Realisation d’une ontologie pour une organisation de gestion des compétitions

I – Etats des lieux :

1. Spécifications
2. Les besoins fonctionnels

On tient a realiser une ontologie pour une organisation qui gère des compétitions , avec les besoins suivants :

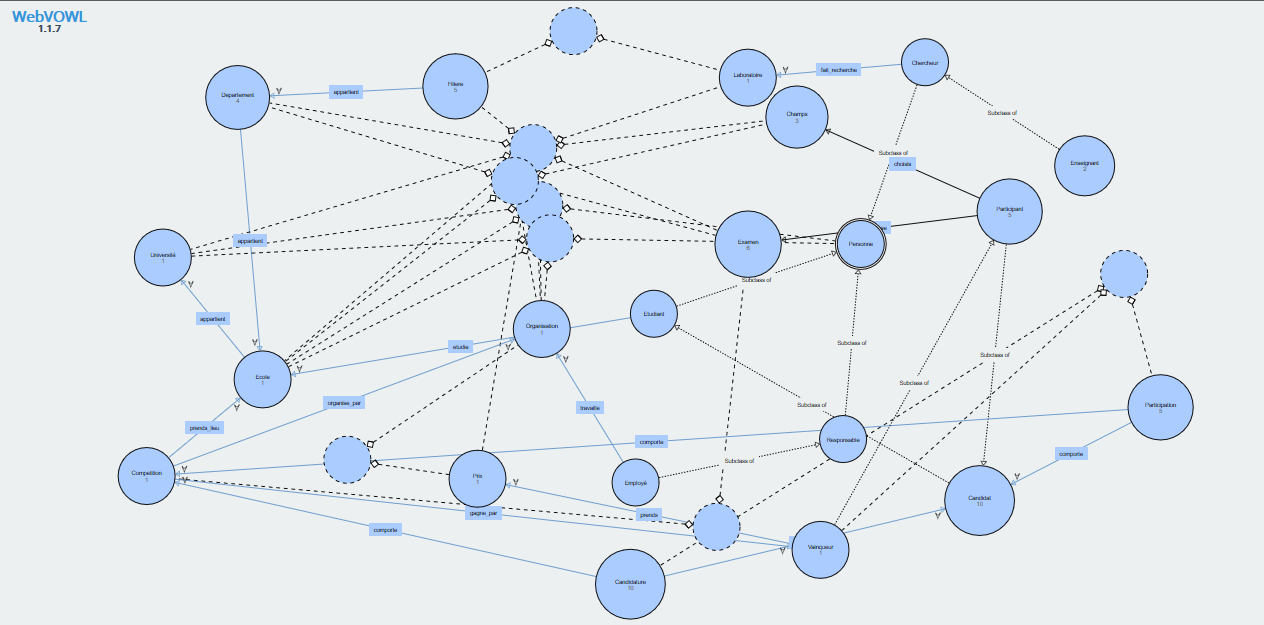
* Une organisation est géré par un a plusieurs responsables
* Un responsable peut etre un fondateur , co-fondateur , adherent ou un membre d’honneur
* Le fondateur crée l’organisation
* L’organisation fait travailler de un a plusieurs employés
* Un employé travaille a une seule organisation
* Un employé peut etre un ingenieur ou un technicien
* Un employé est lui aussi un responsable
* Un responsable est une personne
* Une personne connait une ou plusieurs autres personnes
* Une organisation peut organiser un a plusieurs compétitions
* Une compétition est géré par seule organisation
* Une compétition se déroule en une a plusieurs journées
* Une compétition propose un a plusieurs champs
* *(un champs c-a-d le domaine dont le participant veut passer l’examen )*
* Un champs contient de un a plusieurs examens
* Un etudiant est une personne
* Un etudiant etudie a une seule ecole
* Un etudiant appartient a une seule filiere
* Une ecole contient de un a plusieurs départements
* Un departement appartient a une seule ecole
* Un departement contient de un a plusieurs filières
* Une filiere appartient a un seule departement
* Une ecole appartient a une seule université
* Une université comporte de un a plusieurs ecoles
* Un chercheur est une personne
* Un chercheur fait la recherche dans un seul laboratoire
* Un laboratoire contient de un a plusieurs chercheurs
* Un enseignant est un chercheur
* Un enseignant enseigne dans un a plusieurs ecoles
* Un laboratoire appartient a une seule ecole
* Une ecole contient de un a plusieurs laboratoires
* Un candidat est un etudiant
* Un candidat dépose une seule a plusieurs candidatures
* Un candidature comporte une seule compétition
* Un participant est un candidat
* Un participant participe une a plusieurs participations
* Une participation comporte une seule compétition
* Un participant passe un seul examen
* Un vainqueur est un participant
* Un vainqueur gagne une a plusieurs compétitions
* Une compétition est gagné par un seul vainqueur
* Un vainqueur prends un seul prix

On tient aussi a avoir les attributs suivantes

* Une personne avec *: nom\_famille , prenom , cin , datenaissance , ville , pays , genre , age , titre , email*
* Un responsable avec : *emploi*
* Un etudiant avec : *cne , niveau\_etudes*
* Une organisation avec : *nom , pays , type , domaine , date\_creation , ville*
* Une compétition avec : *periode* , *cible* , *durée*
* Une journée avec : *date* , *activité*
* Un champs avec : *nom* , *domaine*
* Un examen avec : *nom* , *domaine* , *durée*
* Un prix avec : *forme* , *nom* , *type* , *valeur*
* Un vainqueur avec : *saison*
* Une candidature , participation avec : *saison*
* Une ecole avec : *nom , domaine , ville , pays*
* Une université avec : *nom , ville , pays*
* Une departement avec : *nom , domaine*
* Une filiere avec : *nom , domaine , specialité*
* Un laboratoire avec *: nom , specialité*

1. Conception
2. Le graphe

*// Merci de le visualiser encore avec protégé car le graphe est plus grand d’etre visible en seul image*

Visualisation avec WebVOWL :

On a choisit WebVOWL pour sa réduction des instances afin d’assurer une meilleur visualisation

*// Vous trouverez l’image du graphe avec WebVOWL et Protégé*

II – Description d’ontologie :

1. Les classes
2. La liste des classes utilisés

Organisation

Ecole

Université

Personne : Responsable : ( Fondateur , Co-Fondateur , Adherent , Membre d'honneur ), Employé : (Technicien , Ingenieur) , Chercheur : (Enseignant) , Etudiant : Candidat : Participant : Vainqueur

Université

Departement

Filiere

Prix

Champs

Examen

Journée

Compétition

Candidature

Participation

Laboratoire

1. Caractéristiques des classes

* Etudiant et Enseignant sont disjointes , on peut pas avoir héritant des deux

*// respectés avec les restrictions*

* un Candidat est un Etudiant qui dépose Candidature
* un Participant est un Candidat qui participe a une Participation
* un Vainqueur est un Participant qui gagne une Compétition
* un Chercheur est une Personne qui fait la recherche a un Laboratoire
* un Enseignant est un Chercheur qui enseigne a Ecole

1. Les Propriétés
2. La liste des propriétés utilisés

*// Les cardinalités sont respectés avec les restrictions avec des commandes* ***Manchester***

Personne (connait only Personne) or (connait some Personne)

Chercheur fait recherche a only Laboratoire

Etudiant etudie a only Filiere

Organisation (gere par only Responsable ) or (gere par some Responsable )

Responsable (gere only Organisation ) or (gere some Organisation )

Fondateur (cree only Organisation ) or (cree some Organisation )

Employé travaille a only Organisation

Organisation (fait travailler only Employé ) or (fait travailler some Employé )

Organisation (organise only Competition) or (organise some Competition)

Competition organise par only Organisation

Competition (se deroule en only Journée) or (se deroule en some Journée)

Competition (propose only Champs) or (propose some Champs)

Champs (contient only Examen) or (contient some Examen)

Competition gagné par only Vainqueur

Vainqueur prends only Prix

Vainqueur (gagne only Competition) or (gagne some Competition)

Candidature comporte only Competition

Competition (comporte only Candidature) or (comporte some Candidature)

Participation comporte only Competition

Competition (comporte only Participation) or (comporte some Participation)

Candidat (depose only Candidature) or (depose some Candidature)

Candidature comporte only Candidat

Candidat ( participe a only Participation) or (participe a some Participation)

Participation comporte only Candidat

Etudiant etudie a only Ecole

Ecole appartient a only Université

Université (comporte only Ecole) or (comporte some Ecole)

Ecole (comporte only Departement) or (comporte some Departement)

Departement appartient a only Ecole

Departement (comporte only Filiere) or (comporte some Filiere)

Filiere appartient a only Departement

Competition prends lieu only Ecole

Enseignant (enseigne a only Ecole) or (enseigne a some Ecole)

Laboratoire appartient a only Ecole

Ecole (contient only Laboratoire) or (contient some Laboratoire)

Laboratoire (comporte only Chercheur) or (comporte some Chercheur)

1. Caractéristiques des propriétés

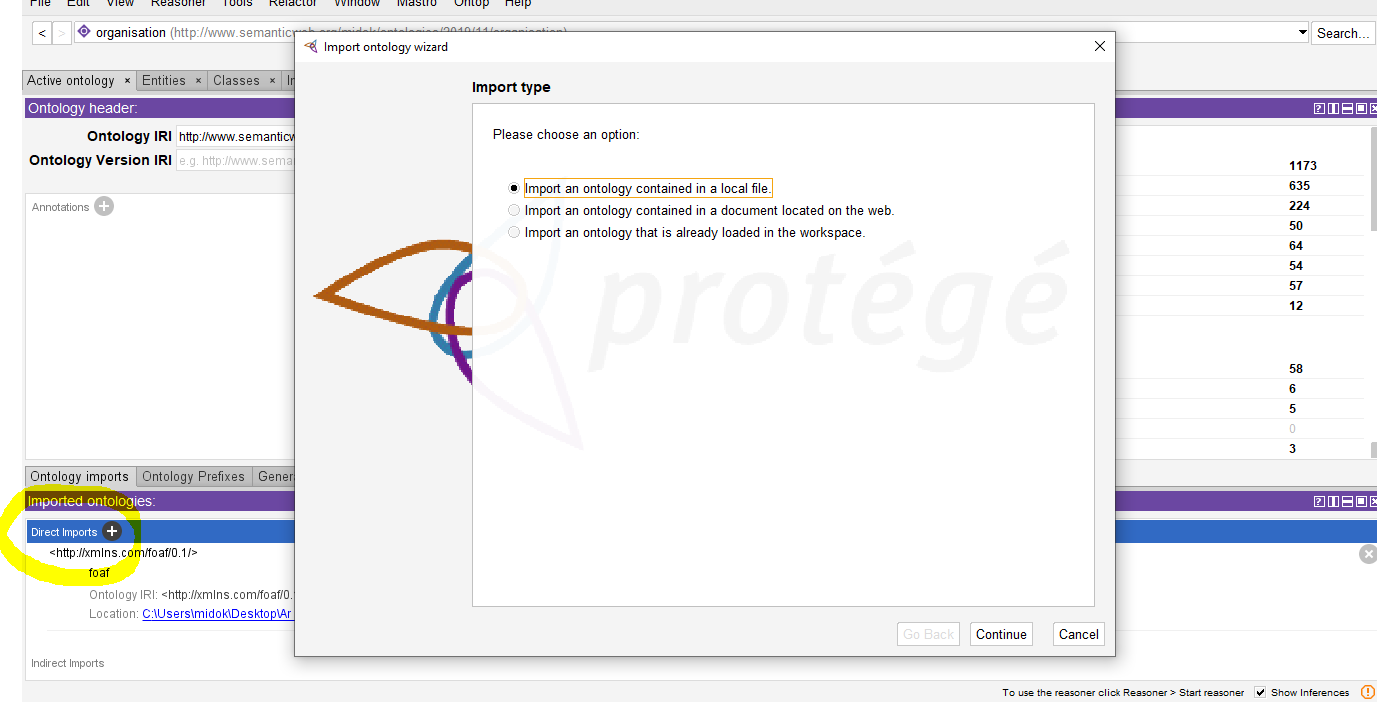
* gere et gere\_par sont inversés
* travaille et fait\_travailler sont inversés
* organise et organise par sont inversés
* gagne et gagné par sont inversés
* appartient et comporte sont disjointes
* enseigne et etudie sont disjointes

1. Les Propriétés de données
2. La liste des propriétés de données utilisés

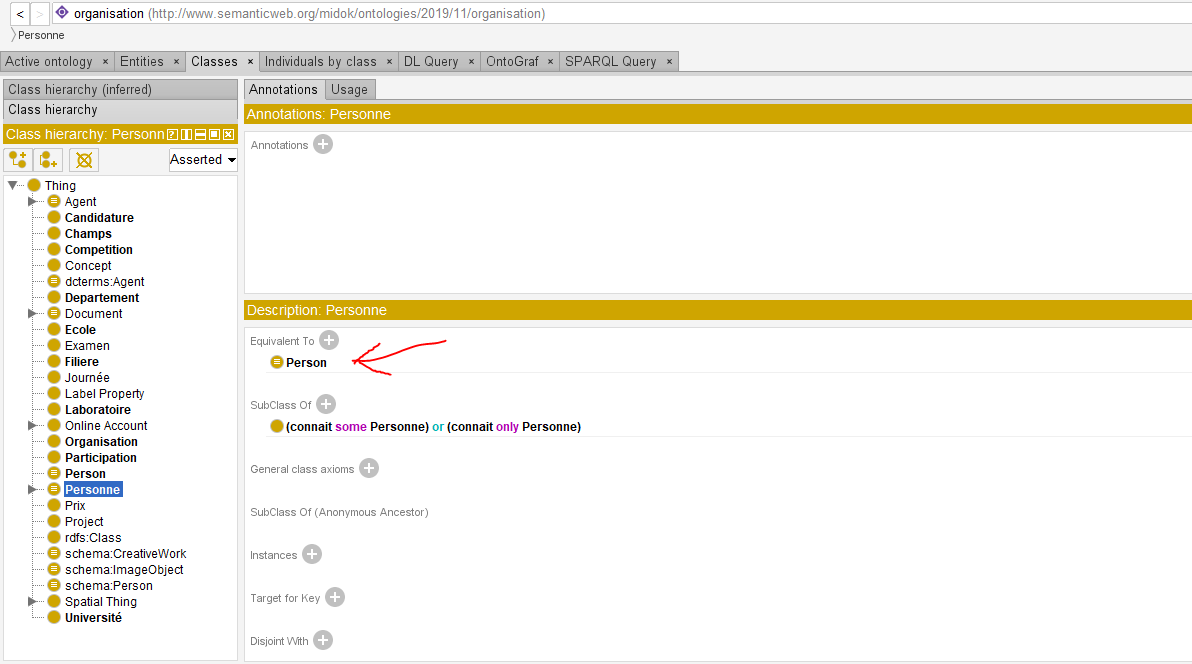
Sont les attributs cités précédemment contenant comme **domain** les classes a qu’elle ils appartiennent , et comme **range** les types de données correspondants ( String , Integer .. etc )

III - Integration de FOAF

La premiere etape est de telecharger de fichier RDF de l’ontologie sociale FOAF et de l’intégrer dans notre ontologie

* Sous Protégé on va choisir direct import , et importer le fichier RDF de FOAF

L’ontologie sociale de FOAF contient plusieurs classes propriétés d’objets et de données , le but ici c’est de l’intégrer , c’est pourquoi on a choisit de faire **l’équivalence** entre nos classes , propriétés d’objets et propriétés de données avec ceux necessaire de FOAF

* Dans un objet donnée , on choisit équivalent a et on precise l’objet correspondant

1. Les classes équivalents

* Personne ⬄ **foaf :Person**

1. Les propriétés d’objets équivalents

* Connait ⬄ **foaf :knows**

1. Les propriétés de données équivalents

* age ⬄ **foaf :age**
* datenaissance ⬄ **foaf :birthday**
* genre ⬄ **foaf :gender**
* prenom ⬄ **foaf :lastName**
* nom\_famille ⬄ **foaf :name**
* titre ⬄ **foaf :title**

*// La raison de faire l’équivalence c’est de garder le langage dont on a fait notre ontologie*

IV – Requetes SPARQL

*// Protégé ne prends en charge que les SPARQL query et pas les SPARQL updates c’est pourquoi on a utilisé le serveur* ***Apache Jena Fuseki*** *pour faire aussi les requetes d’update*

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

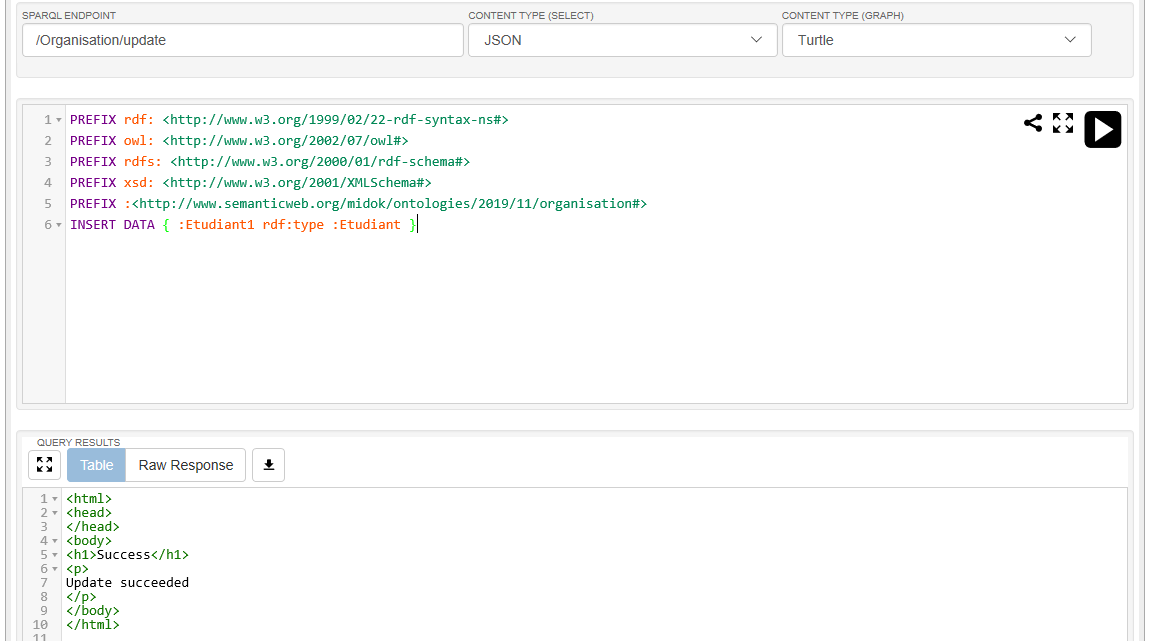
PREFIX owl: <http://www.w3.org/2002/07/owl#>

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

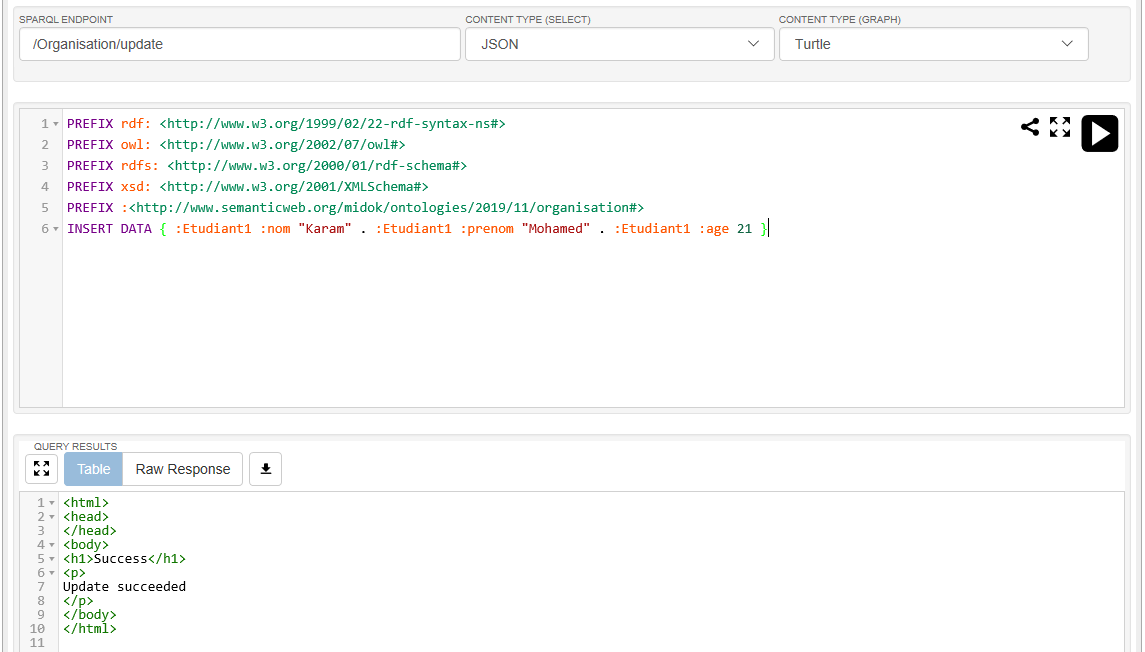
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX :<http://www.semanticweb.org/midok/ontologies/2019/11/organisation#>

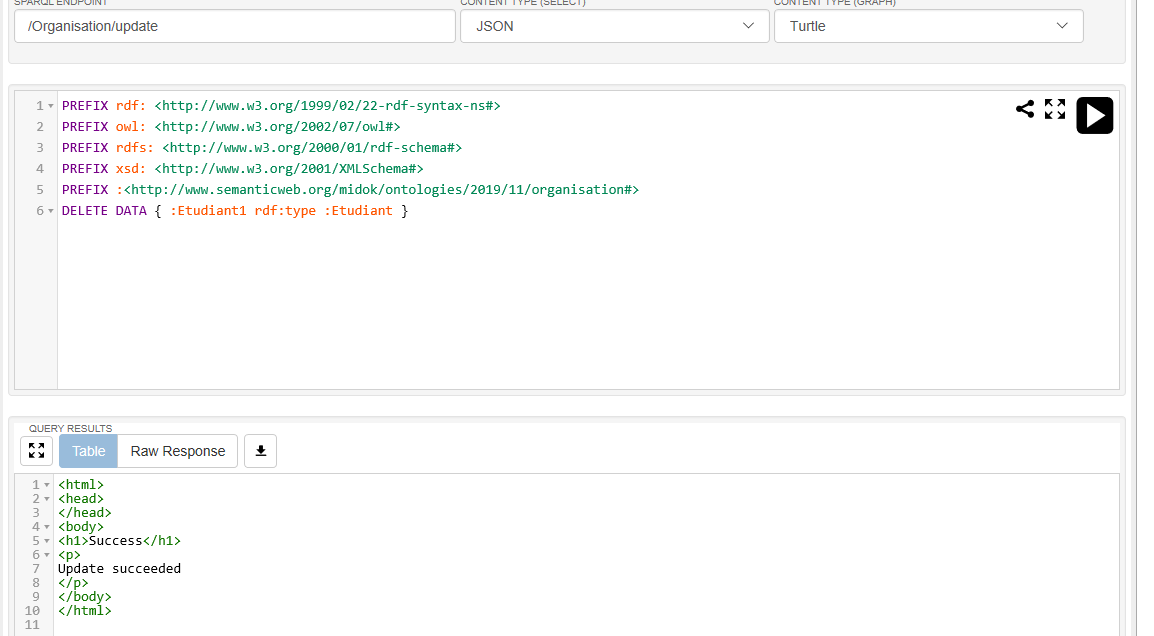
* Ajouter un etudiant :

INSERT DATA { :Etudiant1 rdf:type :Etudiant }

* Ajouter des propriétés de données a cet etudiant :

INSERT DATA { :Etudiant1 :nom "Karam" . :Etudiant1 :prenom "Mohamed" . : Etudiant1 :age 21 }

* Supprimer cet etudiant :

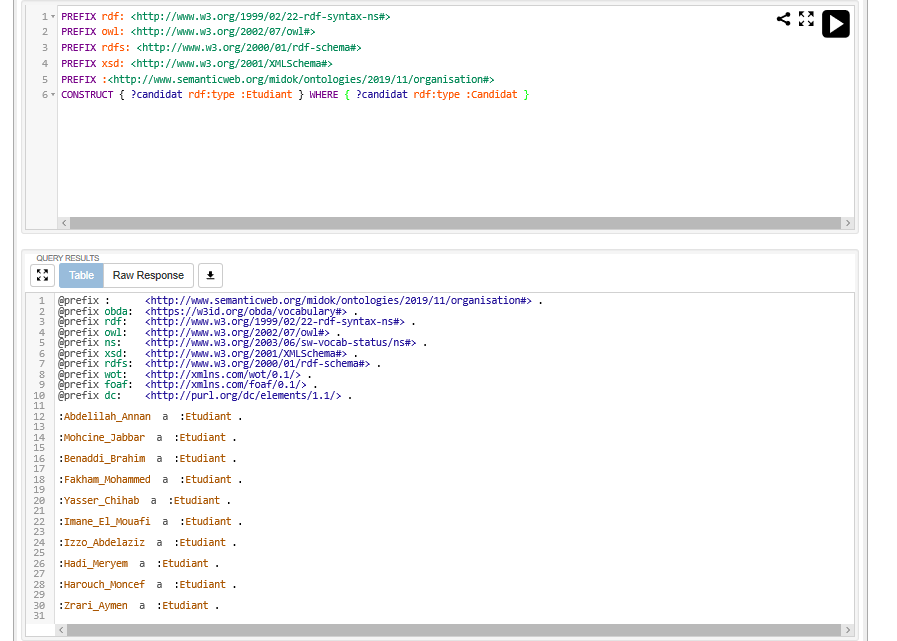
DELETE DATA { :Etudiant1 rdf:type :Etudiant }

* Verifier l’existence des etudiants :

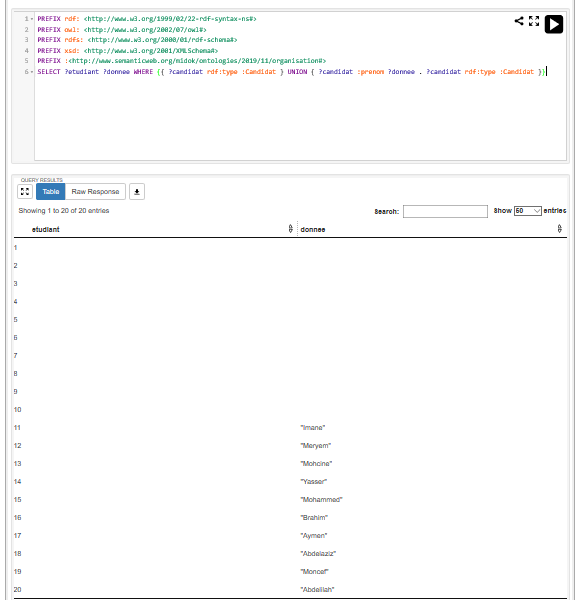
ASK { ?etudiant rdf:type :Etudiant }

* Construire un graphe d’un ensemble d’etudiants a partir des instances des candidats :

CONSTRUCT { ?candidat rdf:type :Etudiant } WHERE { ?candidat rdf:type :Candidat }



* Afficher l’union de deux requetes ( affichage des candidats ) + ( affichage de prénoms des candidats ) :

SELECT ?etudiant ?donnee WHERE {{ ?candidat rdf:type :Candidat } UNION { ?candidat :prenom ?donnee . ?candidat rdf:type :Candidat }}

* Afficher les participants dont les valeurs des prénoms sont ( Abdelilah , Imane ) :

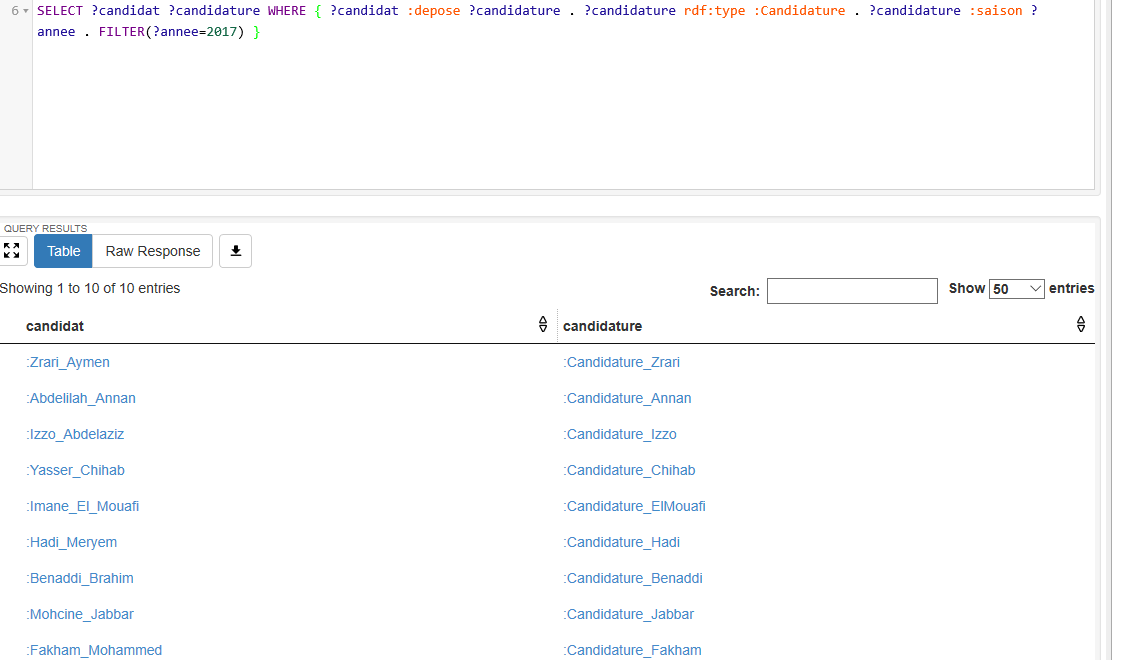
SELECT ?participant ?nom ?prenom WHERE { ?participant rdf:type :Participant . ?participant :nom\_famille ?nom . ?participant :prenom ?prenom} VALUES ?prenom { "Abdelilah" "Imane" }

* Afficher les candidats ayant déposés leurs candidatures en 2017

SELECT ?candidat ?candidature WHERE

{ ?candidat :depose ?candidature . ?candidature rdf:type :Candidature . ?candidature :saison ?annee

. FILTER(?annee=2017) }



* Afficher pour chaque examen passé , combien de fois il était choisis par un participant

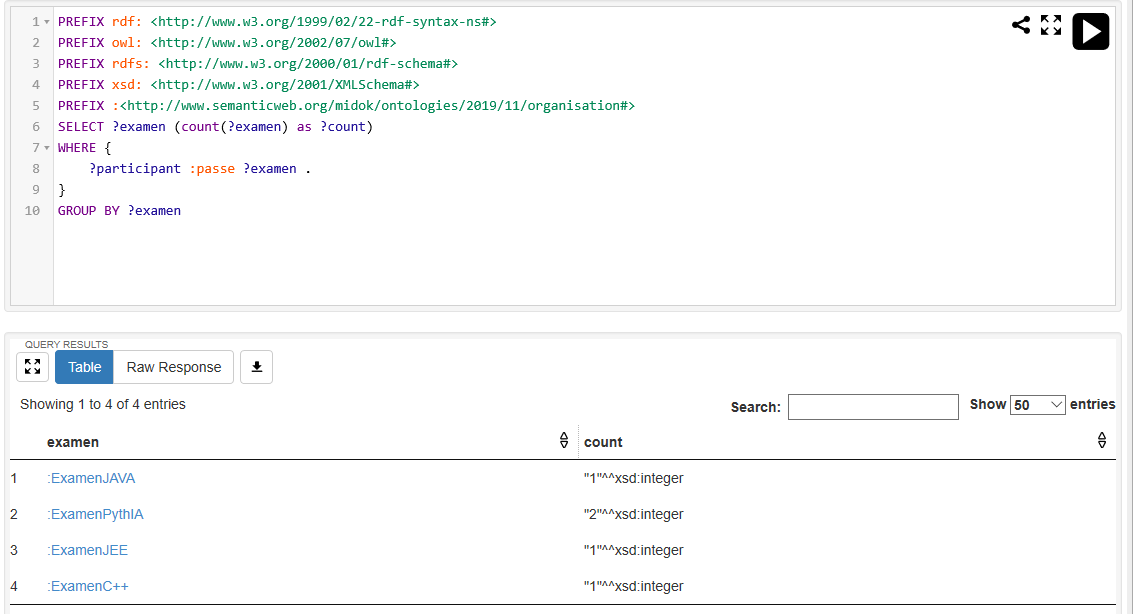
SELECT ?examen (count(?examen) as ?count)

WHERE {

?participant :passe ?examen .

}

GROUP BY ?examen



V – Time2Code :

Dans cette ontologie on s’est interessé a créer une compétition qui s’appele **Time2Code** qui est organisé par **GeniusADev** , pour cela on a crée plusieurs instances nécessaires pour l’édition de 2018 de cette compétition