**Rapport Projet OrientDB**

1. Présentation du logiciel

Objectif : Base de données type document et graphes

Type de données : données avec des relations entre les éléments. Format de fichier : fichier json

Langage de requête : SQL

1. 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>)

1. Python

Il existe un package de Python permettant l’utilisation de OrientDB directement sur Python. Ce package est extrêmement complexe et requiert de bonnes connaissances de programmation. Malgré de nombreuses heures de recherche passée à découvrir le package nous avons décidé de ne pas l’utiliser. En effet, le package présente deux modules : « Client » et « OGM ». Le module « Client est le module de « base ». Il permet la réalisation de taches 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 serveur rendant très compliqué la connexion. Il faut modifier des paramètres de classe des objets ??? . Le « bug » est reporté à de nombreuses reprises sur le web. Nous avons tenté plusieurs stratégies de résolution du bug. Malgré tout, aucune n’a été 100% efficace sur l’ensemble de nos machines.

La deuxième difficulté que présente PyOrient est la non-lisibilité du résultat des requêtes sur Python directement. Une fois le résultat stocké dans une variable, il est nécessaire d’utiliser des outils de code comme Flask ou Django pour lire les résultats. Cela nous semblait ambitieux et de plus, pour un résultat incertain car nous n’aurions certainement pas obtenus de graphes mais des lignes de résultats ?. L’utilisation de l’ORM est encore plus complexe car réservée aux bases de taille importante et aux grands graphes.

1. Requêtes :

Le langage de requête est le extended SQL. Certains éléments de langage ont en effet été ajouter afin de pouvoir mieux manipuler les relations entre «Vertex ». particularité ? Dans le serveur en ligne, 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 un simple tableau ou l’on peut sélectionner des éléments.

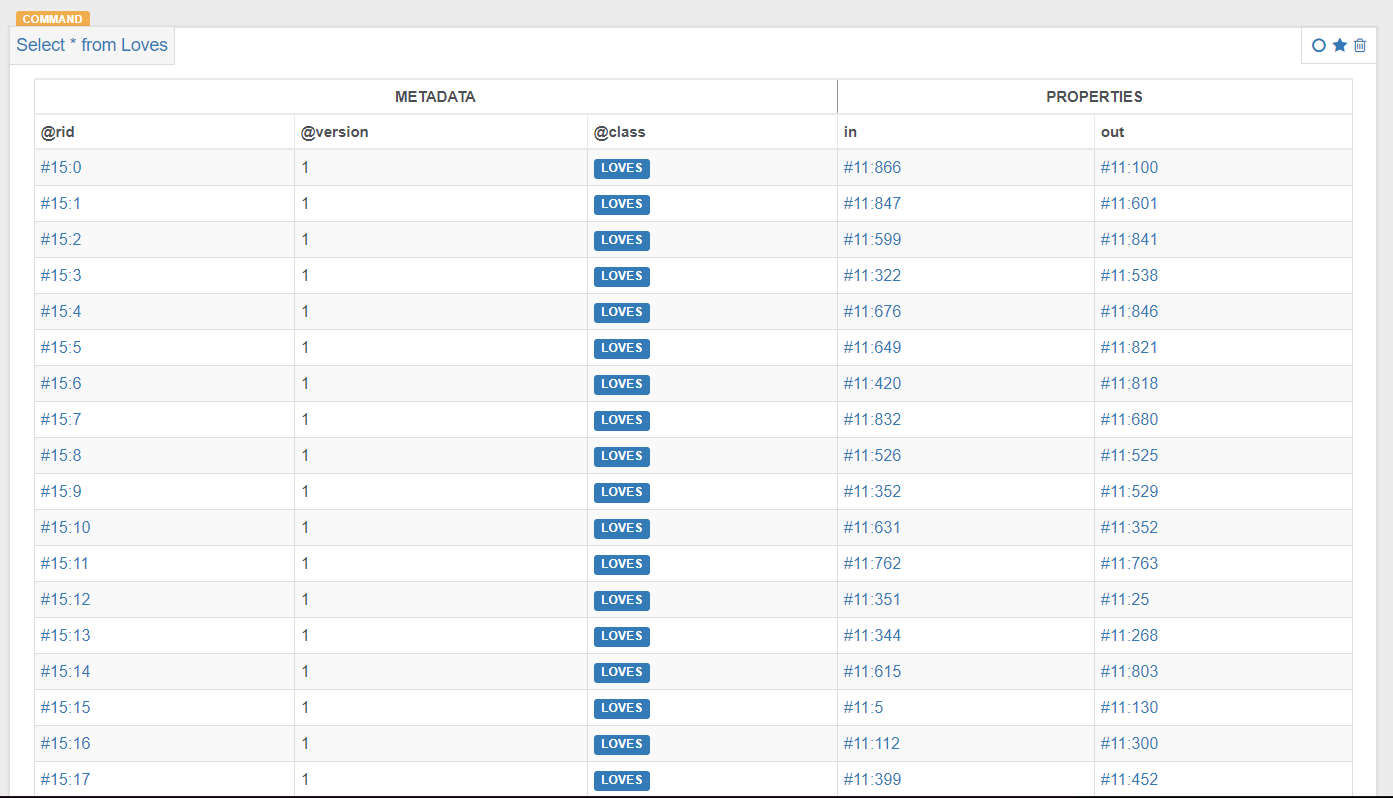
*Who loves who?*

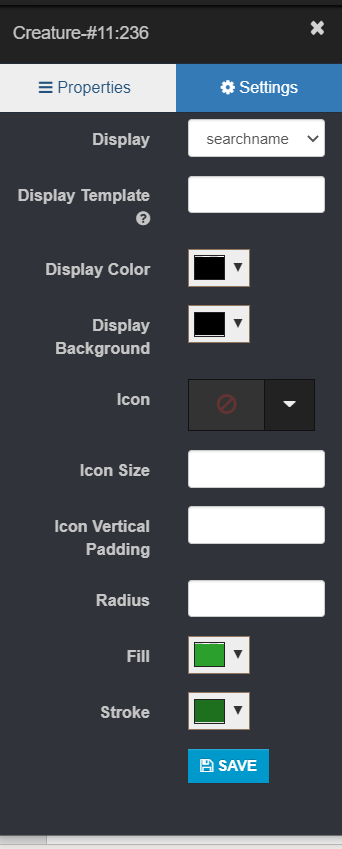
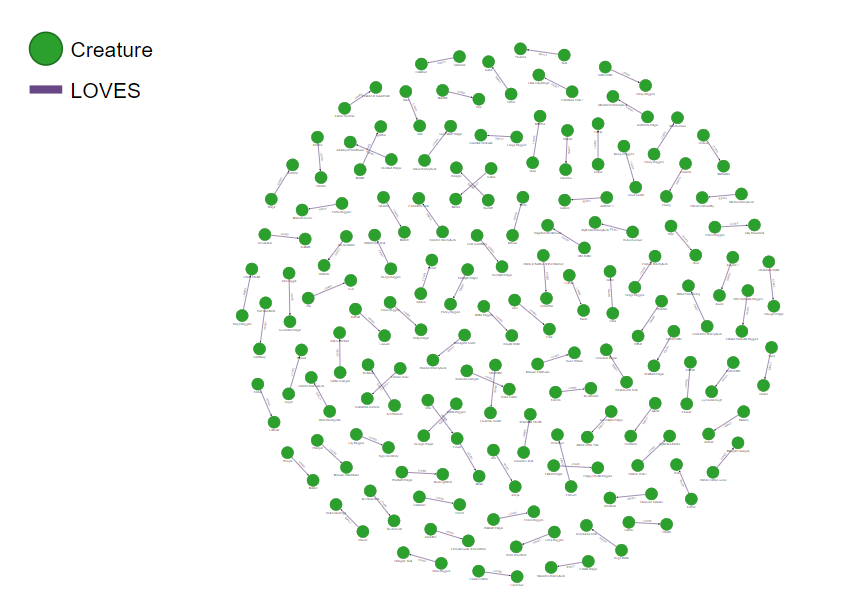
select \* from Loves

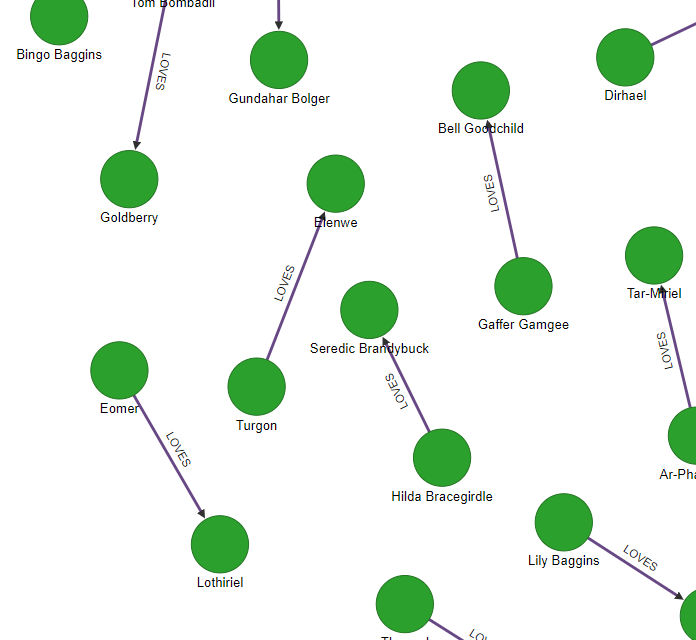
Sortie graphe :

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

Sortie Tableau de données :



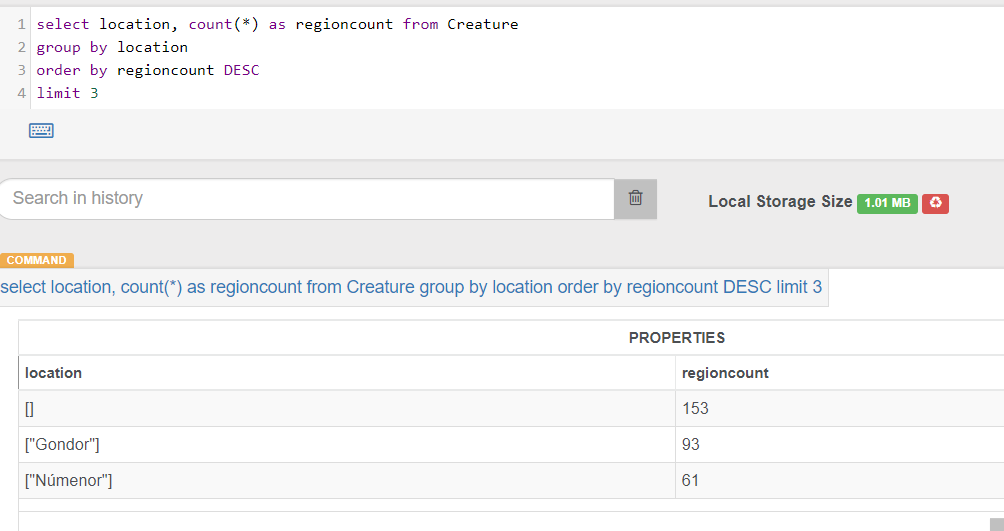


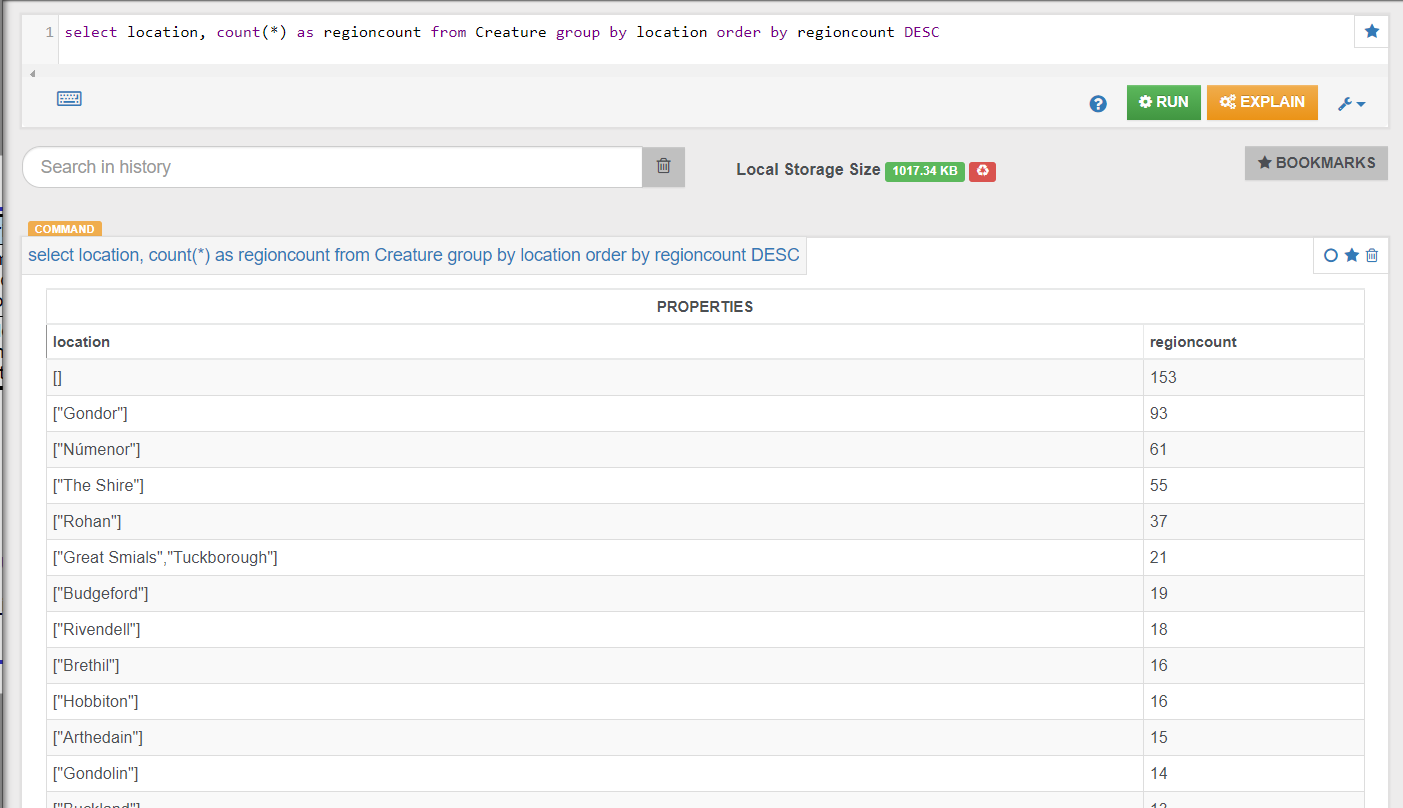


On observe les caractéristiques des relations : élément 1 de la relation, élément 2.

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.

*Quelles sont les trois régions les plus peuplées?*



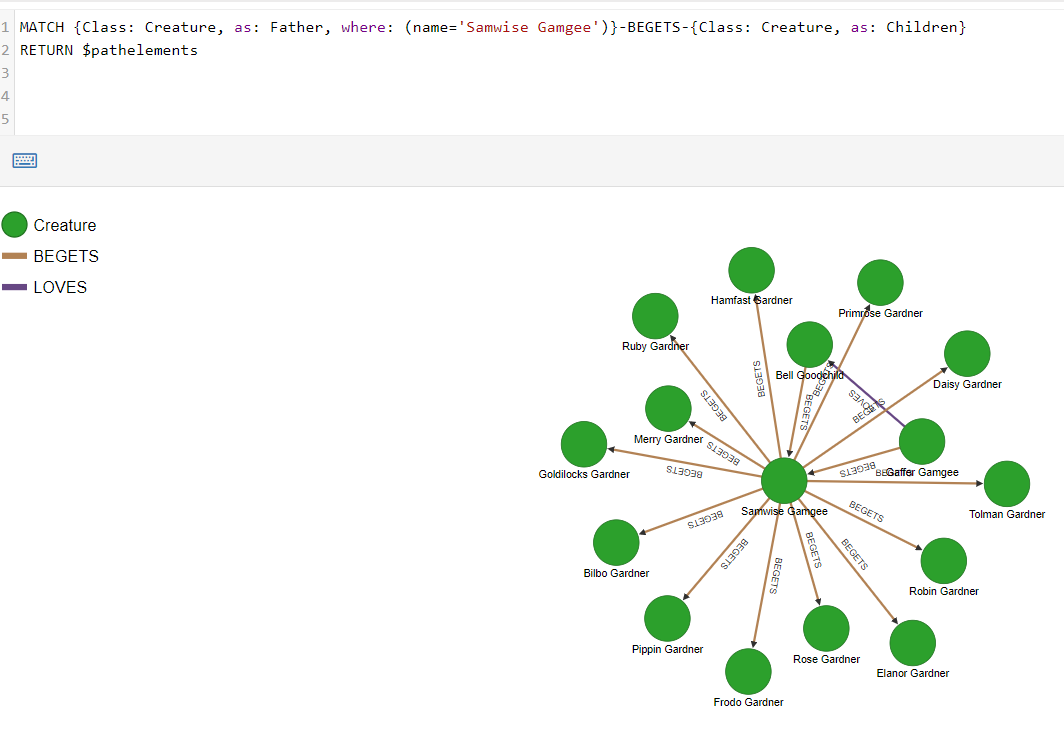


Notion de distance/chemin: Les relations peuvent être désignées par le nom MATCH. L’ajout de $Pathelements permet ‘afficher le graphe relatif à tous les éléments que l’on a désigné ainsi que le relation. La flèche permet de polariser la relation. Ainsi le résultat à la requête suivante est faux :

*Qui sont les enfants de Sam Gangee ?*

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

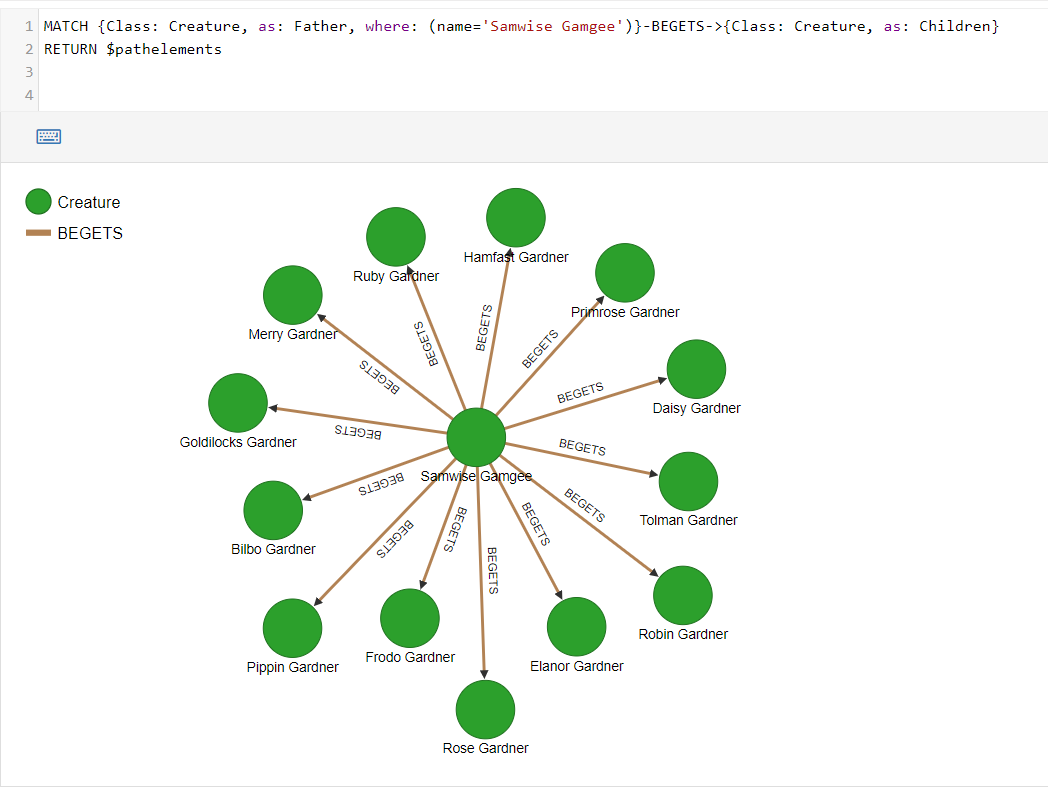
RETURN $pathelements



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



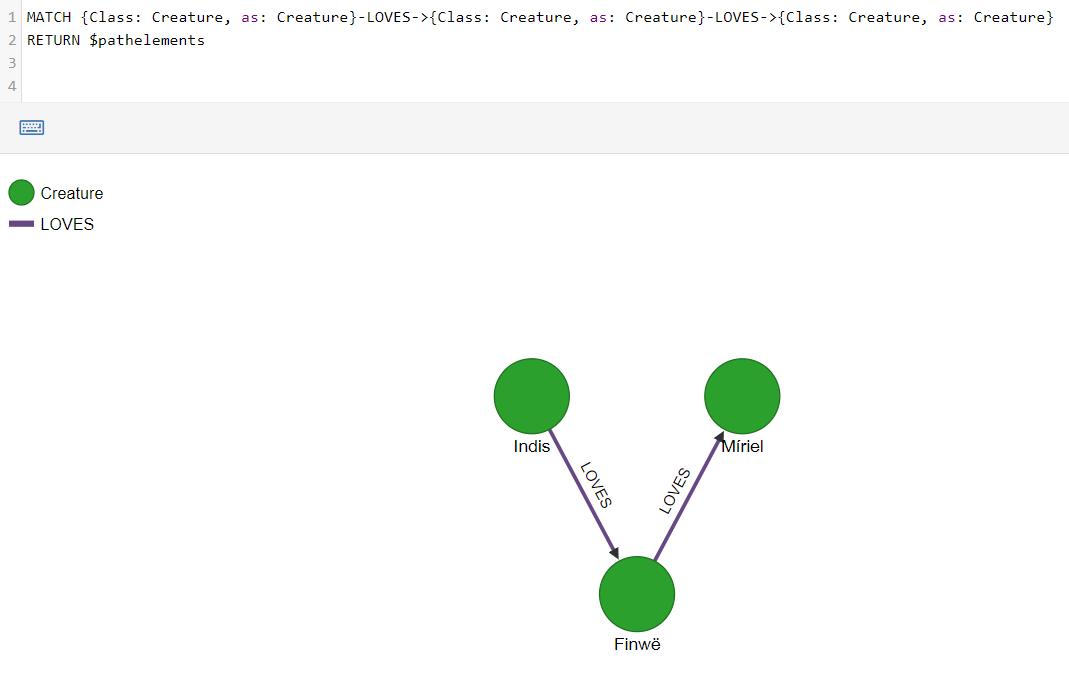
Compter le nombre d’enfants :



*Is there a love triangle?*

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

*RETURN $pathelements*



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. des fonctions très utiles

SELECT $path as path

FROM (TRAVERSE outE(), inV() FROM (SELECT @rid FROM Creature WHERE name='Aragorn II'))

WHERE @rid==(SELECT @rid FROM Creature WHERE name='Isildur')

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

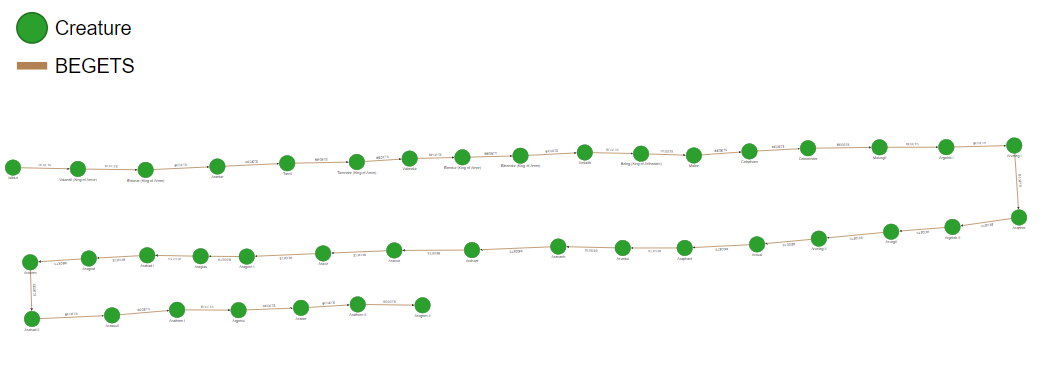
LET

$from = (SELECT FROM Creature WHERE name='Aragorn II'),

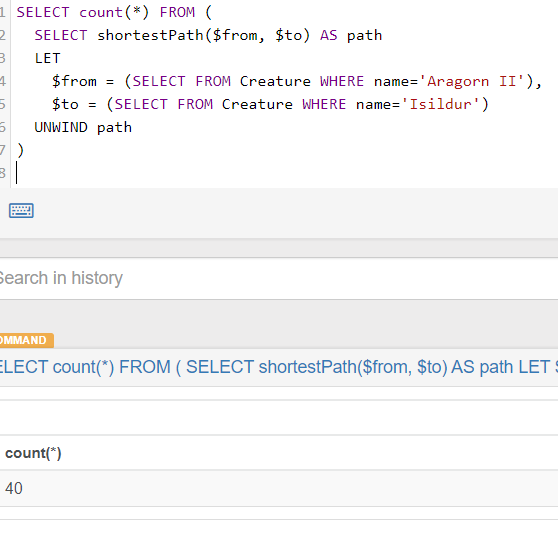
$to = (SELECT FROM Creature WHERE name='Isildur')

UNWIND path

)



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



*Quelle est la plus grande relation de filiation qui existe?*

*SELECT $path as path*

*FROM (TRAVERSE outE(), inV() FROM (SELECT @rid FROM Creature WHERE name='Aragorn II'))*

*WHERE @rid==(SELECT @rid FROM Creature WHERE name='Isildur')*