**Rapport Projet OrientDB**

1. Présentation du logiciel
2. Objectif 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 sont 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 interface utilisateurs des deux logiciels sont d’ailleurs très similaires.

OrientDb est aussi orientée 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. 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.

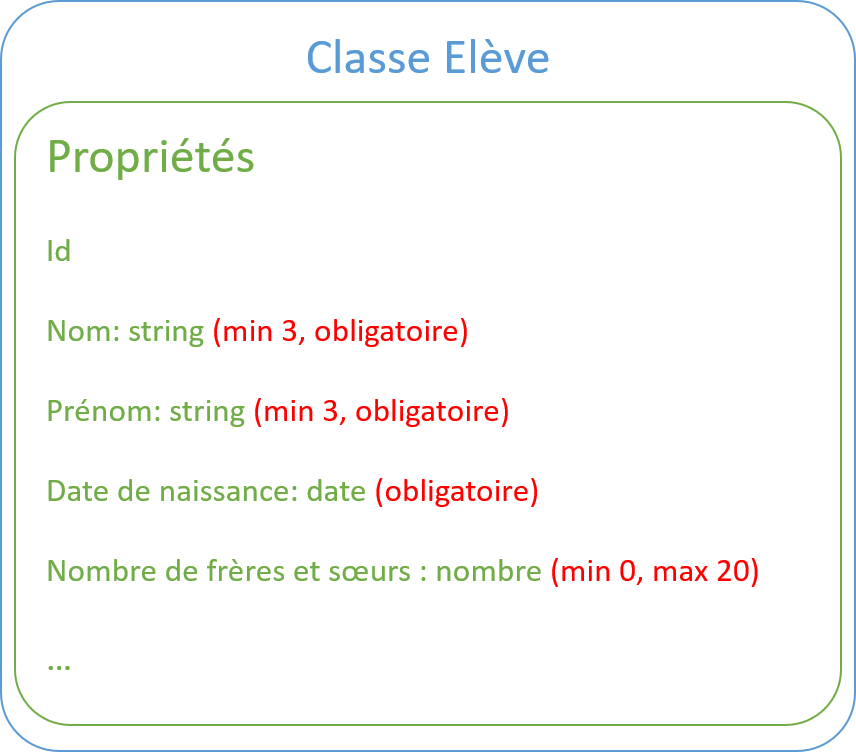
Noyau 1

Noyau 2

Arc

1 relié à 2

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 soit 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

1. Langage de requête

Le langage de requête

1. Manuel d’utilisation

* Installation
* Mis en route
* …

Langage de requête : SQL étendu pour répondre aux exigences graphe

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.