4. Une implémentation et une évaluation utilisant une base de connaissances d'enseignement en santé.

2 Travaux connexes

Notre travail se situe à l'intersection de plusieurs domaines de recherche dans l'accès aux données sémantiques. Nous analysons les approches clés pour rendre les requêtes SPARQL accessibles aux experts du domaine, en mettant l'accent sur la conception d'interfaces et les solutions guidée par ontologie.

2.1 Interfaces de Requêtes Visuelles et Tabulaires

Les interfaces de requêtes visuelles pour SPARQL ont évolué vers un équilibre entre expressivité et facilité d'utilisation. Des systèmes récents comme Sparnatural (Francart, 2023) démontrent comment les interfaces modernes peuvent abstraire la syntaxe SPARQL grâce à des visualisations configurables et guidée par une ontologie. Bien que les approches basées sur les graphes dominent les outils du Web sémantique, les utilisateurs restent attachés aux interfaces tabulaires dans les tâches d'exploration de données, particulièrement parmi les analystes ayant une expérience des logiciels statistiques. Cependant, l'application des paradigmes tabulaires aux requêtes sémantiques reste limitée, la plupart des efforts étant concentrés sur le nettoyage des données plutôt que sur la construction de requêtes.

2.2 Approches guidée par Ontologie

Le rôle des ontologies dans les interfaces de requêtes s'est révélé crucial pour accompagner les experts du domaine. Des systèmes comme Sparklis (Ferré, 2016) démontrent comment les approches interactives peuvent exploiter les connaissances ontologiques pour guider la construction des requêtes tout en préservant l'expressivité de SPARQL. En pratique, les experts du domaine s'appuient généralement sur un ensemble limité de modèles de requêtes qu'ils adaptent et combinent, suggérant ainsi l'intérêt d'un guidage piloté par l'ontologie. Cependant, des défis majeurs subsistent dans l'équilibre entre la complexité de l'ontologie et l'utilisabilité, particulièrement lors du travail avec des bases de connaissances à grande échelle.

Les approches actuelles font face à trois limitations principales : l'intégration de l'ontologie avec les interfaces tabulaires, le passage à l'échelle de la construction interactive de requêtes tout en maintenant la réactivité, et le support de requêtes analytiques complexes sans nécessiter d'expertise SPARQL. Notre travail aborde ces défis à travers une nouvelle combinaison de modèles d'interaction tabulaire et de guidage des requêtes guidé par ontologie, détaillée dans les sections suivantes.

3 Conception et Implémentation du Système

Notre framework répond aux défis identifiés dans la Section 2. Bien qu'instancié sur l'ontologie OntoSIDES pour l'enseignement en santé, sa conception générique permet une adaptation à d'autres domaines et bases de connaissances.

3.1 Architecture du Système

L'architecture du framework proposé tend à rendre les données sémantiques accessibles aux utilisateurs non experts. L'architecture intègre OntoSIDES en l'augmentant des propriétés inverses générées automatiquement (par exemple, *teaches* $\rightarrow$ *is_taught_by*), permettant une navigation complète du graphe. Ces propriétés inverses assurent la connectivité de n'importe quel nœud vers son nœud d'origine, ce qui facilite l'exploration complète du graphe de connaissances.

L'architecture implémente une Interface Tabulaire comme couche d'interaction utilisateur. Cette interface communique avec le Constructeur de Requêtes SPARQL, qui transforme les sélections des utilisateurs en requêtes exécutables. Le système gère la complexité des données RDF en générant automatiquement des requêtes SPARQL appropriées. De plus, le composant serveur web permet aux utilisateurs de conserver leurs résultats d'exploration pour des analyses futures.

3.2 Interface de Requêtes Guidée par Ontologie

La correspondance entre les représentations tabulaires et graphiques suit une structure formelle : 1. Chaque colonne représente une classe RDF ou une propriété de type de données 2. Les relations entre colonnes correspondent aux propriétés de l'ontologie ou aux propriétés inverses 3. Les sélections des utilisateurs créent des chemins dans le graphe RDF sous-jacent

4. Les chaînes de propriétés entre les colonnes forment la base de la génération des requêtes SPARQL.

Les relations cycliques dans le graphe sont gérées en maintenant un ensemble de nœuds visités pendant le parcours, empêchant les boucles infinies tout en permettant des chemins circulaires contrôlés lorsqu'ils sont explicitement demandés.

Le processus de construction de requêtes est illustré à travers une séquence d'exploration en trois étapes d'exploration, voir Fig. 1. L'utilisateur désire les étudiants qui ont travaillés sur des questions dont la spécialité est renseignée. Initialement, comme montré dans la Figure 1a, le nœud "étudiant" et son réseau de connexions sont affichés dans la base de connaissances, tandis que la Figure 1b démontre comment la sélection d'"étudiant" peuple automatiquement la deuxième colonne avec les connexions directes et indirectes. Dans la deuxième étape, la Figure 1c révèle le chemin d'"étudiant" à "question" à travers des nœuds intermédiaires, et la Figure 1d montre comment la sélection de "question" peuple la troisième colonne avec ses nœuds connectés, simplifiant le chemin complexe montré dans le graphe. Enfin, la Figure 1e affiche le chemin complet d'"étudiant" à "spécialité" dans la base de connaissances, et la Figure 1f présente la vue tabulaire finale après la sélection de "spécialité", complétant la séquence d'exploration. Cette progression démontre comment le système exploite des relations dans l'ontologie en une interface tabulaire intuitive tout en préservant les connexions sémantiques entre les entités.

3.3 Génération et Exécution des Requêtes

Lorsque les utilisateurs effectuent des sélections dans la liste déroulante à chaque étape, deux actions sont déclenchées : premièrement, une nouvelle colonne est créée et remplie avec les nœuds de la liste déroulante suivante en fonction du nœud sélectionné. Deuxièmement, une requête est automatiquement générée et exécutée en arrière-plan pour récupérer les données pertinentes basées sur la sélection actuelle. Le système emploie l'Algorithme 1 pour remplir les colonnes suivantes en fonction de la classe actuellement sélectionnée. Le graphe est parcouru selon la méthode de recherche en largeur avec une valeur seuil de 2, ce qui signifie qu'il progressera de deux niveaux à partir du nœud actuel.

Deux types de requêtes SPARQL sont générés :

— Requêtes d'échantillonnage aléatoire : Utilisent une combinaison de la clause `LIMIT` et de la fonction `RAND()` pour récupérer de petits sous-ensembles aléatoires de données. L'échantillonnage assure la diversité en utilisant une sélection aléatoire de type SQL avant d'appliquer les limites, réduisant ainsi le biais vers un sous-ensemble particulier des données.
— Requêtes de jeu de données complet : Récupèrent l'ensemble complet des résultats, exécutées uniquement à la demande de l'utilisateur.

Ce système performance vitesse et exhaustivité, permettant aux utilisateurs d'explorer rapidement les données tout en maintenant l'accès aux jeux de données complets.

3.4 Composant de Visualisation

Le système inclut une fonctionnalité de visualisation graphique implémentée avec `Cytoscape.js`[1], offrant des aperçus plus approfondis des relations entre les nœuds. La visualisation génère dynamiquement deux types de graphes : 1. Des graphes de connexions qui visualisent toutes les connexions possibles vers et depuis le nœud sélectionné, révélant à la fois les liens directs et de second niveau pour fournir une vue d'ensemble complète du contexte du nœud. 2. Des graphes de chemins de requête qui cartographient visuellement le processus de construction de requêtes, montrant les nœuds sélectionnés et leurs relations.

En exploitant la technologie AJAX, le système peut demander et recevoir des données du serveur sans recharger la page, assurant ainsi une expérience utilisateur fluide et réactive.

4 Évaluation

Nous évaluons notre approche sur la base de connaissances RDF OntoSIDES qui combine les entrainements et les évaluations des étudiants en médecine de la plateforme SIDES entre 2012 et 2021 pour toutes les facultés de médecine françaises, enrichis de ressources pédagogiques, des liens vers les référentiels nationaux comme internationaux. OntoSIDES joue le rôle d'un lac de données pour les scientifiques et les chercheurs en éducation. OntoSIDES est construit sur la base d'une ontologie légère composée de 52 classes, 50 propriétés et 10 règles. La base de connaissances comprend plus de 12 milliards de triplets décrivant les activités réalisées par plus de 145 000 étudiants.

OntoSIDES est stocké dans un Triplestore Virtuoso 07.20.3232 avec 32 cœurs, utilisant un processeur Intel Xeon(R) Gold 6144 @ 3.50GHz, 566 Go de RAM, et sept disques de 2 téraoctets pour les tâches d'E/S parallèles.

Des requêtes de complexité croissantes sont disponibles sur un dépot git[2] qui décrit la construction des requêtes, les requêtes SPARQL résultantes, et des mesures de performance.

Bien que le système soit conçu pour gérer la génération de prévisualisations et l'exécution complète des requêtes dans des délais raisonnables, les premiers retours des utilisateurs indiquent que des temps de réponse plus rapides—en particulier pour les opérations de découverte de chemins—pourraient encourager des sessions d'exploration plus longues et plus riches. Une analyse préliminaire suggère que la réduction du temps de parcours de graphes et du temps de rendu des visualisations pourrait potentiellement améliorer l'engagement des utilisateurs. De plus, il est pertinent d'examiner la relation entre le temps d'exécution des requêtes SPARQL et les seuils de patience des utilisateurs. Nous avons identifiés des métriques d'évaluation dérivées de notre diagramme de séquence UML[2] avec contraintes temporelles pour fournir une base exploratoire de la réactivité du système et des schémas d'engagement des utilisateurs. Ces considérations soulignent l'importance d'optimiser des composants critiques du système comme GraphTraversal et SPARQLExecutor[2].

1. `https://js.cytoscape.org/`
2. `https://gricad-gitlab.univ-grenoble-alpes.fr/bakhtyam/Tabular-Query-Builder`

5 Conclusion

Ce travail présente une nouvelle approche de génération de requêtes SPARQL, combinant à la construction d'interfaces tabulaires guidé par une ontologie pour assurer une exploration intuitive des bases de connaissances RDF complexes. Le système, démontré sur la base de connaissances OntoSIDES, équilibre efficacement l'utilisabilité et l'expressivité des requêtes tout en répondant aux défis clés de la gestion des données à grande échelle.

Des expérimentations avec une cohorte d'utilisateurs est en cours de définition. Les tavaux futurs s'orientent vers l'intégration des grands modèles de langage (LLM) avec ce framework interactif. Cette synergie offrirait aux utilisateurs à la fois la flexibilité des LLM et la précision des interfaces guidées par l'ontologie, garantissant ainsi un meilleur contrôle sur les résultats des requêtes.

Références

Ferré, S. (2016). Sparklis : An expressive query builder for sparql endpoints with guidance in natural language. *Semantic Web 8*, 405–418, doi: 10.3233/SW-150208.

Francart, T. (2023). Sparnatural : A visual knowledge graph exploration tool. In *The Semantic Web : ESWC 2023 Satellite Events : Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings*, Berlin, Heidelberg, pp. 11–15. Springer-Verlag, doi: 10.1007/978-3-031-43458-7_2.

Garwal, A. et B. e. a. Swathi (2022). Natural language query formalization to sparql for querying knowledge bases using rasa. *Journal of Prog Artif Intell 11*, 193–206.

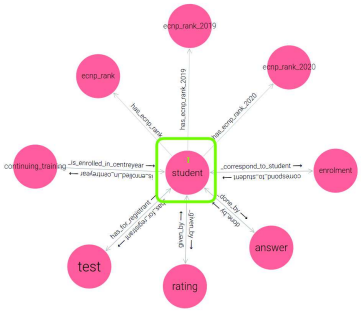
Haag, F., S. Lohmann, S. Siek, et T. Ertl (2015). Queryvowl : A visual query notation for linked data. In F. Gandon, C. Guéret, S. Villata, J. Breslin, C. Faron-Zucker, et A. Zimmermann (Eds.), *The Semantic Web : ESWC 2015 Satellite Events*, Cham, pp. 387–402. Springer International Publishing.

Palombi, O., F. Jouanot, N. Nziengam, B. Omidvar-Tehrani, M. Rousset, et A. Sanchez (2019). Ontosides : Ontology-based student progress monitoring on the national evaluation system of french medical schools. *Artif. Intell. Medicine 96*, 59–67, doi: 10.1016/j.artmed.2019.03.006.

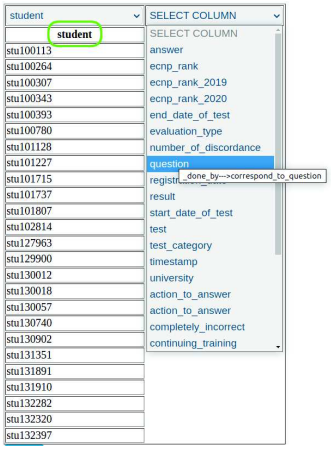
Summary

This article introduces a novel approach to SPARQL query construction, combining the exploration of a knowledge base in a tabular format with a domain ontology to guide the user. This approach addresses challenges faced by scientists who are not experts in knowledge bases when exploring complex and large-scale datasets. While SPARQL allows expressive queries over RDF graphs, formulating them remains difficult for users. Our solution preserves the tabular representation familiar in data science while leveraging an ontology to guide the definition of columns within the tabular view. The system generates relevant data samples, enabling users to iteratively refine their queries while, maintaining system responsiveness. We applied our approach to the OntoSIDES health education knowledge base.

Étape 1 : Sélection initiale de la classe "étudiant" montrant la structure du graphe et la vue tabulaire correspondante

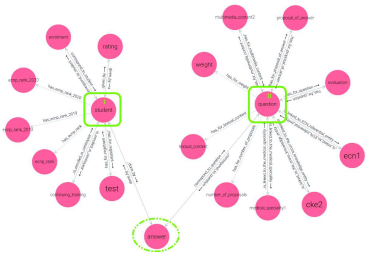


(a) Le nœud "étudiant" et son réseau de connexions dans la base de connaissances.

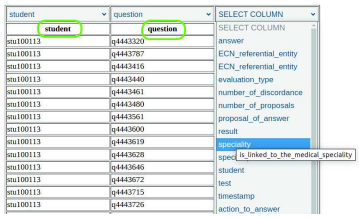


(b) La sélection d'"étudiant" remplit la deuxième colonne avec les connexions directes et indirectes.

Étape 2 : Sélection du nœud "question" montrant la représentation simplifiée du chemin

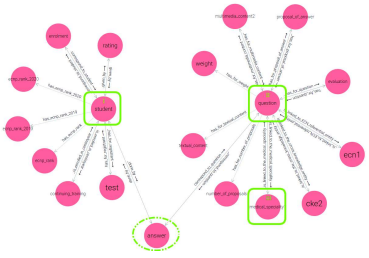


(c) Chemin d'"étudiant" à "question" à travers les connexions intermédiaires.

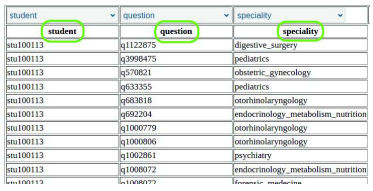


(d) La sélection de "question" révèle ses nœuds connectés dans la troisième colonne.

Étape 3 : Chemin d'exploration complet avec la sélection finale de "spécialité"



(e) Chemin complet d'"étudiant" à "spécialité" dans la base de connaissances.



(f) Vue tabulaire finale après la sélection du nœud "spécialité".

FIG. 1 – *Séquence en trois étapes montrant la transformation des relations du graphe en un tableau.*

Algorithme 1 Parcours d'un graphe RDF en utilisant le "Breadth-First Search"

Entrée :
$start_node$: La classe/nœud de départ pour le parcours.
$threshold_value$: La distance maximale (nombre d'arêtes) depuis le nœud de départ que nous voulons parcourir. C'est la valeur seuil.
$graph$: Le graphe RDF est représenté comme un dictionnaire, où les clés sont les nœuds/-classes et les valeurs sont des listes de paires représentant les propriétés et leurs nœuds connectés, qui peuvent être soit des classes soit des types de données. Pour chaque paire, le premier élément est le nom de la propriété et le second élément est le nœud/classe/type de données auquel il se connecte.
Sortie :
$result$: Une liste de tuples $(property\ path,\ class/datatype,\ level\ in\ the\ graph)$, où chemin est une liste des noms de propriétés du nœud de départ au nœud courant, classe/type de données concerne le nœud courant, et niveau est le niveau de profondeur du nœud courant.
$visited$: Un ensemble de toutes les *classes/datatypes* dans les chemins.
Objectif : Parcourir un graphe RDF en utilisant le BFS et retourner la liste des tuples $(property\ path,\ class/datatype,\ level\ in\ the\ graph)$ et l'ensemble de toutes les classes/-types de données dans le chemin, étant donné un nœud de départ, une valeur seuil et un graphe RDF.

Initialiser un ensemble appelé $visited$ pour suivre les nœuds visités.
Initialiser une liste appelée $queue$ pour implémenter le BFS.
Initialiser une liste appelée $result$ pour stocker les tuples de $(property\ path,\ class/datatype,\ level\ in\ the\ graph)$.
Ajouter le $start_node$ à la $queue$ et le marquer comme visité en l'ajoutant à l'ensemble $visited$.
tant que la $queue$ n'est pas vide **faire**
    Retirer le prochain nœud de la $queue$.
    si le niveau du nœud est inférieur ou égal à la $threshold_value$ **alors**
        Ajouter le nœud à la liste $result$ avec son chemin et son niveau dans le graphe.
    fin si
    pour chaque classe voisine ou littéral de données adjacent n au nœud courant **faire**
        si la propriété entre le nœud courant et n est une propriété de type de données **alors**
            définir n = un nom proxy pour la propriété de type de données.
        fin si
        si n n'a pas été visité avec la propriété courante **alors**
            Ajouter n à la $queue$ et le marquer comme visité en l'ajoutant à l'ensemble $visited$.
            Ajouter le nom de la propriété connectant le nœud courant à n au chemin.
            Ajouter n, le niveau incrémenté et le chemin mis à jour au $result$.
        fin si
    fin pour
fin tant que
retourne $result, visited$