Mapping Abstract Queries to Big Data Web Resources for On-the-fly Data Integration and Information Retrieval

Hasan M. Jamil
Department of Computer Science
University of Idaho
Moscow, Idaho, USA
jamil@uidaho.edu

***Abstract*—The emergence of technologies such as XML, web services and cloud computing have helped, the proliferation of databases and their diversity pose serious barriers to meaningful information extraction from these "big databases". Research in intention recognition has also progressed substantially, yet very little has been done to recognize query intents to search, select, map and extract responses from such enormous pools of candidate databases. Query mapping becomes truly complicated particularly in scientific databases where tools and functions are needed to interpret the database contents, semantics of which are usually hidden inside the functions. In this paper, we present a declarative meta-language, called *BioVis*, using which biologists potentially are able to express their "intentional queries" with the expectation that a mapping function μ is able to accurately understand the meaning of the queries and map them to the underlying resources appropriately. We show that such a function is technically feasible if we can design a schema mapping function that can tailor itself according to a knowledgebase and recognize entities in schema graphs. We offer this idea as a possible research problem for the community to address.**

I. Introduction

Recent interest in NoSQL [1] and keyword [2] queries emerges at a time when biologists are struggling to make sense of the huge amount of disparate data being generated across the globe and made available for harvesting. While debate persists [3] on whether such approaches will ultimately help, researchers have seized these opportunities to demonstrate their potential [4]. These attempts are aimed at easing access to information by "naive users" without the sophistication to form complex structured queries, or having to deal with the heterogeneities of the underlying information sources, and are not truly conducive to forming effective computational pipelines in the life sciences domain. In life sciences, querying goes beyond just being able to access information from a repository and involves stitching together a complex web of resources and tools in a meaningful way, the feasibility of such approaches still remains as an open question.

In this paper, our goal is two fold. First, we propose a querying architecture, called *BioVis*, in which biologists are able to think entirely in terms of their application semantics and express their information needs using only universally accepted domain terms and concepts. Once an application is described, the BioVis system is expected to convert the application into a resource specific query pipeline in BioFlow [5], the host language of LifeDB [6], with minimal user participation. Second, the mapping process we advocate for query mapping, fundamentally relies on a more "smart" schema matching algorithm based on sophisticated graph alignment techniques. In our vision, this graph matching technique is capable of recognizing entity equivalence in the query and database schema graphs to remove ambiguities. This is a challenge that we believe, if resolved, will enable large scale data integration in the age of big data science. In our mind, big data also means an extremely large number of moderately big databases that are distributed across the globe and that are impossible for the user to search, find, select and query. Systems must be developed to map a conceptual query to be automatically mapped to appropriate databases to return an accurate response by the system.

II. Related Research

In its core, BioVis focuses on user intent matching with database extension by aligning user query, high level domain model and real life database to the extent possible using the best-effort assumption [7]. From this standpoint, requirement mapping [8], intention discovery from click logs [9], and query intention recognition [10] research complement ours. The two other research projects that come closest to our approach are query aiding and cooperative query answering [11], and mapping query intent for data integration [12].

Although it has the look and feel of BioVis functionality, research in matching user requirement with web services [8] does not accept user view of the applications or map queries to arbitrary databases; nor it is flexible enough. The reliance on web service platform is also limiting because not too many life sciences resources are web service compliant. On the other hand, click log analysis [9] based intention recognition is after the fact and is used for entirely different application types, such as e-commerce where information channeling is the key. Finally, intention recognition in [10] does not assist in mapping user queries to database queries in any way. The most interesting and closest approach to BioVis is the imprecise query mapping in [12]. BioBike actually accepts somewhat tentative user query and maps it on a database extension by correcting the query based on the scheme. However the range and type of mapping allowed are much weaker and more stringent, only applies to homogeneous database schema that

978-1-4799-3481-2/14/$31.00 © 2014 IEEE

are locally accessible, and does not truly attempt to understand user intentions.

III. BioVis by Example

Consider an application in which a user is trying to find out the name of a human growth control protein, its starting location in a chromosome, and visualize its structure using a suitable software. She has, at her disposal, two large tables shown partially in figures 1(a) and 1(b), and the Protein Data Bank (PDB) web site at `http://www.pdb.org` shown schematically in figures 1(d) and 1(e). One way she can compute this query is to first write the following query

```
select P.protName, G.species
from GeneTable as G, ProteinTable as P
where G.geneName = P.gName and G.species =
   "human"
```

to collect the protein name and species pairs in the table *Temp*. Once collected, she can then manually submit each row in this table to the input form in figure 1(d) and collect the returned responses from the response page in figure 1(e). Alternatively, she can write a small Perl script to manage the submission and collection of the results. In either case, she is able to create the table in figure 1(f) as her intermediate table by submitting each row in *Temp* before executing the following SQL query as her final response to compute the table in figure 1(g).

```
select P.protName, G.start, F.struct
from GeneTable as G, ProteinTable as P,
   FunctionTable as F
where G.geneName = P.gName and P.protName
   = F.pName and F.func = "growth factor"
```

The complexity associated with this simple query is formidable from a computational standpoint, especially when the tables in figure 1(a) and 1(b) are dynamically collected from other online sources. Users need to resolve the schema heterogeneity and parse the web forms to gather the internal structures and variable names (i.e., *pName* as opposed to the displayed *Protein Name*), and so on. Though not difficult once the structure of the tables and forms are understood, the semantic relationships among all the variables and attributes will need to be correctly interpreted. The question is: Is it possible for a user to write the same query in a very abstract level without knowing any lower level details, being required to write any Perl script or manually submit all relevant data at the *PDB* web site, yet compute the above queries automatically to yield the table in figure 1(g)?

The answer to this question perhaps rests on what is an abstract query and how high are the level and "schema" independent queries we allow. Our goal is to show that it is possible to capture the query conceptually with the aid of an interpretive structure provided by the user. This interpretive structure serves as her view of her application world that shows relationships among the conceptual objects, and their properties. Figure 2(a) describes one such concept graph in which rectangles represent objects, ellipses represent properties, and lines show relationships. Lines with arrows show direction, and lines without them show symmetric relationships. In this figure, *consists* is an association object as it connects three objects through three unlabeled symmetric relationships. The relationship *interact* between two *proteins* is labeled and symmetric, while the relationship *encode* between *protein* and *gene* is labeled but is directional[1].

In the remainder of this paper, we will show that the BioVis query below

```
compute proteinname, startloc, structure
from gene, protein, sequence
when name = "human" and function =
   "growth factor";
```

formulated based on the conceptual structure in figure 2(a), can be translated into the query below, in a new data integration query language called *BioFlow*, with identical effectiveness, but without the effort needed to compute it using traditional means. The important point to note here is that the translation is user transparent and fully automatic, and the user query is solely based on the user view of the application, as captured in the concept graph.

```
create view Temp as
extract pName, struct, func
using matcher ontomatch wrapper fastwrap
   filler carbon
for G.species = organism and
   P.protName = pName
from http://www.pdb.org
submit Proteins as P, GeneTable as G
where G.geneName = P.geneName and
   G.species = "human";

select P.protName, G.start, T.struct
from GeneTable as G, Proteins as P, Temp as T
where G.species = "human" and T.func =
   "growth factor" and P.protName = T.pName
   and G.geneName = P.geneName;
```

The overall idea for the translation is quite simple. Since the conceptual graph captures the semantic relationships the user perceives to be correct at an object level, our goal will be to map this network onto a real extensional database that not only includes a concrete database, but also a set of descriptions of online resources – databases and tools – that can be used to compute tables and can be incrementally added to the concrete database. Putting it differently, the external resources are viewed as functions that when supplied with inputs from the local database, produce additional tables in a defined and predictable way. If we adopt a best effort integration model [7], we can accomplish this mapping by adapting a labeled graph matching algorithm. Once mapped, we are in a position to identify the objects in the underlying structure by employing a suitable version of shortest distance based object identification technique. Once these relationships are known, re-formulating the query is rather simple.

[1]Technically, it means looking from the object *gene* we can know it relates to a *protein*, but from the *protein*'s point of view, it has no relationship with *gene*. Instead, if a symmetric relationship is intended, a bidirectional line is more appropriate. In fact this view is an adaptation of and consistent with the earlier data models such as SDM and DAPLEX which served as bases for many modern query languages.

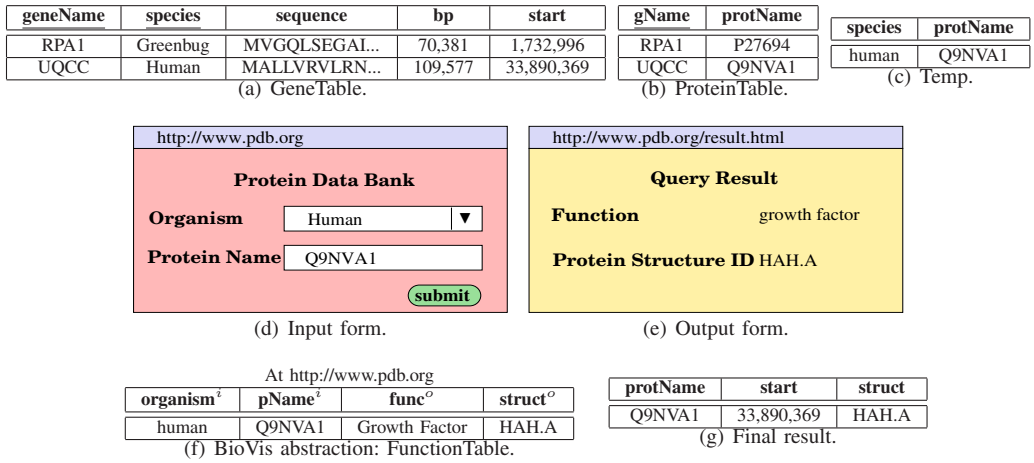


Fig. 1. Gene information tables.

IV. BioVis System Architecture

The overall BioVis architecture in figure 2(b) shows a reactive BioVis to BioFlow query translation component *translator BB*, which is the focus of this paper. While the system is more dynamic and reacts to missing information by activating autonomous information acquisition processes through the *ProMap* subsystem based on the query being translated, for the purpose of this paper, we will assume that all needed information about the web resources have been collected and represented in the system appropriately. The figure also shows various toolboxes available to help users form (*AutoStruct*) or synthesize (*Query Synthesizer*) target queries, and to tweak translated BioFlow queries to adjust parameters to account for translation errors and uncertainties. The ultimate execution is always in SQL/XQuery with remedial processing and control code execution over database management systems such as MySQL and eXist.

V. BioVis Formal Model

The data model of BioVis is a quintuple $\mathcal{S} = \langle \mathcal{C}, \mathcal{R}, \mathcal{M}, \mathcal{D}, \mu, \Sigma, F \rangle$ such that $\mathcal{C}$ is a *concept graph*, $\mathcal{R}$ is a set of relation schemes, $\mathcal{M}$ is a set of expressions that describe elements in $\mathcal{R}$ in relation to the database instances $\mathcal{D}$, and $\mathcal{F}$ is a set of functions over the elements $\mathcal{C}, \mathcal{R}, \mathcal{M}$ and $\mathcal{D}$, μ is a function that maps meta-expressions over $\mathcal{C}$ to nested relational expressions over $\mathcal{R}$ and $\mathcal{D}$, and Σ is a function that returns a precedence graph over a subset of functions in $\{\mu\} \cup \mathcal{F}$.

The concept graph is a structure involving *objects* described, *properties* and *relationships* among them using first-order terms in $\mathcal{T}$. All objects and properties are named, but not all relationships need to be labeled. Two objects are *symmetrically* related if they both are related to each other, and they are *identically* related if the relationship labels are identical (including empty labels). An object in a concept graph is called an *association object* if it is symmetrically and identically related to more than one object. Association objects help establish relationships among other objects which is not possible through symmetric binary relationships.

The meta-data component $\mathcal{M}$ is a set of function descriptions, each of the form $\langle \mathcal{I}, f_\tau, L, \theta, \rho \rangle$ where $\mathcal{I}$ is a set of triples of the form $\langle A, T, d \rangle$ where A is the name of the argument/attribute, T is the type, and d is a binary value indicating if A is an input value or output attribute meant to capture the input-output behavior of online data transformation functions. f_τ is meant to state the type of the function or service this function represents, i.e., a legacy web interface, a web service, or a client server. L is the location or URL of the function or service, and θ is a Boolean condition involving elements in $\mathcal{I}$ designated as input values, i.e., $d =$ "i" as opposed to $d =$ "o". Finally, ρ is also a binary value that states if the returned response for every input is a scaler value (i.e., a tuple) or a relational value (i.e., a set of tuples).

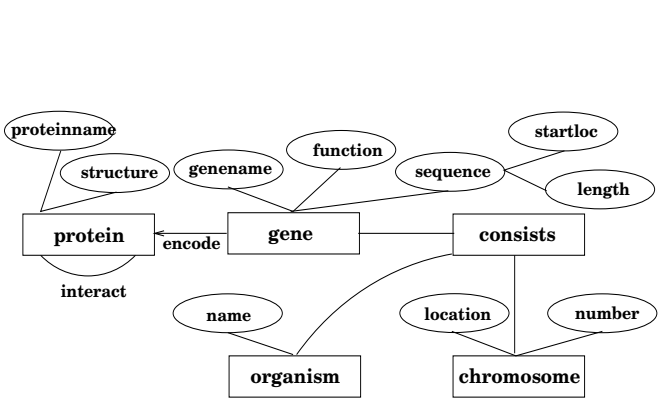
A. BioVis Meta-Language

Unlike BioFlow, we allow two types of statement in BioVis. The first set of statements, called the *conceptual queries*, are expressed over the concept graphs where we use objects for relation names, and properties for attribute names. Since the concept graph allows nesting of attributes, the structure naturally conforms with NFNF relation schemes, e.g., the *length* and *startloc* attributes of the property *sequence* in the concept *gene* in figure 2(a). The *relational queries*, on the other hand, are expressed over the scheme $\mathcal{R}$ and instance $\mathcal{D}$. Technically, hybrid queries involving concepts and relations are possible by partitioning the concept and schema terms. For simplicity, we will focus only on conceptual queries and not on such hybrid queries.

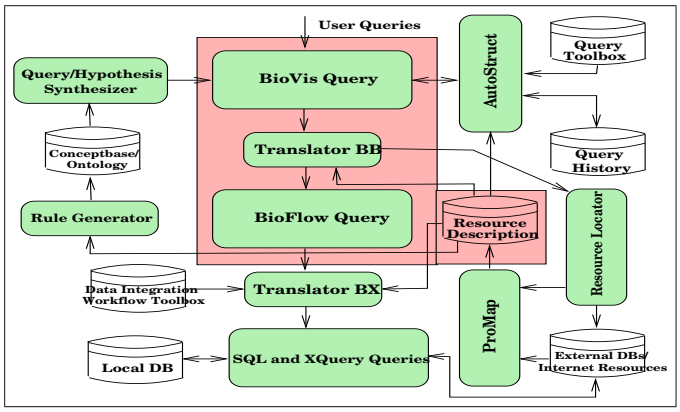
The query language of BioVis has a SQL-like structure, where f_is are functions or operations in $\mathcal{F}$ including empty operations, and c_l, P_j and B_os are all terms in the concept structure $\mathcal{C}$.

compute $f_1(P_1), \ldots, f_k(P_k)$
from $c_1, \ldots, c_m$
when θ
group by $B_1, \ldots, B_n$
having ϕ;

We impose a restriction on the admissible terms in the BioVis language in the interest of an intuitive relationship among the underlying objects, as follows. An admissible BioVis statement requires that



(a) Concept graph example.



(b) Major components of BioVis system.

Fig. 2. Concept graph and BioVis architecture.

- the terms in $c_i \cup P_j \cup B_k$ form a forest in $\mathcal{C}$ where the root of each tree is a term in c_i, no leaf nodes are in c_i, and every P_j is a property of one c_i, i.e., P_js may be from c_is, and
- θ, ϕ, and B_ks do not involve terms in c_i.

This restriction intuitively ensures that each c_i is either an entity set in ER sense, or is a relation valued attribute in nested relational sense, and each P_j is either a relation valued attribute or a property in an entity set. Therefore, the BioVis statement in section III is admissible. Similar to the relational model, it is assumed that relationships among the concepts in the from list exist, i.e., relationship *encode* between *gene* and *protein*, and the relationship of *sequence* as a property of the object *gene*. The flexibility we are seeking here is that users need not express implied relationships explicitly and include only the relevant concepts in the query with the assumption that the underlying concept graph already has captured the semantic relationships that the user subscribes to. The system's job is now to extract the implied relationships in the query and map the entire query into an extensional BioFlow database query.

B. Mapping Abstract Queries to Extensions

The BioVis to BioFlow translator in figure 2(b) is captured in the object mapping function μ and sequencer function Σ in structure $\mathcal{S}$. Using a graph matching method, function μ delivers the best candidate in the form of a correspondence graph G_c involving elements in $\mathcal{R}$ for each of the objects and implied relationships in the from clause list. Such matches may result in a complex select-project-join query in BioFlow as the concept graph objects may be spread across multiple tables in $\mathcal{R}$. However, in the process of transforming the correspondence graphs into a BioFlow query, format conversion and table restructuring may be required to make queries executable and semantically meaningful. The sequencer function Σ ensures this compatibility by augmenting G_c with additional interleaving "action" nodes with functions in $\mathcal{F}$.

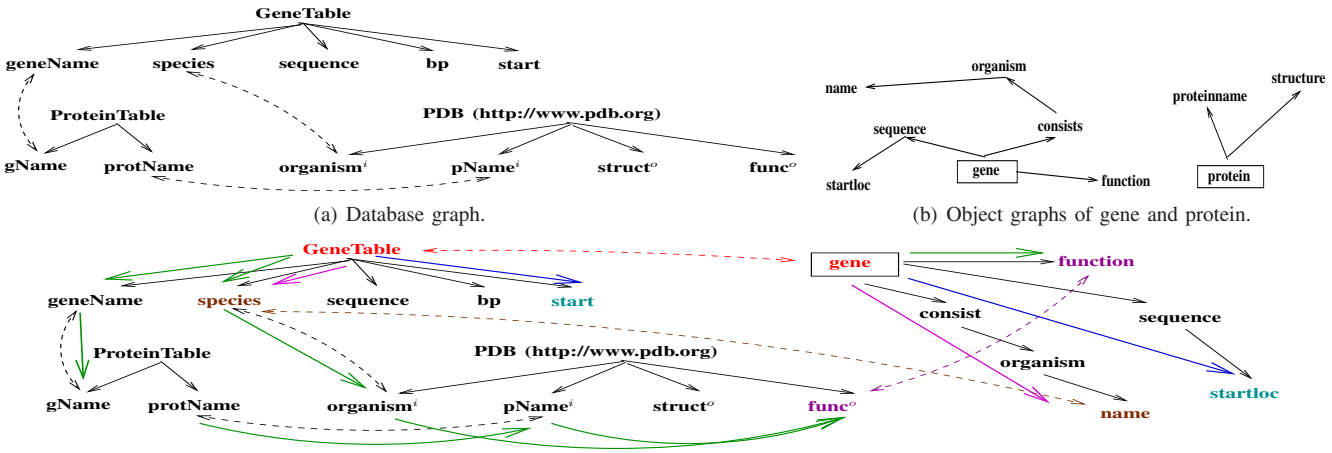
1) Graph Matching based Query Transformation: Recent research on schema matching (e.g., [13], [14]) and graph matching (e.g., [15]) have witnessed interesting progress. However, only a handful of research has exploited their combined strengths to address schema heterogeneity [16], [17] through mapping. It is precisely this combined strength that we exploit, especially the *similarity flooding* [17] method, in order to define our BioVis to BioFlow translator.

We first construct a directed acyclic graph involving the schemes in $\mathcal{R}$ and function descriptions in $\mathcal{M}$ as follows. For every relation scheme $R \in \mathcal{R}$, we create a tree with the relation name as the root and attribute names as the leaf nodes. Also, for every function description in $\mathcal{M}$, we create a tree with the function name as the root, and input attributes as leaves. Furthermore, we create leaves for every output attribute if the output designation ρ is scalar, if not, we create another subtree with the output attributes as its leaves. This is required for two reasons: if the table is a NFNF relation, or if a web function returns a table in response to a form submission resulting in a NFNF relation. If for two relations, a set of attributes of one is a foreign key of the other relation, we merge the attributes in the two corresponding trees. The graph constructed is denoted as $forest(\mathcal{R})$. Note that the concept graph is already a forest when we treat objects as parents and properties as children, denoted analogously as $forest(\mathcal{C})$, with additional edges between objects. For the example database in section III, the trees are shown in figure 3.

In the first step, we perform a global term matching from object graphs to database graph in the context of keeping term relationships. We thus establish the correspondences *gene* $\rightarrow$ *GeneTable, function* $\rightarrow$ *func*^{o}, *startloc* $\rightarrow$ *start*, and *name* $\rightarrow$ *species* shown respectively in red, purple, teal and brown terms[2]. The paths that lead to properties *name, startloc* and *function* of *gene* in the object graph are shown in purple, blue and green respectively. While the attributes *species* and *start* are readily available in *GeneTable*, attribute *function* is not. Instead, this attribute is computable from the *PDB* site if we supply the *species* from *GeneTable* and *protName* from *ProteinTable* by establishing correspondence of gene objects via *geneName* in *ProteinTable*. This entire process is carried out by object mapping function μ. A similar process will map the object *protein* in the from list in our example.

In the second step, we consolidate all the object graphs into one graph so that mapping among the object graphs can be shared. For example, the object map of *protein* in the graph in

[2]We follow the convention in [18] where *func*^{o} means *func* is an output attribute of an interface while *pName*^{i} represents an input attribute.



(a) Database graph.

(b) Object graphs of gene and protein.

(c) Correspondence graph for object gene.

Fig. 3. Correspondence through graph matching (better viewed in color).

figure 4 shows that the submission process in *PDB* for *function* attribute of *gene* can be shared with the *structure* attribute of *protein*. So the goal would be to remove implementation of the *structure* attribute of *protein* and include it in the graph for *function* attribute for *gene*.

As the final step, we check the semantic disparities of the attributes in terms of format requirements, and their structure in terms of flat table versus nested relations. Format disparities are dealt with by introducing format conversion functions as an intermediate step before a query. These functions are modeled in ways similar to a web function and matched based on their input output behavior. Descriptions of such functions are also stored in the database in a way similar to web resources from which database graphs can be created as in step one.

C. Query Translation

Once the graph in figure 4 is constructed, the join path, select list attributes (shown in mustard) and where clause attributes (shown in bubble gum) are already mapped out. Although we treat *PDB* as a table, it cannot just write the following SQL query to compute the desired response

> select *D.pName, D.struct, G.start*
> *GeneTable as G, Proteins as P, PDB as D*
> where *G.geneName = P.geneName* and
> *D.organism = G.species* and *P.protName = D.pName* and *G.species = "human"* and *D.func = "growth factor"*;

because *PDB* is not a ground table. Besides, computation of *PDB* is required for both the where clause condition *func="growth factor"*, and for output *struct* in the select list making its use tricky and precarious. Moreover, it excludes the possibility of using *PDB* in a nested subquery. A simple and straightforward method is to pre-compute *PDB* as a temporary relation and then use it to complete the query, as we have shown in section III.

VI. Observations and Challenges

The key observation is that if the user query follows some form of universally agreed upon view of entities and relationships (as in figure 2(a)) that a system is able to recognize, then domain knowledge about the sites, resources, and data can be leveraged to aid resource specific unambiguous schema mapping. In particular, it should be clear that the relationship heuristics exploited in most graph matching techniques for schema matching are too limited and obvious (i.e., child of a node is related to the parent) and will likely fail to recognize an entity drawing attributes from seemingly unrelated multiple schema trees as shown in figures 3 and 4. For example, reconstruction of the entity *gene* required mapping *func^o* to *function* that was not part of *genetable*. Also, while a matcher could potentially map *gene* to *genetable*, it is not clear how it will establish the mapping of *species* to *name*. But in this application, the connections are semantically appropriate.

We believe that the key to addressing such "smart" schema mapping rests on recognizing inter graph relationship toward entity recognition [19]. Technically, the challenge is reconstructing the entities from the schema graphs following unique relationship paths (as shown using colored arrows in figures 3 and 4) given a set of conceptual entity graphs, and a set of distributed database schema graphs. These relationship paths should be computable using database operations if not already available in a single scheme. It is interesting to note that existing schema matching systems are usually capable of matching the labels in figures 3 and 4 accurately, but are not smart enough to make the overall conceptual connections contextually on a case by case basis.

VII. Conclusions

Research in generating concrete applications from rudimentary and abstract user descriptions has been gaining traction in recent years [20], [21] in prominent ways. Such an approach is becoming increasingly feasible with the maturity of several cognate technologies that can be assembled into one coherent system, as we have demonstrated in this paper. Application of semantic nearness and approximation is so inherent in biological systems that query processing systems such as BioVis are natural candidates for cooperative query answering. For example, the BioBike system [11] uses many biological axioms to aid query formulation and to return responses that are intentional, and are indirectly deducible

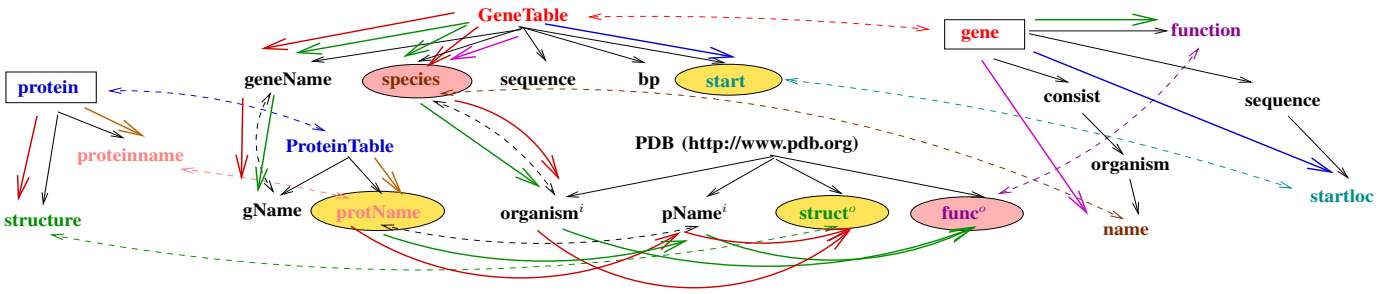


Fig. 4. Consolidation of object maps (better viewed in color).

from an extensional database. By properly tweaking the BioVis component systems and adapting techniques proposed in AutoPipe [22], we believe we can map the query in section III onto the BioVis scheme shown in figure 5 just as accurately, to return a response identical to the BioFlow query in that section. Note that in this figure, the rounded schemes are external resources similar to *PDB*, the similarity of the objects are far from obvious and the semantics are far more complex. If we consider format differences between *sequence* in *SequenceTable* and seq^i in *ProteinMapTable*, in addition to schema and structural heterogeneity, the interpretation becomes more complex because we now will have to deduce that even though we do not have a direct join based relationship across tables, we can still compute the query using the functions.

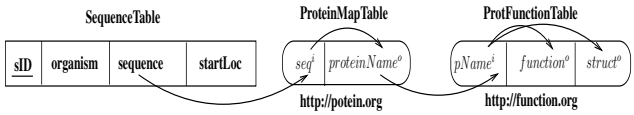


Fig. 5. Intentional query mapping.

Most of the components described in this paper exist in isolation and the overall mapping process has been tested by stitching the components together manually. We strongly believe that BioVis allows a customized view of the underlying resources on a per query basis by allowing the change in interpretation by either changing the concept graph, using different mapping modules and interpretive tools, or altering the interpretation altogether by substituting compiler plug-ins while keeping the resources as they are. Such an approach respects resource autonomy, allows asynchronous meta-data gathering, and best practice of candidate modules as they become available.

Acknowledgment

This research was supported in part by Idaho National Laboratory grant FFK039 and a University of Idaho Office of Research and Economic Development grant.

References

[1] J. Pokorný, "NoSQL databases: a step to database scalability in web environment," in *iiWAS*, 2011, pp. 278–283.

[2] J. Pound, A. K. Hudek, I. F. Ilyas, and G. E. Weddell, "Interpreting keyword queries over web knowledge bases," in *CIKM*, 2012, pp. 305–314.

[3] M. Stonebraker, "SQL databases v. NoSQL databases," *Commun. ACM*, vol. 53, no. 4, pp. 10–11, 2010.

[4] J. Pound, I. F. Ilyas, and G. E. Weddell, "Quick: Expressive and flexible search over knowledge bases and text collections," *PVLDB*, vol. 3, no. 2, pp. 1573–1576, 2010.

[5] H. M. Jamil, A. Islam, and S. Hossain, "A declarative language and toolkit for scientific workflow implementation and execution," *IJBPIM*, vol. 5, no. 1, pp. 3–17, 2010.

[6] H. M. Jamil, "Designing integrated computational biology pipelines visually," *TCBB*, vol. 10, no. 3, pp. 605–618, May/June 2013.

[7] W. Shen, P. DeRose, R. McCann, A. Doan, and R. Ramakrishnan, "Toward best-effort information extraction," in *SIGMOD*, 2008, pp. 1031–1042.

[8] S. Assar and K. Aljoumaa, "Matching user requirements with query formulations in intentional service oriented computing," in *COMPSAC Workshops*, 2011, pp. 476–481.

[9] H. Duan, E. Kiciman, and C. Zhai, "Click patterns: an empirical representation of complex query intents," in *CIKM*, 2012, pp. 1035–1044.

[10] M. Mendoza and J. Zamora, "Building decision trees to identify the intent of a user query," in *KES (1)*, 2009, pp. 285–292.

[11] J. Elhai, A. Taton, J. P. Massar, J. K. Myers, M. Travers, J. Casey, M. Slupesky, and J. Shrager, "BioBIKE: A web-based, programmable, integrated biological knowledge base," *Nucleic Acids Research*, vol. 37, no. Web-Server-Issue, pp. 28–32, 2009.

[12] A. Telang, S. Chakravarthy, and C. Li, "Querying for information integration: How to go from an imprecise intent to a precise query?" in *COMAD*, 2008, pp. 245–248.

[13] T. H. Nguyen, H. Nguyen, and J. Freire, "PruSM: a prudent schema matching approach for web forms," in *CIKM*, 2010, pp. 1385–1388.

[14] E. Peukert, J. Eberius, and E. Rahm, "A self-configuring schema matching system," in *ICDE*, 2012, pp. 306–317.

[15] S. Amin, R. L. Finley, Jr., and H. M. Jamil, "Top-k similar graph matching using TraM in biological networks," *TCBB*, vol. 9, no. 6, pp. 1790–1804, 2012.

[16] Z. Zhang, H. Che, P. Shi, Y. Sun, and J. Gu, "Multi-labeled graph matching - an algorithm model for schema matching," in *ASIAN*, 2005, pp. 90–103.

[17] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A versatile graph matching algorithm and its application to schema matching," in *ICDE*, 2002, pp. 117–128.

[18] A. Calì and D. Martinenghi, "Conjunctive query containment under access limitations," in *ER*, 2008, pp. 326–340.

[19] G. Papadakis, E. Ioannou, C. Niederée, and P. Fankhauser, "Efficient entity resolution for large heterogeneous information spaces," in *WSDM*, 2011, pp. 535–544.

[20] Y. L. Sun, T. J. Harmer, A. Stewart, and P. Wright, "Mapping application requirements to cloud resources," in *Euro-Par Workshops (1)*, 2011, pp. 104–112.

[21] E. Toch, A. Gal, and D. Dori, "Automatically grounding semantically-enriched conceptual models to concrete web services," in *ER*, 2005, pp. 304–319.

[22] H. M. Jamil, "AutoPipe: A toolbox for systems biology workflow query synthesis," in *ISBRA*, May 2012.