

|  |
| --- |
| Introduction au web des données (IWD) |
| Rapport miniprojet |

Dépôt du projet  : <https://github.com/Riga-git/IWD>

Rendu le : 01.06.2020

Rédigé par : Luca Rigazzi

Professeur : J. Tscherrig

# Table des matières

[Table des matières 2](#_Toc42014532)

[1 Introduction 3](#_Toc42014533)

[2 Partie 1 3](#_Toc42014534)

[2.1 Graphe 3](#_Toc42014535)

[2.2 Description des éléments 4](#_Toc42014536)

[2.2.1 Classes 4](#_Toc42014537)

[2.2.2 Propriété objet 4](#_Toc42014538)

[2.2.3 Propriété 5](#_Toc42014539)

[3 Partie 2 6](#_Toc42014540)

[4 Partie 3 6](#_Toc42014541)

[5 Partie 6 9](#_Toc42014542)

[5.1 Requête 1 : Get furniture 9](#_Toc42014543)

[5.2 Requête 2 : Get advance courses 10](#_Toc42014544)

[5.3 Requête 3 : Get number of people in school 11](#_Toc42014545)

[5.4 Requête 4 : Get latin badsed language courses 12](#_Toc42014546)

[5.5 Requête 5 : Get teached languages 13](#_Toc42014547)

[5.6 Get max student capacity 14](#_Toc42014548)

[6 Partie 8 15](#_Toc42014549)

[6.1 Site 1 : hotel – booking.com 15](#_Toc42014550)

# Introduction

Le sujet choisi pour ce projet est la modélisation d’une école en tant qu’institution. Plus précisément il s’agit d’une école de langue.

# Partie 1

## Graphe

La figure 1 montre la modélisation en RDFS.

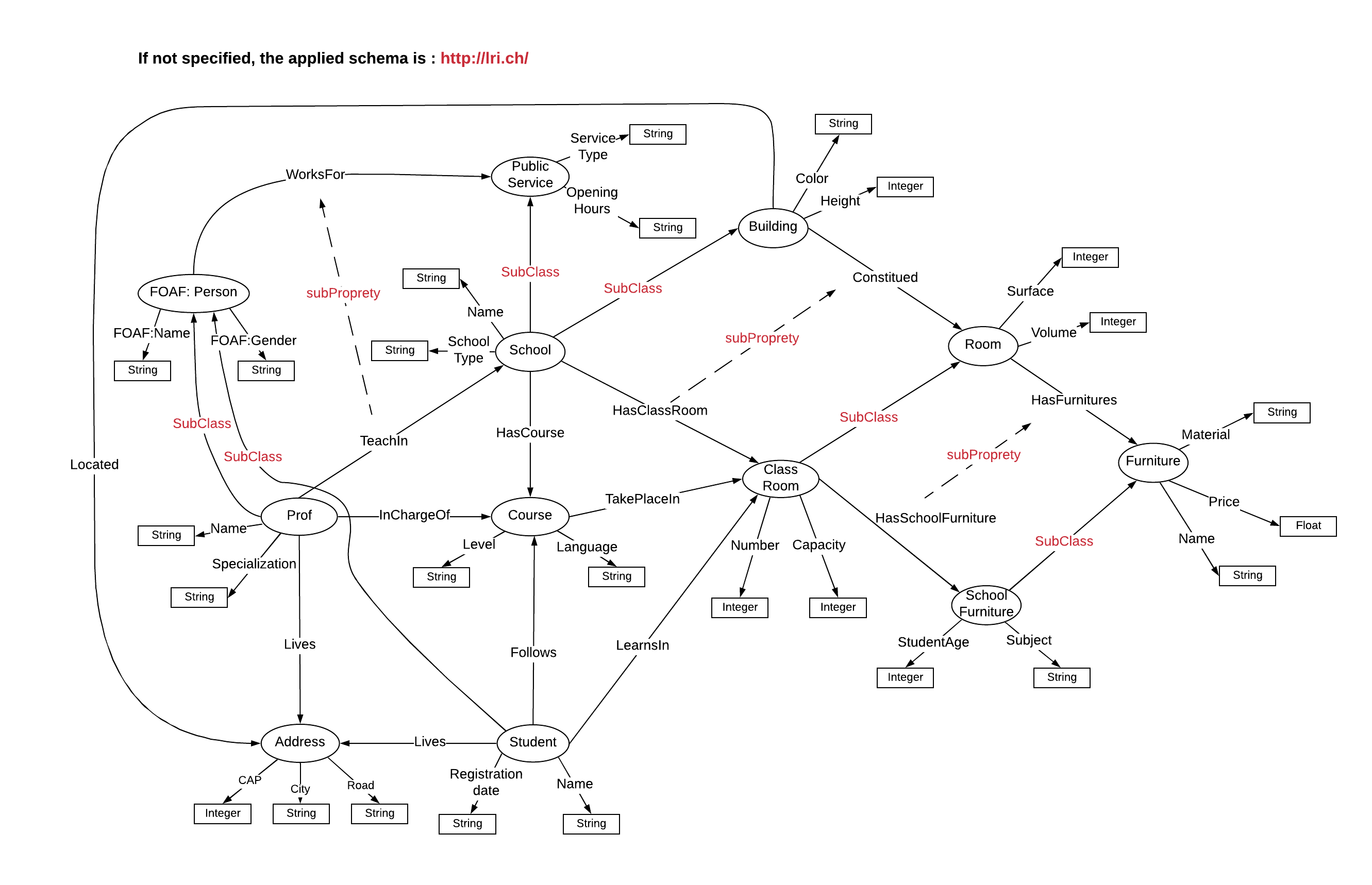


Figure 1: Rdfs pour une école de langue

## Description des éléments

### Classes

|  |  |
| --- | --- |
| **Classe** | **Description** |
| Person | La classe « Person » défini un individue selon la sémantique indique sur  « <https://schema.org/Person> » |
| Prof | La classe « Prof » défini un professeur qui enseigne dans l’école. |
| Address | La classe « Address » défini l’adresse d’une personne ou d’un bâtiment |
| School | La classe « School » représente le sujet de l’ontologie. |
| Course | La classe « Course » défini les différents cours de langue qui sont enseignés dans l’école (ex. : Allemand, Anglais, …) . |
| Student | La classe « Student » défini les étudiants de l’école. |
| PublicService | Indique un quelconque service public (Ecoles, Transport, …) |
| Bulding | Peut définir un bâtiment de tout genre. |
| Room | Peut définir une pièce ou locale de tout genre. |
| Class Room | Cette classe définie une salle de cours. |
| Furniture | Peut définir n’importe quel type de mobilier. |
| SchoolFurniture | Cette classe permet de décrire du mobilier scolaire conçu pour une utilisation scolaire. |

### Propriété objet

|  |  |  |  |
| --- | --- | --- | --- |
| **Propriété** | **Description** | **RDFS : Domain** | **RDFS : Range** |
| WorksFor | Indique qu’une classe de type personne est lié à une classe type PublicService par une relation de travail. | Person | PublicService |
| Located | Emplacement (adresse) d’un immeuble | Building | Address |
| Lives | Domicile d’un prof ou d’un étudiant | Student Prof | Address |
| HasFornitures | Indique que la salle est équipée avec du mobilier | Room | Furniture |
| HasCourse | Indique que l’école a des cours (de langue) | School | Course |
| HasClassRoom | Indique que l’école a des salles de cours | School | ClassRoom |
| Follows | Un étudiant suis un cours de langue. | Student | Course |
| LearnsIn | Un étudiant étudie dans une salle de classe | Student | ClassRoom |
| Constitued | Indique que un bâtiment est constitué de salle. | Building | Room |
| TeachIn | Indique qu’in prof enseigne dans l’école | Prof | School |
| TakePlaceIn | Indique que le cours a lieu dans une salle de cours | Course | ClassRoom |
| InChargeOf | Un professeur est chargé de donner un cours. | Prof | Course |

### Propriété

|  |  |
| --- | --- |
| **Propriété** | **Description** |
| Name | Name selon «<https://schema.org/Thing> » |
| Gender | Sexe selon « <https://schema.org/Person> » (male, femelle) |
| Specialisation | Professeur de Français, Professeur de Anglais |
| CAP | CAP |
| City | Ville |
| Road | Adresse (Rue + N°) |
| RegistrationDate | Date d’inscription ou date de début des cours dans l’école |
| SchoolType | Définie le tipe de l’école (De langue, roffessionelle, …) |
| ServiceType | Définie le tipe de service (Ecole, transport publique, …) |
| Language | Défini la langue enseignée dans le cours |
| OpeningHours | Date d’ouverture de l’établissement |
| Capacity | Nombre de personne que la salle de cours peut accueillir |
| Number | Numero de la salle de cours (Ex. Salle 103) |
| Color | Couleur de l’objet en question |
| Height | Hauteur de l’objet |
| Surface | La surface en m2 |
| Volume | Le volume en m3 |
| SudentAge | L’Age des étudient cible. Par exemple des tables pour des enfants de 6 ans ne sont pas le même que pour une personne de 25 ans. |
| Subject | La matière enseignée dans la salle où se trouve le mobilier. Des tables pour la salle de chimie doivent être résistent aux substances alors que les table de dessin ont une retro-illumination. |
| Price | Le prix de l’objet |
| Material | Le matériel constituant l’objet (Bois, acier, …) |

# Partie 2

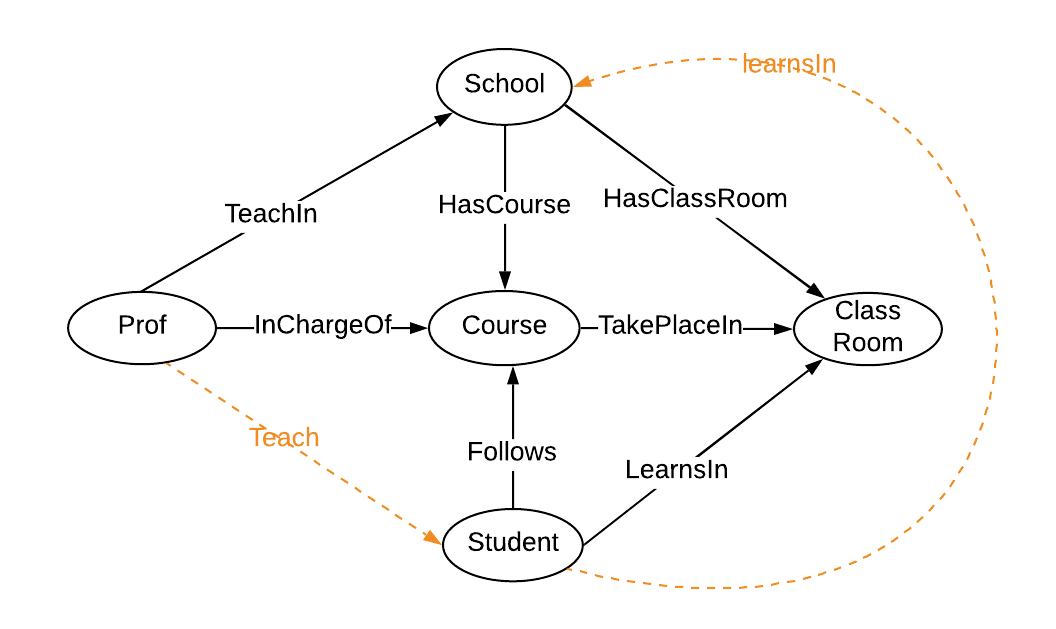


Figure 2: Extrait du schéma avec exemples d’inférence

La figure 2 montre un extrait du schéma complet de l’ontologie. Sur cette image il est possible observer plusieurs exemples d’inférence. Les étudiant il ne sont pas directement lier à l’école ou les professeurs par une propriété objet. Cependant il est possible trouver les étudiant de l’école en passant par le cours ou la salle de classe. Le même raisonnent peut-être applique entre les professeurs et les étudiant. Un moteur d’inférence sera capable de créer « virtuellement » les propriétés ***Teach*** et ***learnsIn***.

# Partie 3

Les images suivantes montrent des exemples de data que l’on pourrait trouver dans les triplestore. Le schéma a été divisé en plusieurs partie pour des raison d’espace et lisibilité.

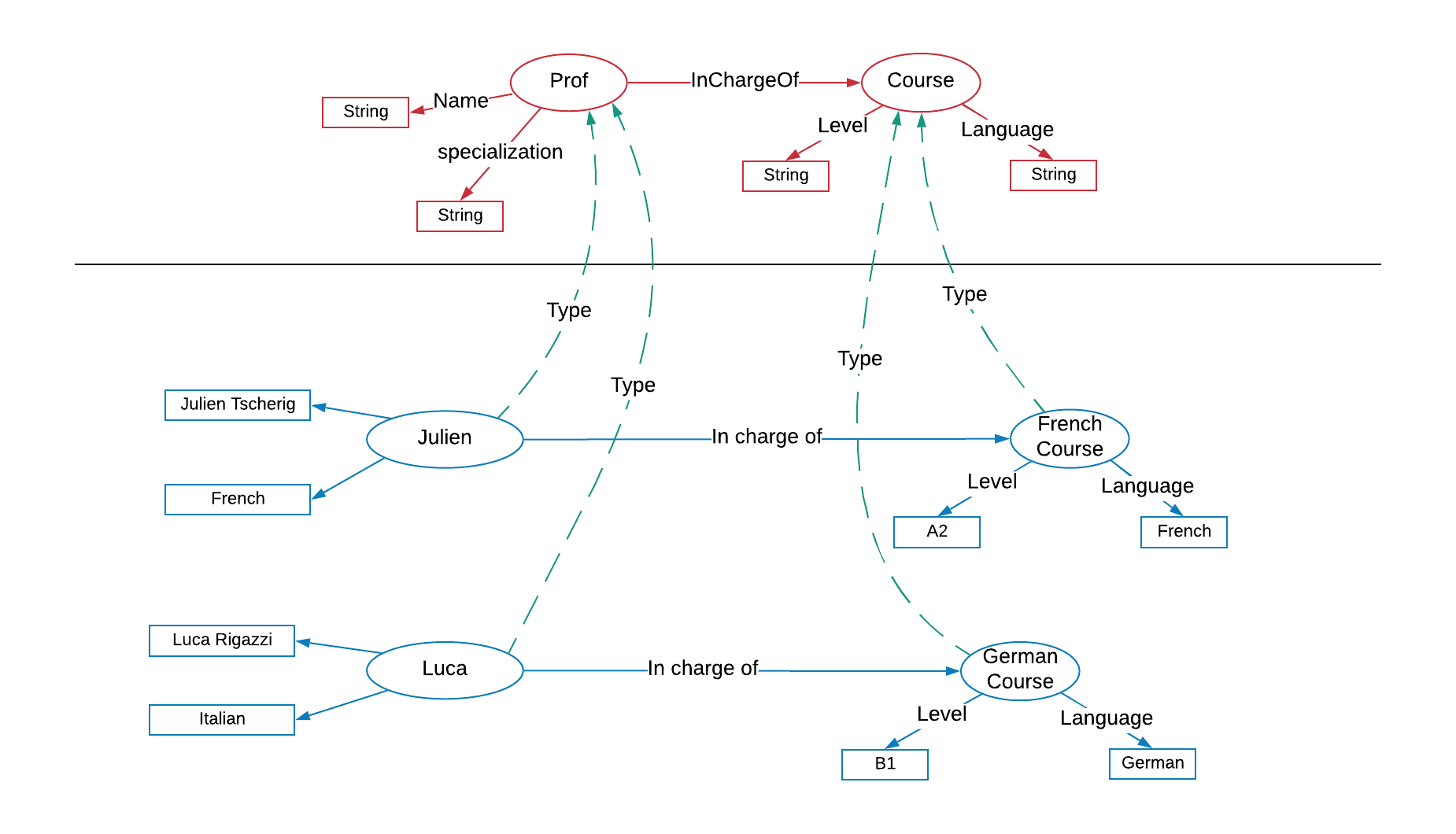


Figure 3 : Instances prof et cours

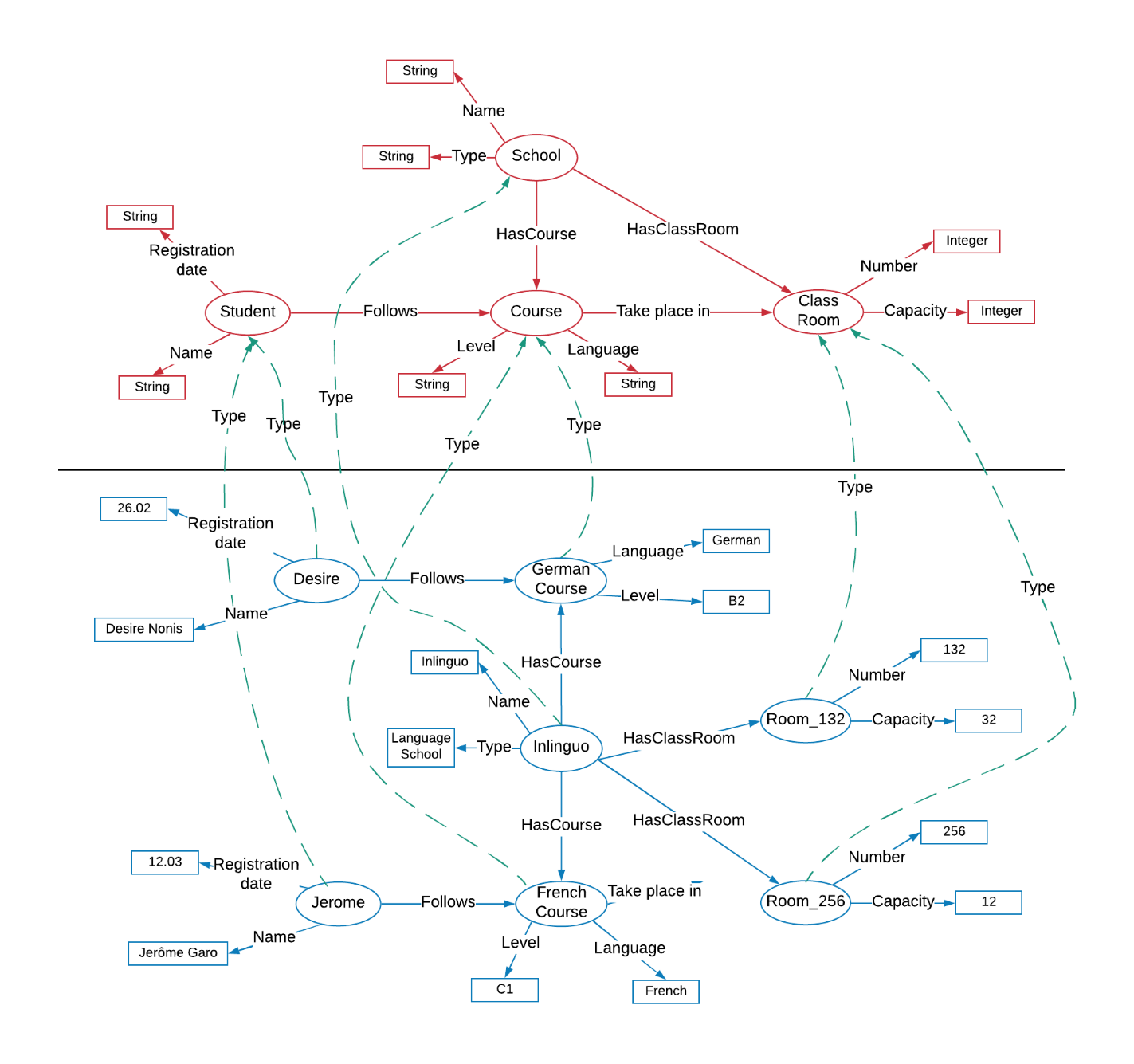


Figure 4 : Instances Etudiants, cours, salle de classe et de l'école

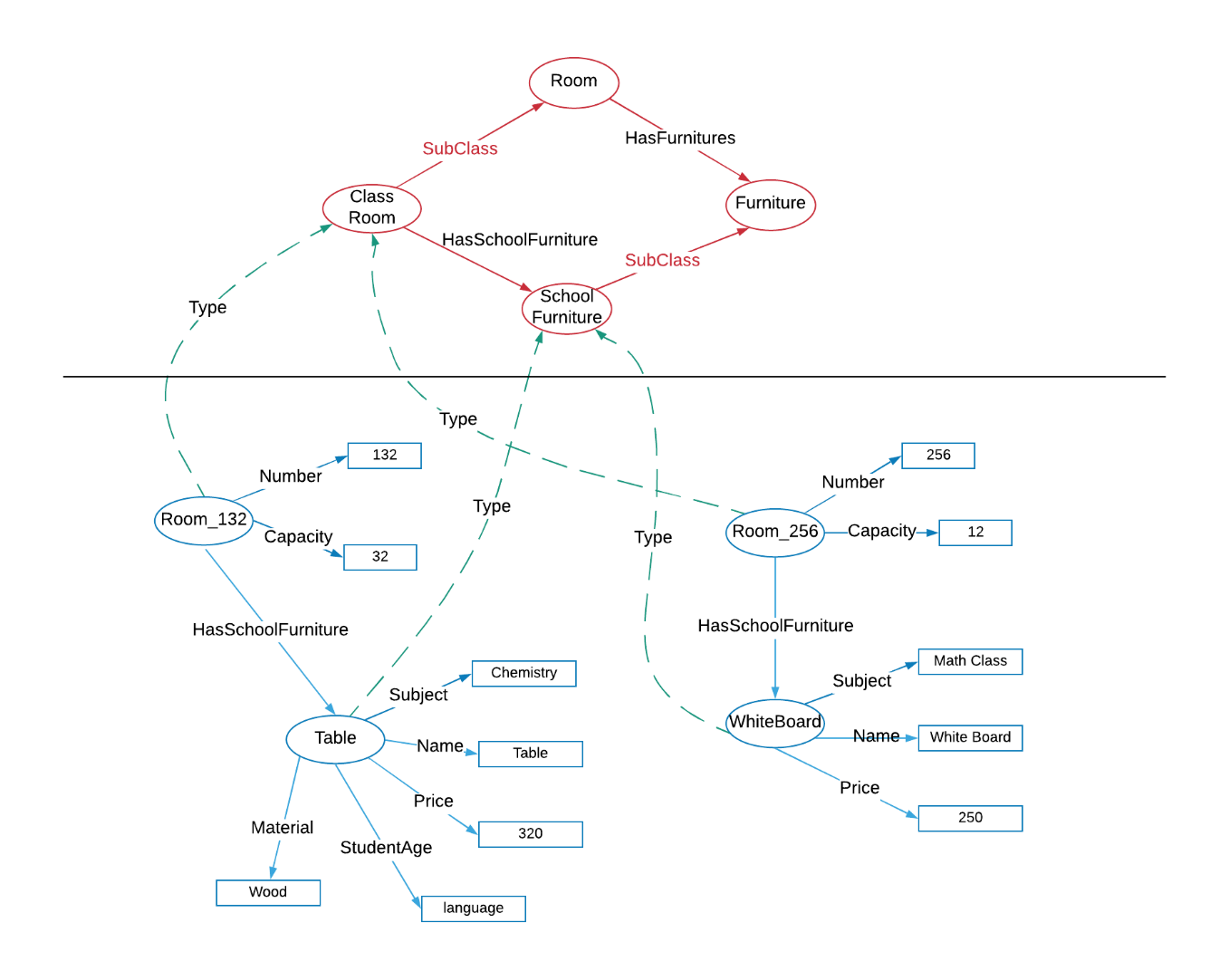


Figure 5 : Instances Mobilier et salles de classe

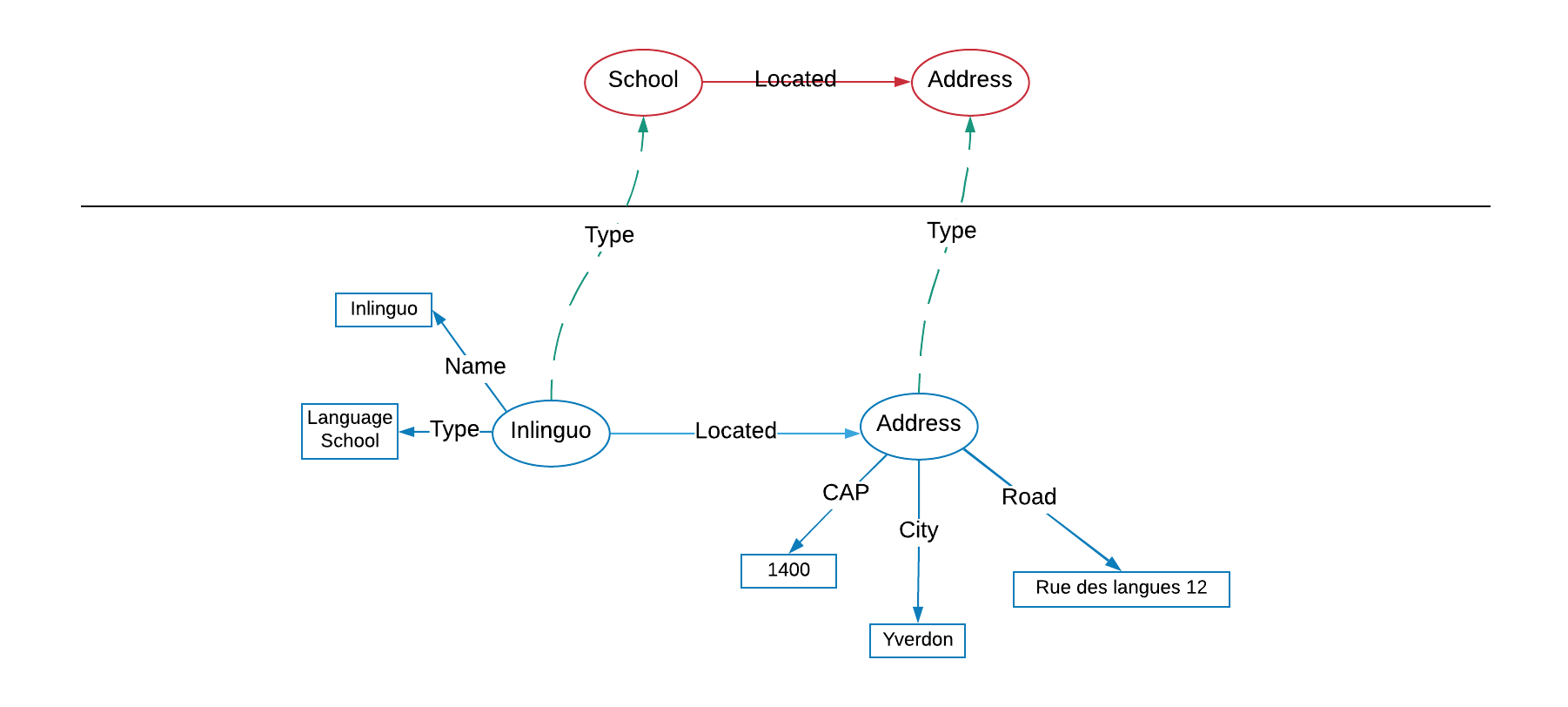


Figure 6 : Instances Ecole et adresse de l'école

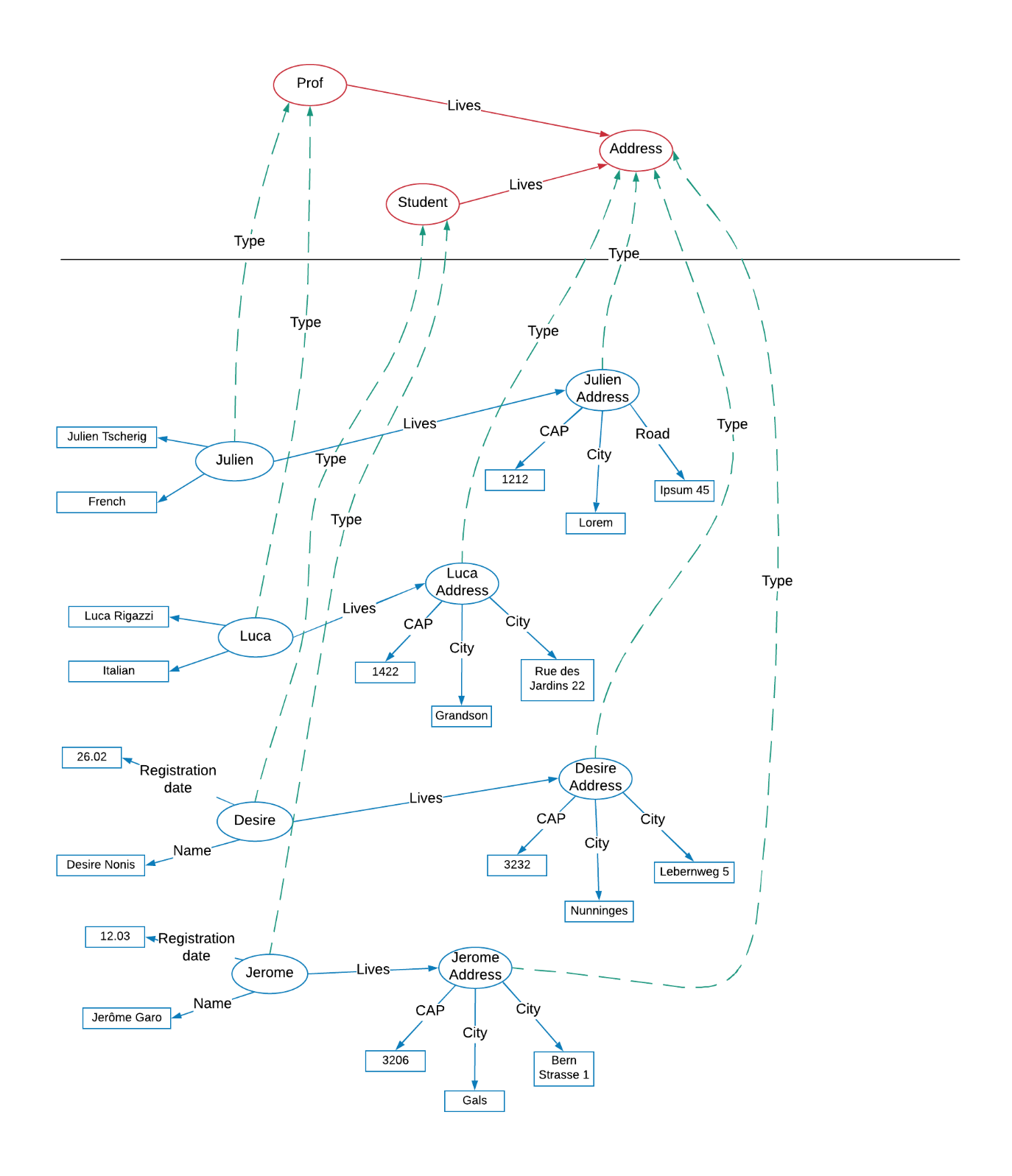


Figure 7: Instances Personnes et de leurs adresses

# Partie 6

## Requête 1 : Get furniture

Cette requête permet de récupérer la totalité du mobilier de l’école ; par exemple en vue d’un inventaire.

Pour trouver tous les meubles de l’école il est nécessaire chercher dans le « Triple Stores » le pattern illustré par la figure 8. Si la propriété ***price*** existe, elle sera affichée avec le nom de l’objet.

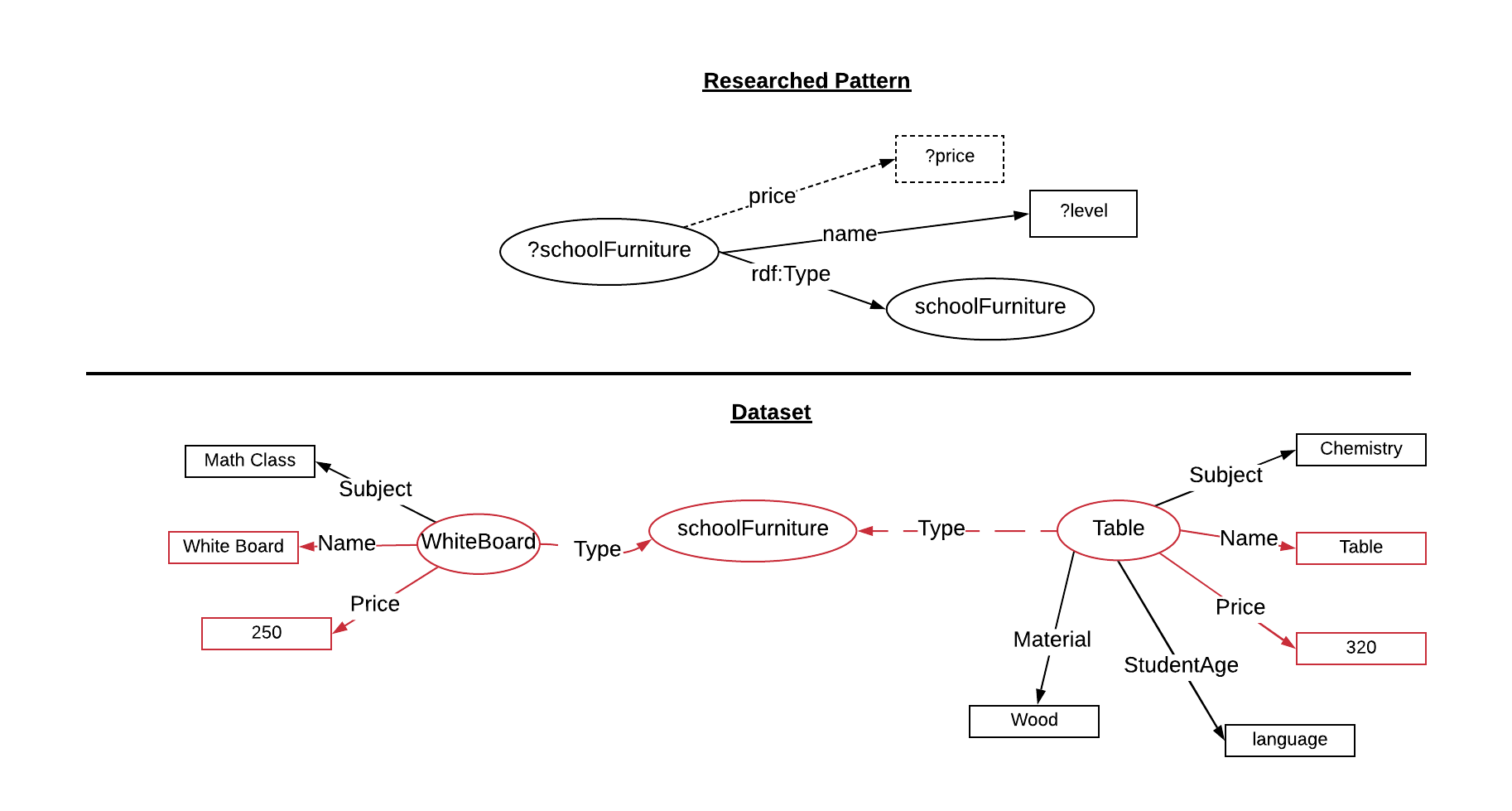


Figure 8 : Construction requête SPARSQL

La requête SPARSQL pour résultat est la suivante :

**PREFIX** rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

**PREFIX** rdfs: <http://www.w3.org/2000/01/rdf-schema#>

**PREFIX** lri: <http:/lri.ch/>

**SELECT** ?name ?price

**WHERE** {

?schoolFurniture rdf:type lri:SchoolFurniture .

?schoolFurniture lri:name ?name .

**OPTIONAL** {

?schoolFurniture lri:price ?price .

}

}

Le résultat obtenu avec les données du triple store utilisé pour ce projet est montré ce de suite.

|  |  |
| --- | --- |
| White Board | 250 |
| Table | 320 |

## Requête 2 : Get advance courses

Cette requête permet d’obtenir tous les cours de langue de niveau avancé (C1 et C2).

Pour trouver tous les cours recherchés il est nécessaire chercher dans le « Triple Stores » le pattern illustré par la figure 9. Le « pattern » permet de trouver tous les cours de l’école ayant la propriété ***level***. Pour sélectionner seulement les cours de niveau avancé, un filtre est appliqué.

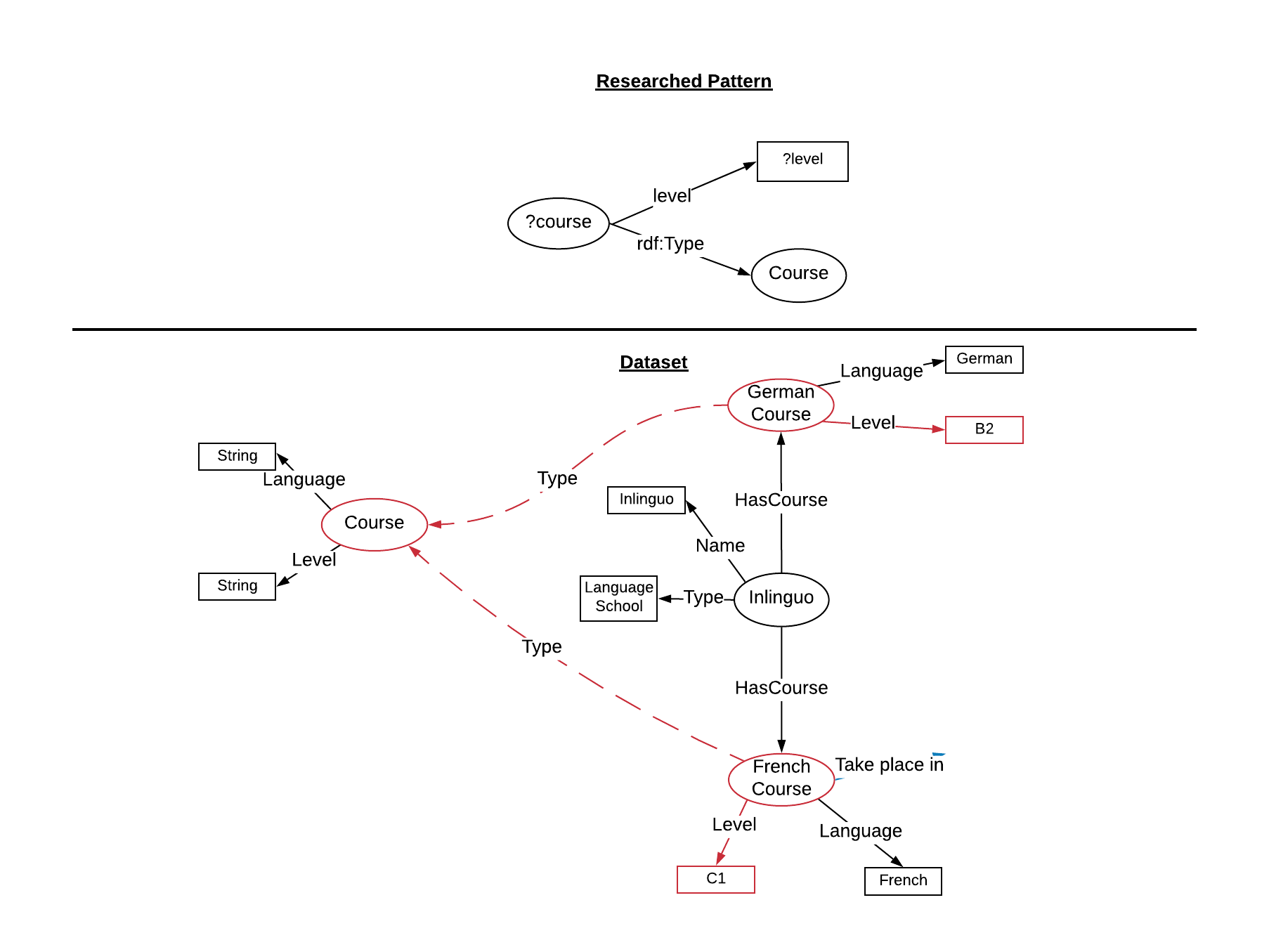


Figure 9 : Construction requête SPARSQL

La requête SPARSQL pour résultat est la suivante :

**PREFIX** rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

**PREFIX** rdfs: <http://www.w3.org/2000/01/rdf-schema#>

**PREFIX** lri**: <http://lri.ch/>**

**SELECT** ?course ?level

**WHERE** {

?course rdf:type lri:Course .

?course lri:level ?level .

**FILTER** (?level = "C1" || ?level = "C2")

}

Le résultat obtenu avec les données du triple store utilisé pour ce projet est montré ce de suite.

|  |  |
| --- | --- |
| FrenchCourse | C1 |

## Requête 3 : Get number of people in school

Cette requête permet d’obtenir le nombre de personne potentiellement présentes dans l’école.

Pour trouver ce chiffre il est nécessaire chercher tous les professeurs ainsi les étudiants de l’école.

Ensuite il est nécessaire de compteur le nombre de « tuples » qui correspondent au « pattern » indiqué par la figure 10.

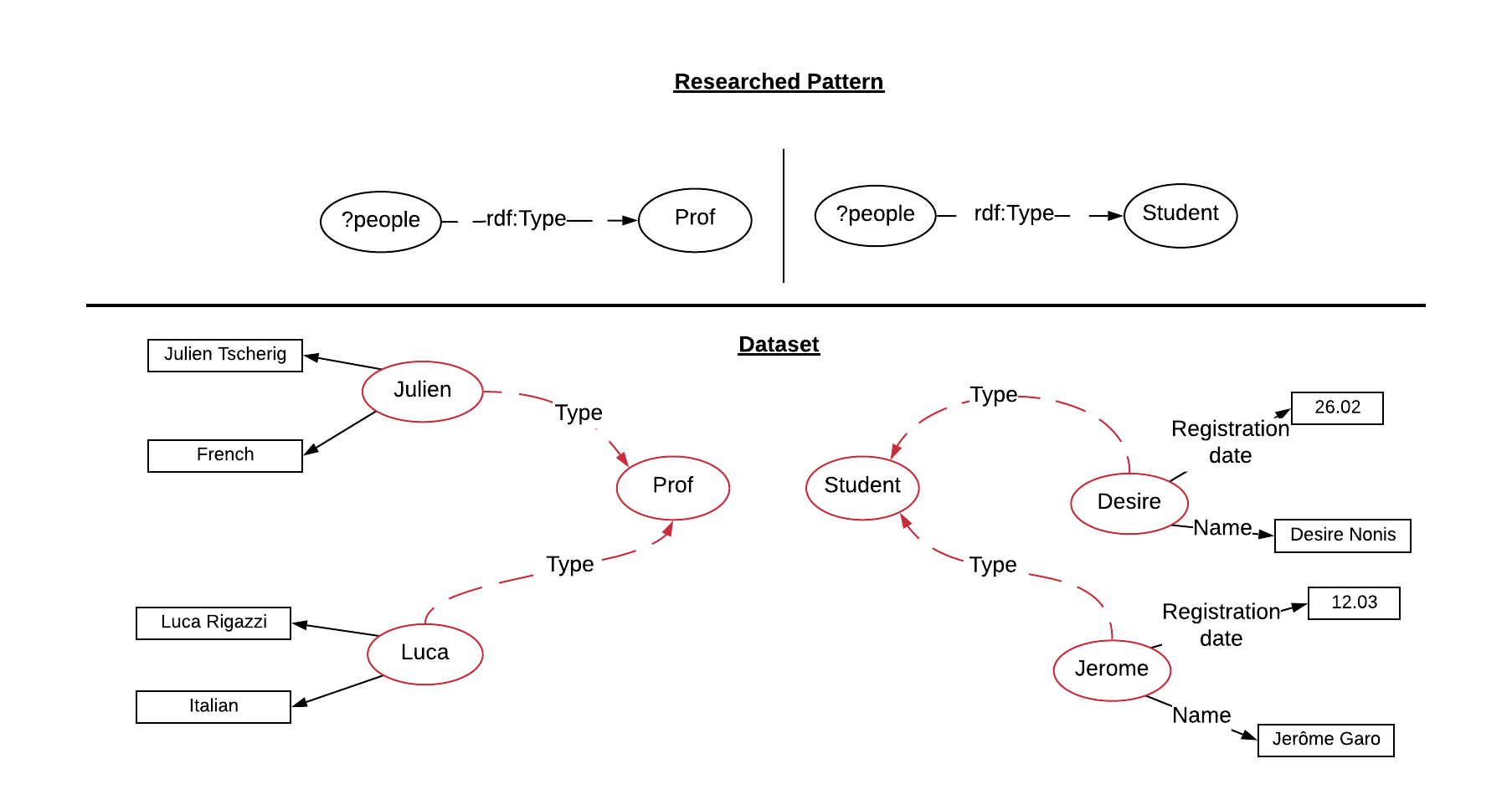


Figure 10 : Construction requête SPARSQL

La requête SPARSQL pour résultat est la suivante :

**PREFIX** rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

**PREFIX** rdfs: <http://www.w3.org/2000/01/rdf-schema#>

**PREFIX** lri**:** <http://lri.ch/>

**SELECT** (count(?people) as ?numberOfPeople)

**WHERE** {{?people rdf:type lri:Student} **UNION** {?people rdf:type lri:Prof}}

Le résultat obtenu avec les données du triple store utilisé pour ce projet est montré ce de suite.

|  |
| --- |
| 4 |

## Requête 4 : Get latin badsed language courses

Cette requête permet d’obtenir les cours basés sur une langue avec une racine latine.

Dans ce cas français, italien et espagnol.

Les triples doivent correspondre à un des « patterns » indiqué à la figure 11. Cette dernière permet de montrer que le cours d’allemand satisfait que la partie du « pattern » décrivant le type mais non celle de la propriété data. Pour cette raison il ne résultera pas dans le résultat de la requête SPARSQL.

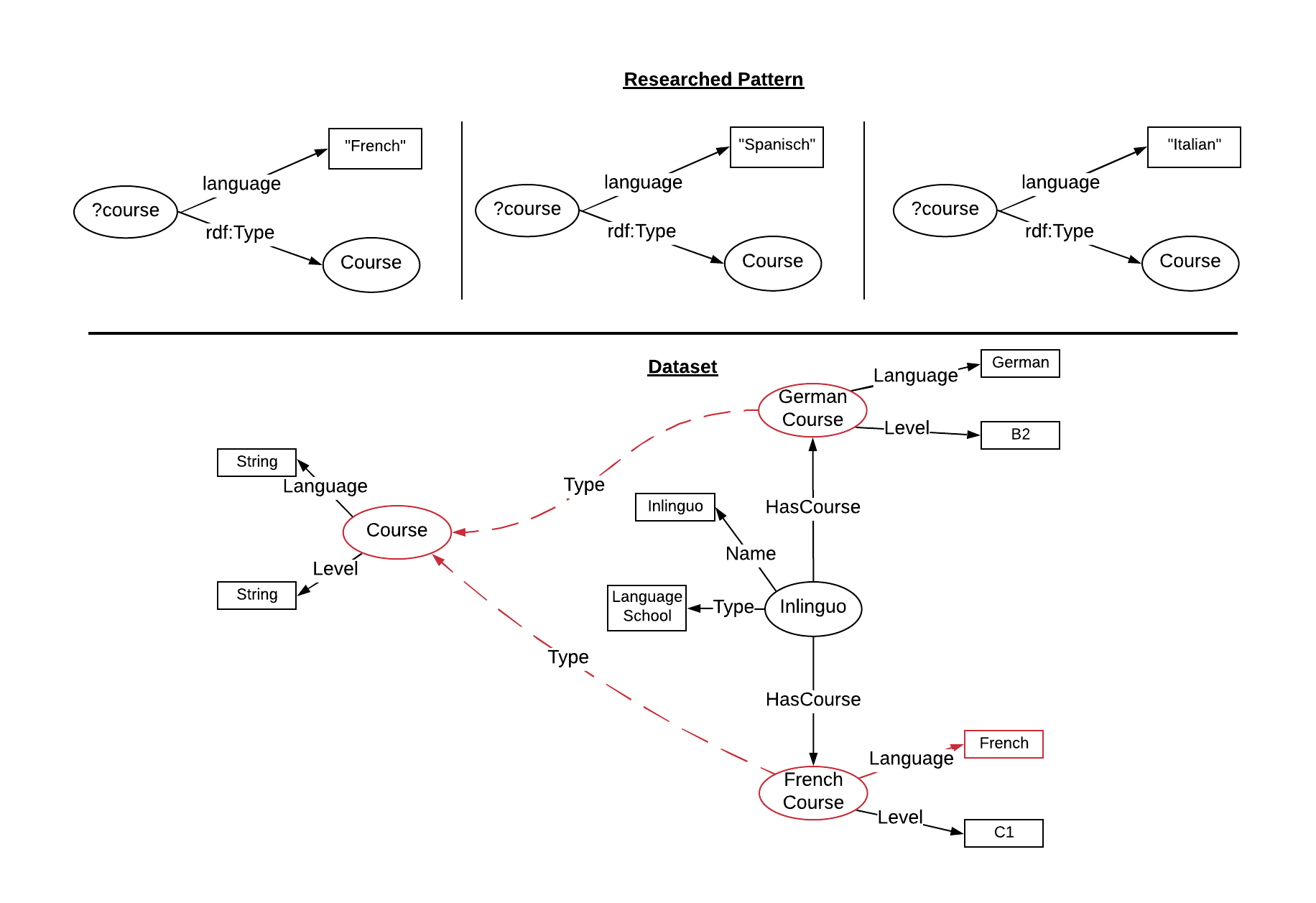


Figure 11 : Construction requête SPARSQL

La requête SPARSQL pour résultat est la suivante :

**PREFIX** rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

**PREFIX** rdfs: <http://www.w3.org/2000/01/rdf-schema#>

**PREFIX** lri**:** <http://lri.ch/>

**SELECT** (count(?people) as ?numberOfPeople)

**WHERE** { {?course rdf:type lri:Course . ?course lri:language "French" }

**UNION**

{?course rdf:type lri:Course . ?course lri:language "Italian"}

**UNION**

{?course rdf:type lri:Course . ?course lri:language \"Spanisch\"}

}

Le résultat obtenu avec les données du triple store utilisé pour ce projet est montré ce de suite.

|  |
| --- |
| FrenchCourse |

## Requête 5 : Get teached languages

Cette requête permet d’obtenir toutes les langues enseignées dans l’école. Le « pattern » de la figure 12 permet de trouver tous les cours de langue. Si plusieurs cours de la même langue existent ils apparaitront dans le résultat de la requête. Pour supprimer le doublon il est possible employer le « MODIFIER» « DISTINCT ».

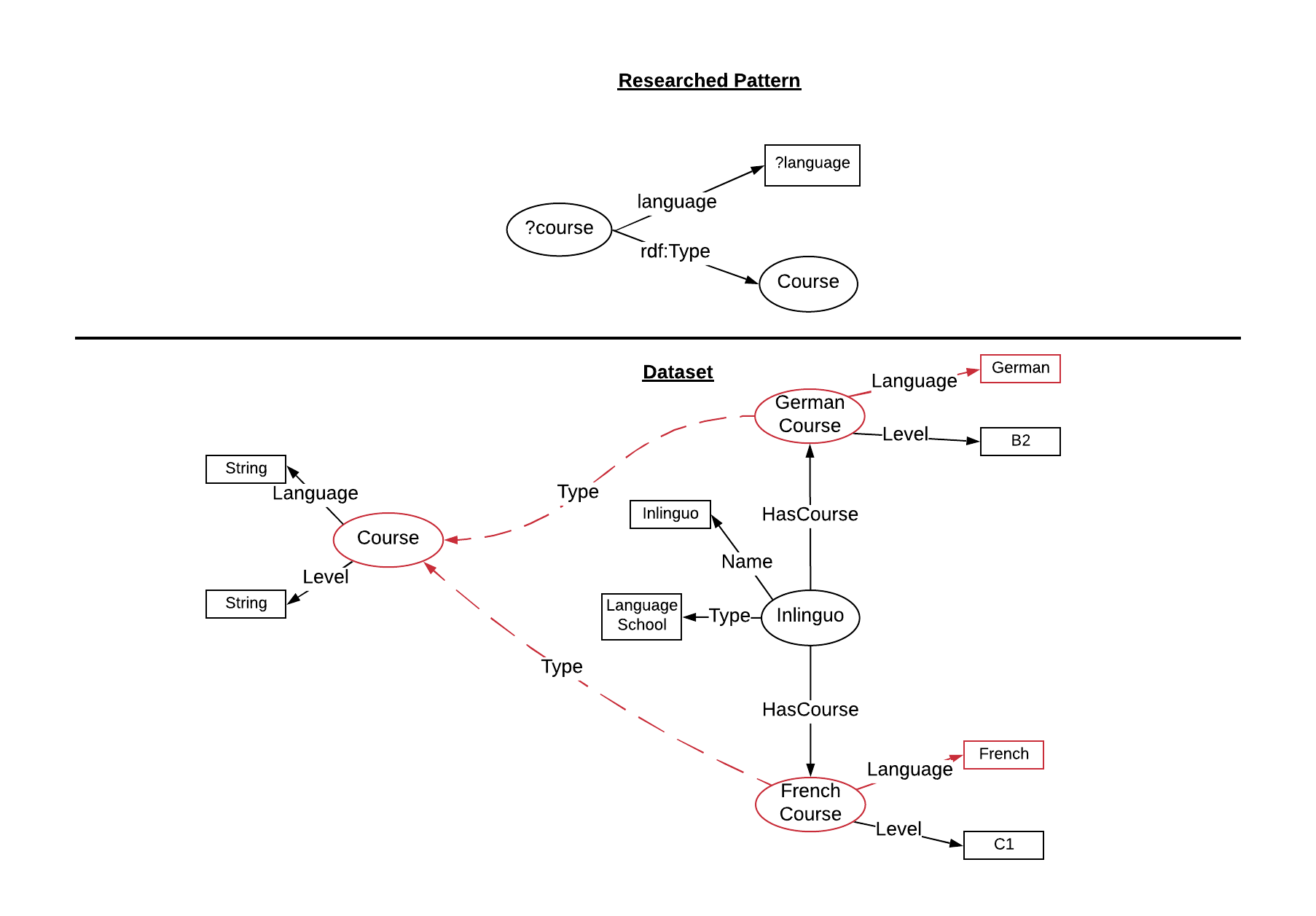


Figure 12 : Construction requête SPARSQL

La requête SPARSQL pour résultat est la suivante :

**PREFIX** rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

**PREFIX** rdfs: <http://www.w3.org/2000/01/rdf-schema#>

**PREFIX** lri**:** <http://lri.ch/>

**SELECT DISTINCT** ?language

**WHERE** {

?course rdf:type lri:Course .

?course lri:language ?language .

}

Le résultat obtenu avec les données du triple store utilisé pour ce projet est montré ce de suite.

|  |
| --- |
| German |
| French |

## Get max student capacity

Cette requête permet d’obtenir la capacité maximale d’étudiant que l’école peut accueillir en fonction de la capacité des salles de cours. La figure 13 montre le « pattern » nécessaire à obtenir la capacité de chacune des salles de cours de l’école ayant la propriété ***capacity***. La somme de ces résultats permet de trouver la capacité maximum de l’école.

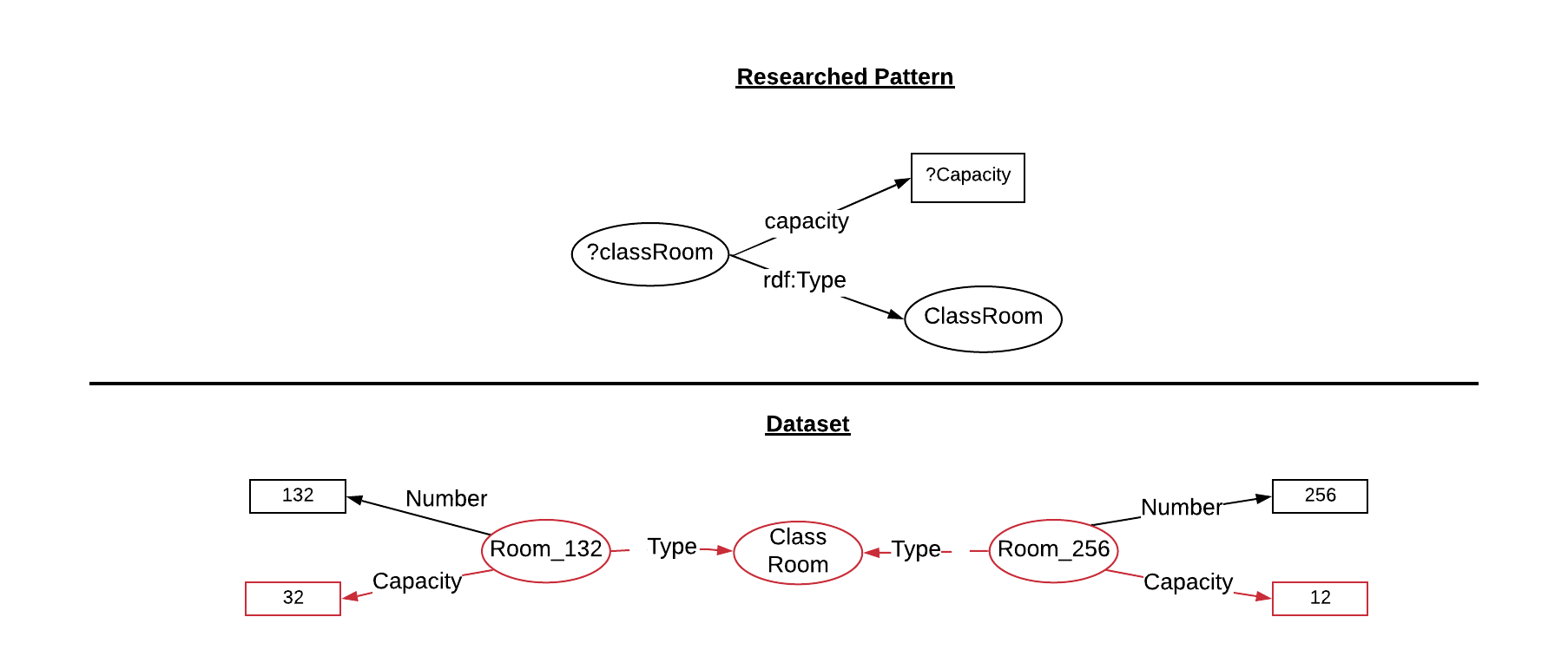


Figure 13 : Construction requête SPARSQL

La requête SPARSQL pour résultat est la suivante :

**PREFIX** rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

**PREFIX** rdfs: <http://www.w3.org/2000/01/rdf-schema#>

**PREFIX** lri**:** <http://lri.ch/>

**SELECT** (**SUM**(?capacity) as ?totalCapacity)

**WHERE** {

?classRoom rdf:type lri:ClassRoom .

?classRoom lri:capacity ?capacity .

}

Le résultat obtenu avec les données du triple store utilisé pour ce projet est montré ce de suite.

|  |
| --- |
| 44 |

# Partie 8

## Site 1 : hotel – booking.com

url : <https://www.booking.com/hotel/ch/microhotel.fr.html>

Les “data properties” affichées en **gras** correspondent aux valeurs trouvées sur le site. Les champs trop longs ont été abrégés par une description en *italique*.

Le site en question est un site connu de réservation en ligne. Le contenu proposé concerne les hôtels disponibles dans le monde entier. La figure 14 montre le graphe des « linked-data » selon le schéma présent sur « Schema.org » pour un hôtel en particulier. La figure 15 montre le résultat dans la recherche de google. Il est possible remarquer que google n’utilise pas forcement les donné spécifié par la propriété objet aggregationRating. Cette dernière défini la note de l’hotel selon les critères de booking alors que dans le « snippet » proposé par google, une note basée sur une autre échelle est proposée. Les données décrivant l’adresse, le nom de l’hôtel et l’implacement corrispondent à celles présentes dans le graphe.

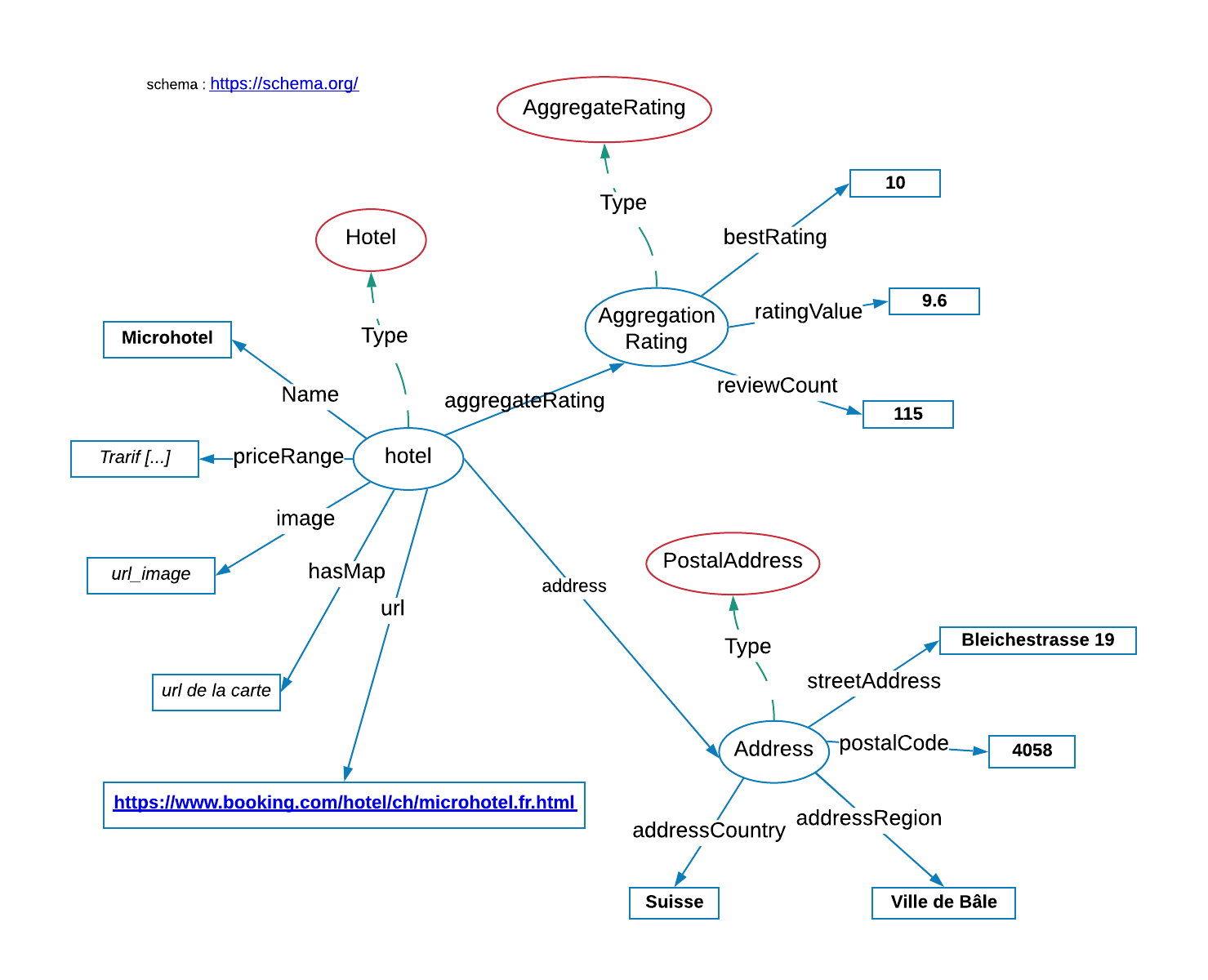


Figure 14 : graphe booking.com basé sur schema.org

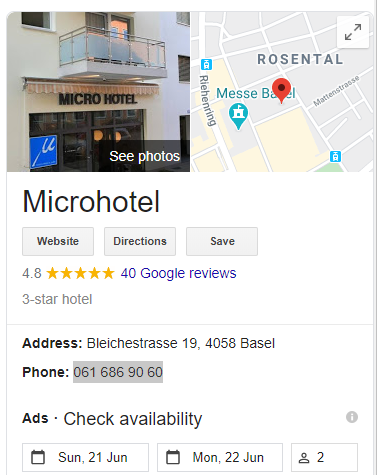


Figure 15: Rendu dans la page de recherche de google

## Site 2 Recipe : marmiton.org

url : <https://www.marmiton.org/recettes/recette_camembert-a-la-braise_14609.aspx>

Les “data properties” affichées en **gras** correspondent aux valeurs trouvées sur le site. Les champs trop longs ont été abrégés par une description en *italique*.

Le site en question est un site connu de recettes de cuisine en ligne. Le contenu proposé concerne des recettes de cuisine de tous genres. La figure 16 montre le graphe des « linked-data » selon le schéma présent sur « Schema.org » pour une recette en particulier. Le graphe présenté permet à une machine de « comprendre » qu’il s’agit d’une recette de cuisine et de spécifier les détails de cette dernière. Grace à ces données, google est capable de présenter un résumé comme montré par la figure 17. Il est possible remarquer que la note, une partie de la démarche, l’image et le temps de préparation affiché par google correspondent à ceux décrites dans les « linked-data » présentes sur le site.

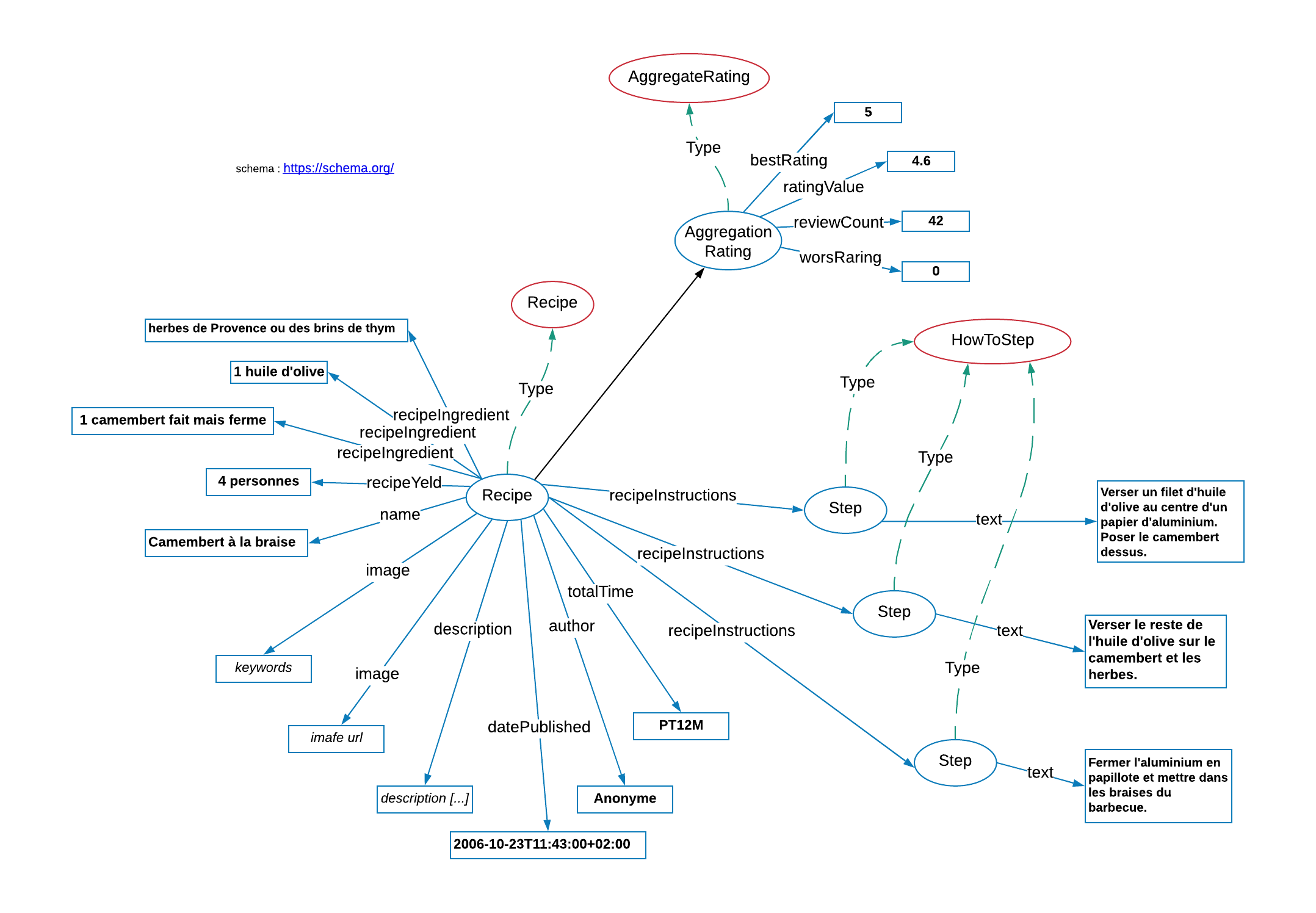


Figure 16 : graphe marmiton.org basé sur schema.org



Figure 17 : Rendu dans la page de recherche de google

## Site 3 : Music event : songkick.com

Url : <https://www.songkick.com/concerts/39491482-wishbone-ash-at-z7-konzertfabrik?utm_medium=organic&utm_source=microformat>

Les “data properties” affichées en **gras** correspondent aux valeurs trouvées sur le site. Les champs trop longs ont été abrégés par une description en *italique*.

Après le schéma « Hotel » et « Recipe » montrés par les chapitres précédents, ce site permet de montrer un exemple d’utilisation du schéma de type MusicEvent. Grace aux schéma présenté, un moteur de recherche comme google est capable d’extraire les « linked-data » et les afficher à l’utilisateur de la façon montré par la figure 18. Il est possible observer que les data Group.Name, Address et startDate sont présentés sous la forme montré figure 19.

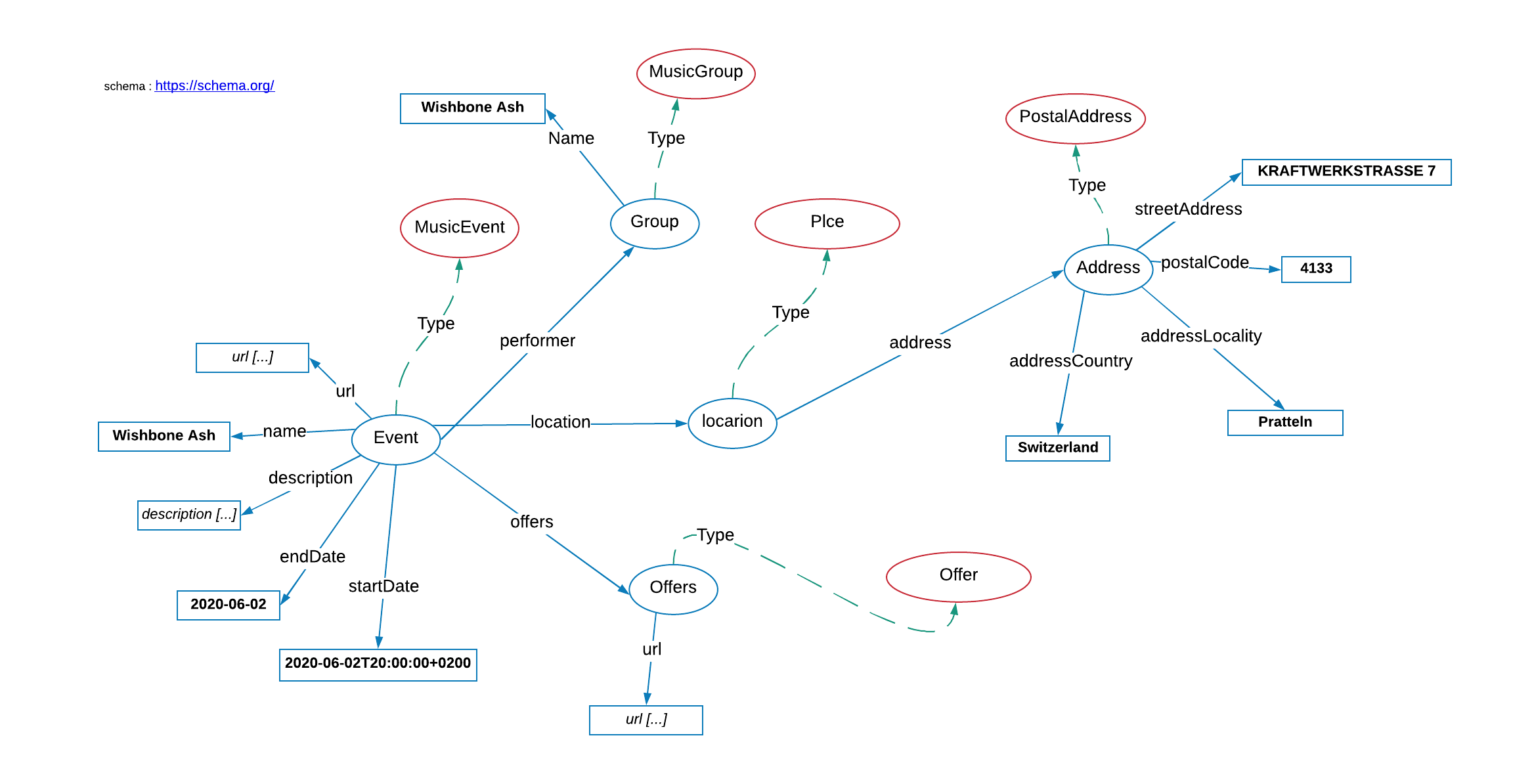


Figure 18 : graphe songkick.com basé sur schema.org

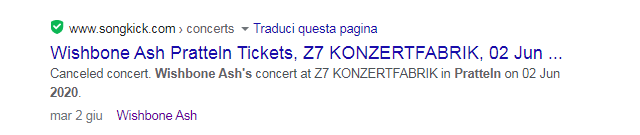


Figure 19 : Rendu dans la page de recherche de google

# Concluions

Un projet structuré de la sorte, permet d’exercer à tous les niveaux et donc très appréciable. Pour les partie 1, 2, 3 et 5 j’ai trouvé la taille de l’ontologie demandée correcte pour pouvoir afficher et exercer différentes propriétés objet, sous classe , sous propriété etc…  
Pour ce qui concerne la partie 4, j’e l’ai trouvé très répétitive et très longue à dessiner. J’aurais aimé employer ce temps pour exercer des requêtes SPARSQL sur un site comme Dbpedia. Ceci aurait de plus permis d’effectuer des requêtes plus compliquées que celle possible avec la partie 7. Pour ce qui concerne le cours, je trouve qu’une introduction au web des données est nécessaire dans une formation comme le MAS-RAD. Malgré la situation non idéale, j’ai trouvé que le déroulement du cours a été très satisfaisant.