Universidad Central de Venezuela
Facultad de Ciencias Escuela
de Computación
Licenciatura en Computación

Base de Datos NoSQL

Proyecto Fase 2:
Mundo Potter: Neo4J Edition

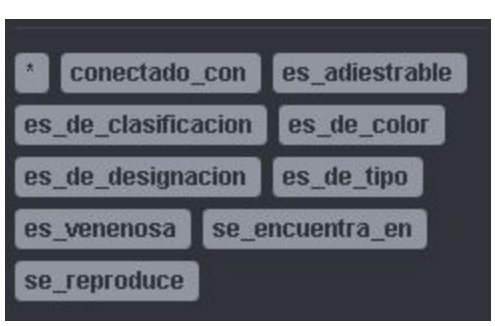
Abelardo Moreno C.I: 18.002.106
David Fernández C.I: 24.313.648
Alberto Suarez C.I: 23.107.093

- **Diseño de la BD**

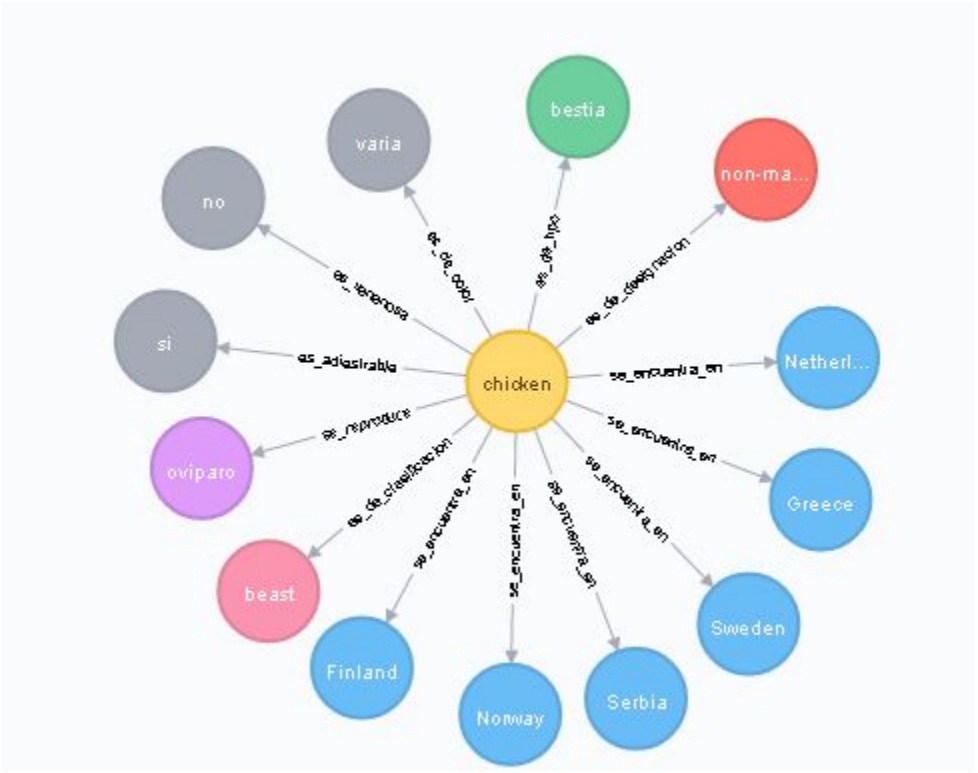
Para el diseño de la BD de Neo4j se decidió crear el siguiente tipo de nodos para poder facilitar el recorrido en las consultas requeridas.

El nodo Criaturas posee todas las propiedades de la criatura que se encontraban en MongoDB pero que no fueron transformadas a un tipo de nodo, mientras que los demás nodos sólo poseen una propiedad de nombre.

Además se crearon las siguientes relaciones



Con las que se realizan las conexiones entre los nodos Criatura con todos los demás y la conexión entre los nodos Región



- **Script**

Para la realización del script se decidieron usar las bibliotecas pymongo y py2neo gracias a su facilidad y gran documentación.

El script posee 4 grandes fases, la primera la de extracción, donde obtenemos todos los datos relevantes de la BD y los almacenamos en variables.

```
#obtenemos las criaturas, clasificacion, tipos, designacion, reproduccion, adiestrable, venenosa y color
for collection in collection.find():

    criaturas_lista.append(collection)
    clasificacion.add(collection.get("clasificacion",'unknown'))#si la criatura no tiene clasificacion, se coloca unknown por defecto
    tipo.add(collection.get("tipo",'unknown'))
    designacion.add(collection.get("designacion",'unknown'))
    reproduccion.add(collection.get("reproduccion",'unknown'))
    for i in range(len(criaturas_lista)):
        if "caracteristicas" in criaturas_lista[i]:

            c = criaturas_lista[i]["caracteristicas"][0]

        else:
            c={}
        adiestrable.add(c.get("adiestrable",'unknown'))
        venenosa.add(c.get("venenosa",'unknown'))
        color.add(c.get("color",'unknown'))
```

En la 2da fase debemos crear los nodos a partir de los datos extraídos

```
#se preparan los nodos de color
for i in range(len(color_lista)):

    nodo=Node("Color",color=color_lista[i])
    color_nodos.append(nodo)

#se preparan los nodos de criaturas
for i in range(len(criaturas_lista)):

#creacion de los nodos criatura,clasificacion, tipo, designacion, reproduccion, adiestrable, venenosa y color
for j in range(len(clasificacion_lista)):
    sgraph.create(clasificacion_nodos[j])
```

En la 3era fase creamos las conexiones entre los nodos, basándose en el modelo de la BD que teníamos en MongoDB

```
#creacion de las relaciones entre criaturas y clasificacion
for i in range(len(criaturas_lista)):
    criatura_relacion = criaturas_nodos[i]
    rela=criaturas_lista[i].get("clasificacion","unknown")
    for j in range(len(clasificacion_lista)):
        if clasificacion_lista[j]==rela:
            clasificacion_relacion = clasificacion_nodos[j]
            relacion = Relationship(criatura_relacion, "es_de_clasificacion", clasificacion_relacion)
            sgraph.create(relacion)
```

Por último en la 4ta fase es donde creamos los nodos Región, los cuales son generados en el script y no son extraídos de mongo, luego esos nodos son asignados de forma aleatoria en una relación con los nodos Criatura, tomando en cuenta las validaciones del enunciado

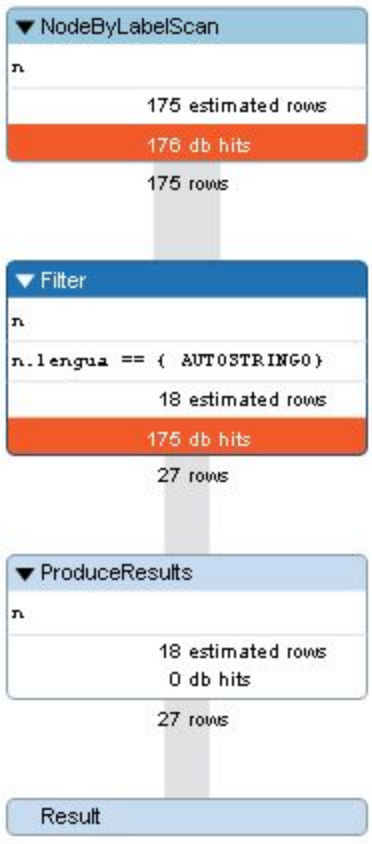
```
#creacion de los paises de europa
for i in paises:
    nodo=Node("Region",nombre=i)
    region_nodos.append(nodo)
    sgraph.create(nodo)

#agregar criaturas a paises de forma random
for i in range(len(criaturas_lista)):
    lista=[]
    verificar=0
    rango=random.randint(0,10)
    for j in range(rango):
        numero=random.randint(0,40)
        if((numero not in lista) and (contadores[numero] <10)):
            lista.append(numero)
            contadores[numero]=contadores[numero]+1
            criatura_random= criaturas_nodos[i]
            paises_random= region_nodos[numero]
            insert= Relationship(criatura_random, "se_encuentra_en", paises_random)
            sgraph.create(insert)
            verificar=verificar+1
    if(verificar<1):
        numero2=random.randint(0,40)
        criatura_random= criaturas_nodos[i]
        paises_random= region_nodos[numero2]
        insert= Relationship(criatura_random, "se_encuentra_en", paises_random)
        sgraph.create(insert)

#relaciones entre paises
for i in range(40):
    km=random.randint(1,100)
    insert1=Relationship(region_nodos[i], "conectado_con", region_nodos[i+1], distancia=km)
    sgraph.create(insert1)
    insert2=Relationship(region_nodos[i+1], "conectado_con", region_nodos[i], distancia=km)
    sgraph.create(insert2)
```

- **Explain de una Consulta**

La consulta a analizar es **PROFILE MATCH (n:Criatura) WHERE n.lengua='humano' RETURN n** que nos retorna las criaturas que hablan la lengua humana



Se observa que que se obtuvieron todas las criaturas (175) en el primer paso, luego estas fueron filtradas por las que tuviesen una propiedad de “lengua” igual a “humano”, obteniendo 27 criaturas. Sobre esas 28 no se tuvieron que realizar más operaciones, por lo que ese es el resultado.