# Créer un Knowledge Graph à partir d'un wiki

Rapport Web Sémantique

**Fait par :**
BENJELLOUN Reda
BOUGASSAA Ali

Date : 21 janvier 2025

# Table des matières

**1 Introduction** **2**

**2 Extraction de l'infobox et génération du graphe RDF** **2**
2.1 Bulbazaur's infobox . . . . . 2
2.2 Création du schéma SHACL pour les Pokémon . . . . . 3
2.3 Génération des données RDF . . . . . 5
2.4 Validation des données avec SHACL . . . . . 6

**3 Extraction des pages Bulbapedia** **7**
3.1 Utilisation de l'API MediaWiki . . . . . 7

**4 Intégration des données multilingues** **9**
4.1 Enrichissement avec les labels internationaux . . . . . 9

**5 Extraction du contenu détaillé via l'API MediaWiki** **10**
5.1 Architecture d'extraction . . . . . 10
5.2 Génération des triplets RDF . . . . . 11
5.3 Enrichissement sémantique . . . . . 11

**6 Commandes SPARQL** **11**
6.1 Nombre de triplets . . . . . 11
6.2 Définition des préfix . . . . . 12
6.3 Requête 1 : Pokémon et capacités avec liens . . . . . 12
6.4 Requête 2 : Moves avec image et liens . . . . . 13
6.5 Requête 3 : Régions avec le plus grand nombre de lieux . . . . . 14
6.6 Requête 4 : Lieux et Régions avec liens . . . . . 15

**7 Conclusion** **16**

# 1 Introduction

Les graphes de connaissances (Knowledge Graphs - KG) sont des structures essentielles pour représenter des informations organisées et interconnectées de manière sémantique. Ils jouent un rôle fondamental dans les domaines du web sémantique, des systèmes de recommandation et des moteurs de recherche. Des initiatives comme DBpedia et YAGO ont démontré le potentiel de ces graphes en exploitant des ressources comme Wikipédia pour structurer et relier des données riches et diversifiées.

Ce projet vise à construire un graphe de connaissances (KG) à partir de Bulbapedia, une encyclopédie dédiée à l'univers des Pokémon, contenant plus de 50 000 articles. L'objectif est de capturer autant que possible le contenu de ce wiki, en créant des entités interconnectées et en traduisant les informations en triples RDF exploitables. Ces triples reflètent les infoboxes, les relations entre les pages, ainsi que les liens externes pertinents.

Pour garantir une utilisation optimale et une interopérabilité avec d'autres systèmes, le vocabulaire du KG s'appuiera sur des ontologies reconnues comme *schema.org* ou celle de DBpedia. Le graphe sera également enrichi de labels multilingues, validé par des schémas dérivés des modèles de Bulbapedia, et exposé via un point d'accès SPARQL permettant des requêtes sophistiquées. Enfin, le KG inclura des liens vers d'autres graphes de connaissances et des ressources en ligne pour renforcer la connectivité de ses entités.

En résumé, ce projet s'inscrit dans la continuité des efforts visant à structurer et à relier les connaissances à l'échelle du web, tout en s'appuyant sur une ressource unique et thématique, Bulbapedia, pour démontrer l'efficacité et l'utilité des graphes de connaissances.

# 2 Extraction de l'infobox et génération du graphe RDF

## 2.1 Bulbazaur's infobox

Pour construire un KG à partir du contenu de Bulbapedia, la première étape consiste à extraire les informations structurées d'une page spécifique en wikitext. Nous avons pris comme exemple la page de *Bulbizarre* (Bulbasaur). En accédant à l'onglet *"view source"* de cette page, nous avons récupéré le wikitext contenant l'infobox associée, spécifiée par le modèle `{{Pokémon Infobox}}`.

Le contenu du wikitext a été sauvegardé dans un fichier texte nommé `bulbasaur_infobox.txt`. Nous avons utilisé la bibliothèque Python `mwparserfromhell` pour parser le contenu et identifier l'infobox.

Quand l'infobox est trouvée et permet d'extraire ses paramètres sous forme de paires clé-valeur. Ces données structurées constituent la base pour la génération des triples RDF.

Une fois les données de l'infobox extraites, nous les avons utilisées pour créer des triples RDF. Chaque paramètre de l'infobox est converti en un triplet *sujet-prédicat-objet*, où le sujet est l'entité `Bulbasaur`, les prédicats correspondent aux attributs de l'infobox, et les objets aux valeurs associées.

```
@prefix pokemon: <http://example.org/pokemon#> .
@prefix schema1: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

pokemon:Bulbasaur a pokemon:Pokemon ;
    pokemon:height "0.7"^^schema1:Float,
        "0.7"^^xsd:float ;
    pokemon:japaneseName "フシギダネ"@ja ;
    pokemon:type1 "Grass"@en ;
    pokemon:type2 "Poison"@en ;
    pokemon:weight "6.9"^^xsd:float ;
    schema1:name "Bulbasaur"@en .
```

FIGURE 1 – Bulbasaur .ttl

Cette étape permet de transformer le contenu non structuré du wiki en une représentation formelle et exploitable, posant les bases du graphe de connaissances.

## 2.2 Création du schéma SHACL pour les Pokémon

Pour assurer la cohérence et la validation des données extraites des infoboxes Pokémon, nous avons développé un schéma SHACL (Shapes Constraint Language) complet. Ce schéma définit formellement la structure et les contraintes que doivent respecter les données de chaque Pokémon dans notre graphe de connaissances.

| Parameter | Allowed values | Notes |
|---|---|---|
| `name` | *String* | The Pokémon's name in English. |
| `jname` | *String* | The Pokémon's name in Japanese. |
| `tmname` | *String* | Translation of the Japanese name. Use the trademarked version if available. |
| `jtranslit` | *String* | Japanese name in rōmaji. |
| `category` | *String* | Category name, from the Pokédex. |
| `ndex` | *Number* | National Pokédex number |
| `forme` | `2` • `3` • `4` • `5` • `6` | **Optional**. Number of forms the Pokémon has. **Don't** use if the Pokémon only has one form. |
| `form1` | *String* | **Optional**. Name of the first form. **Don't** use if the Pokémon only has one form. |
| `form2` | *String* | **Optional**. Name of the second form. |
| `form3` | *String* | **Optional**. Name of the third form. |
| `form4` | *String* | **Optional**. Name of the fourth form. |
| `form5` | *String* | **Optional**. Name of the fifth form. |
| `form6` | *String* | **Optional**. Name of the sixth form. |
| `image` | *Filename* | Main image of the Pokémon. Should **only** be used if the image is not at the default location. **Don't** include `File:` . |
| `image2` | *Filename* | **Optional**. Image of the second form / first Mega Evolution. **Don't** include `File:` . |
| `image3` | *Filename* | **Optional**. Image of the third form / second Mega Evolution. **Don't** include `File:` . |
| `image4` | *Filename* | **Optional**. Image of the fourth form. **Don't** include `File:` . |
| `image5` | *Filename* | **Optional**. Image of the fifth form. **Don't** include `File:` . |
| `image6` | *Filename* | **Optional**. Image of the sixth form. **Don't** include `File:` . |
| | **Types** | |
| `type1` | `Normal` • `Fire` • `Fighting` • `Water` • `Flying` • `Grass` • `Poison` • `Electric` • `Ground` • `Psychic` • `Rock` • `Ice` • `Bug` • `Dragon` • `Ghost` • `Dark` • `Steel` • `Fairy` • `Unknown` | The Pokémon's type. Use the first listed in the Pokédex, where applicable. Defaults to `Unknown` . |
| `type2` | `Normal` • `Fire` • `Fighting` • `Water` • `Flying` • `Grass` • `Poison` • `Electric` • `Ground` • `Psychic` • `Rock` • `Ice` • `Bug` • `Dragon` • `Ghost` • `Dark` • `Steel` • `Fairy` • `Unknown` | **Optional**. The Pokémon's second type. |

FIGURE 2 – Extrait du tableau contenant les parametres du template Pokemon

— Des types de données spécifiques (xsd :string, xsd :integer, xsd :decimal)
— Des cardinalités minimales et maximales (sh :minCount, sh :maxCount)
— Des descriptions explicatives pour chaque champ
— Des valeurs admissibles pour certains champs (sh :in)
Parmi les propriétés principales définies, on trouve :
— Les informations d'identification (nom, numéro Pokédex)
— Les caractéristiques physiques (taille, poids)
— Les attributs de combat (types, capacités)
— Les statistiques de base et les valeurs d'effort (EV)
— Les informations de reproduction (groupes d'œufs)

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix pokemon: <http://example.org/pokemon#> .

pokemon:PokemonTemplateShape
    a sh:NodeShape ;
    sh:targetClass pokemon:Pokemon ;
    sh:property [
        sh:path pokemon:name ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:description "The Pokémon's name in English." ;
    ] ;
    sh:property [
        sh:path pokemon:jname ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:description "The Pokémon's name in Japanese." ;
    ] ;
    sh:property [
        sh:path pokemon:tmname ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:description "Translation of the Japanese name. " ;
    ] ;
    sh:property [
        sh:path pokemon:jtranslit ;
        sh:datatype xsd:string ;
        sh:maxCount 1 ;
        sh:description "Japanese name in rômaji. " ;
    ] ;
    sh:property [
        sh:path pokemon:category ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:description "Category name from the Pokédex." ;
    ] ;
    sh:property [
        sh:path pokemon:ndex ;
        sh:datatype xsd:integer ;
```

FIGURE 3 – Extrait du schéma SHACL pour la validation des données Pokémon

Ce schéma SHACL joue un rôle crucial dans notre pipeline d'extraction en :
— Validant la conformité des données extraites
— Assurant la cohérence du graphe de connaissances
— Facilitant la détection d'erreurs dans le processus d'extraction
— Documentant la structure attendue des données

## 2.3 Génération des données RDF

Après avoir défini notre schéma SHACL, nous avons procédé à l'extraction et à la transformation des données Pokémon en format RDF. Ce processus s'est déroulé en plusieurs étapes clés :

### 2.3.1 Extraction des données de base

La première étape consiste à extraire les informations de base des Pokémon à partir d'une liste structurée. Pour chaque Pokémon, nous récupérons :

— L'identifiant national et régional du Pokédex
— Le nom du Pokémon
— Le stade d'évolution
— Les types (primaire et secondaire)
— Les formes alternatives si elles existent

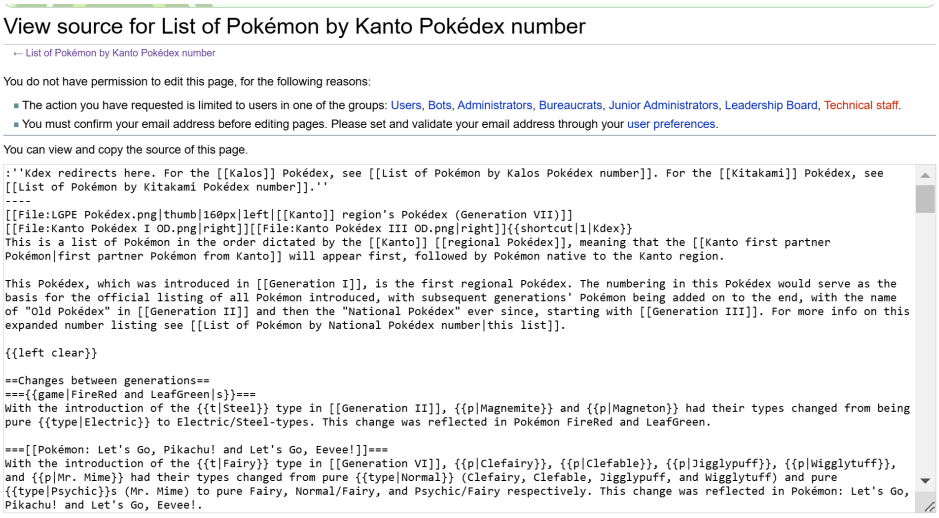


FIGURE 4 – View source pour la liste de Pokémon

### 2.3.2 Création du graphe RDF initial

Une fois les données extraites, nous procédons à la création d'un graphe RDF en utilisant les espaces de noms suivants :

— `pokemon:` pour les entités et propriétés spécifiques aux Pokémon
— `rdfs:` pour les labels et commentaires
— `xsd:` pour les types de données
— `owl:` pour les relations sémantiques
— `bulbapedia:` pour les liens vers les pages Bulbapedia

### 2.3.3 Enrichissement du graphe

Le graphe initial est ensuite enrichi avec des propriétés additionnelles :

— Ajout des capacités (abilities) avec leurs différentes variantes
— Liaison avec le template d'infobox Pokémon
— Création de liens vers les pages Bulbapedia correspondantes
— Ajout des groupes d'œufs et autres caractéristiques

### 2.3.4 Optimisation et normalisation

La dernière étape consiste à optimiser et normaliser les données :
— Remplacement des labels RDFS par des propriétés schema.org pour une meilleure interopérabilité
— Création de liens entre les Pokémon et leurs capacités spécifiques
— Normalisation des noms et identifiants pour éviter les caractères spéciaux
— Validation des données générées contre le schéma SHACL précédemment défini

```
pokemon:Abra a pokemon1:Pokemon ;
    pokemon1:ability1 <http://example.org/pokemon/ability/Synchronize> ;
    pokemon1:ability2 <http://example.org/pokemon/ability/Inner_Focus> ;
    pokemon1:abilityd <http://example.org/pokemon/ability/Magic_Guard> ;
    pokemon1:body "06" ;
    pokemon1:catchrate "200" ;
    pokemon1:category "Psi" ;
    pokemon1:color "Brown" ;
    pokemon1:eggcycles "20" ;
    pokemon1:egggroup1 "Human-Like" ;
    pokemon1:egggroupn "1" ;
    pokemon1:evsa "1" ;
    pokemon1:evtotal "1" ;
    pokemon1:experienceYield 62 ;
    pokemon1:friendship "70" ;
    pokemon1:g4exp "75" ;
    pokemon1:genderCode 63 ;
    pokemon1:generation 1 ;
    pokemon1:height-ftin "2'11\"" ;
```

FIGURE 5 – Exemple de triplets RDF générés pour un Pokémon

Cette approche méthodique nous permet de générer un graphe de connaissances riche et cohérent, prêt à être utilisé pour des requêtes SPARQL et des analyses plus poussées.

## 2.4 Validation des données avec SHACL

Une fois les données RDF générées, nous avons procédé à leur validation en utilisant SHACL. Cette étape cruciale permet de s'assurer que les données respectent les contraintes définies dans notre schéma.

### 2.4.1 Processus de validation

La validation SHACL s'effectue en plusieurs étapes :
— Chargement du graphe RDF contenant les données Pokémon
— Chargement du schéma SHACL définissant les contraintes
— Exécution du validateur avec le moteur d'inférence RDFS
— Génération d'un rapport détaillé de validation

Les résultats de la validation ont confirmé la conformité de nos données avec le schéma défini, validant ainsi notre approche de modélisation et d'extraction des données.

```
Validation conforme : True
Rapport de validation :
Validation Report
Conforms: True
```

FIGURE 6 – Résultats de la validation SHACL pour les différentes catégories

### 2.4.2 Extension aux autres catégories

Le processus de génération de RDF et de validation SHACL a été reproduit pour les autres composantes essentielles de l'univers Pokémon :

**Capacités (Moves)** Nous avons appliqué le même processus d'extraction et de transformation pour générer un graphe RDF complet des capacités Pokémon. Cette transformation inclut les mouvements d'attaque, leurs caractéristiques et leurs effets, aboutissant à la création du fichier `moves_output.ttl`.

**Talents (Abilities)** Les talents des Pokémon ont également été convertis en RDF, en préservant leurs descriptions et leurs effets dans le jeu. Ces données ont été structurées et validées pour produire le fichier `ability_kg.ttl`.

**Lieux (Locations)** Les données géographiques des différentes zones du monde Pokémon ont été transformées en RDF, créant ainsi une cartographie complète des lieux dans le fichier `locations_kg.ttl`.

**Régions** Les informations sur les régions du monde Pokémon ont été converties en suivant la même méthodologie, générant le fichier `regions_output.ttl` qui structure l'organisation géographique du monde Pokémon.

Chaque catégorie a suivi le même workflow rigoureux :

1. Extraction des données depuis Bulbapedia
2. Transformation en format RDF avec les namespaces appropriés
3. Validation via un schéma SHACL spécifique
4. Génération d'un fichier .ttl validé

Cette approche systématique garantit la cohérence et l'interopérabilité de l'ensemble des données dans notre graphe de connaissances.

# 3 Extraction des pages Bulbapedia

## 3.1 Utilisation de l'API MediaWiki

Pour enrichir notre graphe de connaissances, nous avons mis en place un processus d'extraction systématique des pages Bulbapedia en utilisant l'API MediaWiki. Cette

approche nous permet de créer des liens entre les différentes représentations d'un même Pokémon, suivant les bonnes pratiques établies par DBpedia et YAGO.

### 3.1.1 Processus d'extraction

Le processus d'extraction s'effectue en plusieurs étapes :

1. Interrogation de l'API MediaWiki avec le paramètre `list=allpages` pour récupérer la liste exhaustive des pages
2. Filtrage des pages pour ne conserver que celles concernant les Pokémon (identifiées par le suffixe "(Pokémon)")
3. Gestion de la pagination via le paramètre `continue` pour assurer une extraction complète

### 3.1.2 Transformation en RDF

Pour chaque page Pokémon identifiée, nous générons un ensemble de triplets RDF en utilisant plusieurs espaces de noms :

— `wiki:` pour les URLs de Bulbapedia
— `resource:` pour les ressources Pokémon
— `pokemon:` pour les entités Pokémon
— `foaf:` pour les liens vers les pages web
— `owl:` pour les relations sémantiques

### 3.1.3 Structure des triplets générés

Pour chaque Pokémon, nous créons une structure à trois niveaux :

1. La page wiki (`wiki:NomPokemon`)
2. La ressource (`resource:NomPokemon`)
3. L'entité Pokémon (`pokemon:NomPokemon`)


FIGURE 7 – Exemple de structure des triplets RDF pour le Pokémon Abra

Cette structure permet de distinguer clairement :
— La page web qui décrit le Pokémon
— La ressource qui représente le concept du Pokémon
— L'entité Pokémon elle-même
Les relations entre ces éléments sont établies via des prédicats appropriés :
— `pokemon1:describes` lie la page à la ressource
— `owl:sameAs` lie la ressource à l'entité Pokémon
— `foaf:page` lie la ressource à la page web

# 4 Intégration des données multilingues

## 4.1 Enrichissement avec les labels internationaux

Pour enrichir notre graphe de connaissances avec des informations multilingues, nous avons intégré les données du fichier `pokedex-i18n.tsv`, qui contient les noms et descriptions des Pokémon dans différentes langues.

### 4.1.1 Sources de données

Les données multilingues proviennent de deux sources principales :
— Le fichier TTL existant contenant les entités Pokémon
— Le fichier TSV contenant les traductions dans différentes langues

### 4.1.2 Langues supportées

Notre graphe de connaissances intègre les traductions dans dix langues principales :
— Japonais (ja) et sa romanisation (ja-Latn)
— Coréen (ko)
— Chinois (zh)
— Français (fr)
— Allemand (de)
— Espagnol (es)
— Italien (it)
— Anglais (en)
— Tchèque (cs)

### 4.1.3 Processus d'intégration

L'intégration des données multilingues s'effectue en plusieurs étapes :

1. Chargement du graphe RDF existant
2. Lecture et traitement du fichier TSV de traductions
3. Association des identifiants Pokémon avec leurs traductions
4. Mise à jour des triplets avec les tags de langue appropriés

```
schema1:name "Abra",
    "Abra"@de,
    "Abra"@en,
    "Abra"@es,
    "Abra"@fr,
    "Abra"@it,
    "ケーシィ"@ja,
    "캐이시"@ko,
    "凯西"@zh ;
```

FIGURE 8 – Exemple de triplets RDF multilingues pour un Pokémon

#### 4.1.4 Structure des données multilingues

Pour chaque Pokémon, nous générons des triplets utilisant :
— Le prédicat `schema:name` pour les noms
— Des tags de langue standardisés (ISO 639-1)
— Une distinction claire entre le japonais standard et sa romanisation
Cette approche permet :
— Une identification claire de la langue de chaque nom
— Une meilleure interopérabilité avec d'autres bases de connaissances
— Une facilité d'accès aux informations dans la langue souhaitée

# 5 Extraction du contenu détaillé via l'API MediaWiki

## 5.1 Architecture d'extraction

Pour enrichir notre graphe de connaissances avec des informations détaillées, nous avons développé un système d'extraction modulaire utilisant l'API MediaWiki de Bulbapedia. Cette architecture s'articule autour d'une classe abstraite `WikiContentExtractor` et de ses implémentations spécifiques pour chaque type de contenu.

#### 5.1.1 Types de contenus extraits

Notre système extrait cinq catégories principales de données :
— Pokémon (avec le suffixe "(Pokémon)")
— Capacités (avec le suffixe "(move)")
— Talents (avec le suffixe "(ability)")
— Lieux (avec le suffixe "(location)")
— Régions (avec le suffixe "(region)")

#### 5.1.2 Processus d'extraction

Pour chaque entité, le processus d'extraction comprend :

1. **Récupération du wikitext** :
   — Extraction du code source de la page
   — Analyse des templates utilisés
   — Collecte des images associées
   — Identification des liens internes et externes
2. **Analyse des infobox** :
   — Parsing des templates d'infobox
   — Nettoyage des valeurs (suppression du markup wiki)
   — Structuration des données en paires clé-valeur
3. **Extraction des métadonnées** :
   — Catégories de la page
   — Structure des sections
   — Propriétés spécifiques
   — Liens interwiki

## 5.2 Génération des triplets RDF

Les données extraites sont transformées en triplets RDF selon plusieurs espaces de noms :

— `pokemon:` pour les entités principales
— `wiki:` pour les pages Bulbapedia
— `resource:` pour les ressources extraites
— `foaf:` pour les images et représentations
— `dcterms:` pour les métadonnées

## 5.3 Enrichissement sémantique

Le processus d'enrichissement crée plusieurs types de relations :
— Relations entre pages et ressources (`pokemon:describes`)
— Liens vers les images (`foaf:depiction`)
— Références aux autres pages (`dcterms:references`)
— Catégorisation du contenu (`dcterms:subject`)
— Structure des sections (`pokemon:hasSection`)

# 6 Commandes SPARQL

## 6.1 Nombre de triplets

Voyons tout d'abord le nombre total de triplets qu'on a créés.


Figure 9 – Le nombre total de triplets

## 6.2 Définition des préfix

```
PREFIX poke: <https://pokemonkg.org/ontology#>
PREFIX po: <http://purl.org/ontology/po/>
PREFIX pokemon: <http://example.org/pokemon/>
PREFIX ability: <http://example.org/pokemon/ability/>
PREFIX location: <http://example.org/pokemon/location/>
PREFIX region: <http://example.org/pokemon/region/>
PREFIX resource: <http://example.org/pokemon/resource/regions/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX schema: <http://schema.org/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
```

## 6.3 Requête 1 : Pokémon et capacités avec liens

Cette requête récupère les noms des Pokémon et de leurs capacités, tout en établissant des liens vers des ressources externes, notamment DBpedia. Elle filtre uniquement les noms en anglais et retourne des relations telles que :

- Le Pokémon est lié à ses capacités (`pokemon1:ability1`).
- Chaque capacité et chaque Pokémon ont un lien vers une ressource correspondante sur DBpedia (`owl:sameAs`).
- Des liens externes supplémentaires associés aux Pokémon sont également inclus (`rdfs:externalLink`).

**Utilité** : Cette requête illustre l'interconnexion entre des entités dans le graphe et des bases de connaissances externes, ce qui renforce l'interopérabilité.

```
SELECT ?pokemonName ?abilityName ?abilityDBpedia ?pokemonDBpedia ?externalLink
WHERE {
  ?pokemon a pokemon:Pokemon ;
           schema:name ?pokemonName ;
           pokemon:ability1 ?ability .

  ?ability schema:name ?abilityName .

  FILTER (lang(?pokemonName) = "en" && lang(?abilityName) = "en")

  ?pokemon owl:sameAs ?pokemonDBpedia .
  FILTER(STRSTARTS(STR(?pokemonDBpedia), "http://dbpedia.org/resource/"))

  ?ability owl:sameAs ?abilityDBpedia .
  FILTER(STRSTARTS(STR(?abilityDBpedia), "http://dbpedia.org/resource/"))

  ?pokemon rdfs:externalLink ?externalLink .
}
LIMIT 10
```

| pokemonName | abilityName | abilityDBpedia | pokemonDBpedia | externalLink |
|---|---|---|---|---|
| 1"Abra"@en | "Synchronize"@en | <http://dbpedia.org/resource/Synch... | <http://dbpedia.org/resource/Abra> | <https://bulbapedia.bulbagarden.net/wiki/Kadabra_(Pok%C3%A9mon)> |
| 2"Abra"@en | "Synchronize"@en | <http://dbpedia.org/resource/Synch... | <http://dbpedia.org/resource/Abra> | <https://bulbapedia.bulbagarden.net/wiki/Poliwrath_(Pok%C3%A9mon)> |
| 3"Abra"@en | "Synchronize"@en | <http://dbpedia.org/resource/Synch... | <http://dbpedia.org/resource/Abra> | <https://bulbapedia.bulbagarden.net/wiki/Bulbapedia:Project_Pok%C3%A9dex> |
| 4"Abra"@en | "Synchronize"@en | <http://dbpedia.org/resource/Synch... | <http://dbpedia.org/resource/Abra> | <https://bulbapedia.bulbagarden.net/wiki/Bulbapedia:Projects> |
| 5"Abra"@en | "Synchronize"@en | <http://dbpedia.org/resource/Synch... | <http://dbpedia.org/resource/Abra> | <https://bulbapedia.bulbagarden.net/wiki/File:Project_Pok%C3%A9dex_logo.p... |
| 6"Abra"@en | "Synchronize"@en | <http://dbpedia.org/resource/Synch... | <http://dbpedia.org/resource/Abra> | <https://bulbapedia.bulbagarden.net/wiki/List_of_Pok%C3%A9mon_by_Nation... |
| 7"Aerodactyl"@en | "Rock Head"@en | <http://dbpedia.org/resource/Rock_... | <http://dbpedia.org/resource/Aerod... | <https://bulbapedia.bulbagarden.net/wiki/Aerodactyl_(Pok%C3%A9mon)> |
| 8"Aerodactyl"@en | "Rock Head"@en | <http://dbpedia.org/resource/Rock_... | <http://dbpedia.org/resource/Aerod... | <https://bulbapedia.bulbagarden.net/wiki/Alakazam_(Pok%C3%A9mon)> |
| 9"Aerodactyl"@en | "Rock Head"@en | <http://dbpedia.org/resource/Rock_... | <http://dbpedia.org/resource/Aerod... | <https://bulbapedia.bulbagarden.net/wiki/Beedrill_(Pok%C3%A9mon)> |
| 10"Aerodactyl"@en | "Rock Head"@en | <http://dbpedia.org/resource/Rock_... | <http://dbpedia.org/resource/Aerod... | <https://bulbapedia.bulbagarden.net/wiki/Blastoise_(Pok%C3%A9mon)> |

FIGURE 10 – Exemple de résultats pour la requête Pokémon et Capacities

## 6.4 Requête 2 : Moves avec image et liens

Cette requête extrait les caractéristiques des mouvements (*moves*) des Pokémon, notamment :

- Le nom du mouvement, sa catégorie de dégâts (`pokemon:damagecategory`), et sa génération (`pokemon:gen`).
- Une image associée au mouvement (`pokemon:gameimage`).
- Des liens vers des descriptions plus détaillées sur Bulbapedia (`rdfs:seeAlso`) et DBpedia (`owl:sameAs`).

**Utilité** : Elle fournit une vue enrichie des mouvements avec des images et des liens explicatifs, permettant de naviguer facilement entre les bases de connaissances.

```
SELECT ?moveName ?damageCategory ?generation ?gameImage ?bulbapediaLink ?dbpediaLink
WHERE {
  ?move a pokemon:Move ;
        schema:name ?moveName ;
        pokemon:damagecategory ?damageCategory ;
        pokemon:gen ?generation ;
        pokemon:gameimage ?gameImage ;
        rdfs:seeAlso ?bulbapediaLink ;
        owl:sameAs ?dbpediaLink .

  FILTER (lang(?moveName) = "en")
}
```

| moveName | damageCategory | generation | gameImage | bulbapediaLink | dbpediaLink |
|---|---|---|---|---|---|
| 1"Absorb"@en | Special | I | <https://bulbapedia.bulbagarden.net/wiki/... | <https://bulbapedia.bulbagarden.net/wiki/Absor... | <http://dbpedia.org/resource/Abs... |
| 2"Accelerock"@en | Special | I | <https://bulbapedia.bulbagarden.net/wiki/... | <https://bulbapedia.bulbagarden.net/wiki/Accel... | <http://dbpedia.org/resource/Acc... |
| 3"Acid"@en | Special | I | <https://bulbapedia.bulbagarden.net/wiki/... | <https://bulbapedia.bulbagarden.net/wiki/Acid_... | <http://dbpedia.org/resource/Acid> |
| 4"Acid Armor"@en | Special | I | <https://bulbapedia.bulbagarden.net/wiki/... | <https://bulbapedia.bulbagarden.net/wiki/Acid_... | <http://dbpedia.org/resource/Acid... |
| 5"Acid Armor"@en | Special | I | <https://bulbapedia.bulbagarden.net/wiki/... | <https://bulbapedia.bulbagarden.net/wiki/Acid%... | <http://dbpedia.org/resource/Acid... |
| 6"Acid Downpour"@en | Special | I | <https://bulbapedia.bulbagarden.net/wiki/... | <https://bulbapedia.bulbagarden.net/wiki/Acid%... | <http://dbpedia.org/resource/Acid... |
| 7"Acid Downpour"@en | Special | I | <https://bulbapedia.bulbagarden.net/wiki/... | <https://bulbapedia.bulbagarden.net/wiki/Acid_... | <http://dbpedia.org/resource/Acid... |
| 8"Acid Spray"@en | Special | I | <https://bulbapedia.bulbagarden.net/wiki/... | <https://bulbapedia.bulbagarden.net/wiki/Acid_... | <http://dbpedia.org/resource/Acid... |
| 9"Acid Spray"@en | Special | I | <https://bulbapedia.bulbagarden.net/wiki/... | <https://bulbapedia.bulbagarden.net/wiki/Acid%... | <http://dbpedia.org/resource/Acid... |
| 10"Acrobatics"@en | Special | I | <https://bulbapedia.bulbagarden.net/wiki/... | <https://bulbapedia.bulbagarden.net/wiki/Acrob... | <http://dbpedia.org/resource/Acr... |

FIGURE 11 – Exemple de résultats pour la requête Moves

## 6.5 Requête 3 : Régions avec le plus grand nombre de lieux

Cette requête identifie les régions contenant le plus grand nombre de lieux dans l'univers Pokémon. Elle :

— Compte le nombre de lieux (`pokemon:Location`) associés à chaque région (`pokemon:Region`).
— Retourne les noms des régions et le nombre total de lieux associés.
— Trie les résultats par ordre décroissant de nombre de lieux.

**Utilité** : Elle permet d'analyser la densité des lieux dans chaque région, utile pour identifier les régions les plus riches en contenu.

```
SELECT ?regionName (COUNT(?location) AS ?locationCount)
WHERE {
  ?location a pokemon:Location ;
            pokemon:region ?region .

  ?region a pokemon:Region ;
          schema:name ?regionName .
}
GROUP BY ?regionName
ORDER BY DESC(?locationCount)
LIMIT 5
```

| regionName | locationCount |
|---|---|
| 1"Unovský"@cs | "38"^^<http://www.w3.org/2001/XMLSchema#integer> |
| 2"Einall"@de | "38"^^<http://www.w3.org/2001/XMLSchema#integer> |
| 3"Hoenn"@de | "38"^^<http://www.w3.org/2001/XMLSchema#integer> |
| 4"Hoenn"@en | "38"^^<http://www.w3.org/2001/XMLSchema#integer> |
| 5"Unova"@en | "38"^^<http://www.w3.org/2001/XMLSchema#integer> |

FIGURE 12 – Exemple de résultats pour les régions et lieux

## 6.6 Requête 4 : Lieux et Régions avec liens

Cette requête relie les lieux (`pokemon:Location`) à leurs régions respectives (`pokemon:Region`) et ajoute des liens externes pour chaque entité. Elle :

— Récupère les noms des lieux et des régions.
— Inclut des liens vers DBpedia (`owl:sameAs`) et Bulbapedia (`rdfs:seeAlso`).
— Limite les résultats aux noms en anglais.

**Utilité** : Cette requête met en avant l'interconnexion des lieux et des régions avec des sources externes, offrant une navigation enrichie entre les différentes ressources.

```
SELECT ?locationName ?regionName ?regionLink ?dbpediaLink ?bulbapediaLink
WHERE {
  ?location a pokemon:Location ;
            schema:name ?locationName ;
            pokemon:region ?region ;
            owl:sameAs ?dbpediaLink ;
            rdfs:seeAlso ?bulbapediaLink .

  ?region a pokemon:Region ;
          schema:name ?regionName ;
          owl:sameAs ?regionLink .

  FILTER (lang(?regionName) = "en" && lang(?locationName) = "en")
}
LIMIT 10
```

| locationName | regionName | regionLink | dbpediaLink | bulbapediaLink |
|---|---|---|---|---|
| 1"Abandoned Ship"@en | "Hoenn"@en | <http://dbpedia.org/resource/Hoenn> | <http://dbpedia.org/resource/Abandoned_Ship> | <https://bulbapedia.bulbagarden.net/wiki/Abandoned_Ship> |
| 2"Abundant Shrine"@en | "Unova"@en | <http://dbpedia.org/resource/Unova> | <http://dbpedia.org/resource/Abundant_Shrine> | <https://bulbapedia.bulbagarden.net/wiki/Abundant_Shrine> |
| 3"Ancient Tomb"@en | "Hoenn"@en | <http://dbpedia.org/resource/Hoenn> | <http://dbpedia.org/resource/Ancient_Tomb> | <https://bulbapedia.bulbagarden.net/wiki/Ancient_Tomb> |
| 4"Artisan Cave"@en | "Hoenn"@en | <http://dbpedia.org/resource/Hoenn> | <http://dbpedia.org/resource/Artisan_Cave> | <https://bulbapedia.bulbagarden.net/wiki/Artisan_Cave> |
| 5"Battle Chateau"@en | "Kalos"@en | <http://dbpedia.org/resource/Kalos> | <http://dbpedia.org/resource/Battle_Chateau> | <https://bulbapedia.bulbagarden.net/wiki/Battle_Chateau> |
| 6"Battle Maison"@en | "Kalos"@en | <http://dbpedia.org/resource/Kalos> | <http://dbpedia.org/resource/Battle_Maison> | <https://bulbapedia.bulbagarden.net/wiki/Battle_Maison> |
| 7"Battle Resort"@en | "Hoenn"@en | <http://dbpedia.org/resource/Hoenn> | <http://dbpedia.org/resource/Battle_Resort> | <https://bulbapedia.bulbagarden.net/wiki/Battle_Resort> |
| 8"Battle Royal Dome"@en | "Alola"@en | <http://dbpedia.org/resource/Alola> | <http://dbpedia.org/resource/Battle_Royal_Dome> | <https://bulbapedia.bulbagarden.net/wiki/Battle_Royal_Dom... |
| 9"Battle Subway"@en | "Unova"@en | <http://dbpedia.org/resource/Unova> | <http://dbpedia.org/resource/Battle_Subway> | <https://bulbapedia.bulbagarden.net/wiki/Battle_Subway> |
| 10"Battle Tree"@en | "Alola"@en | <http://dbpedia.org/resource/Alola> | <http://dbpedia.org/resource/Battle_Tree> | <https://bulbapedia.bulbagarden.net/wiki/Battle_Tree> |

FIGURE 13 – Exemple de résultats pour les régions et lieux avec des liens

# 7 Conclusion

Ce projet a permis de créer un graphe de connaissances complet et riche à partir du contenu de Bulbapedia, d'intégrer des données RDF multilingues et de les valider avec SHACL. Les commandes SPARQL développées permettent d'explorer le graphe et de répondre à des requêtes complexes, notamment sur les relations entre Pokémon, leurs capacités, leurs mouvements, et leur localisation.

Les résultats obtenus mettent en évidence l'interopérabilité des données avec d'autres graphes comme DBpedia. Ces travaux posent les bases pour des applications avancées, comme les systèmes de recommandation ou les outils de recherche enrichie. Une extension future pourrait inclure l'ajout de données en temps réel ou l'exploration des relations dynamiques entre entités.