

Secure Simultaneous Search of Distributed, Heterogeneous Bioinformatics Databases

Gregory L. Vert

Computer Science and Engineering Department
University of Nevada
Reno NV, USA

Nic V. Herndon

Computer Science and Engineering Department
University of Nevada
Reno NV, USA

Abstract – *The field of bioinformatics has experienced rapid development during the last few years. The volume of data generated at universities and private firms has created a new problem for the bioinformatics researcher: how to retrieve the data from several distributed, heterogeneous databases and how to ensure that the retrieved data is securely transmitted in a computationally efficient fashion. Several attempts to integrate the databases or to create a graphical user interface for constructing SQL queries have met with limited success. This paper presents a model for searching and retrieving similar genomic data from multiple data sources and transmitting the data with a low cost authentication technique referred to as BioRLE. Retrieved information is presented in a user friendly interface.*

Keywords – Bioinformatics Search, Database Integration, Schema Integration.

1. Introduction

Bioinformatics researchers are faced with the challenging task of retrieving genetic data for analysis from distributed heterogeneous databases. Bioinformatics data generated at several universities and private firms are made available to researchers either as downloadable flat files or through web interfaces to local relational databases. Thus, a bioinformatics researcher has to become familiar with relational databases and to several different web interfaces. Moreover, the lack of data standardization has led each data provider to create their own naming convention. Therefore, the same data may be found under different names in different databases or the same name may be used for different data in different databases. This adds considerable work overhead for bioinformatics researchers in analyzing the data and generating hypotheses.

Several methods attempted to reduce the effort to retrieve bioinformatics data by: combining all the databases in a federated database [1], relating the databases using schema integration [2], collecting all

the data from the remote databases and creating a new local database with the collected data augmented with XML semantic tags [3, 9], using a conceptual interface rather than a logical interface for databases [4], using a graphical query interface [5, 7], or using a natural language to query the databases [6, 8].

Each of the above mentioned approaches had its drawbacks. For the federated database approach only databases that conformed to the data model could be integrated. With schema integration a lot of effort was consumed in fixing data representation conflicts, which most of the times required the assistance of a person with special knowledge in the bioinformatics area – a bioinformatics domain expert. With XML integration the size of the integrated database was several times larger than the data stored due to the overhead of the metadata stored for each piece of data. And finally, the natural language interface to pose database queries produced results that were not satisfactory.

This paper examines these attempts and presents a new alternative solution that can make bioinformatics data retrieval easier. The remaining of this paper is organized as follows: section 2 presents current work in more detail, section 3 develops a conceptual alternative solution, section 4 discusses a low cost method for secure transmission of the data, and section 5 describes the details of the implementation. Finally section 6 presents the results of the implementation of a small prototype based on this research, and section 7 presents future work.

2. Related work

Much work has been done previously on how to best retrieve bioinformatic data and deal with the idiosyncrasies of how it is stored and managed. Kemp et al. [1] developed a federated architecture and mediator, based on the functional data model database, P/FDM, to integrate access to heterogeneous, distributed biological databases. A federated database system is a type of meta-database management system

which transparently integrates multiple autonomous database systems into a single federated database. The role of the mediator was to process queries posed to the federation's integration schema. The mediator held the metadata describing the integration schema and the external schemas of each of the federation's data sources. It stored data descriptions of the participating databases which were expressed in a common data model. The functional data model allowed the query of the databases using a concept familiar to programmers – calling functions that return the data through the referenced parameters, as in Figure 1. In this example, `ec_number` is an attribute of the `enzyme` entity in one of the federated databases, `swissprot_entries` is an entity in the same database, `protein_name` is an attribute of the `protein` entity in the same database, and `def` and `acc` are attributes of the `swisprot_entries` entity. Entities were used to represent conceptual objects and the functions represented the properties of an object. The mediator translated the queries, posed through a graphical interface, translated them and then passed them to the external databases. When it received the results it fused and returned them to the user through the graphical user interface. One of the problems with this approach would be encountered when the schema of one of the federated databases is changed. In this case, the data contained in that database would not be retrieved until the metadata of that database is updated in the federated architecture. Another disadvantage of this approach is that the federated databases have to adopt a “shared data model across participating sites.”

```
for each e in enzyme such that ec_number(e) =
"1.1.1.1"
  for each s in swissprot_entries(e)
print(protein_name(e), def(s), acc(s));
```

Fig. 1 – Example of a query posed on the federated architecture and mediator implemented by Kemp et al. [1]

The process of schema integration has been explored by Ram and Ramesh [2]. They introduced and described the steps required for schema integration: schema translation, pre-integration, schematic interschema relationships generation, and integrated schema generation. By using the blackboard architectures – based on AI systems – they were able to facilitate the communication between human and computational agents and created a semi-automated schema integration process. One aspect of this implementation, schematic interschema relationships generation, involves the assistance of a domain expert to identify similar objects with the same representation, distinguish between dissimilar objects with similar representations, and suggest a new representation. Any changes to the databases' schemas

would require the assistance of the bioinformatics domain expert to reintegrate the database.

Wong and Shui, and Direen et al. presented a different approach in which the data from remote databases was stored in local XML documents along with the semantics of that data [3, 9]. Then, a search of these documents could retrieve the relevant data. This approach eliminated the need to know in advance the semantics of the data stored in database, which was needed in the traditional design of the relational databases. An example of a XML database entry garnered from the NCBI genebank presented by Direen et al. is shown in Figure 2. One major drawback of XML databases is the size of the database. Each piece of information has to be stored with additional metadata which generates additional data to be stored in the database, increasing the size of the database.

```
<genbank_entry>
  <header>
    <locus bp="741" type="mRNA"></locus>
    <definition>
      Homo sapiens cystic fibrosis
      transmembrane
      conductance regulator...
    </definition>
    <accession>NM_000492</accession>
    <version>NM_000492.2
GI:6995995</version>
  </header>
  <Comment>
    This record has been curated by NCBI...
    The protein encoded by this gene is a
    member
    of the super family of ATP-binding
    cassette...
  </Comment>
  <features>
    <feature chromosome="7"></feature>
    <feature map="7q32.1"></feature>
    <feature gene="CFTR"></feature>
    <feature
product="ABC_membrane"></feature>
    <feature
note="ABC_transporter"></feature>
    <feature note="ATP-binding"></feature>
  </features>
  <base_count a="1886" c="1181"
...></base_count>
  <origin>
    aattggaagcaaatacattcacacgaggtc...
  </origin>
</genbank_entry>
```

Fig. 2 – Example of XML database entry presented by Direen et al. [9]

Siau et al. claim, based on the results of their experiment, that users perform better interaction with databases with the conceptual interface, expressed in terms of entities, objects, relationships, and attributes, rather than the logical interface, with queries specified in terms of abstract structures, such as records and

tables [4]. In their experiment they split the test subjects into two groups. The first group received a training booklet with concise description of the relational data model and illustrations of the query by example, QBE, query language, while the second group's booklet contained a concise description of the Entity Relationship (ER) model and the full query language designed for the ER model, VKQL (Visual Knowledge Query Language). Then an initial test was administered in which the subjects had to answer ten questions and were allowed to use the training material to answer the questions. After each question they were asked to rate the confidence with which they answered the question. Two weeks after the initial test, the subjects were given a retention test in which they were asked to answer the same ten questions as in the initial test. Shortly after the retention test was administered, the subjects took a practice test and if they had any questions they were answered by the trainer. Immediately after the practice session, the subjects were asked to take the test again.

Aversano et al., and Polyviou et al. confirmed the results of Siau et al. [5, 7]. Aversano et al. designed and implemented an iconic query system which enabled novice users to learn the relational data model and the textual query language SQL through the use of icons. Polyviou introduced and formally defined query by browsing (QBB) – a visual query language based on the desktop user interface paradigm. Using QBB the user could formulate and pose queries by navigating the database schema tree similar to how users find files with a graphical interface for file systems.

Silverman et al., and Dekleva investigated the use of natural language in querying databases. Silverman et al. described an agent that used restricted natural language to help users search product catalogs which achieved significant accuracy rates, while Dekleva studied the practicality of natural language querying by evaluating the level of correct interpretation and found that the researchers who have experimented with natural language query systems doubt their usefulness and found SQL to be superior. In an experiment performed by Jarke et al. the results indicated that “subjects were about two to three times more likely to get correct or partially correct output in SQL than they were in natural language.” The experiment compared a prototype natural language interface that mapped the queries into SQL with a SQL interface. Users of the natural language interface attended training which concentrated on the underlying philosophy of a natural language system, after which, users from each group had to formulate queries with the SQL and natural language interface, respectively.

To summarize, the above mentioned approaches failed to generate considerable results from several reasons. They limited the types of databases that could be integrated, as in the federated database approach. They focused on fixing data representation conflicts which consumed a lot of effort of a domain expert, as in the schema integration. They considerably increased the size of the integrated database with XML metadata, in XML integration. And finally, they failed to produce satisfactory results when using natural language interface to pose database queries.

3. Search of Distributed Databases

The administrators of bioinformatics databases stored at universities and private research firms usually limit the access of the users of their web interfaces to retrieve only. This prevents the corruption of the data. Considering that almost all bioinformatics researchers accessing the remote databases use only the search functionality, the solution that we propose to make the use of bioinformatics databases more user-friendly is an application that allows the users to do simultaneous searches on the distributed databases.

Compared to the previous mentioned approaches, this allows the integration of databases regardless of the data model, presenting the user the raw data from the remote databases without trying to solve data representation conflicts, in less time due to the smaller size of the integrated database, and with a user-friendly interface as explained below.

This application has two main components: the back end interface which allows the administrator to add data from the remote databases to the search engine and the front end which allows user to pose queries to the search engine. The front end interface is similar to Google's interface, as shown in Figure 3.



Fig. 3 – Front end interface of the search engine

Using this interface the users can do searches on the DNA sequences and/or keywords related to the DNA sequences. For example, the user can do a search

on a DNA string and the drought keyword to find if the DNA string is related to the defense mechanisms implemented by plants to defend against draught.

The backend component is a standalone application. It is used to add data from remote databases to the search engine in two stages. In the first stage, the DNA sequences from remote database are indexed and inserted into a local database along with a URL of the remote database. To implement this, if the data is stored in a relational database, we would query each table and retrieve the whole content of the table, then parse the content and search for DNA strings. Once a DNA string is found, it is added to the local database along with the URL of the remote database. For example, to add the DNA string AAAATTGCCCC we would perform the following steps, which reference the tables described in Figure 4:

- If the string does not exist in the DNA table we will add it:
insert into DNA
set DNA_String = 'AAAATTGCCCC';
- Get the generated ID of the string from the DNA table, which we will use in the following step:
select DNA_ID from DNA
where DNA_String = 'AAAATTGCCCC';
- Add the DNA string (the ID of the DNA string) along with the URL to the URL table:
insert into URL
set Keyword = DNA_ID,
URL = 'url_of_the_database';

If the data is stored in a flat file, we would parse it and search for DNA strings. Once a DNA string is found, we would add it to the local database along with the URL of the flat file. Also, the remaining data from that database relating to the DNA sequences is indexed as well and inserted into a local database in a separate table also containing the URL of the remote database, using a parsing process similar to the one used to extract DNA strings. This allows a rapid integration of remote databases.

This is important in two cases: when data from a new database is added to the search engine, or when the schema of a remote database has been updated. In either case, the users of the search engine would be able to find right away that the data of interest is available in some database and would get the URL of the database. However, in this case the user would have to figure out how to retrieve the data from the remote database.

In the second stage, the SQL query that is used to retrieve the data from the remote database, based on the metadata of the remote database, is added to the local search engine allowing the users to retrieve the data with the click of the mouse. This is a separate process because, unlike the first stage, it cannot be fully automated and it requires the human assistance.

For example, to add the SQL query that would retrieve the DNA string AAAATTGCCCC we would add the SQL string to the URL table:

```
update URL
set Query = 'query to retrieve the DNA string
from the remote database'
where
Keyword = 'DNA.DNA_ID of the
DNA string' and
URL = 'url_of_the_database';
```

4. Secure Transmission

Once data has been identified for retrieval from the meta database, the question becomes how to transmit the data in a fashion that will be secure and can be utilized for authentication of the data upon arrival at its source.

Typically, Public Key Encryption or Secure Hashing is utilized for authentication and encryption of transmitted data. However, these methods are computationally intensive which can be a problem with bioinformatic data. The problem stems from the sheer volume of bioinformatic data that can be retrieved in a single query. As an example, the BLAST database contains 6 billion DNA nucleotide base pairs. All of which in theory could be retrieved in a single query for transmission. The time to encrypt such a large volume of data would be prohibitively high. Therefore, what is needed is a technique that can provide a fast method for doing some simple authentication. For this purpose we propose the use of Run Length Encoding (RLE) to create a data reduced signature of retrieved data that can be used for authentication. This technique we have named BioRLE.

The BioRLE method is fairly straight forward: for all adjacent based pairs in a sequence, reduce the data to a type descriptor and a count descriptor. For instance, given the sequence:

AAAATTGCCCC

the BioRLE authentication signature would be:

A4T2GC4

In essence, this approach is a data reduction method that can be used for transmission of smaller sequences of data and for authentication. In the above example we have a 40% reduction in the amount of data that can be transmitted and a signature that is 60% the size of the original sequence.

The authentication signature is transmitted separately from the original sequence information. Upon arrival of the original sequence and its signature,

the signature of the arrived sequence is computed and compared to the signature. If they match, then the transmitted sequence is assumed to be unmodified and therefore authenticated.

The above approach renders 100% authentication and is our initial proposal for security of data, however we would like to find methods for reducing the signature even further. We are considering signature generation for regions of a particular sequence that may be significant from a biochemical standpoint, e.g. not all sequences code for useful genetic information. Alternatively, we are considering encoding signatures utilizing codons, which are tri-nucleotide DNA sequence units, and a similar scheme (every DNA sequence is composed of four nucleotides: Guanine, Adenine, Thymine, Cytosine). All of this will be the subject of future work.

5. Implementation details

The local database contains the tables listed in Figure 4.

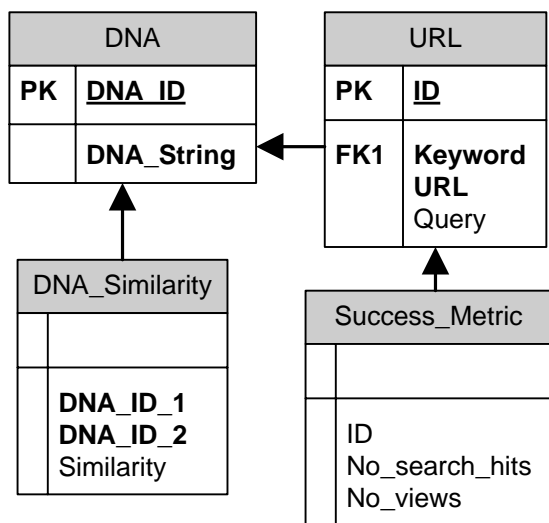


Fig. 4 – Database tables

The DNA table contains the DNA fragments and the generated DNA_ID. The URL table contains the keywords or the DNA_ID of DNA strings; the URL of the database, flat file, or research paper that contains that keyword or DNA string; and the query to retrieve all the data related to the keyword or DNA string from the remote database – for the flat files or research papers this attribute is set to NULL; as well as a generated ID. The DNA_Similarity table contains the IDs of 2 DNA strings and the percentage of similarity between the two strings. We are going to investigate the similarity of DNA strings in future research.

And finally, the Success_Metric table contains the URL.ID of the keyword or DNA string along with the

number of times that ID was returned by searches and the number of times the information returned by the search was actually viewed. The ratio No_views/No_search_hits gives us a success measure – the higher the ratio the higher the number of users that viewed the retrieved data and thus, the higher the probability that the source of the data is reliable.

The following steps are taken when a new database, flat file, or research paper is added to the list of searchable databases. First, any DNA strings found that are not present in the local database are added to the DNA table; the IDs of the newly added DNA strings are inserted in the URL table along with the URL of the remote database; a record is added to the Success_Metric table. Then an application that determines the similarity between the DNA strings, such as BLAST, is run and the results are added to the DNA_Similarity table. If the DNA string is already present in the local database only the URL table is updated with a record containing the DNA_ID and the URL of the database containing the DNA string. Similarly, the data relating to the DNA strings is added to the URL and Success_Metric tables. When a search query ran against the local database returns any data from a remote database which was incorporated using only the above described first step it contains only the URL of where that data can be found. In this case the user uses the web interface of the remote database to retrieve the data.

In the second step, the administrator of the local database determines the table relationships of the newly added remote database and generates the SQL queries to retrieve all the related data and update the URL table with the SQL queries. After this step, if a search query ran against the local database returns any data from a remote database it will also return all the data related to it. In this case the user would not have to learn the web interfaces of the remote databases to retrieve the data – because the local database automatically retrieves it for him/her.

6. Results

The quality metrics is a measure that indicates the confidence that the user should place on the retrieved data. In other words, if there are multiple sources of the retrieved data, how can a user determine which data is more reliable. The quality metrics, Q_{metric} can be determined based on a number of criteria. For the purposes of this research we define them to be:

- the credibility of the source of the remote data – we consider the data stored at research universities with prestigious bioinformatics department to be more reliable than the data stored at private firms,
- the reputation of the users with access to update that database – we consider the data to be less reliable

when there is a large number of users with write permissions to the data, especially if the users are from outside the institution that owns the data,

- the reputation of the database administrators, the security applied on the database, and the update policy implemented by the administrators – the more secure the database is the more reliable the data it contains is.

An additional quality metrics can be calculated from the Success_Metric table. The ratio No_views / No_search_hits gives an indication on how often the returned search data was actually viewed by the users. The higher the number of users that viewed the data, the higher is the quality and the credibility of the data. For example, a ratio value above 75% would indicate that the users that viewed the source of the data considered it to be reliable and viewed the data.

7. Conclusion

Compared with federated databases this approach has several advantages. First, for a database to become federated, the process of database integration is very complex. It requires the close collaboration of computer scientists and a bioinformatics domain expert to analyze the semantics of the data and find ways to fix discrepancies, such as having the same data store under different names in different databases. When the schema of a remote database is updated, which can happen quite often the database integration process has to be redone, otherwise the search queries posed using the old schema would return no results, and during the process of reintegration, the data from the remote database is not available to the users of the federation. With the search engine, the computer scientist can focus on retrieving the data without concern about its semantics and the bioinformatics researcher can focus on analyzing and interpreting the data without the need to learn and understand relational databases. Second, in addition to data contained in the remote databases, research papers can be included in the search engine as well. This allows a researcher to retrieve not only raw data but also related research papers. Moreover, by implementing the

8. Future work

One major aspect that influences the speed of processing of the search queries is the size of the local databases. By indexing the DNA strings and all the data related to them from several remote databases, the size of our local database could potentially become very large. We are considering fragmenting the database and performing parallel searches on each fragment. Additionally, we are considering running

the application that determines the similarity of two DNA strings in parallel on multiple machines (i.e., on each database fragment), which would reduce the time necessary to determine the similarity of a newly added DNA string or a search string to all the existing strings in the local database.

For data security we will investigate two methods for reducing the size of the signature: generate the signature only for the DNA sequences that are significant from the bioinformatics point of view and generating the signatures based on codons or a similar approach.

References

- [1] G.J.L. Kemp, N. Angelopoulos, P.M.D. Gray, "Architecture of a mediator for a bioinformatics database federation," *IEEE Transactions on Information Technology in Biomedicine*, Vol. 6, No. 2, Jun 2002, pp.116 – 122
- [2] S. Ram, V. Ramesh, "A blackboard-based cooperative system for schema integration," *IEEE Expert*, Vol. 10, No. 3, Jun 1995, pp. 56 – 62
- [3] R.K. Wong, W.M. Shui, "Utilizing multiple bioinformatics information sources: an XML database approach," *Proceedings of the IEEE 2nd International Symposium on Bioinformatics and Bioengineering Conference*, Nov 2001, pp. 73 – 80
- [4] K.L. Siau, H.C. Chan, K.K. Wei, "Effects of query complexity and learning on novice user query performance with conceptual and logical database interfaces," *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, Vol. 34, No. 2, Mar 2004, pp. 276 – 281
- [5] L. Aversano, G. Canfora, A. De Lucia, S.Stefanucci, "Understanding SQL through iconic interfaces," *Proceedings of the 26th Annual International Computer Software and Applications Conference*, Aug 2002, pp. 703 – 708
- [6] B.G. Silverman, M. Bachann, K. Al-Akharas, "Do what I mean: online shopping with a natural language search agent," *IEEE Intelligent Systems*, Vol. 16, No. 4, Jul-Aug 2001, pp. 48 – 53
- [7] S. Polyviou, G. Samaras, P. Evripidou, "A Relationally Complete Visual Query Language for Heterogeneous Data Sources and Pervasive Querying," *Proceedings of the 21st International Conference on Data Engineering*, Apr 2005, pp. 471 – 482
- [8] S.M. Dekleva, "Is natural language querying practical?," *ACM SIGMIS Database*, May 1994, Vol. 25, No. 2, pp. 24 – 36
- [9] H. Direen, C. Brandin, M. Jones, C. Hedgpeth, D. Shin, "Knowledge management through a fully extensible, schema independent, XML database," *Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Oct 2001, Vol. 4, pp. 3676 – 3679