Rapport du Projet HMSN204

Mathieu Blaison, Sean Laidlaw

Introduction

Au début du projet nous avons tout d'abord eu une phase de réfexion initiale dsur le projet. Nous avons ainsi établi le diagramme de Gantt suivant.

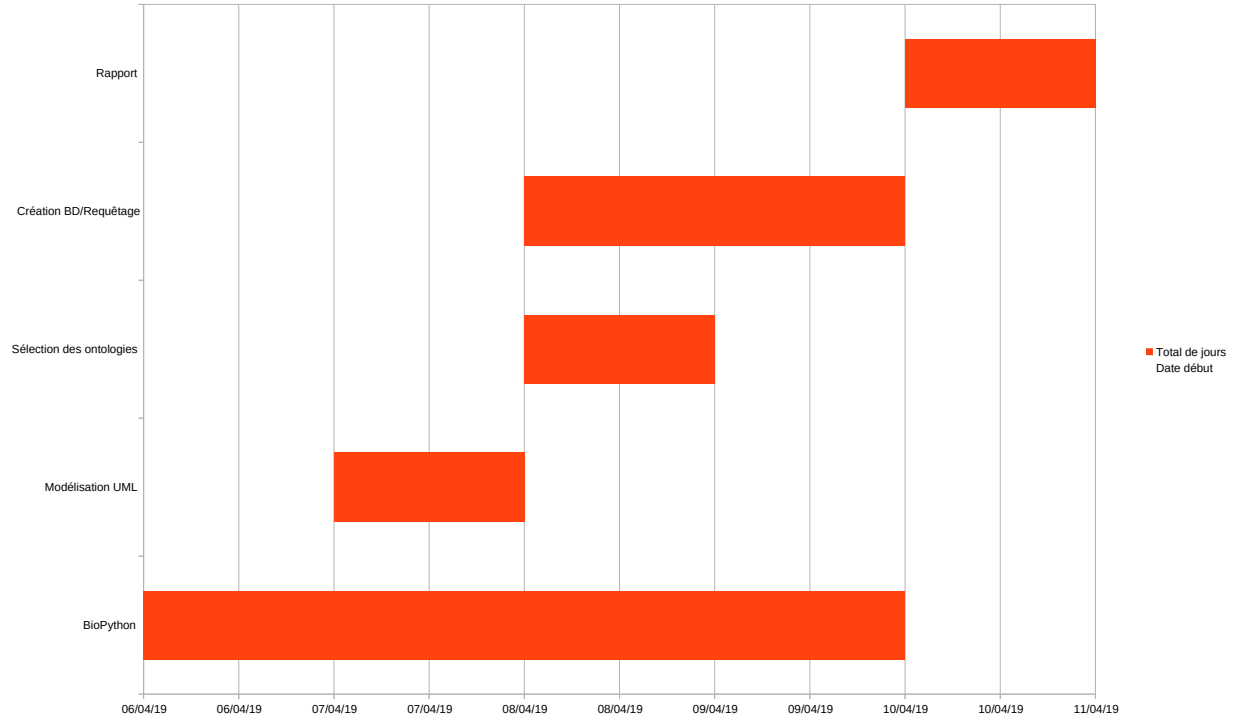

Fig. 1 : Diagramme de Gantt détaillant le partage des responsabilités par rapport au temps

De cette réflexion initiale surle projet nous avons pu en établir le diagramme de cas d'utilisation suivant.

Traitements Biopythons et manipulation des séquences

Obtention des sequences

Le jeu de données étant collectionné afin de mieux comprendre les mécanismes physiologiques liés à la croissance des plantes, on a choisi d'étudier le gène SEX1, qui a un rôle dans le dégradation de l'amidon.

Afin de pouvoir démontrer les résultats, le code utilisé, et les explications, on a choisi d'utiliser un Jupyter Notebook. Dans ce dernier nous avons écrit une fonction qui obtient le fichier fasta via une

requête Entrez correspondant à l'ARN messager.

Pour l'obtention des séquences mutés cependant, il y avait un choix à faire. En effet, en recherchant la base de données NCBI avec la requête `SEX1[Gene] AND Arabidopsis thaliana[Organism]`, on constate qu'il y a 3 ARNm différents répondant à ces critères. En regardant de plus près cependant, on voit que ce sont des variants d'épissage. Alors que le plupart des aligneurs modernes sont *splice-aware*, l'aligneur que nous avons construit effectue un alignement globale à la base de l'algorithme Needleman-Wunsch et donc ne l'est pas. Ceci veut dire qu'on ne pourrait pas correctement aligner les variants d'épissage contre notre séquence sauvage. Afin d'obtenir une séquence contre lequel on pourrait aligner notre fasta sauvage, on a consulté la base de donnée dbSNP qui détail le liste des SNP de la gène SEX1, leurs localisations dans l'ARN messager, et les identifiants de ces SNP. Un peu de vimscript plus tard, et 1/4 des SNP détaillés sur dbSNP étaient appliqués à la fasta du SEX1 sauvage. Le fasta mutant produit, a ensuite été ajouté au projet GitHub afin que le notebook Jupyter puisse le telecharger pour le transformer en objet Seq et l'aligner contre la séquence sauvage.

Tab. 1 : Liste des identifiants dbSNP utilisés pour creer le fasta mutant

Used dbSNP IDs			
rs1105066589	rs1103971843	rs1095089377	rs1100808719
rs1095780989	rs1095659046	rs1097407346	rs1097236347
rs1105152302	rs1097124183	rs347038182	rs346885812
rs1101762250	rs1100942745	rs1106840358	rs346897346
rs1099291378	rs1102995172	rs1096948278	rs1099609436
rs1104510198			

Alignement Globale

Afin d'aligner les deux séquences, on a utilisé les dataframe de la librarie Pandas. Étant l'objet qui s'approchait le plus d'un tableau, et permettant l'écriture d'une de ses cases avec seulement les coordonnés de la case, il a paru d'être la meilleur solution.

Le déroulement de l'alignement se passe dans plusieurs étapes. D'abord il y a l'initiation du tableau, ou le script parcours la longueur des deux séquences convertant chaque nucléotide d'une en colonne et chaque nucléotide de l'autre en ligne.

Ensuite, pour la remplissage du tableau les cases ont été remplis avec les valeurs correspondant au statut de match ou mismatch de la nucléotide, ainsi que le score de ses cases voisins d'en haut, à gauche, et en diagonale. En plus de déposant le score dans la case du premier dataframe, une symbole représentant la direction à été inséré dans un autre tableau (<, , ou |) afin que la fonction suivante puisse suivre le chemin parcouru.

Une fois le tableau rempli, le chemin de retour a été tracé par la fonction traceback qui exploite le modèle de double tableau afin de se baser à la fois sur le score le plus haut, et puis de se déplacer dans la direction qui à amené à ce score. Ensuite avec un alignement obtenu les positions de variations sauvage/mutés, le taux de GC, et la traduction en séquence dacides aminés sont données.

Optimisations

L'algorithme utilisé pour l'alignement globale souffre de la problème de complexité. En effet, la génération et la remplissage du tableau augmente quadratiquement avec les longeurs de sequences fournis. Où m est le longeur de seqX et n le longueur de seqY, la complexité de temps d'une alignement par cette méthode serait de l'ordre de O(m*n). Pour les séquences recherchés dans ce projet, les longueurs de séquence était à l'ordre du millier, rendant l'alignement très lent.

Afin d'essayer de contourner le problème, des optimisations ont été recherchés. Python n'étant pas la plus vite des langages, nous a amener à utiliser la librairie cython qui convertiras certains de nos fonctions en code C afin de profiter d'une meilleure performance. De plus, on a essayé également de mettre certains fonctions en cache, pour pas qu'il soit recalculé à chaque fois, et enfin on a essayé la librairie JIT qui fait une compilation à la volée. Ce dernier a donnée les résultats les plus prometteurs, mais n'as pas fonctionné avec la fonction le plus lourd de l'alignement car il n'est pas compatible avec pandas.

Tab. 2 : Comparaisons d'optimisations et performances

Optimisation	Alignment Time (s)
None	18.6
Function Caching	18.16
Cython	18.02
JIT Compilation	18.22
Combined	18.55

Base de données

Réflexion sur la base de données

On a eu le choix d'un des SGBD vu ce semestre : chado/postgresql (module phenotype), relationnel seul (oracle, postgresqlou mysql), neo4j, couchdb, triplestore jena/rdf.

Notre choix s'est ainsi porté sur la base de données Chado. Celle-ci est conçue spécialement pour gérer les représentations complexes de données biologiques. Chado va également posséder des modules permettant de gérer spécifiquement les relations entre termes ontologiques. Pour créer sa base

de données Chado utilise le système de gestion PostgreSQL. Ayant deja des bases en SQL , ce choix nous a ainsi paru judicieux pour pouvoir réaliser une implémentation correcte de la base de données avec un temps limité et la taille de notre groupe.

Modélisation de la base de données

Avant de nous lancer dans la création de notre base de donnée il faut d'abord la penser. Pour ce faire le meilleur moyen est de se représenter la base de données visuellement à l'aide d'un shéma UML et plus particulièrement d'un diagramme de classes. Notre bases de données se divise en "deux" parties. Une gérant les données issues des expérimentations des M1 BFP et l'autre gérant les termes d'ontologies. Un lien est fait entre ces 2 parties au niveau de la table phénotype. Avec le recul cette manière de faire n'était probablement pas la meilleure et des améliorations doivent être apportés principalement pour la partie gérant l'ontologie.

Nous retrouvons ansi avec une base de données corrspondant au diagramme de classe suivant :

Ontologie

Il a ainsi fallu penser à la manière d'intégrer l'ontologie dans notre base de données . L'ontologie va représenter des concepts (comme une caractéristique phénotypique) et les relations que ces concepts peuvent partager. Dans notre cas on s'est intéressé à rechercher les relations entre les termes ontologiques constituant les phénotypes observés et mesurés par les étudiants BFP. Pour ce faire ces phénotypes ont été "décomposé" en plusieurs et leur terme ontologique ont été associé en une relation. Par exemple, la longueur de la racine primaire va corrrespondre à une longueur de racine (possédant son terme ontologique) qui ferait partie (is part of) d'une racine primaire(possédant aussi son propre terme ontologique). ces données générées seront ensuite insérées dans notre base de données. Ces données peuvent être récupérées grâce à de nombreuses ressources regroupant les termes d'ontologies appartenant à un même "groupe".

Implémentation bases de données et peuplement

Création des tables

Afin de créer les tables un script SQL de création de ces tables à été écrit. Celui-ci va constituer dans un premier temps en la destruction des tables et de leurs contraintes dans le cas à l'aide d'un drop table. Il va ensuite permettre de construire ces tables en affectant les clefs primaires et étrangères appropriés à chaque table.

Afin de générer les tables nécessaires à l'ontologie, le module cv de Chado à été utilisé. Je me rends compte cependant à l'écriture de ce rapport que j'ai mal compris et mal utilisé ce module cv, notamment en n'utilisant pas la table cv_term relationship.

Insertion des données dans les tuples

Afin de ne pas avoir à insérer les données issues du document csv qui nous a été fourni manuellement dans nos tuples une réorganisation de celles-ci a été effectué. Celle-ci ont ainsi été redistribuées sur

plusieurs fichier csv correspondant chacun à une table donnée. On peut ensuite insérer les données contenues dans ces fichier dans nos tuples à l'aide de la commande `\\copy` . Les données ayant rapport à l'ontologie ont été insérées "manuellement" à l'aide de 'insert values into'.

Requêtes

Des requêtes ont été généré afin d'interroger notre base de données. Ces reqûetes ont été regroupées dans un script . Ces requêtes m'ont permis de me rendre compte que certains éléments de la base de donnée auraient mérités à être pensé autrement afin de pouvoir faire des requêtes plus précise et intéressantes au niveau de l'ontologie.

Exemple de requête effectué :

Observe le potentiel effet du milieu sur la taille moyenne de l'aire racinaire

```
select m.type, avg(ph.Root_area)
from milieu m, boite b, plant p, phenotype ph
where m.id_boite = b.id_boite and b.plant_id = p.plant_id and p.plant_id = ph.plant_id
group by m.type ;
```

Tab. 3 : Résultat de la requête

gene_name	count
Col_0	29
nrt	76
sex1	11
pgm	11

Conclusion :

Ainsi nous avons été en mesure de générer une base de donnée permettant de faire des requêtes pour la consultation et potentiellement l'interprétation des résultats obtenus par les M1 BFP . Cette base de données permet aussi de consulter les relations d'ontologies des phénotypes mesurés. Cependant cette ontologie n'a pas été implémenté de manière optimale par manque de comprhéension et de temps pour rattraper les erreurs. Les séquences nucléiques des gènes o,t pu être récupéreés depuis le NCBI puis elles ont pu être analysées àl'aide de BioPython pour nous fournir des informations qui auraient pu être utilisées dans notre base de données.

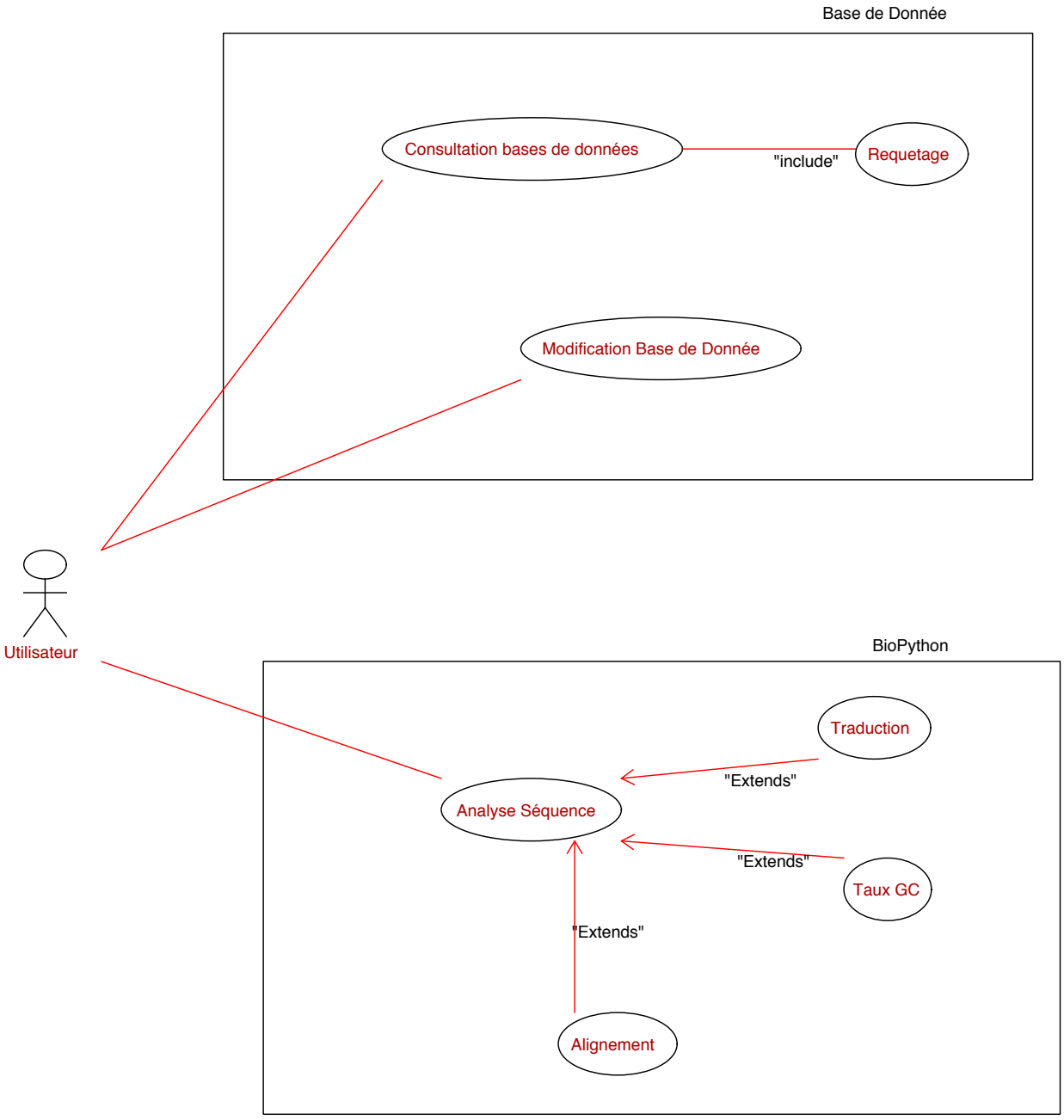

FIG. 2 : Diagramme de Cas d'Utilisation

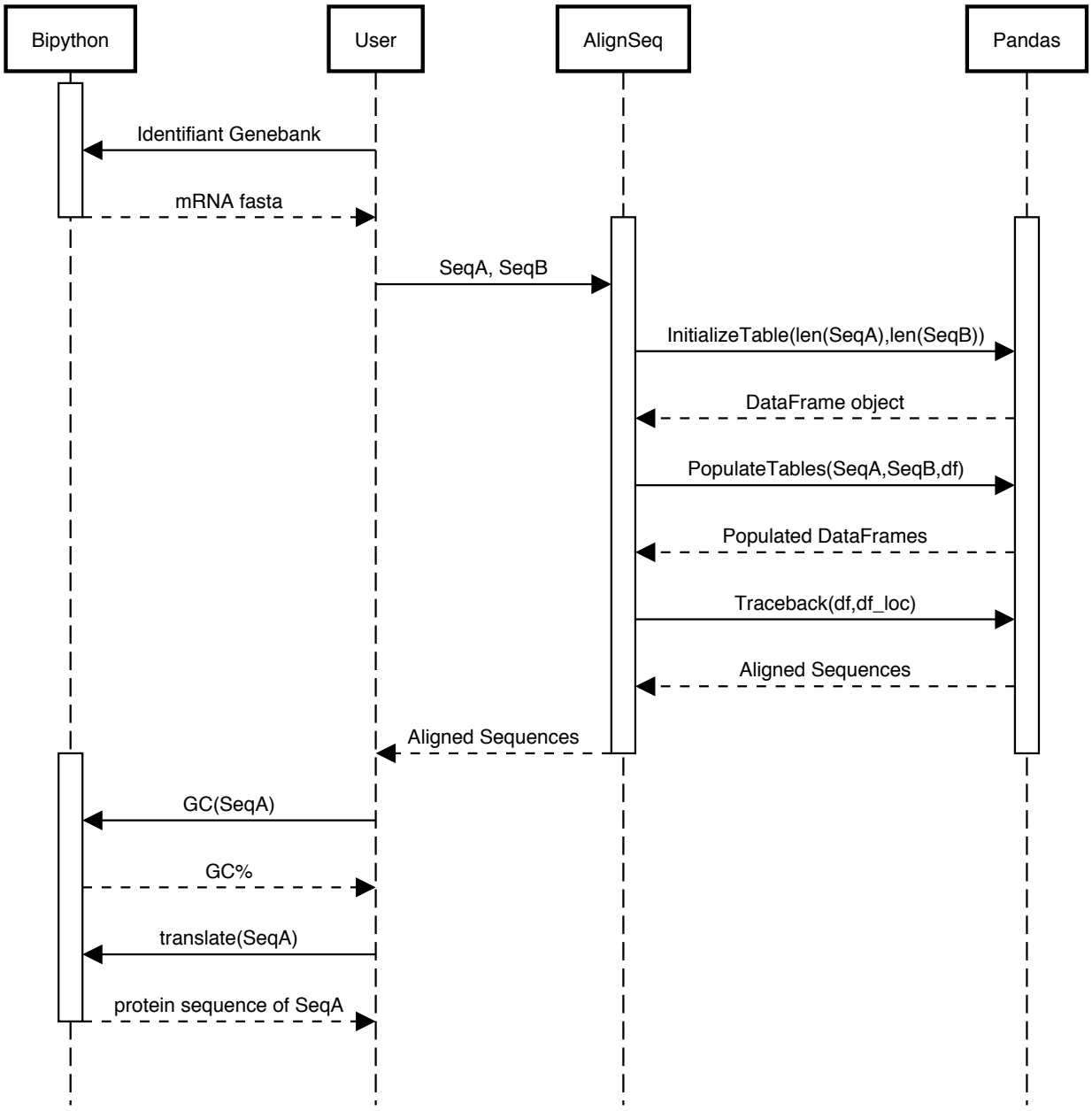

FIG. 3 : Diagramme de séquence pour alignement globale

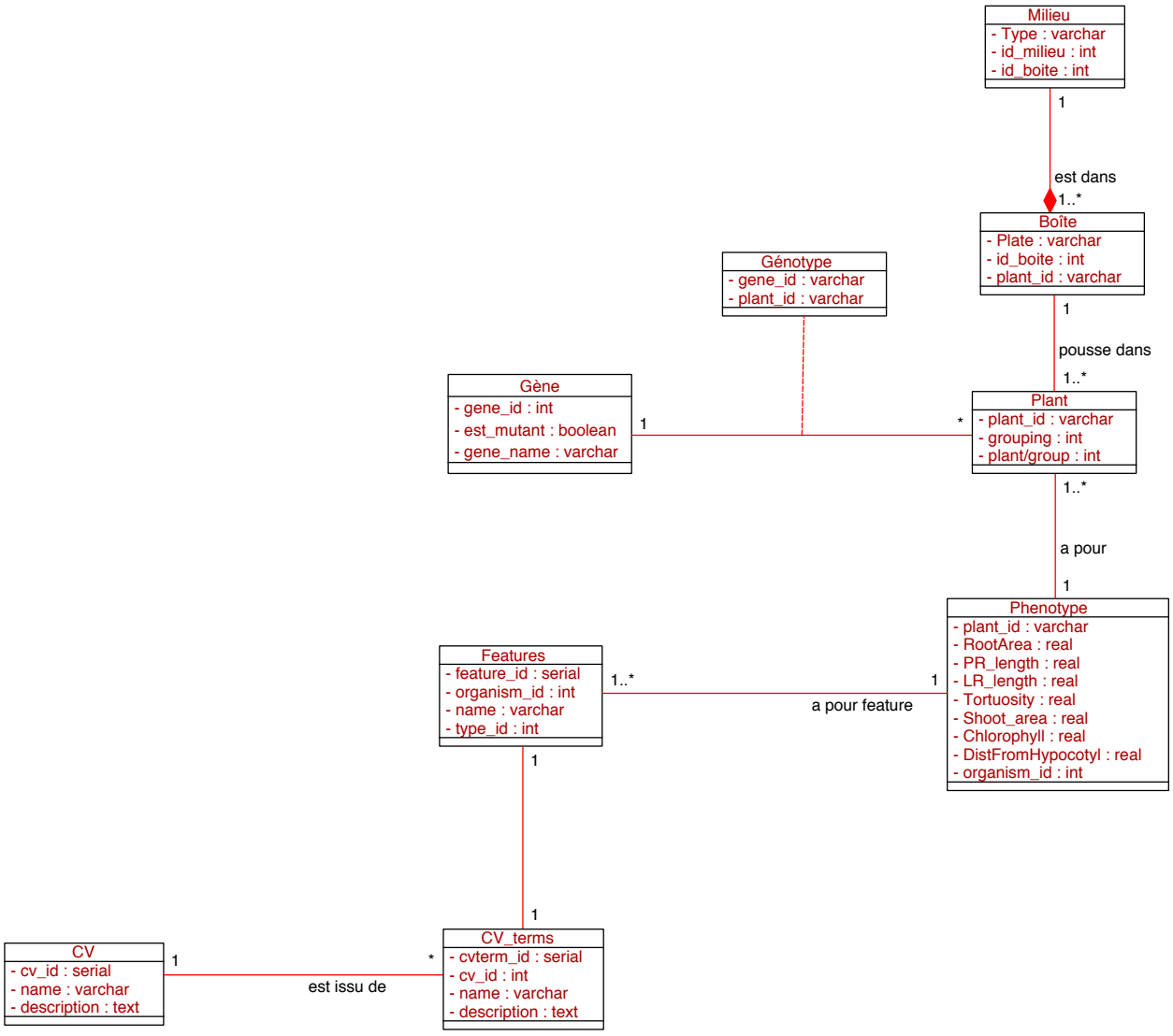

FIG. 4 : Diagramme de classe