Requêtes alternatives dans le contexte d'un entrepôt de données génomiques

Christine Froidevaux*, Frédéric Lemoine*

*LRI, CNRS UMR 8623, Université Paris Sud 11, France
{chris,lemoine}@lri.fr,

Résumé. Afin d'aider les biologistes à annoter des génomes, ce qui nécessite l'analyse, le croisement, et la comparaison de données provenant de sources diverses, nous avons conçu un entrepôt de données de génomique microbienne. Nous présentons la structure globale flexible de l'entrepôt et son architecture multi-niveaux et définissons des correspondances entre ces niveaux. Nous introduisons ensuite la notion de requête alternative et montrons comment le système peut construire l'ensemble des requêtes alternatives à une requête initiale. Pour cela, nous indroduisons un mécanisme d'interrogation qui repose sur l'architecture multi-niveaux, et donnons un algorithme de calcul des requêtes alternatives.

1 Introduction

Avec l'entrée dans l'ère post-génomique, l'avancée du séquençage de génomes et l'utilisation de plus en plus massive d'expériences à haut débit produisent une quantité gigantesque de données biologiques. La conception de systèmes de gestion de données pour stocker et interroger cette information devient cruciale, en particulier dans le domaine de l'annotation fonctionnelle des génomes, qui consiste en l'attribution d'une fonction biologique aux produits de chaque gène. Cette tâche est indispensable pour savoir quels gènes sont impliqués dans certains processus (e.g la pathogénicité pour les génomes microbiens).

C'est dans ce contexte que nous avons conçu l'entrepôt de données génomiques Microbiogenomics[1], dont l'objectif est de rassembler des données de génomique microbienne, pour l'annotation fonctionnelle (ou la ré-annotation) de génomes microbiens (Lemoine et al., 2007). Pour réaliser cette tâche d'annotation, les biologistes ont besoin d'une grande variété de données (telles que des données fonctionnelles, d'homologie, de voies métaboliques, etc.) qui se trouvent dans diverses sources de données dispersées sur le web. Leur travail consiste à naviguer dans les sources de données, trouver des gènes / protéines homologues à leurs gènes / protéines d'étude, comparer les données qui proviennent de ces différentes sources et finalement prendre une décision quant à la fonction de leurs protéines d'intérêt.

Notre objectif est de pouvoir effectuer des calculs sur les données, ainsi que d'appliquer des techniques de fouille de données telles que l'extraction de règles d'associations. C'est pourquoi nous avons choisi une architecture d'entrepôt de données, bien adaptée à ces tâches. Notre entrepôt est spécifique et ne suit pas la définition classique d'un entrepôt de données de

[1] http ://microbiogenomics.u-psud.fr

(Inmon, 2005) dans la mesure où il intègre de nombreuses sources, tout en les laissant indépendantes les unes des autres. Ces sources proposent des données de différentes provenances, et n'ayant pas subi les mêmes traitements : ces données peuvent par conséquent fournir des informations redondantes ou conflictuelles. L'entrepôt permet à l'utilisateur d'exploiter ces complémentarités et ces divergences, ainsi que les différents points de vue sur les données qu'offrent les multiples sources. C'est cette dernière caractéristique que nous allons exploiter à des fins de ré-annotation, par la mise au point d'un module d'interrogation pour l'entrepôt, afin de répondre aux requêtes des biologistes, en tenant compte de leurs besoins en terme de consultation des sources.

Parmi les différentes approches pour l'intégration de données, on trouve les systèmes médiateurs tels que Tambis (Stevens et al., 2000), K2/Kleisli (Davidson et al., 2001) qui utilisent la notion de vue et systèmes pairs-à-pairs tels que Piazza (Halevy et al., 2003). Une autre approche intègre les sources de données en les rapatriant localement dans des entrepôts génomiques tels que BioWarehouse (Lee et al., 2006), Biozon (Birkland et Yona, 2006), GEDAW (Guérin et al., 2005), GUS (Davidson et al., 2001) ou Biomart (Durinck et al., 2005) dans le domaine de la génomique. Enfin, d'autres approches reposent sur la notion de chemins et graphes des entités, approches dîtes navigationnelles telles que Biomediator (Shaker et al., 2004), BioNavigation et BioGuide (Cohen-Boulakia et al., 2006). Notre approche est originale puisque nous nous situons dans une approche entrepôt, en offrant la possibilité de construire des requêtes mixtes et d'obtenir des réponses aux différentes requêtes alternatives.

Dans une première partie, nous décrivons brièvement le contenu de l'entrepôt, en terme des données avec lesquelles nous travaillons ainsi que du schéma relationnel de l'entrepôt. Dans la deuxième partie, nous présentons l'architecture multi-niveaux sur laquelle repose notre module d'interrogation. Dans une troisième partie nous présentons la manière dont l'entrepôt peut être interrogé, introduisons la notion de requêtes alternatives, et donnons les algorithmes que nous avons mis au point pour manipuler les requêtes. Enfin, nous présentons un exemple biologique pour montrer l'intérêt pratique d'une telle approche.

2 Contenu de l'entrepôt

La structure bas niveau de l'entrepôt de données est présentée ci-dessous. Nous exposons tout d'abord l'ensemble des sources choisies et montrons l'organisation de ces dernières dans l'entrepôt. Cette structure correspond au niveau des bases de données dans notre architecture générale (voir fig 1 page 4).

2.1 Données biologiques

Les données présentes dans l'entrepôt sont de différentes provenances. Nous intégrons une version relationnelle des principales sources de données publiques utiles pour l'annotation fonctionnelle : RefSeq, Genome Reviews, Uniprot, et Kegg, ainsi que des sources de données locales : sous-parties d'AGMIAL (Bryson et al., 2006), Genopage (Cohen-Boulakia et al., 2002), Syntebase et Orenza (Lespinet et Labedan, 2006).

Nous pouvons classer ces données en trois catégories. La première concerne les *données primaires*, c'est-à-dire les données brutes stockées dans les sources, qui n'ont pas subi de calculs (par exemple les données de génome, séquences protéiques, voies métaboliques, réactions

enzymatiques, etc.). Les données de la deuxième catégorie sont appelées *données secondaires* et résultent de programmes d'alignement de séquences protéiques tels que Blast et Darwin, qui visent à calculer la similarité entre les protéines. La troisième catégorie de données est constituée par les données *tertiaires*, qui sont le résultat de calculs plus complexes, effectués à partir des données primaires et secondaires, tels que la détection de la conservation de l'ordre des gènes, le calcul des relations d'orthologie entre les protéines, et les calculs de phylogénies. De manière générale, nous pouvons définir la relation d'orthologie en disant que deux protéines sont orthologues si elles dérivent d'une protéine ancestrale commune dans leur unique ancêtre commun. Les deux protéines sont donc le fruit d'un évènement de spéciation.

Les données à intégrer dans l'entrepôt ont été choisies en concertation avec les biologistes de l'IGM (Institut de Génétique et Microbiologie - CNRS/université Paris Sud), et de l'équipe MIG (Mathématique, Informatique et Génome - INRA) qui ont exprimé leurs besoins en termes d'annotation et ré-annotation fonctionnelle.

2.2 Schéma global de l'entrepôt

L'entrepôt est au format relationnel et est implémenté sous PostgreSQL[2]. Comme indiqué précédemment, le *schéma global* de l'entrepôt garde les sources de données indépendantes. En effet, pour chaque source intégrée, nous stockons sa version relationnelle de manière indépendante dans l'entrepôt. Les avantages d'un tel schéma global sont les suivants : 1) chaque source peut être mise à jour indépendamment des autres, 2) il est possible d'ajouter facilement une source de données, et par conséquent nous pouvons, non seulement ajouter une nouvelle source de données publique, mais aussi personnaliser l'entrepôt d'après les besoins des utilisateurs en ajoutant des sources de données locales privées par exemple, et 3) il conserve les complémentarités et les divergences que l'on trouve dans les sources de données.

L'entrepôt est centré sur les protéines, il est donc particulièrement important de pouvoir interroger aisément cette entité biologique. Cependant, toutes les sources de données n'utilisent pas le même identifiant pour décrire une même protéine. Pour réaliser l'intégration sémantique des données, nous avons construit des associations entre certaines tables relationnelles des sources, quand cela était possible, au moyen de tables de liens indépendantes. Pour les protéines par exemple, pour calculer ces associations entre les sources, nous avons tout d'abord considéré les références croisées qui existent déjà dans les sources et qui associent entre elles les protéines de différentes sources qui n'ont pas le même identifiant mais représentent la même instance. Pour confirmer ces associations, nous avons exécuté une série d'alignements (Blast) avec des seuils adaptés à la détection de l'identité ou la très forte similarité de séquence, qui assure la correspondance entre les sources pour cette entité en particulier.

Le *schéma global* de l'entrepôt est défini comme suit :

$$Sch(Wh) = \left\{ \bigcup_{i=1}^{n} Sch(BD_i) \bigcup Sch(LBD) \right\}$$

Où n est le nombre de sources intégrées, $Sch(BD_i)$ est le schéma de la source BD_i, et $Sch(LBD)$ est le schéma de la base de données de liens (qui contient les tables de liens)

[2]http ://www.postgresql.org/

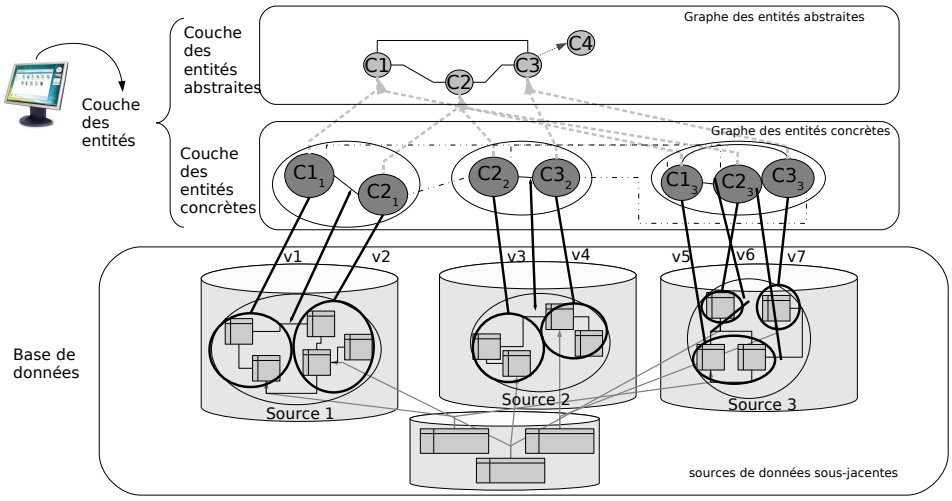

FIG. 1 – *Architecture de l'entrepôt*

3 Modélisation des entités présentes dans l'entrepôt

Nous avons conçu un module d'interrogation dédié qui présente plusieurs caractéristiques spécifiques. La première est motivée par le fait qu'interroger un tel schéma relationnel est difficile à cause non seulement de sa taille en terme de nombre de tables, mais aussi car une entité est présente dans plusieurs sources. Ce schéma n'est pas donné ici du fait de sa taille et de sa complexité. Nous voulons aider l'utilisateur à construire ses requêtes. La seconde est la possibilité de proposer à l'utilisateur des requêtes *alternatives* qui sont des requêtes similaires à sa requête initiale, et qui peuvent lui donner des résultats intéressants car inattendus, et qui concernent le même sujet. Troisièmement, ce module permet à l'utilisateur d'exprimer ses requêtes sans avoir à préciser quelle source doit être interrogée. Pour cela, nous avons introduit la couche des entités sur laquelle repose le module d'interrogation et qui est constituée de deux sous-couches. L'architecture globale du système est montrée fig 1. Elle est composée de la couche des entités et des bases de données sous-jacentes. L'introduction de ces différentes couches permet une indépendance entre le stockage des entités et la façon dont l'utilisateur peut les percevoir.

3.1 Couche des entités abstraites

La *couche des entités abstraites* consiste en un modèle conceptuel qui contient des entités biologiques élémentaires. Puisque nous voulons un modèle dédié et spécialisé, et qu'aucun modèle consensuel n'existe, nous avons conçu notre propre modèle conceptuel, dans une démarche ascendante, en partant du contenu des sources, en collaboration avec les biologistes de MIG et de l'IGM. Ce modèle est composé d'entités, de propriétés attachées à ces entités, et de liens biologiques les associant, et peut être vu comme un graphe.

Plus formellement, le *graphe des entités abstraites* est un multigraphe orienté étiqueté $G_{ae} =< N_a, A_a, l_a >$, où $N_a = N_{ae} \cup N_{ap}$ (N_{ae} et N_{ap} disjoints) dénote l'ensemble des sommets, et $A_a = A_{a\_isa} \cup A_{as} \cup A_{ap} \subseteq N_a \times N_a$ l'ensemble des arêtes. Les sommets peuvent être classés en deux catégories, les *entités abstraites* N_{ae} (par exemple Protéine, ou Enzyme dans fig 1), et les propriétés N_{ap} telles que la séquence pour une protéine (non représentées dans fig 1). Les arêtes relient deux entités abstraites, ou une entité abstraite et une propriété. $A_{a\_isa} \subset N_{ae} \times N_{ae}$, est l'ensemble des liens isa associant deux entités abstraites (représentés par une flèche gris clair pointillée dans la fig 1, par exemple, le lien entre C_3 et C_4). $A_{as} \subset N_{ae} \times N_{ae}$ est l'ensemble des liens biologiques entre les entités abstraites (représentés en traits pleins fins dans figure, par exemple le lien *Existe-dans* qui associe une Protéine et un Organisme), et finalement, $A_{ap} \subset N_{ae} \times N_{ap}$ est l'ensemble des arêtes entre les entités abstraites et les propriétés (non représentés dans fig 1). l_a est une fonction d'étiquetage telle que : 1) Les arêtes a de A_{as}, sont étiquetées par $type(a) \in L_{biol}$ désignant un nom de lien biologique, 2) Les arêtes p de A_{ap} sont étiquetées par $type(p) ='' APourPropriete''$, 3) Les arêtes s de A_{s_isa} sont étiquetées $type(i) ='' isa''$, 4) Les sommets v_1 de A_{ae} sont étiquetés par $type(v_1) \in L_{abstraite}$ désignant un nom d'entité biologique et 5) Les sommets v_2 de A_{ap} sont étiquetés par $type(v_2) \in L_{prop}$ désignant le nom de la propriété.

3.2 Couche des entités concrètes

Sous la couche des entités abstraites, nous introduisons la *couche des entités concrètes*. Elle contient les entités présentes dans une source, que nous appelons *entités concrètes* (dans le même esprit que les sources entités de Cohen-Boulakia et al. (2007)). Elles sont représentées par des cercles gris foncés dans fig 1. Par exemple, "UniprotProtéine" est l'entité concrète désignant une protéine présente dans la source Uniprot.

Il y a plusieurs catégories de liens dans la couche des entités concrètes. La première catégorie est constituée des liens biologiques associant deux entités concrètes de la même source de données (par exemple, une protéine de Uniprot *Existe-dans* le génome d'un organisme de Uniprot, lignes pleines noires de fig 1). Chacun de ces liens correspond à un chemin (de taille supérieure ou égale à un) dans la couche des entités abstraites, et à une jointure (par clé primaire/clé étrangère ou par l'intermédiaire d'une table de liens) dans la source sous-jacente. La deuxième catégorie de liens est formée par les associations entre les entités concrètes et leurs propriétés (par exemple, la sequence associée d'une protéine de Uniprot). Les liens inter-sources forment la troisième catégorie de liens. Ils correspondent à une association entre deux entités concrètes n'appartenant pas à la même source, et à une association par l'intermédiaire d'une table de liens de LBD dans la source sous-jacente (lien entre les protéines de Uniprot et les protéines de Kegg par exemple). Nous définissons le *graphe des entités concrètes* comme un multigraphe non orienté étiqueté :

$G_c =< N_c, A_c, l_c >$, avec $N_c = N_{ce} \cup N_{cp}$ (N_{ce} et N_{cp} disjoints) et $A_c = A_{cs} \cup A_{cp} \cup A_{dl}$. N_{ce} contient les sommets qui représentent les entités concrètes, et N_{cp} les sommets qui représentent les propriétés. $A_{cs} \subset N_{ce} \times N_{ce}$ contient les liens biologiques entre deux entités concrètes de la même source, $A_{cp} \subset N_{ce} \times N_{cp}$ les liens entre les entités concrètes et leurs propriétés, et $A_{dl} \subset N_{ce} \times N_{ce}$ les liens directs entre deux entités concrètes n'appartenant pas à la même source. l_c est une fonction d'étiquetage telle que : 1) Les arêtes a_1 de A_{cs} sont étiquetées par $type(a_1) \in L_{biol}$ désignant le nom du lien biologique, 2) Les arêtes a_2 de A_{cp} sont étiquetées $type(a_2) ='' APourPropriete''$, 3) Les arêtes a_3 de A_{dl} sont étiquetées

$type(a_3) ='' \ LienDirect''$, 4) Les sommets s_1 de N_{ce} sont étiquetés $type(s_1) \in L_{concret}$ désignant le nom de l'entité concrète, et 5) Les sommets s_2 de $N_{cp} \in L_{prop}$ sont étiquetés $type(s_2)$ désignant le nom de la propriété.

3.3 Couche des entités

La *couche des entités* regroupe les deux sous-couches des entités abstraites, et des entités concrètes et est représentée par le *graphe général des entités*, composé des deux graphes définis ci-dessus, avec en plus des liens entre eux (voir fig1).

À chaque entité concrète correspond une entité abstraite dans la couche des entités abstraites, et une vue sur une source de données sous-jacente, qui exprime comment récupérer des instances de l'entité en question. Les entités concrètes sont mises en correspondance avec les entités abstraites au moyen d'une relation isa (Par exemple, UniprotProtéine *isa* Protéine). Les propriétés d'une entité concrète dans une source sont les propriétés de ses parents dans la hiérarchie isa (où les liens isa induisent une relation transitive), et éventuellement des propriétés propres à cette entité dans cette source.

Le *graphe général des entités* est un multigraphe graphe orienté étiqueté $G =< N, A, l >$, où $N = N_a \cup N_c$ et $A = A_a \cup A_c \cup A_{c\_isa} \cup A_{ms}$, où A_a et A_c sont les ensembles d'arêtes des graphes d'entités abstraites et d'entités concrètes.

$A_{c\_isa} \subset N_{ce} \times N_{ae}$ contient les liens isa qui connectent chaque entité concrète à son entité abstraite correspondante. $A_{ms} \subset N_{ce} \times N_{ae}$ contient les liens biologiques entre une entité concrète et une entité abstraite, qui dérivent des liens biologiques entre entités abstraites. Par exemple si on a Protéine *Existe-dans* Organisme (dans A_{as}), et UniprotProteine *isa* Protéine (dans A_{c_isa}) alors on aura UniprotProteine *Existe-dans* Organisme (dans A_{ms}). l est une fonction d'étiquetage qui associe la même étiquette que précédemment aux sommets et arêtes connus, et pour les autres 1) Les arêtes a_1 de A_{ms} sont étiquetées par $type(a_1) \in L_{biol}$, 2) Les arêtes a_2 de A_{c_isa} sont étiquetées $type(a_2) ='' \ isa''$.

Correspondances et Contraintes sur le graphe.

- On note ϕ la fonction de correspondance partielle qui associe une entité concrète à un couple (entité abstraite, source). Par construction, chaque entité concrète correspond à une et une seule entité abstraite.
- Nous construisons une fonction de correspondance ϕ_2 entre les arêtes de A_{cs} (liens biologiques de la couche des entités concrètes) et les arêtes A_{as} de la couche des entités abstraites. À chaque arête de A_{cs} correspond un chemin formé d'arêtes consécutives de A_{as}. Cette fonction est établie manuellement. Nous pouvons par exemple avoir le lien **Une Protéine de Swissprot** - Existe dans - **Un Organisme de Swissprot** ; qui correspond au chemin : - **une Protéine** - Traduite depuis - **Un ARNm** - Transcrit depuis - **un gène** - Dans un - **Organisme**.
- Pour chaque paire de sommets c_1, c_2 de N, s'il existe une arête $a = (c_1, c_2)$ de $A_{c\_isa} \cup A_{s\_isa}$ alors on dit que $type(c_1)$ isa $type(c_2)$, ou $type(c_1) < type(c_2)$. On a alors $type(c_1) \leq type(c_2)$ équivalent à ($type(c_1) < type(c_2)$ ou $type(c_1) = type(c_2)$)
- Une entité concrète hérite les propriétés de ses parents. Par exemple, si une protéine a une séquence, alors une protéine de Uniprot aura une séquence.
- Une entité concrète hérite des liens biologiques de ses parents, (liens de A_{ms}).

4 Interroger l'entrepôt de données

Nous montrons comment la couche des entités est utilisée pour construire des requêtes expressives. Notre module d'interrogation permet d'interroger l'entrepôt de données en termes à la fois des entités abstraites et des entités concrètes. Cela signifie que la requête peut être une combinaison de concepts de la couche des entités abstraites et de concepts de la couche des entités concrètes (*requête mixte*). Elle est transparente pour certaines entités (on ne sait pas quelle source l'entrepôt va interroger), et précise pour d'autres (on spécifie dans quelle source ces entités doivent être considérées).

4.1 Définition des requêtes

Nous exprimons les requêtes en exploitant l'architecture multi-niveaux exposée plus haut. Une requête mixte est représentée par un graphe dont les sommets et les arêtes "sont choisis" dans le graphe général des entités G, de manière proche de (Mugnier et Leclère, 2007).

Graphe de requête mixte. Soit $G =< N, A, l >$ un graphe général d'entités. Un *graphe de requête mixte* G_q sur G, est défini par $< N_q, A_q, l_q >$, avec N_q l'ensemble des sommets, $A_q \subset N_q \times N_q$ l'ensemble d'arêtes, et l_q une fonction d'étiquetage des sommets et des arêtes qui vérifient les propriétés suivantes.

Soit c un sommet de G_q. Il est étiqueté par une paire $(type(c), marker(c))$, où $type(c) \in L_{concret} \cup L_{abstraite} \cup L_{prop}$ et $marker(c) \in \tau \cup \{*\}$, avec τ un ensemble de marqueurs individuels et $*$ le marqueur générique (qui permet par exemple d'interroger l'organisme particulier *E.coli* ou n'importe quel organisme). Les marqueurs individuels permettent d'interroger un individu et non plus le concept générique. Soit r une arête. Elle est étiquetée par $type(r)$, avec $type(r) \in L_{biol} \cup \{''LienDirect'', '' APourPropriete''\}$.

Nous introduisons la contrainte suivante : Une arête n'existe entre deux sommets de G_q que si elle existe dans G. Autrement dit : $\forall r = (x, y) \in A_q, \exists t = (c_1, c_2) \in A - (A_{a\_isa} \cup A_{c\_isa}) tq : type(r) = type(t),\ type(x) = type(c_1),\ type(y) = type(c_2)$.

Les requêtes concernent alors aussi bien des entités abstraites que des entités concrètes, sont sous forme de graphe, et une entité donnée peut figurer plusieurs fois dans la requête. Par ces deux derniers aspects, les requêtes que nous formons sont plus expressives que les requêtes communément exprimées dans les approches à base de chemins.

Exemple de requête mixte : "Quels sont les numéros ec des enzymes de Uniprot présentes dans l'organisme *Buchnera aphidicola* et de leur paralogues ?". (Deux protéines sont paralogues si elles proviennent d'une duplication ancestrale des gènes les codant).

4.2 Construire des requêtes alternatives

Les requêtes alternatives sont définies comme des requêtes ayant la même sémantique biologique, mais étant de formes différentes. Elles permettent d'exploiter la richesse de l'entrepôt en donnant à l'utilisateur plusieurs points de vue. Elles peuvent interroger les mêmes entités dans des sources différentes, et potentiellement d'autres entités, si les liens de la requête initiale restent sémantiquement équivalents. Étant donné une requête mixte G_q, l'ensemble des *requêtes alternatives* de G_q est constitué des *requêtes de bas niveau* ayant la même *abstraction* modulo le graphe général des entités G. Nous précisons ci-dessous ces notions.

Requête de bas niveau. Une requête de bas niveau $q_l =< N_l, A_l, l_l >$ définie sur G est une requête mixte comprenant uniquement des entités concrètes : $\forall n \in N_l$, $type(n) \in L_{concret} \cup L_{prop}$, et $\forall a \in A_l$, $type(e) \in L_{biol} \cup \{''APourPropriete'', ''LienDirect''\}$.
Requête de haut niveau. Une requête de haut niveau $q_h =< N_h, A_h, l_h >$ sur G est une requête mixte contenant uniquement des entités abstraites : $\forall n \in N_h$, $type(n) \in L_{abstraite} \cup L_{prop}$, et $\forall a \in A_h$ $type(e) \in L_{biol} \cup \{''APourPropriete'', ''LienDirect''\}$.
Abstraction. Étant donné une requête mixte q_m et une requête de haut niveau q_h, définies sur G, q_h est l'abstraction de q_m pour G si : 1) Il n'y a aucune propriété attachée aux entités dans q_h ; 2) À chaque sommet s de q_m correspond un sommet e dans q_h tels que $type(e) \in L_{abstraite}$, et si $type(s) \in L_{abstraite}$ alors $type(s) = type(e)$, et sinon si $type(s) \in L_{concret}$ alors $type(s) < type(e)$; 3) À chaque arête a de q_m correspond un chemin c dans q_h tel que $c = \phi_2(a)$ et les types des extrémités de l'arête sont $\leq$ aux types des extrémités du chemin.

La requête abstraite correspondant à la requête mixte de la section 4.1 est : "Quels sont les ec des enzymes traduits à partir d'ARNm eux-mêmes transcrits à partir de CDS placés sur un réplicon du génome de *Buchnera aphidicola*, et de leurs paralogues ?"

Première étape : Requêtes mixtes équivalentes. Nous utilisons ici le mapping ϕ_2. L'objectif est, connaissant les raccourcis sémantiques pouvant exister entre les entités, de calculer l'ensemble des requêtes mixtes équivalentes.

Deux *requêtes mixtes sont équivalentes* si pour chaque chemin élémentaire composant la première, il existe un chemin élémentaire dans la seconde qui lui est équivalent, et vice versa.

Introduisons tout d'abord la notion de point de cassure. Un sommet $n \in N_q$ d'une requête mixte G_q est considéré comme *point de cassure* (sommets 1,5 et 6 dans fig 2) s'il représente une entité et s'il vérifie au moins une des conditions suivantes : 1) il a moins (ou plus) de deux liens avec d'autres sommets représentants des entités pour obtenir des chemins linéaires ; 2) il est lié à au moins une propriété ; 3) il a un marqueur individuel ; 4) il représente une entité concrète (cela signifie que l'utilisateur veut cette entité en particulier).

Les chemins reliant ces points de cassures définissent des *chemins élémentaires* tels que 1-2-3-4-5 et 1-8-7-6 dans fig 2. Deux chemins élémentaires sont *équivalents* modulo le graphe des entités G si leurs requêtes élémentaires associées ont la même abstraction pour G.

L'algorithme que nous proposons prend en entrée une requête mixte, et retourne toutes les requêtes mixtes équivalentes, tenant compte du mapping ϕ_2. La première étape est d'extraire de la requête initiale des *chemins élémentaires* (étape 1 dans fig 2). Ces chemins élémentaires sont définis comme étant les chemins reliant deux sommets considérés comme des points de cassure. Pour chaque chemin élémentaire, nous appliquons l'algorithme spécifié ci-dessous pour trouver tous ses chemins élémentaires équivalents (étape 2 dans fig 2).

L'algorithme prend en entrée un chemin élémentaire. À partir de ce chemin, il construit le graphe contenant tous les chemins élémentaires alternatifs (voir (a),(b),(c) dans fig 2), en utilisant les correspondances ϕ_2, et en ne gardant que les arêtes a dont $\phi_2(a)$ est *inclus* dans le chemin initial. Une fois ce graphe construit, on le parcourt alors partant de la première entité du chemin élémentaire initial, jusqu'à la dernière entité du chemin élémentaire, pour trouver tous les chemins élémentaires équivalents. Un exemple du passage de la requête initiale aux chemins élémentaires équivalents est représenté fig 2.

On reforme finalement un ensemble de requêtes mixtes, en parcourant, pour chaque chemin élémentaire initial, tous les chemins élémentaires équivalents.

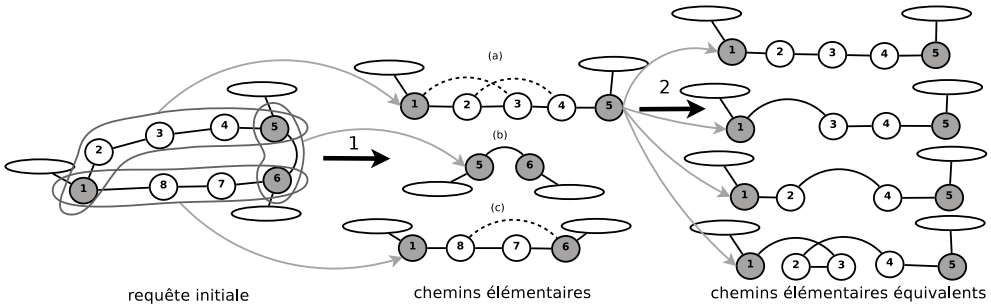

FIG. 2 – *(1) La requête utilisateur est séparée en chemins élémentaires. (2) Tous les chemins élémentaires équivalents pour le chemin (a) sont calculés en utilisant ϕ_2. Les ovales représentent les propriétés attachées aux entités (abstraites ou concrètes) représentées par un cercle. Les cercles gris foncé indiquent les points de cassure, les lignes pleines les liens biologiques dans la requête, et les lignes pointillées les raccourcis définis dans le ϕ_2.*

Deuxième étape : Requêtes alternatives finales. Soit $G_q =< N_q, A_q, l_q >$ une requête mixte sur le graphe des entités G, calculée comme précédemment, le but est d'obtenir toutes les requêtes alternatives de bas niveau. Pour cela, nous construisons le graphe des requêtes alternatives $G_{hq} =< N_{hq}, A_{hq}, l_{hq} >$ (cf. fig 3). Chaque sommet de N_{hq} représente une entité (concrète ou abstraite) ou une propriété ; et les arêtes $A_{hq} \subset N_{hq} \times N_{hq}$ connectent deux entités abstraites, deux entités concrètes, une entité concrète et son entité abstraite correspondante, ou une entité et une propriété.

1) Pour chaque sommet $n \in N_q$ représentant une entité abstraite ($type(n) \in L_{abstraite}$) de G_q, nous construisons un sommet v de N_{hq} (en gris dans fig 3) tel que $type(v) = type(n)$. Pour chaque n' du graphe général des entités G, tq $type(n') \in L_{concret}$ et $type(n') < type(n)$, nous construisons un sommet v' (en blanc dans fig 3) avec $type(v') = type(n')$ et l'arête $a = (n, n')$ liant ces deux sommets (double flèche dans fig 3). À chaque sommet v', nous ajoutons les propriétés de v (dans G_q). Pour chaque sommet n_2 de G_q représentant une entité concrète ($type(n) \in L_{concret}$) de G_q, nous construisons un sommet $v_2 \in N_{hq}$ représentant l'entité abstraite correspondante (en gris dans la figure) tel que $type(v_2) \in L_{abstraite}$ et $type(n_2) < type(v_2)$ de G. Nous appliquons la même règle que précédemment, c'est-à-dire que nous lions v_2 aux sommets représentant les entités concrètes de v_2. Pour chaque arête $a = (s_1, s_2)$ de G_q tel que $type(s_1) \in L_{abstrait}$ et $type(s_2) = L_{concret}$, on construit une arête dans G_{hq}, connectant les sommets représentants les deux entités abstraites correspondantes. Pour chaque arête de G_q connectant deux sommets représentants des entités abstraites, on construit une arête de G_{hq} connectant les deux sommets correspondants (voir fig 3).

2) Une fois le graphe des requêtes alternatives construit, il est parcouru en profondeur, pour trouver toutes les manières de répondre à la requête dans les sources. Il retourne l'ensemble des graphes correspondant aux requêtes alternatives à la requête initiale de l'utilisateur.

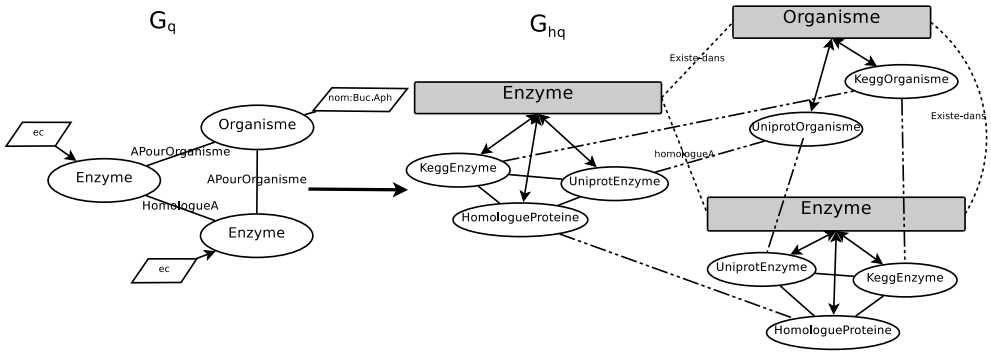

FIG. 3 – *Graphe des requêtes alternatives G_{hq} construit à partir de la requête G_q obtenue à la fin de la première étape. Dans G_{hq}, les lignes pleines représentent les arêtes A_{hq} dont le $type$ est dans $''LienDirect''$, les lignes pointillées sont les arêtes entre deux entités abstraites dont le $type$ est dans L_{biol}, les tiretés sont les arêtes entre deux entités concrètes dont le $type$ est dans L_{biol}, et les double flèches lient une entité abstraite et ses entités concrètes.*

5 Implémentation et exemple

Nous avons implémenté le graphe général des entités G en RDF (Klyne, 2004), avec l'aide de RDFs (Brickley et Guha, 2004). RDFs peut en effet modéliser la hiérarchie des entités, les propriétés, et les associations que nous avons définies. De plus, il est simple et assez expressif pour notre objectif. Nous avons choisi un sous-ensemble de la syntaxe de SparQL (Prud'hommeaux et Seaborne, 2007) pour implémenter nos requêtes, car SparQL est bien adapté à la définition des requêtes mixtes, et que sa syntaxe est simple à apprendre. De plus il existe des frameworks qui permettent de manipuler des requêtes SparQL et des documents RDFs.

Nous reprenons la requête mixte de la section 4.1 pour souligner l'importance de comparer les résultats des requêtes dans des sources de données différentes.

un coût peu élevé dépendant du nombre de chemins élémentaires équivalents dans les mappings, et du nombre d'entités concrètes par entité abstraite dans la requête) 15 requêtes alternatives qui donnent des résultats différents. Le graphe G_{hq} de la figure 3 présente plusieurs de ces requêtes alternatives. Nous obtenons par exemple : "Quels sont les ec. des enzymes de kegg ayant pour organisme Buchnera chez Kegg, et les ec de leurs homologues chez Kegg, qui ont elles-mêmes pour organisme Buchnera chez Kegg", ou encore "Quels sont les enzymes de Uniprot ayant pour organisme Buchnera chez Uniprot, et les ec de leurs homologues de Uniprot, qui ont pour organisme Buchnera chez Uniprot". De manière surprenante, les résultats diffèrent selon les sources que l'on consulte. Par exemple à la protéine *P57220* est associé le ec number $3.6.1.-$ chez Uniprot et aucun ec chez Kegg. De même pour la protéine *P57506*, l'ec number associé chez Uniprot est $3.1.11.-$ et chez Kegg $2.7.7.7$. Il est donc important de disposer de plusieurs façons de récupérer l'information de fonction enzymatique, pour pouvoir comparer, et *a posteriori*, décider d'une stratégie pour une réannotation plus cohérente.

6 Conclusion

Le module d'interrogation que nous proposons exploite la richesse de l'entrepôt de données, et offre à l'utilisateur des voies alternatives intéressantes pour récupérer des données de sources différentes. Il exploite la structure multi-niveaux que nous avons introduite, et qui permet l'expression de requêtes mixtes sous forme de graphe.

La notion de reformulation est présente d'autres travaux. Dans (Lowden et Robinson, 2004), l'objectif est de trouver une requête de coût minimal parmi l'ensemble des requêtes alternatives fournissant exactement le même ensemble de résultats. Dans (Necib et Freytag, 2004), c'est la requête elle-même, exprimée en termes de concepts, qui est reformulée ; les relations sémantiques (isa, etc.) présentes entre les concepts sont alors exploitées dans cette reformulation. Dans notre cas, l'objectif est différent, puisque nous reformulons les requêtes de l'utilisateur, en gardant strictement les mêmes concepts, mais en variant les sources les fournissant, de manière à obtenir de nouvelles requêtes (nos requêtes alternatives) qui donnent des résultats complémentaires selon les sources, que l'on peut comparer.

Cette approche est générique, dans le sens où elle peut être adaptée à d'autres entrepôts relationnels ayant une architecture semblable, c'est-à-dire 1) ayant la possiblité de dégager les graphes des entités abstraites et concrètes, 2) possédant des entités non-unique et 3) ayant une réconciliation sémantique des données via des tables de liens.

7 Remerciements

Ce travail a été partiellement soutenu par l'ANR Masse de données Microbiogenomics. Nous remercions ses participants, en particulier J.F Gibrat (MIG), B. Labedan (IGM). Nous remercions également A. Morgat (SIB), E. Coissac (LEC) et S. Cohen Boulakia (LRI).

Références

Birkland, A. et G. Yona (2006). BIOZON : a system for unification, management and analysis of heterogeneous biological data. *BMC Bioinformatics 7*, 70.

Brickley, D. et R. Guha (2004). Rdf vocabulary description language 1.0 : Rdf schema. *http ://www.w3.org/TR/rdf-schema/*.

Bryson, K., V. Loux, R. Bossy, P. Nicolas, S. Chaillou, M. van de Guchte, S. Penaud, E. Maguin, M. Hoebeke, P. Bessieres, et J. Gibrat (2006). AGMIAL : implementing an annotation strategy for prokaryote genomes as a distributed system. *Nucleic Acids Res 34*(12), 3533–45.

Cohen-Boulakia, S., O. Biton, S. Davidson, et C. Froidevaux (2007). BioGuideSRS : querying multiple sources with a user-centric perspective. *Bioinformatics 23*(10), 1301–3.

Cohen-Boulakia, S., S. Davidson, C. Froidevaux, Z. Lacroix, et M. E. Vidal (2006). Path-based systems to guide scientists in the maze of biological data sources. *J Bioinform Comput Biol 4*(5), 1069–95.

Cohen-Boulakia, S., C. Froidevaux, E. Waller, et B. Labedan (2002). Genopage : A database of all protein modules encoded by completely sequenced genomes. In *JOBIM2002*, pp. 187–193.

Davidson, S. B., J. Crabtree, B. P. Brunk, J. Schug, V. Tannen, G. C. Overton, et C. J. J. Stoeckert (2001). K2/kleisli and gus : Experiments in integrated access to genomic data sources. *IBM Systems Journal 40*(2), 512–531.

Durinck, S., Y. Moreau, A. Kasprzyk, S. Davis, B. De Moor, A. Brazma, et W. Huber (2005). Biomart and bioconductor : a powerful link between biological databases and microarray data analysis. *Bioinformatics 21*(16), 3439–3440.

Guérin, E., G. Marquet, A. Burgun, O. Loréal, L. Berti-Equille, U. Leser, et F. Moussouni (2005). Integrating and warehousing liver gene expression data and related biomedical resources in gedaw. In *DILS*, pp. 158–174.

Halevy, A. Y., Z. G. Ives, D. Suciu, et I. Tatarinov (2003). Schema mediation in peer data management systems. In *ICDE*, pp. 505–516. IEEE Computer Society.

Inmon, W. (2005). *Building the Data Warehouse*. John Wiley & Sons.

Klyne, J. (2004). Resource description framework (RDF) : Concepts and Abstract Syntax, recommandation. *W3C : <http ://www.w3.org/TR/rdf-concepts/>*.

Lee, T., Y. Pouliot, V. Wagner, P. Gupta, D. Stringer-Calvert, J. Tenenbaum, et P. Karp (2006). BioWarehouse : a bioinformatics database warehouse toolkit. *BMC Bioinformatics 7*, 170.

Lemoine, F., E. Coissac, A. Morgat, B. Labedan, et C. Froidevaux (2007). Screening functional genomics data by querying a data-warehouse. DILS'2007, poster.

Lespinet, O. et B. Labedan (2006). ORENZA : a web resource for studying ORphan ENZyme activities. *BMC Bioinformatics 7*, 436.

Lowden, B. G. T. et J. Robinson (2004). Improved data retrieval using semantic transformation. In *DEXA*, Volume 3180 of *LNCS*, pp. 391–400. Springer.

Mugnier, M.-L. et M. Leclère (2007). On querying simple conceptual graphs with negation. *Data Knowl. Eng. 60*(3), 468–493.

Necib, C. B. et J. C. Freytag (2004). Using ontologies for database query reformulation. In *ADBIS (Local Proceedings)*.

Prud'hommeaux, E. et A. Seaborne (2007). SPARQL Query Language for RDF. *W3C : <http ://www.w3.org/TR/rdf-sparql-query/>*.

Shaker, R., P. Mork, J. Brockenbrough, L. Donelson, et P. Tarczy-Hornoch (2004). The biomediator system as a tool for integrating biologic databases on the web. *IIWeb04*.

Stevens, R., P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N. Paton, C. Goble, et A. Brass (2000). TAMBIS : transparent access to multiple bioinformatics information sources. *Bioinformatics 16*(2), 184–5.

Summary

We have designed a data warehouse gathering microbial genomic data to help users to perform genome functional annotation, which necessitates data from various sources to be crossed and compared. We present its flexible global structure and its multi-level architecture, and define mappings between the levels. We then introduce the concept of alternative queries based on the multi-level architecture, and give an algorithm for calculating them.