BIOINFORMATICS

Vol. 24 ISMB 2008, pages i322–i329
doi:10.1093/bioinformatics/btn159

GenoQuery: a new querying module for functional annotation in a genomic warehouse

Frédéric Lemoine[1,2], Bernard Labedan[1] and Christine Froidevaux[2,*]

[1]Institut de Génétique et Microbiologie and [2]Laboratoire de Recherche en Informatique, Université Paris-Sud XI, 91405 Orsay Cedex, France

ABSTRACT

Motivation: We have to cope with both a deluge of new genome sequences and a huge amount of data produced by high-throughput approaches used to exploit these genomic features. Crossing and comparing such heterogeneous and disparate data will help improving functional annotation of genomes. This requires designing elaborate integration systems such as warehouses for storing and querying these data.

Results: We have designed a relational genomic warehouse with an original multi-layer architecture made of a databases layer and an entities layer. We describe a new querying module, GenoQuery, which is based on this architecture. We use the entities layer to define mixed queries. These mixed queries allow searching for instances of biological entities and their properties in the different databases, without specifying in which database they should be found. Accordingly, we further introduce the central notion of alternative queries. Such queries have the same meaning as the original mixed queries, while exploiting complementarities yielded by the various integrated databases of the warehouse.

We explain how GenoQuery computes all the alternative queries of a given mixed query. We illustrate how useful this querying module is by means of a thorough example.

Availability: http://www.lri.fr/∼lemoine/GenoQuery/

Contact: chris@lri.fr, lemoine@lri.fr

1 INTRODUCTION

With the entry in the genomics era, the advances of genome sequencing (700 published microbial genomes in April 2008) and the increasingly massive use of high-throughput approaches have produced a huge amount of data. We urgently need management systems in order to store and query biological information. In particular, this is critical in functional annotation of genomes (Ouzounis and Karp, 2002). Indeed, it is now easy to get the complete sequence of a prokaryotic genome and to detect all its genes (structural annotation). On the contrary, the functional annotation of its putative coding sequences is increasingly difficult, especially for organisms never studied by experimental biology. For instance, it is a challenging goal to reconstruct the complete metabolism of a species using uniquely its genomic sequence (Karp *et al.*, 2005). However, such a reconstruction is crucial to disclose potential drug targets in the case of pathogenic microbes or to exploit novel pathways in the case of species that are potentially useful for bioremediation or bioenergy needs. Such a functional annotation step requires combining various pieces of knowledge and correctly handling heterogeneous data stored in various databases that are either local sources or distributed sources on the web.

Several approaches have been proposed in the field of databases integration. Portals like SRS (Etzold *et al.*, 1996) and EXPASY (Gasteiger *et al.*, 2003) allow users to query easily multiple sources by means of a single website. Path-based systems such as Biomediator (Cadag *et al.*, 2007), Bionavigation and BioGuide (Cohen-Boulakia *et al.*, 2006) are based on cross-references between data sources, in order to navigate from one source to another, and provide the user with ranked paths selected according to her/his preferences. Such systems allow to query easily data sources by means of a more or less expressive language. Mediator systems such as Tambis (Stevens *et al.*, 2000) or K2/Kleisli (Davidson *et al.*, 2001) are designed to query the distant sources through a virtual mediated schema. The query formulated on the mediated schema is further translated into queries over the schemata of the sources and the answers are processed locally. In these non-materialized integration systems, data are not stored locally, but remain in the distant sources. Thus, mediator systems provide up-to-date data, but do not permit complex computations. In particular, data-mining techniques are very difficult to apply on such systems. In peer-to-peer systems like Piazza (Halevy *et al.*, 2003) or Orchestra (Green *et al.*, 2007), data are stored in multiple distant 'peers', which communicate with a few other peers and queries can be formulated over each peer. Finally, in fully materialized systems such as GUS (Davidson *et al.*, 2001), BioWarehouse (Lee *et al.*, 2006), Biozon (Birkland and Yona, 2006), GEDAW (Guérin *et al.*, 2005), Biomart (Durinck *et al.*, 2005) or Columba (Trissl *et al.*, 2005), data are integrated within a warehouse based on a locally constructed schema. While updating warehouses is a challenging task, performing complex computations is easier.

We are expecting a huge amount of prokaryotic genomic sequences (e.g. more than 2.10^6 proteins for the 700 genomes presently available) and many derived heterogeneous data, that must be integrated and on which we need to perform data-mining techniques and other complex computations. Therefore, we decided to gather these data within the Microbiogenomics data warehouse, with an *ad hoc* architecture. In this way, the integrated databases may provide complementary, different or even divergent points of view on the same data, by adding supplementary useful information as discussed subsequently.

Accordingly, we are developing Microbiogenomics to fulfill two main objectives: improving functional annotation or reannotation of microbial genomes and studying molecular evolution of genes and genomes of micro-organisms. To perform these tasks, Microbiogenomics contains a large variety of primary

[*]To whom correspondence should be addressed.

(sequence-based) and secondary (homology-based) data. Primary data (genomes, protein sequences, metabolic pathways, enzymes and bibliographic data) come from different and complementary public sources. We have generated secondary data using various pipelines (Bryson *et al.*, 2006; Le Bouder-Langevin *et al.*, 2002; Lemoine *et al.*, 2007) that have been designed to find out homology relationships between proteins as the result of sequence alignment programs such as Blast (Altschul *et al.*, 1990) and Darwin AllAll (Gonnet *et al.*, 2000).

This article describes GenoQuery, an innovative approach to query data warehouses such as Microbiogenomics. This querying method is based on a concerted use of a few of the concepts and tools developed by the already described approaches. Indeed, GenoQuery uses the notion of *view* from mediator and peer-to-peer systems applied to a local integration of data sources as in warehouses approaches, and uses the notion of *path* and *graph of entities* as in browsing approaches. Moreover, our approach offers a new notion of alternative queries permitting to exploit both matches and conflicts between data sources.

GenoQuery is based on a dedicated two-layers architecture that is outlined on Figure 1. The first layer is made of the different relational databases present in the Microbiogenomics warehouse. The second layer is made of entities organized in two levels. Since biological entities can be considered at two degrees of abstraction, we distinguish the abstract entities level and the concrete entities level. Abstract entities are the biological entities present in the content of the databases, that we consider as relevant, whereas concrete entities are the manifestations of abstract entities in the sources. Both levels are represented by a specific elementary graph, as detailed subsequently, and the graphs of abstract and concrete entities are further combined in the so-called *general graph of entities*. This general graph of entities is further used to build the mixed query graphs. Finally, we propose a first prototype with a friendly user interface allowing complex queries that lead to alternative answers rich in biological information. A prototype of GenoQuery is available at http://www.lri.fr/∼lemoine/GenoQuery/.

2 DATABASES LAYER

2.1 Biological data and databases

The biological databases we integrated into the Microbiogenomics warehouse are major public data sources: RefSeq (Pruitt *et al.*, 2007), Genome Reviews (Sterk *et al.*, 2006), Prose (http://genome.jouy.inra.fr/prose) a relational version of UniProt (Bairoch *et al.*, 2005) and Pareo (http://genome.jouy.inra.fr/pareo/) a relational version of KEGG (Kanehisa *et al.*, 2006). We have added local sources developed by the different labs participating in the Microbiogenomics project: elements of AGMIAL (Bryson *et al.*, 2006), Genopage (Cohen-Boulakia *et al.*, 2002), Syntebase (Lemoine F. *et al.*, manuscript in preparation), Orenza (Lespinet and Labedan, 2006).

These primary and secondary data were then used to generate tertiary data such as conservation of gene order in genomes, orthologous relationships between proteins, phylogenetic relationships using a pipeline recently described (Lemoine *et al.*, 2007).

2.2 Warehouse schema based on independent databases

Microbiogenomics is built in relational format and is implemented under PostgreSQL.[1] We chose the relational format because it is well known and allows expressive queries. Each database remains independent from the others in terms of schema and data. Thus, the data are not fully reconciled, allowing to consider the different points of view given by the various databases. The advantages of such a global schema are the following: (1) each database can be updated independently from each other, (2) addition of new sources is possible at any time, allowing to personalize the warehouse according to specific and/or new needs by adding a locally developed new data source and (3) the complementarities and divergences about some biological data are kept because they potentially add supplementary useful information.

Microbiogenomics is focusing mainly on protein-related data. However, public databases use different identifiers to describe the same protein. To achieve semantic integration of the data, we built associations between the relational tables of the sources by means of an independent links table that is updated whenever a database is updated or a new database is added.

3 ENTITIES LAYER

3.1 The abstract entities level

The *abstract entities* level is a conceptual model that we have built using a bottom-up approach. We used the concepts present in the sources and the biological associations linking these concepts. This level is represented by a directed graph, the *graph of abstract entities* (Figs 1 and 2).

Vertices are divided into two parts: abstract entities themselves such as `Protein` or `MetabolicPathway` (circles in the top panel of Fig. 2), and properties of these abstract entities (data not shown), such as `name`, `sequence` for the entity `Protein`.

Edges describe three types of links: (1) biological links between two abstract entities, such as `Transcript` - *`CodesFor`* - `Protein` (black lines between abstract entities in Fig. 2), (2) links between the abstract entities and their properties such as `Protein` - *`has For Property`* - `sequence`, (3) hierarchical relations between abstract entities, the *`isa`* links represented as black arrows in Figure 2. For instance, Figure 1 shows an *`isa`* link between C_3 and C_4 (gray arrow), and Figure 2 shows an *`isa`* link between `Enzyme` and `Protein`.

3.2 The concrete entities level

Below the conceptual model, we built the level made of the *concrete entities* present in each database, in a way similar to Cohen-Boulakia *et al.* (2007). Each concrete entity is a view of some abstract entity in a given database, which provides instances of it. For example, `GenopageProtein` is the concrete entity mapped with the abstract entity `Protein`, for which instances can be found in the database Genopage. We define the properties of a concrete entity as the set of properties of its corresponding abstract entity, and possibly other properties specific to the entity in its underlying database. With each concrete entity is associated a query in its underlying database, which expresses how to retrieve instances of the concrete entity in the database (e.g. $v1$ for $C1_1$ in Fig. 1).

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

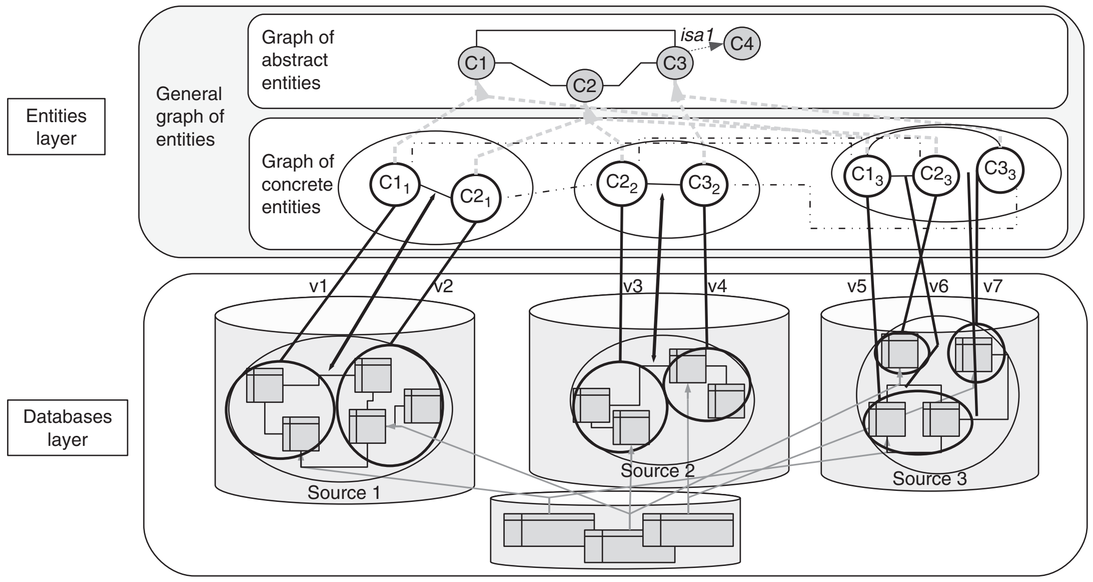

Fig. 1. The multi-layer warehouse architecture.

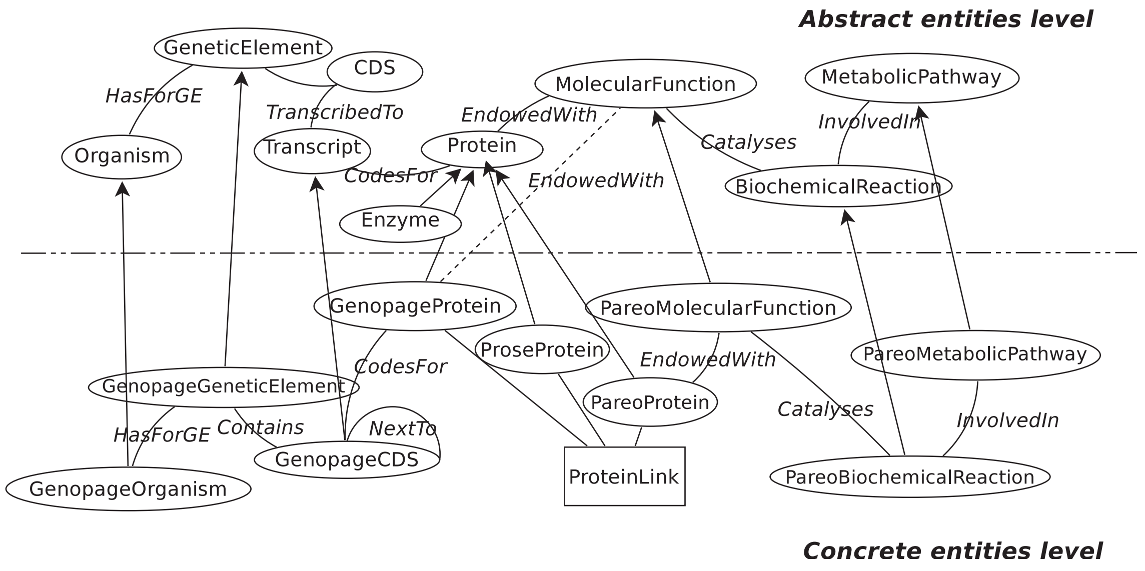

Fig. 2. Part of the general graph of entities. Arrows link concrete entities to their corresponding abstract entity. The dotted line between `GenopageProtein` and `MolecularFunction` is an example of an inherited link.

The concrete entities level is represented by a directed graph, the *graph of concrete entities* (Figs 1 and 2). Its vertices are of three kinds: (1) concrete entities such as `ProseProtein` (circles in the bottom panel of Fig. 2), (2) properties of these entities such as `length` or `amino acid sequence` (data not shown) and (3) link entities, such as `ProteinLink`, which represent the links tables.

Edges linking the vertices are of different kinds. The biological links between two concrete entities (black lines in the bottom panel of Fig. 2) are mapped to a path (possibly of length 1) in the abstract entities level and a join (either a foreign key or links table) in the databases layer. For example, the link `GenopageCDS` - *`CodesFor`* - `GenopageProtein` is mapped to the path of length 2 `CDS` - *`Transcribed To`* - `Transcript` - *`CodesFor`* - `Protein` in the graph of abstract entities. A second kind of link is the association between the concrete entities and their properties. Finally, there are links between a concrete entity and a `link entity`. This last kind of link corresponds to the entities that are present in different sources and for which the links table is involved (lines between squares and circles in Fig. 2). For instance, the links table represented by the link entity ProteinLink (square in Fig. 2) associates GenopageProtein, ProseProtein and PareoProtein. Such links have no mapping in the abstract entities level, but match with the links table in the databases layer (Fig. 1).

3.3 The general graph of entities

The *general graph of entities* that is the basis of our querying module GenoQuery (see subsequently), is a directed graph, where the vertices are the union of the vertices of the graph of abstract entities and the graph of concrete entities (Fig. 2). Its edges are the edges of both graphs, the *isa* links between concrete entities and their corresponding abstract entities, and some additional links between them that are needed for constructing mixed queries (Fig. 2). If some abstract entity is linked to another abstract entity by a biological link, then all the corresponding concrete entities will also be linked to the other abstract entity with the same kind of link. These links are called *inherited biological links*. For example, the association Protein-*EndowedWith*-Function creates the inherited biological link ProseProtein-*EndowedWith*-Function.

4 THE QUERYING MODULE GENOQUERY

We designed a dedicated querying module that uses mechanisms of query reformulation and exploits the architecture of the data warehouse. Presently, GenoQuery queries databases that are mirrored locally. First, *mixed queries* are expressed in terms of abstract as well as concrete entities. Second, the querying module translates these mixed queries into equivalent queries, only expressed in terms of concrete entities (*alternative queries*).

4.1 Defining mixed and alternative queries

4.1.1 Mixed queries We call mixed query a query that searches for both abstract and concrete entities, properties of these entities and relationships between them. The possibility to formulate queries that involve abstract entities, concrete entities or both confers a good expressivity power on the query language of our system. On the one hand, one can feel free from having to specify the data sources of the warehouse that are relevant to a query, using only abstract entities. We call such a query a *high-level* (or *transparent*) query. On the other hand, using concrete entities allows to specify for some given entities the source in which their instances must be searched for. If the query has only concrete entities, we will refer to such a query as a *low-level query*.

A *mixed query graph* is a directed graph where vertices are abstract or concrete entities chosen in the general graph, and edges are associations between these entities, taken from the general graph of entities. It is worth noticing that a given entity can have more than one occurrence in a mixed query.

4.1.2 Alternative queries Every abstract entity in a mixed query has to be first translated into concrete entities to be answered. As the user does not have to specify in which source instances of this abstract entity has to be searched for, the querying module will search for all the sources of the warehouse that can provide the corresponding concrete entities using the *isa* links between concrete and abstract entities (Fig. 1). Moreover, we will exploit a nice feature of the global schema we have designed for the warehouse. Since the reconciliation of the data is not mandatory, it is possible to examine the different points of view of the sources, without forcing the reconciliation of all the data. Indeed, alternative queries could give complementary or divergent data that are potentially interesting for improving functional annotation.

We will show subsequently that links between concrete entities in some databases can correspond to more elaborate paths between concrete entities in other sources, and thus involve other entities not present in the original mixed query.

One mixed query can therefore lead to several alternative queries, which are low-level queries, each one corresponding to the same high-level query as the initial mixed query. This mapping is established using the *general graph of entities*. The reader is referred to *Technical Report* (submitted for publication) for a formal definition in terms of graphs.

Let us give an example of mixed query that we will work with in the following (Fig. 3): 'What are the pairs of proteins that are encoded by neighboring genes in Genopage and which participate in the same metabolic pathway in *Escherichia coli* from Genopage'. Figure 3 shows that the corresponding graph is complex and necessitates multiple sources. GenopageOrganism is the concrete entity for *E.coli* from Genopage, MetabolicPathway is the abstract entity for metabolic pathway, GenopageCDS are the concrete entities for neighboring genes and GenopageProtein is the concrete entity for the proteins encoded by the genes. Note that the last two entities have two occurrences in the mixed query graph. Moreover, the gray entities correspond to the ones returned by the query. Additional entities are present in the mixed query graph (MolecularFunction,BiochemicalReaction and GenopageGeneticElement) although they are not explicitly mentioned in the natural language query. They come from the general graph and are necessary to associate the entities explicitly searched for.

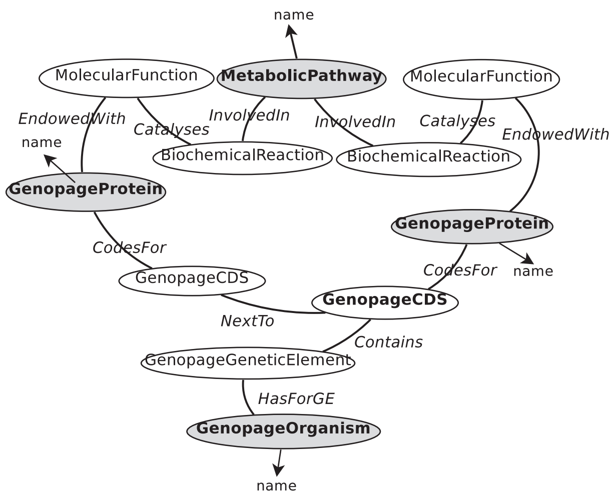

Fig. 3. Graphical representation of the mixed query: 'What are the pairs of proteins encoded by neighboring genes in Genopage which participate in the same metabolic pathway in *Escherichia coli* from Genopage.' The following properties are represented: 'name' for metabolic pathway, 'name' for GenopageProtein and 'name' for GenopageOrganism.

4.2 Building alternative queries

The mixed query graph representing the initial user's query is processed through four steps in order to get the low-level query graphs leading to the alternative queries.

4.2.1 Splitting up the mixed query graph into elementary paths In the mixed query one entity searched for by the user can be linked to more than two entities, so that the mixed query graph can become rather complex. From the initial query we build elementary queries where each entity is only linked to one or two other entities. Such elementary queries are represented by elementary paths in the mixed query graph, where each vertex has at the most two adjacent vertices. To delineate elementary paths, we distinguish in the initial query some entities that play a special role. A vertex representing an entity in the graph is considered as a *breakpoint* if it verifies at least one of the following conditions: (1) it has less (or more) than two links with other entities nodes (they are start points or end points of the query, or they are central in the query being linked to many other entities), (2) it is linked to at least one property.

We then compute all the linear paths beginning and ending with breakpoints in the initial query. In Figure 3, the five breakpoints are in bold: **`GenopageCDS`** because it is linked to more than two other entities, and the shaded entities that are linked to a property. We extract five elementary paths (EPs) from the initial query graph:

- EP 1 (right part of the graph): **Genopage Protein**— *Endowed With*—Molecular Function—*Catalyzes*—Biochemical Reaction— *InvolvedIn*—**Metabolic Pathway**
- EP 2 (left part of the graph): **Genopage Protein**—*Endowed With*—Molecular Function—*Catalyzes*—Biochemical Reaction—*InvolvedIn*—**Metabolic Pathway**
- EP 3: **Genopage CDS**—*NextTo*—GenopageCDS—*Codes For*—**Genopage Protein**
- EP 4: **Genopage CDS**—*CodesFor*—Genopage Protein
- EP 5: Genopage Organism—*Has For GE*—Genopage Genetic Element—*Contains*—**GenopageCDS** (The biological associations are in italic, and the entities in normal font).

4.2.2 Computing intermediate queries Alternative paths are high-level paths (containing only abstract entities), that take into account the mapping between the biological links in the concrete entities level and the paths in the abstract entities level defined in Section 3.2. They are defined as alternative ways of linking abstract entities, while keeping the meaning of the initial query.

Alternative paths can be computed for each elementary path in two phases (Fig. 4): (1) abstraction of the elementary paths and (2) computation of all the corresponding alternative paths. Since the mapping gives a semantic correspondence between an edge in the concrete entities level and a path in the abstract entities level, a path in the general graph has equivalent reformulations. In theory, this step can give rise to a very large number of alternative paths. In practice, however, in the context of Microbiogenomics, the entities graph contains a few entities (about 50). Therefore, the number of different alternative paths generated is low.

For instance, in the elementary path EP4, **`GenopageCDS`** is mapped with the abstract entity `CDS` and **`GenopageProtein`** with `Protein`. The abstract path corresponding to EP4 is: 'CDS—*Transcribed To*—Transcript—*CodesFor*—Protein'. From this

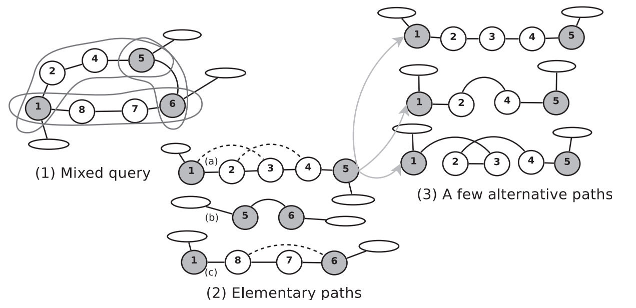

Fig. 4. Splitting up an initial query into a set of elementary paths. For each elementary path we compute all the semantically equivalent alternative paths, using the mappings. The ovals represent the properties, attached to the entities represented by circles. The shaded circles represent the 'breakpoints'. The lines represent biological links in the query, and the dotted lines represent the shortcuts defined in the mapping.

resulting abstract path, we compute all the alternative paths. Here is an example of a resulting alternative path: 'CDS—*Transcribed To a transcript which Codes For*—Protein'. We can see that there is a new link in this alternative path that is a shortcut standing for the first two links in the abstract path.

The set of intermediate queries is made of all the queries that are combinations of the alternative paths computed above. The queries we collected never involve many entities (at the most 15). Therefore, whereas the number of intermediate queries could be very high in theory, this is not the case in practice.

4.2.3 Building querying graphs For each intermediate query, we build a **querying graph** where each abstract entity is linked to all its corresponding concrete entities (Fig. 5). Note that these links are not shown in Figure 5, but the entities linked together belong to the same box. For instance, `MolecularFunction` is linked to `PareoMolecularFunction` and `ProseFunction`. We attach to each concrete entity of the 'querying graph' the properties of the abstract entity corresponding in the intermediate query (data not shown in the figure). Figure 5 further shows how we add three kinds of associations to this querying graph:

(1) Edges between abstract entities of the intermediate query (e.g. the thick black line in Figure 5 linking the boxes Protein and CDS)
(2) Edges between the corresponding concrete entities (e.g. the dotted line linking `ProseCDS` and `ProseProtein`)
(3) Links tables that associate concrete entities of different sources (black circle in the box `Protein` linking `ProseProtein`, `GenopageProtein` and `PareoProtein`).

Intuitively, a querying graph is an undirected graph which regroups all the entities (abstract and concrete) and the links (between concrete or abstract entities) necessary to compute all the alternative queries from an intermediate query. A querying graph is valid (i.e. there is at least one alternative query corresponding to the intermediate query) if (1) it is connected, (2) for each abstract entity there exists a concrete entity and (3) each association of the intermediate query corresponds to at least one association between two concrete entities.

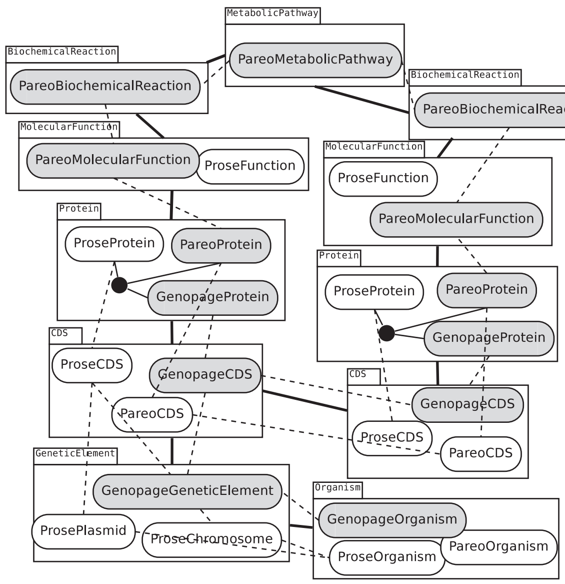

Fig. 5. Example of a querying graph constructed from an intermediate query. The thick black lines represent biological associations between abstract entities of the intermediate query. The dotted lines represent biological associations between concrete entities. Finally, the black circles represent the links tables that associate proteins of different sources (thin black lines). For the sake of readability neither the entities properties nor the association names are represented.

4.2.4 Getting alternative queries The last step is the computation of the set of alternative queries. After traversing the querying graph, all the low-level queries are extracted.

First, we perform a depth first search of the querying graph, and for each edge linking two abstract entities in the querying graph (thick black lines in Fig. 5), we calculate all the ways of instantiating it by associations between concrete entities (dotted lines in Fig. 5) together with their properties. Then, we add associations between concrete entities that are not in the same source, (thin black lines in Fig. 5) in order to link the concrete entities between the sources (e.g. `Proseprotein` and `GenopageProtein`). Eventually, we obtain a set of graphs that involve only concrete entities, and constitute the alternative queries.

The shaded concrete entities together with the dotted lines linking them in Figure 5 constitute the graph of such an alternative query for the running mixed query. This alternative query interrogates the databases Genopage for entities `Protein`, `CDS`, `Organism` and `Genetic Element`, and Pareo for entities `MolecularFunction`, `Biochemicalreaction` and `MetabolicPathway`, respectively.

5 IMPLEMENTATION

5.1 General graph of entities

The Resource Description Framework, RDF (Klyne, 2004), is a simple and expressive language for representing information in the web, and is especially well suited to graph representation. We have thus implemented the general graph of entities in RDF (Klyne, 2004), with the use of RDFs, a RDF Vocabulary Description Language that allows to build concepts and relationships between them. RDFs is able to model `isa` relationships between entities, as well as associations between two entities, or between an entity and its properties. To each abstract or concrete entity corresponds one ‘RDFs class’ and to each property corresponds a ‘literal’. To each `isa` relationship corresponds a RDF statement which indicates that the first class is a subclass of the second one; to each edge associating an entity (abstract or concrete) with its property corresponds an ‘RDFs property that’ has a literal as range; finally to each other edge corresponds a RDFs property that makes the link between two classes.

The mappings between the graph of concrete entities and the graph of abstract entities are expressed in XML, with a specific XML schema describing how the document must be formed.

5.2 Mixed query graphs

There are several ways of expressing mixed queries. First, one can use the graphical interface prototype developed in the Java language (Fig. 6). This interface allows to construct a mixed query graph, guided by the general graph of entities. A second possibility is to write the query using a subset of the SparQL query language (Prud’hommeaux and Seaborne, 2007) over the general graph of entities itself represented in RDFs. Then, the SparQL query can be imported in the interface (click on File/import SparQL query) to be processed by the GenoQuery module.

Using a formal query language allows to save and compare queries, which can be interesting for repeated uses of the warehouse. We chose SparQL because it is well adapted for the definition of queries on RDFs documents, its syntax is relatively simple to learn, and some frameworks currently exist for parsing and handling SparQL queries such as Jena.[2]

5.3 Alternative query graphs

Once the initial query is constructed, one can compute the alternative queries, by clicking on ‘Query/Compute alternative queries’ menu. The alternative queries are displayed as a graph in the right panel. The user can navigate through the alternative queries by clicking on the black arrows. By clicking on the ‘execute’ button, the current alternative query is converted to a SQL query that can be run on the relational warehouse.

6 USING THE QUERYING MODULE

We illustrate the usefulness of GenoQuery by means of the following mixed query. Assume that we are interested in the genes of the bacterium *Mycobacterium tuberculosis* that have been annotated as encoding an enzyme registered in Pareo (KEGG) with a known activity described by an EC number. The abstract entities are `CDS` (linked to the property `name`) and `Organism` (linked to the property `speciesname`), whereas the concrete entities are `PareoProtein` and `PareoBiochemicalReaction` (linked to the property `EC_number`). Using our querying module we recover data from the last updated relational versions of UniProt (Prose) and KEGG (Pareo).

We mention two alternative queries. The first one is ‘What are the genes of *M.tuberculosis* in Pareo that have been annotated as encoding an enzyme in Prose (UniProt) and that display an

[2]http://jena.sourceforge.net/

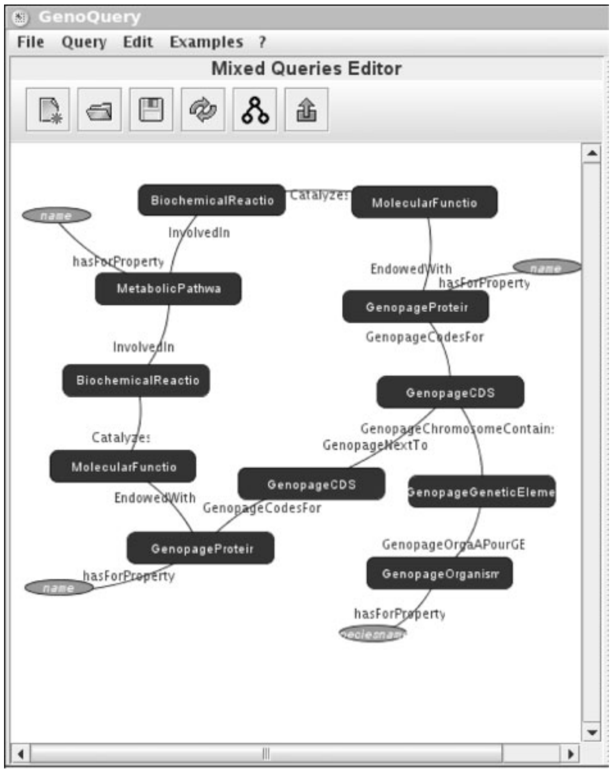

Fig. 6. Snapshot of the prototype developed in the Java language.

EC number in Prose?' The second one is 'What are the genes of *M.tuberculosis* in Pareo that have been annotated as encoding an enzyme in Pareo and that display an EC number in Pareo?'

Results of both alternative queries are presented to the user in a tabular form, with two columns. The first one displays the name of the gene, and the second one the EC number.

We found 1298 gene products associated with an EC number in UniProt (Prose) and only 1089 in KEGG (Pareo). Among them, 102 EC numbers associated with the same gene are different, as detailed subsequently.

We note increasing kinds of dissimilarities about the molecular function. For 57 enzymes, UniProt and KEGG disagree about the last digit, i.e. the precise definition of the biochemical reaction. For instance, the gene Rv1086 (O53434, ZFPP_MYCTU) is annotated as encoding a short-chain Z-isoprenyl diphosphate synthetase (EC 2.5.1.68) corresponding to the first step of decaprenyl diphosphate biosynthesis in UniProt. In KEGG, Rv1086 is viewed as encoding a di-*trans*, poly-*cis*-decaprenyl*cis*transferase (EC 2.5.1.31), one of the step of terpenoid biosynthesis.

The 45 remaining dissimilar EC numbers differ at least at the level of their first digit, i.e. they correspond to very different molecular functions. Among them, we get eight fully annotated EC numbers that differ in their four digits. For instance, Rv1248 (O50463, KGD_MYCTU) is encoding a 2-oxoglutarate decarboxylase (EC 4.1.1.71) according to UniProt. In KEGG, Rv1248 is viewed as encoding an α-ketoglutarate decarboxylase (EC 1.2.4.2).

Thus, GenoQuery allows finding easily major conflicts or minor discrepancies between main public databases.

Ongoing work includes automating the last steps of the results, the analysis of which is currently done manually. This is especially important because we are expecting to see an amplification of these inconsistencies as both automatic and manual annotation processes are becoming increasingly complex.

7 DISCUSSION

We have designed GenoQuery, a module for querying a relational genomic warehouse. GenoQuery is based on an original multi-layer architecture of the warehouse, made of two layers, the entities layer and the databases layer. We have further distinguished two levels in the entities layer: abstract entities that are extracted from the databases and concrete entities that are views of theses abstract entities in the databases. These two levels are linked with `isa` relationships that map concrete entities to abstract entities, and inherited biological links. Each concrete entity is associated with a query in its corresponding database, which allows retrieving its instances in the database. The global schema of the data warehouse is flexible enough to allow easy addition of a new source or update existing ones, as the various data coming from the different databases have been merely gathered in the warehouse, and since we did not remove redundancies or potential conflicts. This kind of integrating schema is close to the approach of Trissl *et al.* (2005), where the data from different sources are never mixed into a single table and each data source is considered as an essentially independent dimension around the central fact (protein structure). Likewise, the content of our warehouse is centered on the fundamental entity Protein. In our approach, semantic integration is achieved thanks to a links table that connects the id's of the same proteins on the basis of their amino acid sequence identity and their gene position. This is crucial to take into account the possible polymorphisms carried by the different sources. Accordingly, one can take advantage of complementary points of view on the same data.

The maintenance of this multi-layer architecture depends on two kinds of modifications. In the case of addition (or deletion) of a source, we have to add/remove concrete entities and links in the concrete entities layer. Similarly, a revision of the relational schema of a source requires the adjustment of the concrete entities layer. Note that updates of the sources in terms of instances do not affect the diverse layers. It is worth noting that this maintenance additional cost is mandatory in order to keep the different points of view given by the sources. Indeed, discussions are opened (Pennisi, 2008) to propose systems that permit to confront points of views.

We have used this architecture to define mixed queries in terms of both abstract and concrete entities, allowing to search for instances of biological entities and their properties in the databases, without requiring specifying in which database they have to be found. In order to exploit the complementarities of the points of view displayed on the warehouse, we have introduced the central notion of this article, alternative queries. Indeed, GenoQuery yields as alternative queries, concrete queries that have the same meaning as the original mixed query, but that may consider the entities in other databases than those specified in the mixed query (if any). We have further shown how to calculate all the alternative queries of a given mixed query and have described the prototype available on the web. The usefulness of the approach has been demonstrated through a thorough example. GenoQuery helps to settle varying interpretations by pinpointing the dissensions between sources.

One original aspect of GenoQuery is the capacity to reformulate an initial query in alternative queries. The concept of query reformulation has been already used in other works but with a different meaning. For example, in the approach of Lowden and Robinson (2004), reformulation of an initial query leads to minimal cost that provide exactly the same set of answers and is merely a semantic query optimization process. In the work of Necib and Freytag (2004), query reformulation exploits the knowledge of an ontology to build more meaningful query answers. In these works, reformulation is performed in the context of a single database. On the contrary, our approach fully exploits the fact that the data come from various sources that give their own point of view in the same warehouse. Accordingly, our reformulation process keeps unchanged the meaning of the initial query while proposing all the ways of consulting the different databases for retrieving the instances of the entities searched for.

Our notion of reformulation can be applied to other integration systems. Actually, GenoQuery is generic enough to be used straightforwardly in other relational genomic data warehouses where reconciliation of data is not fully achieved (redundant and divergent pieces of knowledge are allowed). More precisely, the warehouse architecture should make available several points of view on the data, by keeping all the data from the different databases and mentioning in the warehouse their provenance (from which database they come from), as done in Biowarehouse (Lee *et al.*, 2006), Columba (Trissl *et al.*, 2005) and Ensmart (Kasprzyk *et al.*, 2004). To plug our querying module on such warehouses, it would be necessary to design the entities layer on the top of the warehouse architecture, together with the mapping to the databases layer. Then, our module offers a user-friendly interface for defining mixed queries and allows calculating alternative queries and their translation into SQL queries on the databases.

The notion of alternative queries we have introduced seems to be especially promising to discover new unexpected knowledge from data warehouses such as Microbiogenomics.

ACKNOWLEDGEMENTS

We would like to thank all the participants of the Microbiogenomics project, in particular J.F. Gibrat and his group at MIG, for their help in the design of the global schema of the warehouse. We are also grateful to Anne Morgat (SIB), Eric Coissac (LEC) and Sarah Cohen Boulakia (LRI) for fruitful discussions.

Funding: This work was funded by the CNRS (UMR 8621, UMR 8623), the PPF 'Bioinformatique et BioMathematique' of the Université Paris-Sud and the Agence Nationale de la Recherche (ANR-05-MMSA-0009 MDMS_NV_10). F.L. is a PhD student supported by the French Ministry of Research.

Conflict of Interest: none declared.

REFERENCES

Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Bairoch,A. *et al.* (2005) The universal protein resource (UniProt). *Nucleic Acids Res.*, **33**, D154–D159.

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

Bryson,K. *et al.* (2006) AGMIAL: implementing an annotation strategy for prokaryote genomes as a distributed system. *Nucleic Acids Res.*, **34**, 3533–3545.

Cadag,E. *et al.* (2007) Biomediator data integration and inference for functional annotation of anonymous sequences. In *Pacific Symposium on Biocomputing*, pp. 343–354.

Cohen-Boulakia,S. *et al.* (2002) Genopage : a database of all protein modules encoded by completely sequenced genomes. In *Actes de JOBIM2002, Journées Ouvertes, Biologie, Informatique et Mathématiques*, pp. 187–193.

Cohen-Boulakia,S. *et al.* (2006) Path-based systems to guide scientists in the maze of biological data sources. *J. Bioinform. Comput. Biol.*, **4**, 1069–1095.

Cohen-Boulakia,S., *et al.* (2007) BioGuideSRS: querying multiple sources with a user-centric perspective. *Bioinformatics*, **23**, 1301–133.

Davidson,S.B. *et al.* (2001) K2/kleisli and gus: experiments in integrated access to genomic data sources. *IBM Syst. J.*, **40**, 512–531.

Durinck,S. *et al.* (2005) Biomart and bioconductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics*, **21**, 3439–3440.

Etzold,T. *et al.* (1996) SRS: information retrieval system for molecular biology data banks. *Methods Enzymol.*, **266**, 114–128.

Gasteiger,E. *et al.* (2003) Expasy: the proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Res.*, **31**, 3784–3788.

Gonnet,G. *et al.* (2000) Darwin v. 2.0: an interpreted computer language for the biosciences. *Bioinformatics*, **16**, 101–103.

Green,T.J. *et al.* (2007) Orchestra: facilitating collaborative data sharing. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (*SIGMOD'07*). ACM, New York, USA, pp. 1131–1133.

Guérin,E. *et al.* (2005) Integrating and warehousing liver gene expression data and related biomedical resources in GEDAW. In *Second International Workshop on Data Integration in the Life Sciences (DILS)*, Vol. 3615/2005 of *LNCS*, San Diego, CA, US, pp. 158–174.

Halevy,A.Y. *et al.* (2003) Schema mediation in peer data management systems. In Dayal,U. Ramamritham,K. and Vijayaraman,T.M. editiors, *Proceedings of the 19th International Conference on Data Engineering*, IEEE Computer Society, pp. 505–.

Kanehisa,M. *et al.* (2006) From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Res.*, **34**, D354–D357.

Karp,P.D. *et al.* (2005) Expansion of the BioCyc collection of pathway/genome databases to 160 genomes. *Nucleic Acids Res.*, **33**, 6083–6089.

Kasprzyk,A. *et al.* (2004) Ensmart: a generic system for fast and flexible access to biological data. *Genome Res.*, **14**, 160–169.

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

Le Bouder-Langevin,S. *et al.* (2002) A strategy to retrieve the whole set of protein modules in microbial proteomes. *Genome Res.*, **12**, 1961–1973.

Lee,T. *et al.* (2006) BioWarehouse: a bioinformatics database warehouse toolkit. *BMC Bioinformatics*, **7**, 170.

Lemoine,F. *et al.* (2007) Assessing the evolutionary rate of positional orthologous genes in prokaryotes using synteny data. *BMC Evol. Biol.*, **7**, 237.

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

Lowden,B.G.T. and Robinson,J. (2004) Improved data retrieval using semantic transformation. In Galindo,F. and Takizawa,M. (eds) *DEXA*, *Lecture Notes in Computer Science*, Vol. 3180. Springer, Zaragoza, Spain, pp. 391–400.

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

Ouzounis,C.A. and Karp,P.D. (2002) The past, present and future of genome-wide re-annotation. *Genome Biol.*, **3**, c2001.1–c2001.6.

Pennisi,E. (2008) DNA data. Proposal to 'Wikify' GenBank meets stiff resistance. *Science*, **319**, 1598–1599.

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

Pruitt,K. D. *et al.* (2007) NCBI Reference Sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic Acids Res.*, **35**, D61–D65.

Sterk,P. *et al.* (2006) Genome reviews: standardizing content and representation of information about complete genomes. *OMICS*, **10**, 114–118.

Stevens,R. *et al.* (2000) TAMBIS: transparent access to multiple bioinformatics information sources. *Bioinformatics*, **16**, 184–185.

Trissl,S. *et al.* (2005) Columba: an integrated database of proteins, structures, and annotations. *BMC Bioinformatics*, **6**, 81.