Rapport de Projet : Knowledge Graphs et Raisonnement (NBA)

Bases de Données Spécialisées

Binôme : Chris ESSOMBA, Yvan BANFOU

Lien du dépôt Git du projet : *https://github.com/ChrisEssomba/NBA-Knowledge-Graph-Project*

1. Introduction et objectif du projet

Ce projet s'inscrit dans le cadre du cours Bases de Données Spécialisées et a pour objectif de construire un *knowledge graph* à partir de jeux de données hétérogènes liés à la NBA, de les intégrer sous forme RDF, de les interroger via SPARQL et d'enrichir les connaissances grâce au raisonnement RDFS. L'enjeu principal est de montrer comment des données initialement tabulaires (CSV) peuvent être transformées en un graphe de connaissances cohérent, interopérable et exploitable pour des requêtes avancées.

2. Jeux de Données et Préparation (Sections 0 & 1 du Notebook)

2.1 Objectif

Sélectionner des sources de données pertinentes liées à la NBA et les préparer afin de permettre leur intégration cohérente dans un graphe de connaissances RDF.

2.2 Méthodes

Deux jeux de données libres provenant de la plateforme Kaggle ont été utilisés :

1. **NBA Games Data** : ce jeu de données contient les résultats des matchs, les dates, ainsi que les équipes à domicile et à l'extérieur.[1]
2. **NBA Database** : il regroupe les profils des joueurs, incluant notamment le nom, la taille, l'université fréquentée et les informations de draft.[2]

Le prétraitement a été réalisé en Python à l'aide de la bibliothèque Pandas. Les principales étapes ont consisté à :

-filtrer les colonnes pertinentes (par exemple GAME_DATE_EST, PTS_home, draft_year),

- nettoyer les valeurs manquantes à l'aide de dropna et convertir les types de données (dates, booléens pour HOME_TEAM_WINS),

- extraire et unifier les entités *équipes* (teams) à partir des fichiers joueurs et matchs afin d'assurer la cohérence des identifiants team_id entre les différentes sources.

2.3 Résultats

À l'issue de cette phase de préparation, trois DataFrames propres ont été obtenus : games, players et teams. Cette étape a permis d'identifier 61équipes uniques, ainsi que de préparer plus de 3 000 joueurs et 26 000 matchs, constituant une base fiable et cohérente pour la conversion ultérieure des données en RDF.

3. Conversion de CSV vers RDF (Section 2)

3.1. Objectif

Transformer les données tabulaires en triplets RDF exploitables dans un *triple store*. Pour ce faire, nous nous sommes servis de la documentation da la librairie RDFlib [3].

3.2. Méthodes

Chaque entité est associée à une URI unique dans le namespace nba.

Les correspondances principales sont :

- lignes CSV → ressources RDF,

- colonnes → propriétés RDF,

- valeurs → littéraux ou URI.

Les triplets sont générés automatiquement et stockés dans un graphe RDF unique.

3.3. Résultats

On obtient un graphe RDF décrivant explicitement :

- les joueurs et leurs caractéristiques,

- les équipes et leurs villes,

- les matchs et leurs résultats.

Ce graphe constitue la base de toutes les requêtes SPARQL ultérieures.

4. Requêtes SPARQL (Section 3)

4.1. Objectif

Exploiter le graphe RDF pour répondre à des questions analytiques sur les données NBA. Pour pouvoir apprendre à réaliser des requêtes SPARQL complexes, nous nous sommes servis du cours et d'un article de Wikidata sur le sujet. [4]

4.2. Méthodes et Résultats

1. Requête Fédérée (DBpedia) : Requête de liaison avec le web des données pour récupérer les pages Wikipédia des équipes.

- Résultat : Une liste de ressources externes (url de pages Wikipédia d'équipes) associées à des ressources internes à notre schéma RDF (nom des équipes et leurs nombres de points obtenus à la saison 2018).

2. Filtrage Complexe : La requête 2 récupère les joueurs évoluant au poste de "Center".

- Résultat : Des joueurs comme Kareem Abdul-Jabbar (Lakers) ou Steven Adams (Grizzlies) ont été identifiés.

3. Négation et Chaînes de Caractères : La requête 3 liste les joueurs n'ayant pas étudié dans une école contenant "Chicago".

4. Agrégation : La requête 5 affiches les équipes ayant jouées 100 matchs ou plus lors d'une saison donnée

- Résultat : Les Cavaliers apparaissent avec 109 matchs joués (saison 2017), démontrant la capacité à effectuer des analyses statistiques via SPARQL.

5. Propriétés Booléennes : La requête 6 identifie les matchs remportés par l'équipe jouant à domicile en exploitant la propriété booléenne nba:homeTeamWins.

- Résultat : Sur la saison analysée, environ 58 % des matchs ont été gagnés par l'équipe à domicile, illustrant la prise en compte de types de données booléens dans les requêtes SPARQL.

6. Agrégation Avancée avec UNION : La requête 7 réalise une analyse statistique approfondie des performances des équipes sur une saison, distinguant domicile et extérieur via une clause UNION, et calculant victoires, points marqués/encaissés, taux de victoire et différentiel moyen.

- Résultat : Les Golden State Warriors présentent le meilleur différentiel moyen de points (+11,2) sur la saison étudiée, confirmant la puissance des fonctions d'agrégation (SUM, AVG, COUNT) combinées à HAVING.

7. Expressions de Chemin : La requête 8 identifie l'ensemble des équipes ayant participé à au moins un match, en utilisant l'expression de chemin (nba:homeTeam | nba:visitorTeam) pour capturer indifféremment le rôle de l'équipe.

- Résultat : 61 équipes distinctes ont été retrouvées, correspondant à la totalité des franchises NBA actives, démontrant l'efficacité et la concision des property paths.

8. Graphes Nommés : La requête 9 retourne la liste des graphes nommés présents dans le store (base et inféré).

- Résultat : Deux graphes ont été identifiés : urn:graph:base (données brutes) et urn:graph:inf (données enrichies par inférences RDFS), illustrant la séparation logique entre faits explicites et connaissances déduites.

9. Comparaison Avant/Après Raisonnement : La requête 10 mesure l'impact du raisonnement RDFS en comparant le nombre de triplets utilisant la propriété nba:involvesTeam dans les graphes de base et inféré.

- Résultat : Cette propriété est absente dans le graphe de base (0 triplet), mais apparaît massivement dans le graphe inféré (plus de 26 000 triplets), prouvant que les relations involvesTeam ont été automatiquement déduites à partir de homeTeam et visitorTeam grâce à l'héritage subPropertyOf.

Ces requêtes démontrent la capacité du modèle RDF à répondre à des questions complexes à partir de relations explicites.

5. Raisonnement sur les données RDF (Section 4)

5.1. Construction de l'ontologie RDFS

5.1.1. Objectif

Structurer le graphe via un schéma RDFS afin de permettre des inférences automatiques.

5.1.2. Méthodes

Le fichier schema_rdfs_complet.ttl définit :

- une hiérarchie de classes (ex. nba:Game rdfs:subClassOf nba:Event),

- une hiérarchie de propriétés (rdfs:subPropertyOf),

- des domaines et co-domaines (rdfs:domain, rdfs:range).

Cette ontologie unifie les concepts du domaine NBA et clarifie les relations entre entités.

5.2. Raisonnement avec RDFS

5.2.1. Objectif

Enrichir le graphe RDF en déduisant de nouveaux triplets implicites. Nous nous sommes inspirés de la documentation de la librairie OWLRL[5] et de l'article de oxfordsemantic.[6].

5.2.2. Méthodes

Le raisonnement est effectué à l'aide de la librairie owlrl qui est une alternative à Jena et que nous avons choisis afin d'apprendre à réaliser cette tache sans avoir recours à Java.

5.2.3. Résultats

Après application du raisonnement RDFS, plusieurs enrichissements concrets du graphe sont observés.

Tout d'abord, certaines ressources sont automatiquement typées dans des classes plus générales grâce à la hiérarchie de classes définie dans le schéma. Par exemple, les ressources de type nba:Game sont inférées comme appartenant également à la classe nba:Event via la relation rdfs:subClassOf, sans qu'il soit nécessaire d'ajouter explicitement ce type dans les données initiales.

Ensuite, des relations implicites apparaissent par héritage de propriétés. Lorsqu'une propriété spécifique utilisée dans les données est déclarée comme sous-propriété d'une relation plus générale dans le schéma, les requêtes portant sur cette propriété générale retournent davantage de résultats après inférence. Cela permet, par exemple, d'interroger de manière uniforme des relations similaires sans dépendre de leur nom exact dans les données brutes.

Enfin, le nombre total de triplets augmente après raisonnement (passant de 243669 à 445121), ce qui confirme l'enrichissement automatique du graphe. Cette augmentation se traduit directement dans les résultats des requêtes SPARQL : certaines requêtes exécutées sur le graphe enrichi retournent plus d'entités ou des classifications plus complètes qu'avec les seules données explicites, démontrant l'apport effectif du raisonnement RDFS.

6. Conclusion : Réponse à la question de la section 5

Les requêtes SPARQL combinées au raisonnement RDFS permettent une exploitation plus efficace et plus riche des données NBA intégrées. Par exemple, les requêtes d'agrégation (comptage du nombre de matchs joués par équipe en 2020) ou d'exclusion (équipes dont la ville n'est ni *Utah* ni *New Orleans*) montrent comment SPARQL permet d'exprimer simplement des questions analytiques complexes sur plusieurs entités du graphe.

Le raisonnement RDFS enrichit ces résultats en révélant des connaissances implicites issues du schéma. Ainsi, grâce à la hiérarchie de classes (par exemple nba:Game

rdfs:subClassOf nba:Event), les matchs sont automatiquement reconnus comme des événements sans avoir à expliciter ce type pour chaque ressource. De même, la hiérarchie de propriétés permet d'hériter de relations générales lors des requêtes, ce qui augmente le nombre de résultats pertinents après inférence.

Cependant, le raisonnement RDFS reste limité à des inférences simples (héritage de classes et de propriétés) et ne permet pas d'exprimer des règles plus complexes, comme des conditions dépendant de valeurs numériques ou temporelles (ex. déterminer automatiquement le vainqueur d'une saison). Des améliorations possibles incluent l'utilisation d'OWL pour définir des contraintes plus riches, l'intégration de sources externes comme DBpedia (villes ou franchises), ou encore l'extension du schéma à des concepts plus avancés tels que les statistiques de carrière des joueurs.

En conclusion, ce projet illustre concrètement comment l'association de RDF, SPARQL et RDFS permet d'intégrer, structurer et analyser des données hétérogènes, tout en démontrant l'apport réel du raisonnement sémantique pour enrichir et mieux exploiter un graphe de connaissances.

7. Comment Exécuter le code ?

#1 Dans un nouveau dossier, cloner le repository du projet :

$ Git clone https://github.com/ChrisEssomba/NBA-Knowledge-Graph-Project

2 Créer un environnement virtuel et active le

$ python -m venv .venv

$ .venv\Scripts\activate

#3 Une fois l'environnement activé, installez y toutes les dépendances du projet

$ cd NBA-Knowledge-Graph-Project

$ pip install -r requirements.txt

#4 Exécutez les toutes les cellules du notebook, section après section.

8. Références bibliographiques

[1] « NBA games data ». Consulté le: 31 décembre 2025. [En ligne]. Disponible sur: https://www.kaggle.com/datasets/nathanlauga/nba-games

[2] « NBA Database ». Consulté le: 31 décembre 2025. [En ligne]. Disponible sur: https://www.kaggle.com/datasets/wyattowalsh/basketball

[3] « Overview - RDFLib ». Consulté le: 31 décembre 2025. [En ligne]. Disponible sur: https://rdflib.readthedocs.io/en/stable/

[4] « Wikidata:Tutoriel SPARQL - Wikidata ». Consulté le: 31 décembre 2025. [En ligne]. Disponible sur: https://www.wikidata.org/wiki/Wikidata:SPARQL_tutorial/fr

[5] *owlrl: A simple implementation of the OWL2 RL Profile, as well as a basic RDFS inference, on top of RDFLib. Based mechanical forward chaining.* Python.

[6] « 6. Reasoning in RDFox — RDFox documentation ». Consulté le: 31 décembre 2025. [En ligne]. Disponible sur: https://docs.oxfordsemantic.tech/5.6/reasoning.html