Projet base de données : OrientDB

BONY Audrey

DAVID Oriane

LAGROY DE CROUTTE DE SAINT-MARTIN Anne-Victoire

PAUTREL Léa

ZHAO Junyi

Master 2 Sciences des Données 2020

Christine Largouët

1. Présentation du logiciel

1.1. Objectifs et approche

OrientDB est un système de gestion de base de données (SGBD) **NoSQL** orienté document-graphe.

Un système NoSQL ne respecte pas le paradigme relationnel basé sur la théorie des ensembles régissant les système SQL. Dans un système SQL, la modélisation de la donnée est structurée par un ensemble de tables et de clefs. La structure est réfléchie en amont et fixe. Au contraire un système NoSQL suit ses propres lois de modélisation. Les données n'ont pas à être homogènes et indexées. Cela rend un système NoSQL beaucoup plus flexible.

OrientDB est construit comme un système orienté graphe tout en incluant des caractéristiques de base orientée document.

Les bases orientée graphe obéissent aux lois de la théorie des graphes. Les données sont parcourues sans avoir recours à un index (parcours d'une table) mais en utilisant les arcs (liens) existant entre les nœuds (individus). Le parcours entre nœuds et arcs est très rapide.
Une structure graphe esquive l'utilisation de jointures lors des requêtes qui peuvent être très couteuses surtout lorsqu'on a beaucoup de données.
En ce point, on peut comparer OrientDB à Neo4j. Les interfaces utilisateurs des deux logiciels sont d'ailleurs très similaires.

OrientDB est aussi orienté document (comme MongoDB). Les données sont stockées de façon semi-structurées. En effet, les données sont dans un format particulier (en JSON souvent) mais la structure des données est libre.

Pour conclure, un système NoSQL orienté document-graphe permet une gestion efficace de données massives et hétérogènes tout en jouissant d'une structure flexible.

1.2. Type de données

Une base de données orientée graphe est formée de noyaux (vertex) et d'arcs (edge). Un document est stocké dans un noyau et les arcs forment un pointeur d'un nœud à l'autre.

Chaque document appartient à une **classe** (ce qui se rapproche de la définition d'une table en système relationnel). Une classe structure et qualifie les attributs des documents. En effet, on pourra lui définir, si on le souhaite, des propriétés puis des contraintes. Une **propriété** définira le nom et le type de l'attribut (chaine de caractère, nombre, etc…). A cela on pourra ajouter des **contraintes** sur les

propriétés. On peut par exemple contraindre la forme de l'attribut (nombre de caractères minimum ou maximum, plage de valeurs acceptée) et si l'attribution est obligatoire. En soi les propriétés ne sont pas nécessaires donc un document peut être rempli de façon libre. Cependant un document doit nécessairement avoir un **identifiant**.


```
CREATE CLASS Eleve

CREATE PROPERTY Eleve.nom STRING

ALTER PROPERTY Eleve.nom MIN 3

ALTER PROPERTY Eleve.nom MANDATORY true
```

Dans le model, on définira des **super-classes.** Celles-ci indiquent la nature de l'objet : nœud (vertex) ou arc (edge).

1.3. Langage de requête

Le langage de requête du logiciel OrientDB est en langage SQL étendu afin de répondre aux exigences d'une structure graphe. Ainsi des commandes permettant de naviguer au sein du graphe. Ce langage est relativement basique, ce qui permet à des utilisateurs assez peu expérimentés de pouvoir explorer et utiliser la base de données. Des exemples de requêtes seront donnés plus loin.

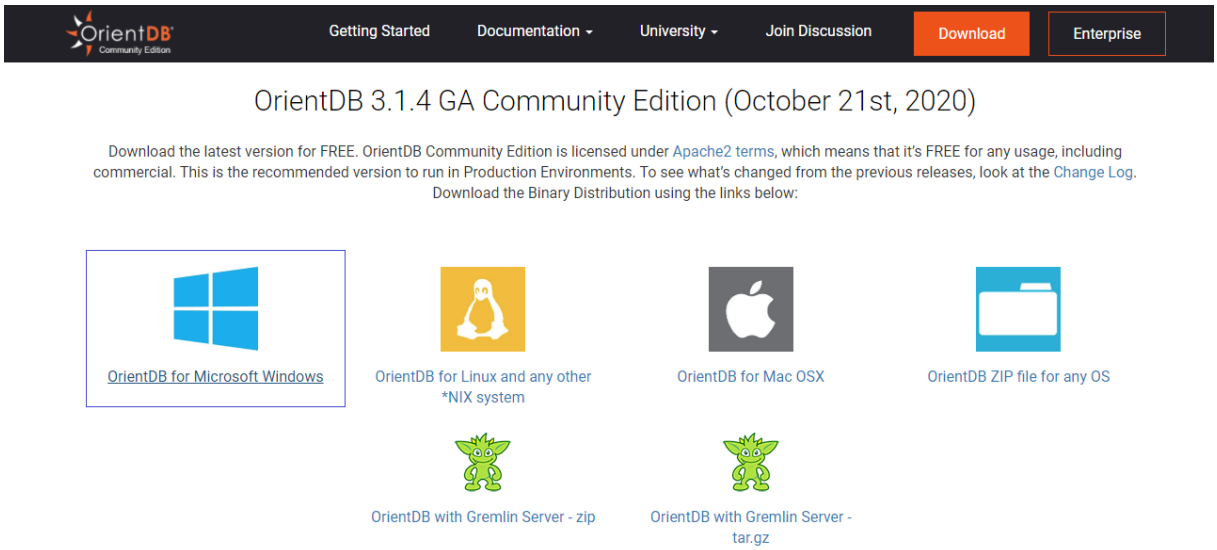
2. Manuel d'utilisation

2.1. Installation

Pour utiliser la SGBD OrientDB, il faut d'abord l'installer sur son ordinateur.

Installation d'OrientDB 3.1.4 sur Windows

Pour l'installer, rendez-vous sur le site https://www.orientdb.org/download et cliquez sur 'OrientDB for Microsoft Windows'.

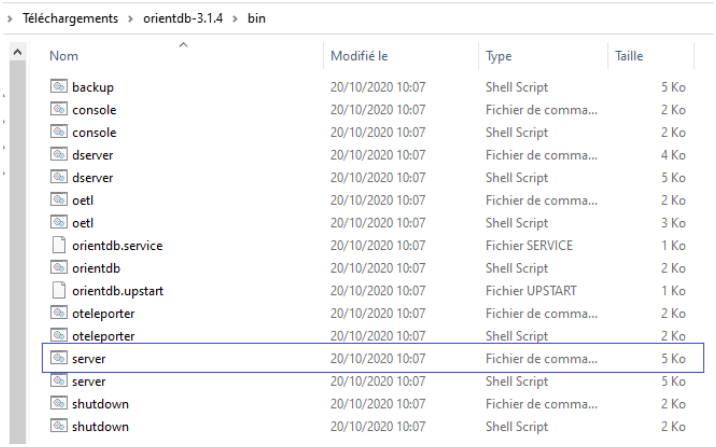


Un fichier ZIP se télécharge alors. A la fin du téléchargement, il suffit de dézipper le dossier et d'en extraire les composantes.

Configuration d'OrientDB 3.1.4

Dans le dossier 'orientdb-3.1.4', allez dans le dossier 'bin'. Ouvrez alors fichier de commande 'server'.

> ***ATTENTION**, il y a 2 fichiers appelés 'server', choisissez bien le fichier de commande (extension .bat) si vous avez un système Windows et le Shell Script (extension .sh) si vous avez un système basé sur Unix (un mac par exemple).*



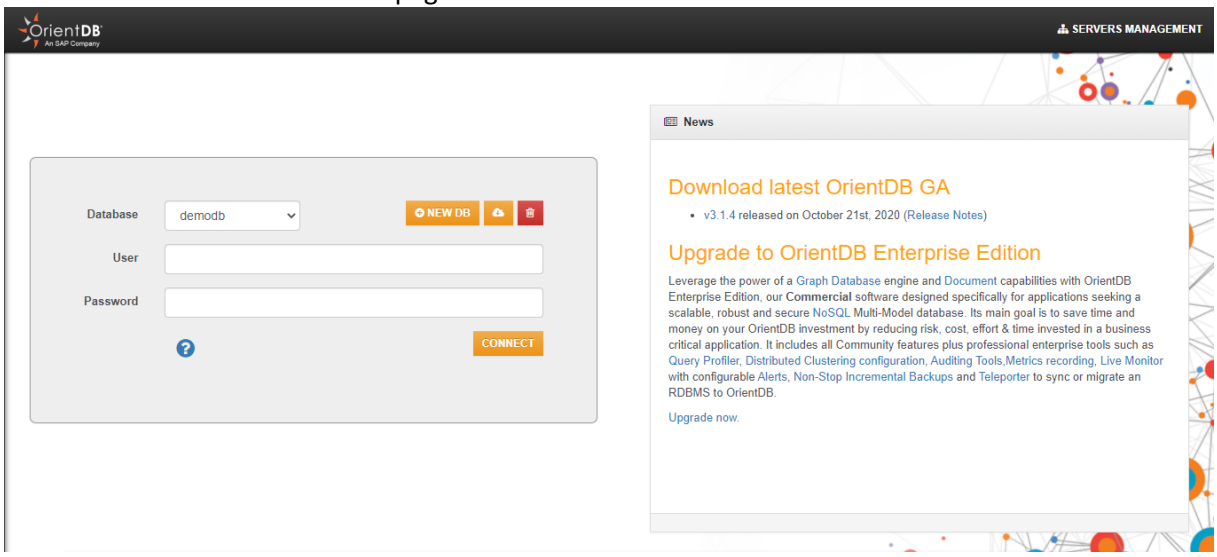
Lorsque vous ouvrez le fichier, on vous affectera par défaut le nom d'utilisateur 'Root' et on vous demandera un mot de passe. Vous pouvez donc saisir un mot de passe.

> ***ATTENTION :** votre mot de passe sera à réutiliser donc notez-le bien ! De plus, ne faites pas 'Enter' sans en saisir sinon cela posera un problème par la suite. En effet, le mot de passe choisi sera aléatoire et il sera difficile de le récupérer.*

Si la fenêtre du serveur se referme directement sans que vous puissiez configurer, c'est que votre version de Java n'est pas à jour. Vous pouvez télécharger la dernière version sur ce lien : https://www.java.com/fr/download/.

Ouverture d'OrientDB Studio

Une fois la configuration du nom d'utilisateur et mot de passe effectuée, la fenêtre d'OrientDB Studio devrait s'ouvrir automatiquement dans votre navigateur. Si elle ne s'ouvre pas, vous pouvez l'ouvrir avec ce lien : http://localhost:2480/studio/index.html#/ après avoir quand même ouvert le server OrientDB. Vous arrivez alors à la page d'accueil suivante :

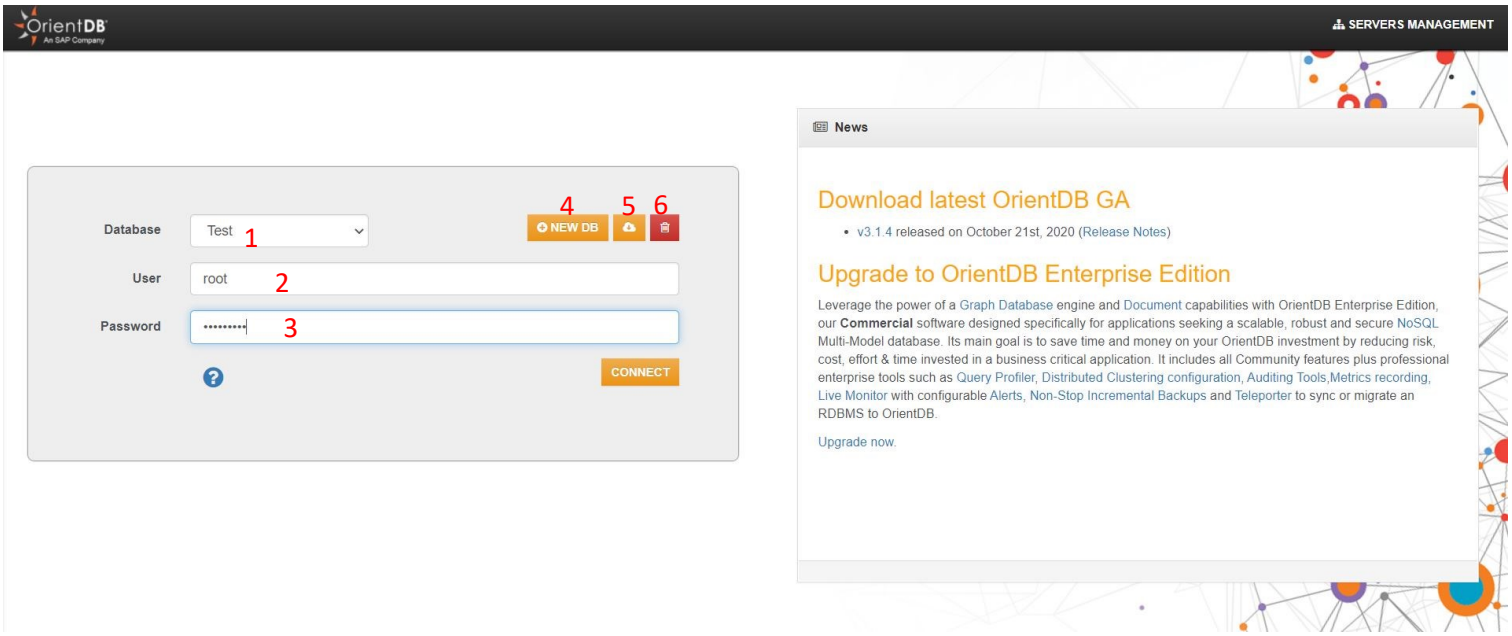


ATTENTION : *Si vous vous connectez pour la seconde fois, il faut rouvrir le fichier serveur (avec l'extension .bat ou .sh selon votre système d'exploitation). Vous pourrez ensuite vous connecter au serveur avec le lien.*

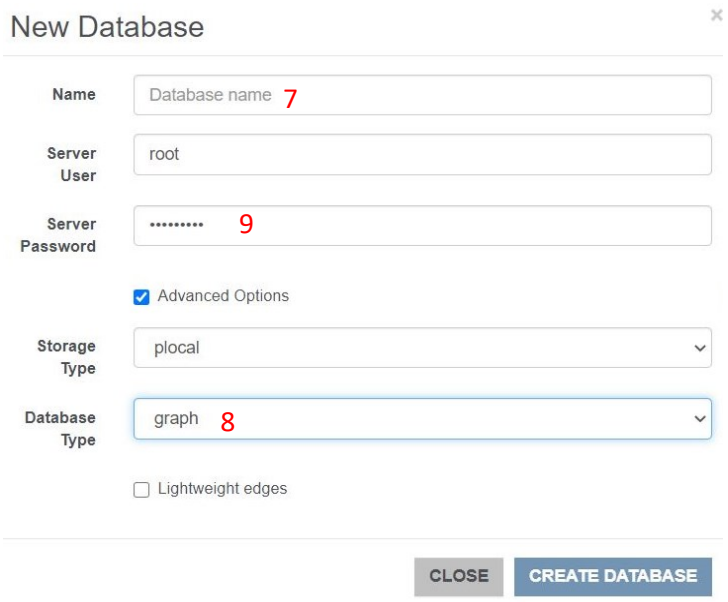
2.2. Utilisation de OrientDB Studio

Un fois sur la page d'accueil de OrientDB Studio, vous pouvez choisir la base de données sur laquelle vous voulez travailler (1), indiquer l'utilisateur « root » (2) et renseigner votre mot de passe (3). Votre mot de passe est celui que vous avez choisi lors de la configuration d'OrientDB.

Vous pouvez aussi créer une nouvelle base de données (4), importer une base de données publique qui existe déjà (5) ou supprimer la base de données sélectionnée (6).



Lorsque vous créez une base de données, vous pourrez lui donner un nom (7), définir son type (document ou graph) (8) ainsi que le mot de passe associé à la base de données (9).



Dans OrientDB, vous avez le choix d'effectuer une action de manière visuelle ou à l'aide d'une ligne de code. L'interface visuelle de manipulation de la base est dans l'onglet *Schema* et l'interface permettant

l'utilisation de ligne de code est dans l'onglet *Browse*. Nous allons à présent détailler quelques actions simples pour modifier votre base de données à l'aide d'OrientDB Studio.

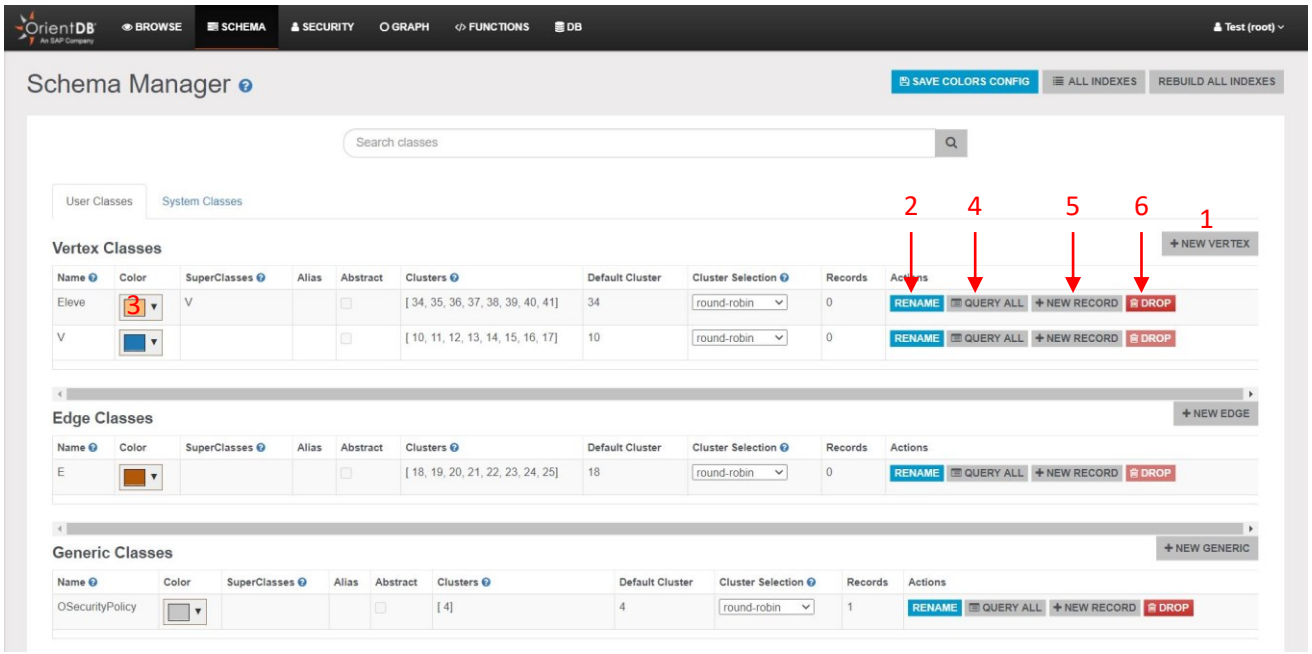
Ajouter, modifier, supprimer une classe de type nœud

Pour créer une classe de type nœud, on peut appuyer sur le bouton + *New Vertex* (1). On aura alors juste à indiquer le nom de la classe qu'on souhaite créer.

La commande équivalente à ces actions est :

```
CREATE CLASS Eleve extends V
```

On peut renommer la classe (2), changer la couleur d'affichage de la classe (3), afficher l'ensemble des documents enregistrés dans la classe (4) (le nombre d'enregistrement est affiché dans la colonne *Records*), ajouter un document (5) ou supprimer la classe (6).



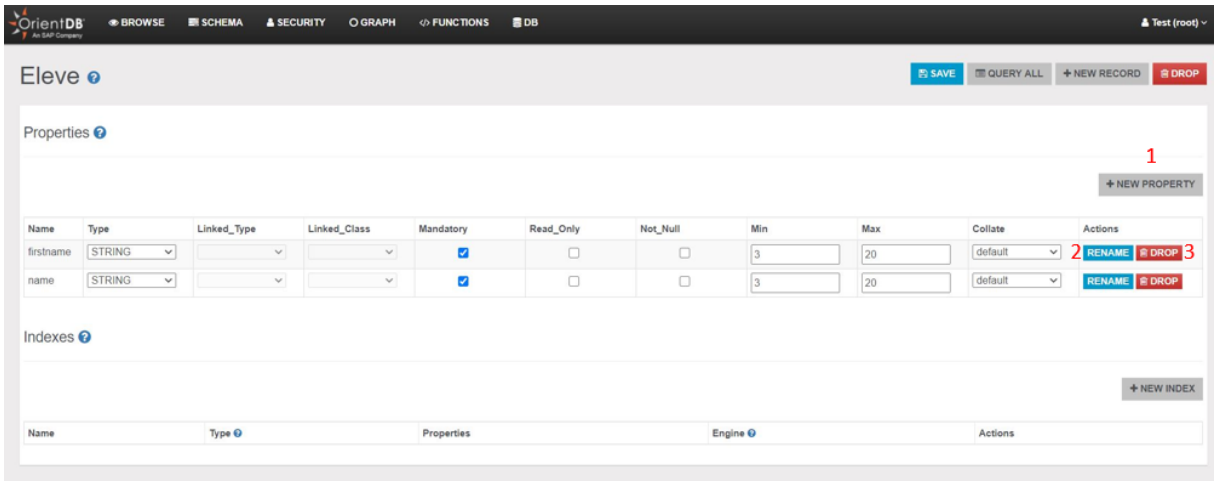
Pour résumer :

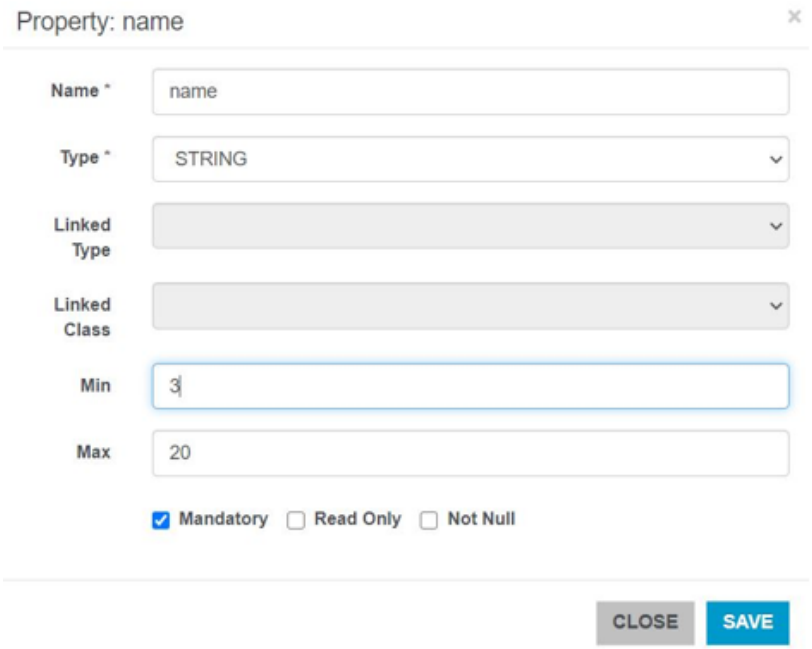
Action	Commande Browse
1	`CREATE CLASS Eleve extends V`
2	`ALTER CLASS Eleve NAME Eleve2`
4	`SELECT * FROM Eleve`
5	`INSERT INTO Eleve SET nom = … , prenom = …`
6	`DROP CLASS Eleve`

Ajouter, modifier, supprimer la propriété d'une classe

On peut ajouter une nouvelle propriété à une classe (1), lorsqu'on appuie sur le bouton +*New property*, une boite de dialogue s'ouvre et on peut indiquer le nom de la propriété, son type et les contraintes choisies.

On peut renommer la propriété grâce au bouton *Rename* (2) ou la supprimer grâce au bouton *Drop* (3) en face de celle-ci.





Action	Commandes
1	`CREATE PROPERTY Eleve.nom STRING`
2	`ALTER PROPERTY Eleve.name NAME “nom”`
3	`DROP PROPERTY Eleve.nom`

Ajouter, modifier ou supprimer une classe de type Edge

Les relations dans OrientDB sont de la superclasse Edge. De la même façon que pour les objets de classes vertex, on peut utiliser le *Browser* (ligne de commandes) ou l’onglet *Schema* (clic-bouton) pour les manipuler.

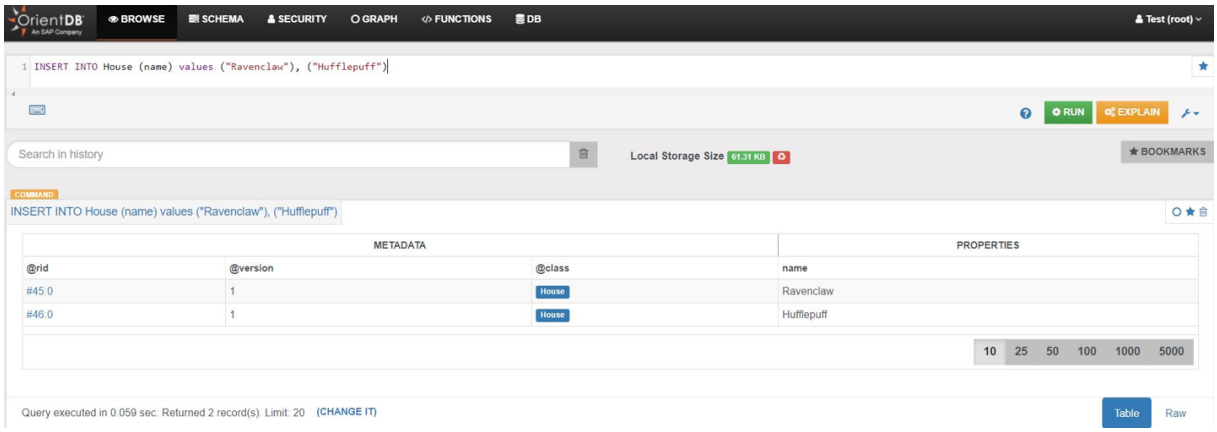
Dans l’onglet *Schema*, il suffit d’appuyer sur le bouton *+NEW EDGE* pour ajouter une classe, de façon assez similaire que pour les classes, on peut renommer la relation, ajouter une nouvelle relation entre deux vertex ou supprimer la relation.

Action	Commandes
Créer	`CREATE EDGE Belongs extends E`
Renommer	`ALTER EDGE Belongs NAME Appartient`
Insérer	`CREATE EDGE Belongs FROM #source TO #target`
Supprimer	`DROP CLASS Belongs`

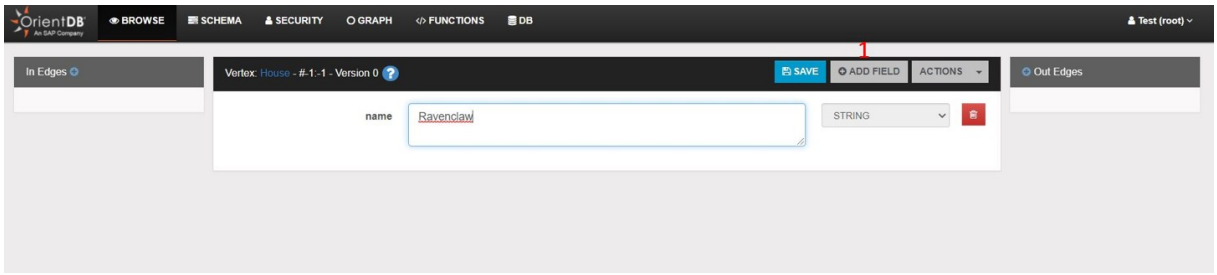
Manipulation des enregistrements de nouvelles données (records)

Dans cette partie, on verra comment on peut ajouter de nouveaux documents et relations entre documents. Pour cela on peut utiliser l'onglet *Browse,* l'onglet *Schema* ou *Graph.*

Pour la partie *Browser*, on a un exemple juste ci-dessous. Avec cette commande on ajoute deux documents de la classe House.



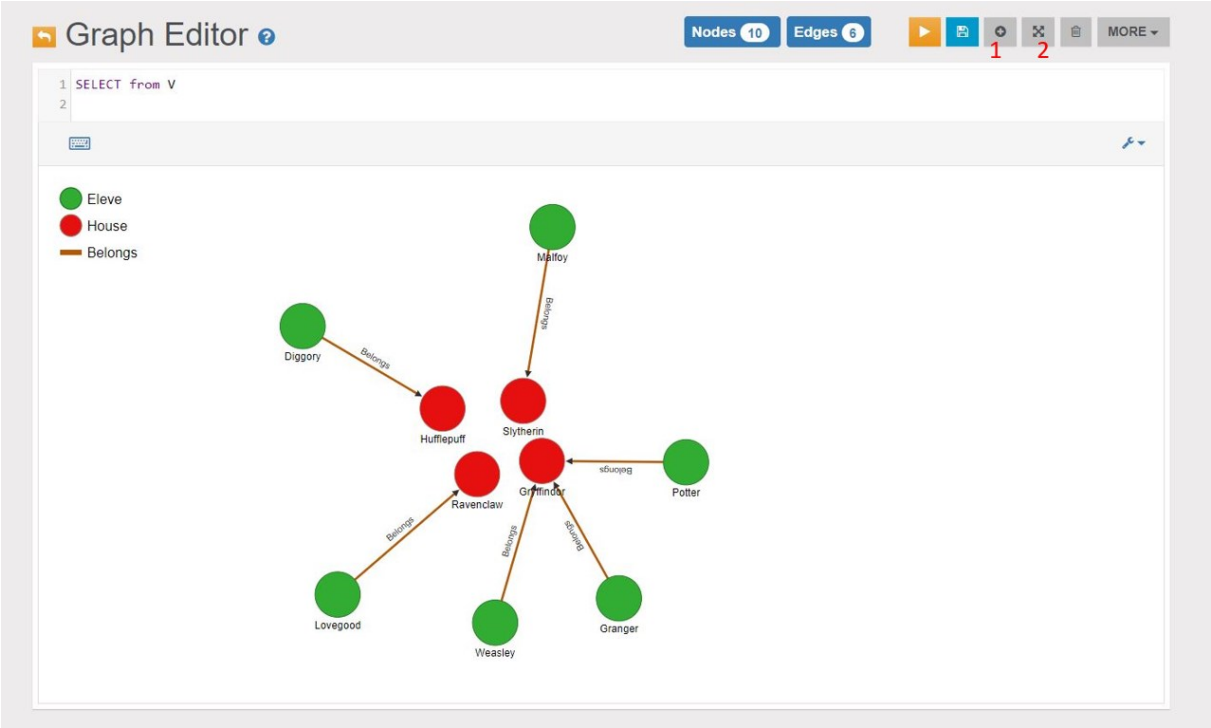
En cliquant sur le nom de classe dans laquelle on veut ajouter un document ou le bouton *New Record* de l'onglet *Schema*, on atteint une nouvelle page qui nous permet d'entrer à la main un nouveau document (1) comme illustré ci-dessous :

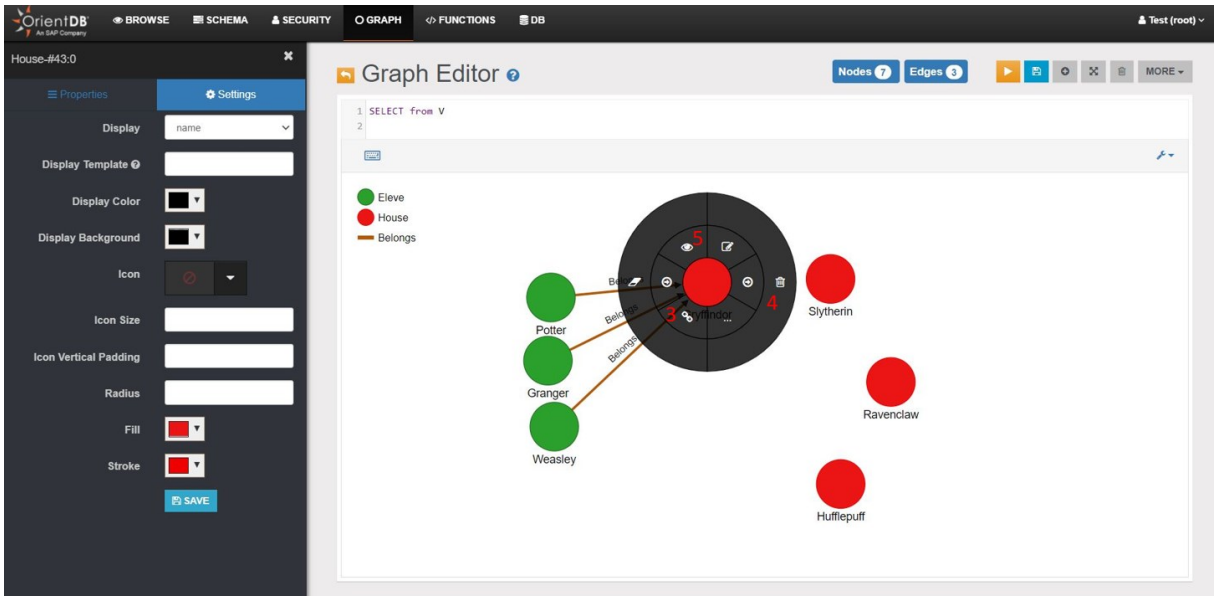


Enfin, la dernière méthode pour ajouter un document est de l'ajouter depuis l'onglet *Graph*.

A partir de cet onglet, on peut ajouter des objets de la classe nœuds (bouton 1, puis remplissage des attributs du document). Le bouton 2 sert juste à visualiser le graphe dans une grande fenêtre.

Lorsqu'on clique sur un nœud, un menu s'ouvre autour de celui-ci. On peut alors ajouter une relation entre ce nœud et un autre (Bouton 3 puis cliquer-glisser vers la destination et choix de la relation). Le bouton 4 permet de supprimer le nœud. Le bouton 5 permet d'accéder aux propriétés et paramètres du nœuds. On peut alors modifier ses informations (onglets *Properties*) et ses paramètres d'affichage comme le texte à afficher, la couleur du nœud, ajout d'une icône etc… (onglet *Settings*)





A présent, vous savez créer et modifier la structure de votre base ainsi que la remplir de document via OrientDB Studio.

2.3. Utilisation du script Python

2.3.1. Avant de lancer le script

Il faut vérifier que :

- ✓ Dans le dossier où est ce script, vous avez bien un dossier 'donnees' avec le document 'data_tolkien.json'
- ✓ Vous avez bien lancé le 'server.bat' de OrientDB
- ✓ Vous avez bien créé une base de données vide locale Neo4j avec Neo4j Desktop, ayant pour user "neo4j" et pour mot de passe "Neo4J"

✓ Vous avez bien fait Start pour cette base de données sur Neo4j Desktop et ouvert le Browser
✓ Vous avez bien installé py2neo
→ Dans la console, installez py2neo en écrivant la ligne suivante :

```
> pip install py2neo
```

✓ Vous avez bien installé pyorient
→ Dans la console, installez pyorient en écrivant la ligne suivante :

```
> pip install pyorient
```

→ Puis mettez pyorient à jour avec la ligne suivante, toujours dans la console :

```
> pip install --upgrade git+https://github.com/OpenConjecture/pyorient.git
```

2.3.2. Déroulé du script

Introduction

Un message s'affiche et il faut rentrer le mot de passe choisi lorsque l'on a installé OrientDB. Les importations nécessaires sont effectuées, et deux fonctions sont définies.

Création de la base de données OrientDB

Après s'être connecté à la session OrientDB ouverte, la base de données est créée en plusieurs temps :

- Création des clusters vides
- Création des classes (Vertex et Edge)
- Création des propriétés des classes
- Insertion des informations (records) dans la base de données

Création de la base de données Neo4j

Un équivalent de la base de données, mais en Neo4j est ensuite créée.

Requêtes

Des requêtes sur la base de données, présentées en 3.2, ont été lancées à la fois sur OrientDB Studio (pour les sorties graphiques principalement) et sur le script Python.

Comparaison OrientDB/Neo4j

Les requêtes sont faites à la fois sur la base de données OrientDB et sur la base de données Neo4j. Les textes des requêtes sont affichés pour comparer le vocabulaire des deux systèmes, et le temps de calcul de chaque requête est mesuré. Deux graphiques permettent de comparer quantitativement ces temps de calculs.

3. Jeu de données

3.1. Présentation du jeu de données

Le jeu provient de la base jeu de données publiques du logiciel OrientDB.

Il s'appelle 'Tolkien-Arda'. Il décrit l'univers des romans de Tolkien (Trilogie du *Seigneur des Anneaux* et *Le Hobbit*)

Database containing Tolkien (Middle-earth) related information like Characters, Locations and Events FROM his different works (http://arda-maps.org)

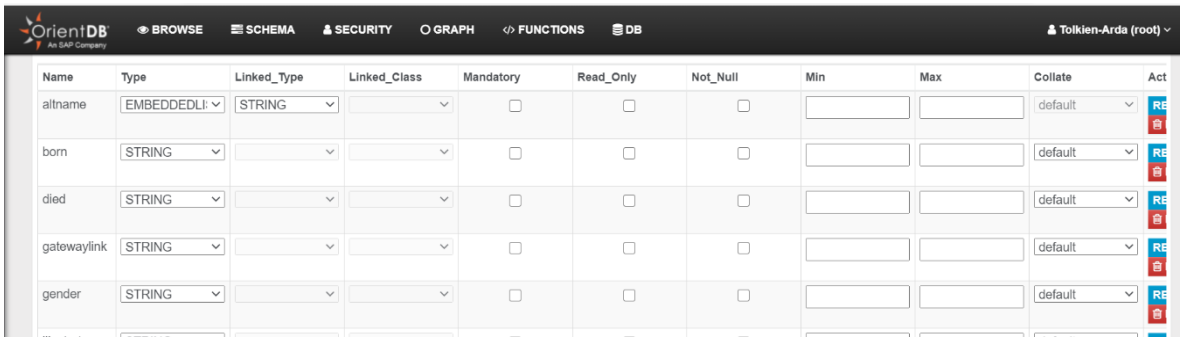
Les classes sont séparées en User Classes et System Classes. Les classes de User Classes servent directement à effectuer des requêtes alors que celles dans System classes tournent en fond et permettent de faire fonctionner la base de données.

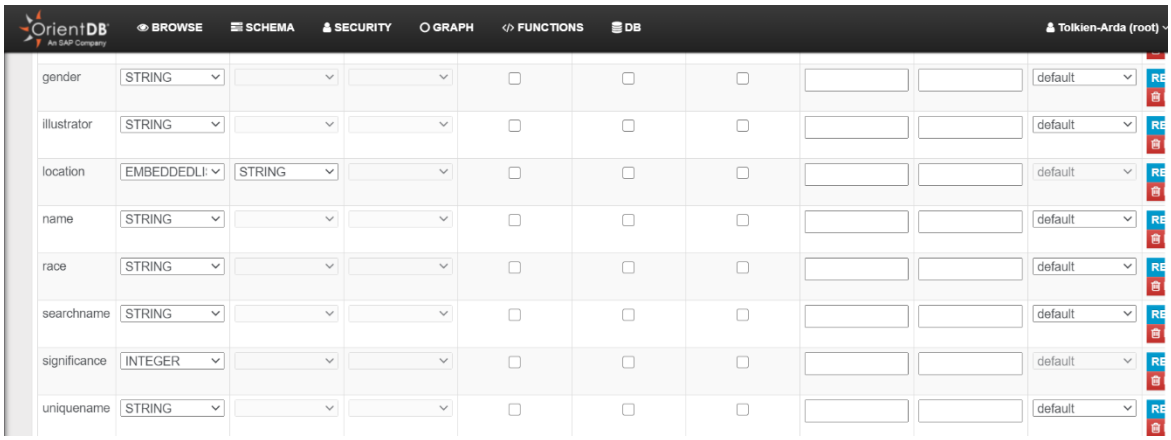
Dans User classes nous avons :

Les noyaux (vertex) sont : AbstractName, Creature, Event, Location, V.

AbstractName et V sont des superclasses. Les classes dans la superclasse AbstractName sont : Creature et Location. Les classes de la superclasse V sont AbstractName et Event.

Noyau	ID	Nombre d'enregistrements	Propriétés
Creature	11	882	altname, born, died, gatewaylink, gender, illustrator, location, name, race, searchname, significance, uniquename.
Location	12	796	age, altname, area, canon, gatewaylink, illustrator, name, searchname, significance, type, uniquename
Event	13	58	description, illustrator, name, uniquename
AbstractName			altname, name, searchname et uniquename.
V	9	0	





Les arcs (Edge Classes) sont BEGETS, HASSIBILING et LOVES qui appartiennent tous à la superclasse E.

La classe BEGETS est la relation de parenté : qui a engendré qui. La classe HASSIBILING indique une relation de fratrie entre 2 Creature. La classe LOVES représente une relation d'amour entre Creature.

Arcs	ID	Enregistrements
BEGETS	14	752
HASSIBLINGS	16	24
LOVES	15	102

Generic classes : ORIDs OSecurityPolicy

System classes

OFunction, OIdentity, ORestricted, ORole, OSchedule, OTriggered, OUser, _studio

3.2. Exemple de requêtes sur OrientDB Studio

Le langage de requête est le SQL. Certains éléments de langage ont en effet été ajoutés afin de pouvoir mieux manipuler les relations entre « Vertex ». Dans le Studio, les requêtes peuvent se faire dans deux onglets distincts selon le type de sortie que l'on veut avoir : « graph » (pour avoir une sortie graph) et « browse » pour un tableau de données.

- Principes simples requêtes SQL :

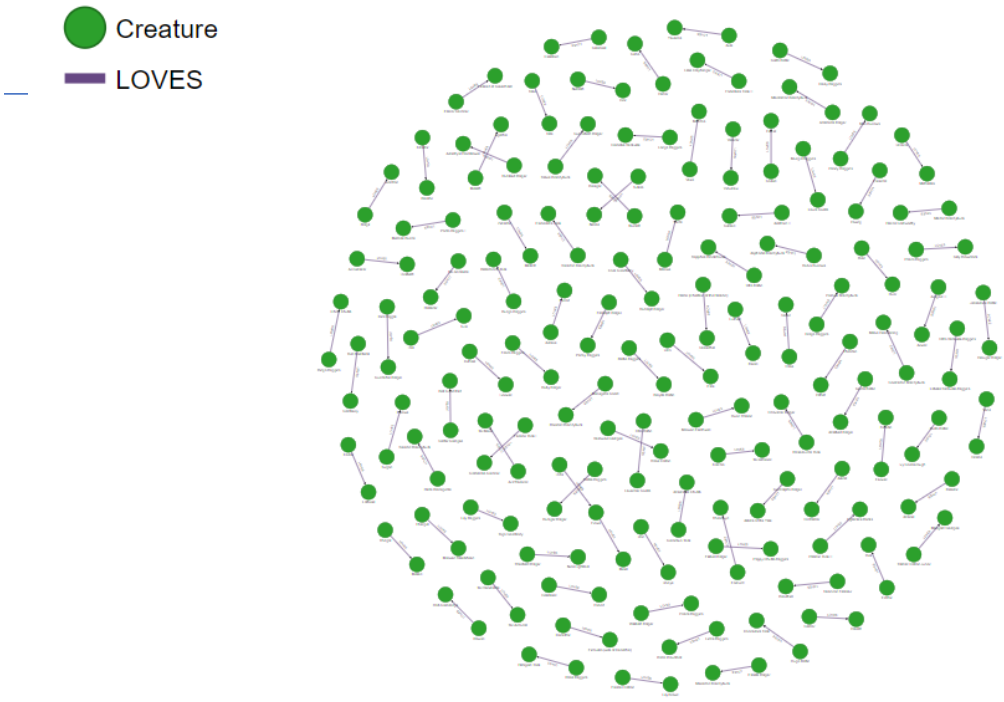
Les requêtes SQL de base s'appliquent. Les relations ou « edges » sont considérées comme une simple table ou l'on peut sélectionner des éléments.

Qui aime qui ?

```
SELECT * FROM LOVES
```

Sortie graphe :

On observe les « vertex » reliés par les « edges » lorsque l'on recherche le « edges »



Sortie Tableau de données :

COMMAND

Select * from Loves

METADATA			PROPERTIES	
@rid	@version	@class	in	out
#15:0	1	LOVES	#11:866	#11:100
#15:1	1	LOVES	#11:847	#11:601
#15:2	1	LOVES	#11:599	#11:841
#15:3	1	LOVES	#11:322	#11:538
#15:4	1	LOVES	#11:676	#11:846
#15:5	1	LOVES	#11:649	#11:821
#15:6	1	LOVES	#11:420	#11:818
#15:7	1	LOVES	#11:832	#11:680
#15:8	1	LOVES	#11:526	#11:525
#15:9	1	LOVES	#11:352	#11:529
#15:10	1	LOVES	#11:631	#11:352
#15:11	1	LOVES	#11:762	#11:763
#15:12	1	LOVES	#11:351	#11:25
#15:13	1	LOVES	#11:344	#11:268
#15:14	1	LOVES	#11:615	#11:803
#15:15	1	LOVES	#11:5	#11:130
#15:16	1	LOVES	#11:112	#11:300
#15:17	1	LOVES	#11:399	#11:452

On observe les caractéristiques des relations : élément 1 de la relation, élément 2 (in et out).

- Agrégations :

On peut également faire des agrégations : comme en SQL normal, il est possible d'appliquer les fonctions mathématiques et les fonctions de regroupement dans Group by. On s'attend ici à des sorties tableaux, ces commandes sont généralement écrites dans la console de l'onglet « browse ».

Quelles sont les trois régions les plus peuplées ?

```
SELECT Location, COUNT(*) as regionCOUNT FROM Creature
GROUP BY Location
ORDER BY regionCOUNT DESC
limit 3
```

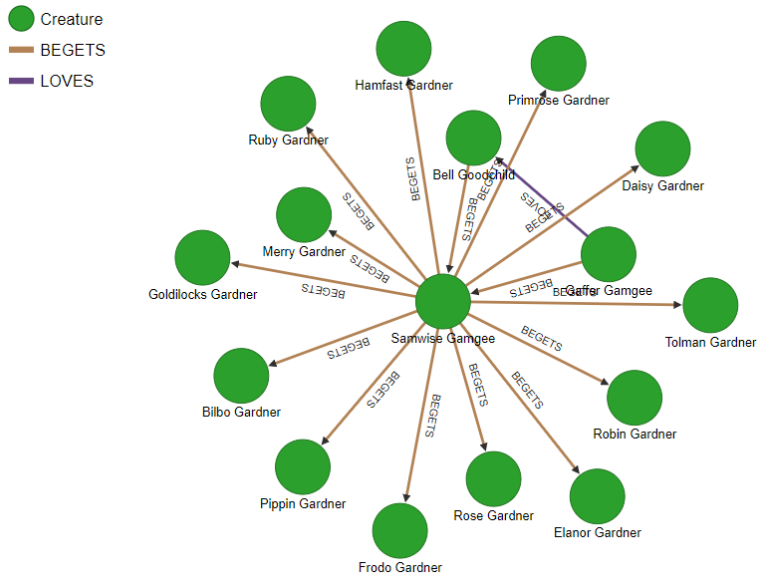
PROPERTIES	
location	**regioncount**
[]	153
["Gondor"]	93
["Númenor"]	61

- Relation *« edges »* entre éléments *« vertex »*

Notion de distance/chemin : Les relations peuvent être désignées par le nom MATCH. L'ajout de $Pathelements permet d'afficher le graphe relatif à tous les éléments que l'on a désigné ainsi que leur relation. La flèche permet de polariser la relation. Ainsi le résultat à la requête suivante est faux :

Qui sont les enfants de Sam Gamgee ?

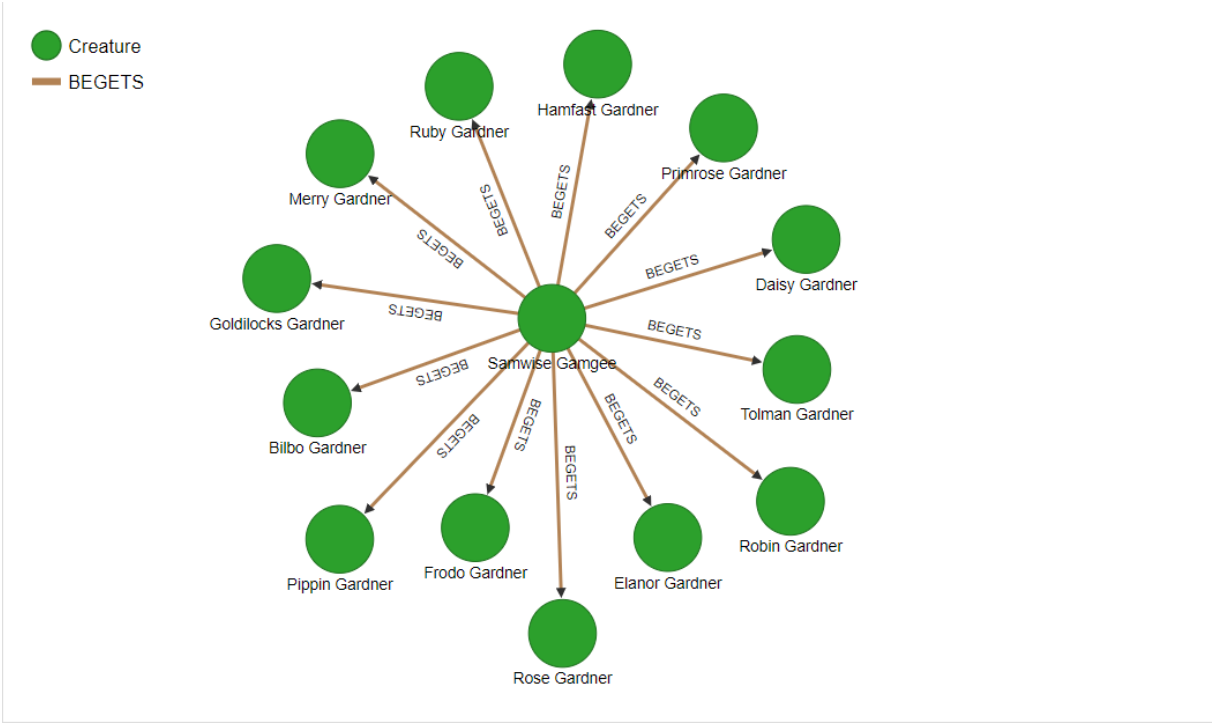
```
MATCH {Class: Creature, as: Father, where: (name='Samwise Gamgee')}-BEGETS-{Class: Creature, as: Children}
RETURN $pathelements
```



Il faut polariser la relation, afin d'avoir uniquement les Creatures que Samwise a engendré et ne pas avoir celle qui ont engendré Samwise (ses propres parents).

```
MATCH {Class: Creature, as: Father, where: (name='Samwise Gamgee')}-BEGETS->
{Class: Creature, as: Children}

RETURN $pathelements
```



Combien d'enfants a Sam Gamgee ? On attend une sortie tableau

Il suffit de modifier le return par :

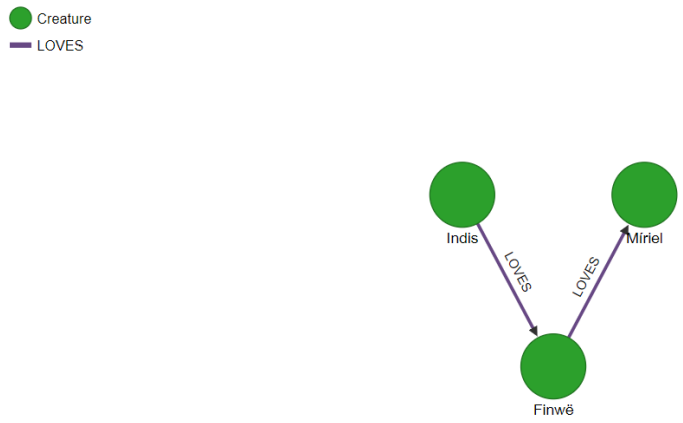
```
RETURN COUNT(Children)
```

On obtient la réponse 13.

Existe-t-il un triangle amoureux ?

```
MATCH {Class: Creature, as: Creature}-LOVES->{Class: Creature, as: Creature}-LOVES-
>{Class: Creature, as: Creature}

RETURN $pathelements
```



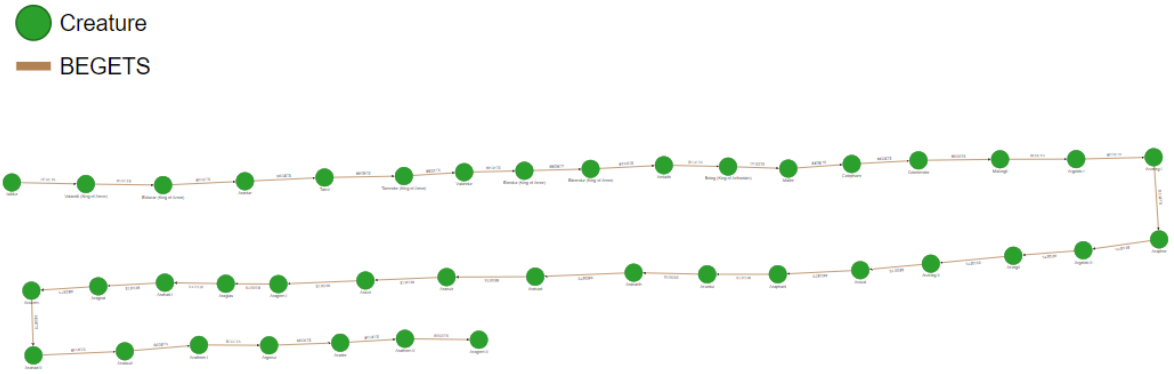
- Notion de chemin

La requête suivante illustre la notion de chemin. En effet, il existe l'élément Path permettent de relier par un chemin les éléments désignées d'un graphe.

La fonction shortestPath permet de désigner le chemin le plus court.

Quels sont les ancêtres qui séparent Aragorn II et Isildur ?

```
SELECT expand(path) FROM (
  SELECT shortestPath($FROM, $to) AS path
  LET
    $FROM = (SELECT FROM Creature WHERE name='Aragorn II'),
    $to = (SELECT FROM Creature WHERE name='Isildur')
  UNWIND path
)
```



Combien de générations les séparent (en comptant les leurs) ? (Sortie tableau)

Il suffit de rajouter un COUNT (*) à partir de la requête précédente. On obtient un résultat égal à 40.

4. Python et PyOrient

Il existe un package de Python permettant l'utilisation de OrientDB directement sur Python : PyOrient. Ce package est complexe et requiert de bonnes connaissances de programmation. En effet, le package présente deux modules : « Client » et « OGM ». Le module « Client » est le module de base. Il permet la réalisation de tâches simple : se connecter à un serveur, manipuler les bases de données et faire des requêtes SQL.

Il y a plusieurs reproches que nous pouvons faire à PyOrient. Tout d'abord, il est très compliqué de se connecter à un serveur. En effet, il existe un problème de version du package. Le « bug » est reporté à de nombreuses reprises sur le web. Nous avons tenté plusieurs stratégies de résolution du bug (importation d'un upgrade de Github).

Un deuxième obstacle rencontré avec PyOrient fut pour la création de la base de données. Nous avions initialement créé l'ensemble de la base de données avec les méthodes de PyOrient Client n'impliquant pas d'écriture de requête SQL, comme par exemple `record_create()` pour ajouter chaque créature dans le Vertex Creature. Cependant, cette solution fonctionnait seulement partiellement. Elle ajoutait

bien les informations de chaque créature, mais lors de l'importation des informations dans les Edges faisant des liens entre les créatures (LOVES, BEGETS, HASSIBLING), aucun lien n'était effectivement créé entre les créatures. Nous avons donc dû remplacer l'utilisation de `record_create()` par l'utilisation de la méthode `command()`, avec une requête de type `CREATE VERTEX Creature SET…` .

La dernière difficulté que présente PyOrient est la non-lisibilité du résultat des requêtes sur Python aisément. Les résultats de requêtes de la fonction `query()` de PyOrient sont dans un format objet. Une fois le résultat stocké dans une variable, il est possible d'y accéder via Python avec des crochets, ou bien d'utiliser des outils comme Flask ou Django pour lire les résultats. L'utilisation de ces méthodes est obligatoire pour la visualisation des graphes. L'utilisation du module « OGM » est encore plus complexe, car réservée aux bases de taille importante et aux grands graphes. Nous n'avons pas pu utiliser de méthode OGM, Django ou Flask, nous n'avons donc pas eu de résultats sous forme de graphes.

5. Conclusion

Nous avons bien appréhendé les différentes facettes de OrientDB. Le langage utilisé basé SQL donc assez facile à prendre en main pour les personnes maitrisant des rudiments en base de données.

Plusieurs aspects d'OrientDB nous ont particulièrement plu. Tout d'abord l'installation du logiciel est relativement facile, et le mode d'emploi proposé est assez complet. L'interface Studio présente de nombreuses fonctionnalités et offre de nombreuses possibilités de personnalisation aux utilisateurs. Le langage de requête est simple. Les mauvais côtés d'une base SQL (nombreuses jointures, requêtes longues…) ne sont pas présents car il s'agit d'un SGBD NoSQL. L'interface propose de visualiser les propriétés des classes et de personnaliser directement les paramètres de la base. Le côté graphe est bien sûr un avantage car cela offre une excellente possibilité de visualisation des données. PyOrient est un outil très puissant, la version OGM, demande un gros investissement en installation mais permettrait la manipulation de grandes bases.

Cependant, certains aspects du logiciel nous ont semblés plus difficiles à maîtriser. La présence de deux consoles pour faire des requêtes (en fonction du type de résultat que l'on veut obtenir) peut porter à confusion. Certains paramètres et certaines rubriques du studio sont très précises et dénotent avec la simplicité du langage (qui paraît être pour des personnes moins expérimentées) car un certain niveau d'expérience en bases de données est nécessaire pour leur pleine maîtrise. Le langage Extended SQL nous a semblé également parfois compliqué à appréhender. En effet, de nouvelles fonctionnalités sont régulièrement mises à jour. Ainsi, il est parfois difficile de trouver la « bonne version » des fonctions. La documentation du logiciel se veut assez complète, cependant elle est par certains aspects mal faite. Elle change régulièrement, certaines informations sont manquantes et il y a relativement peu de bibliographie à part celle qui est officielle. Le package PyOrient est aussi compliqué, avec un bug systématique à l'installation.

Nous avons comparé OrientDB et Neo4J. OrientDB est beaucoup plus difficile à maîtriser que Neo4J mais offre plus de fonctionnalités. Tout d'abord, Neo4J est plus utilisé que OrientBD, il y a donc plus de documentation. Neo4J Desktop est plus lent qu'OrientDB Studio. Sur Python, les temps de calcul sont très instables sur OrientDB comparé à Neo4J. Selon les machines, le plus rapide est soit OrientDB, soit Neo4J. On peut le voir sur les deux figures suivantes :

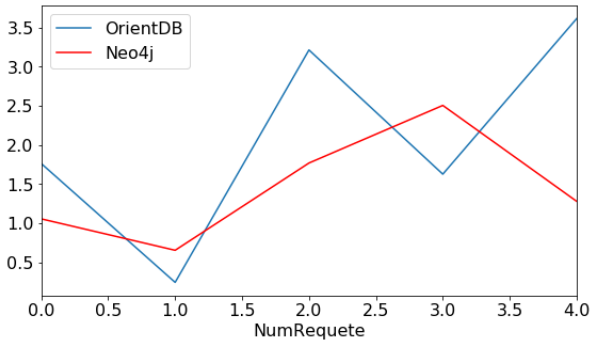


Figure 1 : Evolution du temps de calcul par requête

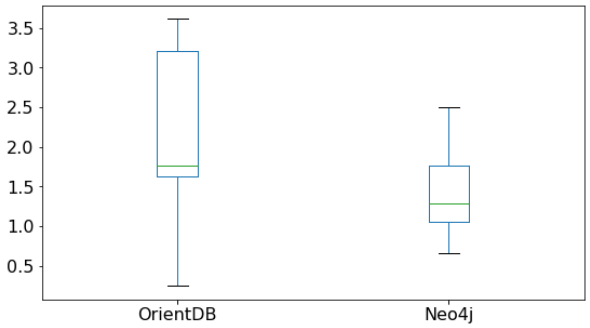


Figure 2 : Diagramme en boîte, comparaison des temps de calcul

6. Sources

Ces sites ont été consultés entre octobre et novembre 2020.

- Site officiel de OrientDB : https://orientdb.org/
- Documentation officielle de OrientDB et PyOrient : http://orientdb.com/docs/3.1.x/
- Github official PyOrient : https://github.com/OpenConjecture/pyorient
- Github forum d'aide bug PyOrient : https://github.com/mogui/pyorient/issues/270
- Tutoriels divers : https://stph.scenari-community.org/contribs/nos/orient1/co/OrientDB-1-exercices_1.html
- Divers forums StackOverflow