XML modeling of miRNA

Sarah Lopez and Z. Chen
Department of Computer Science
University of Nebraska at Omaha
Omaha, NE 68182-0500, USA
zchen@mail.unomaha.edu

1 Introduction

It has been noted that in bioinformatics the development of general purpose applications is difficult and limited to basic tools. This is echoed by the recent development in data mining tools, where the nontrivial gaps between data mining principles and domain-specific applications have been noticed and the need for application-specific data mining systems has drawn much attention. In this paper we report part of our effort in a research project concerning non-coding RNA (ncRNA) genes which produce RNA that does not encode a protein but instead produces a strand of RNA that performs regulatory and control processes [1]. Our objective is twofold: consolidate data in a single database, as well as study important data modeling aspects which can lead to effective data mining tasks in a specific bioinformatics application.

Even within the sub-subject of non-coding RNA (ncRNA), the diversity displayed is immense. Different types of ncRNA have been identified. In our research, we have focused on microRNA, or miRNA, which are genes that are the reverse complement of other genes and are used to regulate or inhibit the expression of the target gene. We have constructed miRNA databases based on three modeling approaches, namely, E-R based relational, XML shredded relational and eXist NXD (Native XML Database). In this paper we focus on the native XML approach.

Because one can create one's own tags and structure in XML, many organizations have started defining XML documents that are domain specific. There are several vertical XML standards developed to support the biological community. The Biopolymer Markup Language (BIOML), the Bioinformatic Sequence Markup Language (BSML), and the Genome Annotation Markup Elements (GAME) are just a few of them. These markups, while highly useful, continue the genetic community's bias towards DNA by not including all information that could be associated with an RNA gene. As a result, the RNAML syntax was developed [2].

2 XML modeling through RNAML

Vertical standards such as RNAML serve two purposes: a data exchange format description as XML was originally developed to support; and an information model of the subject material (11). Modeling data with XML's hierarchical structure is more intuitive and often reflects the real world better than tables and columns or entities and relationships do. Some of the advantages modeling information with XML are:

- Heterogeneity – different documents can contain different data fields.
- Extensibility – new types of data can be added as discovered.
- Flexibility –data fields can vary in size and configuration from document to document.
- Self-describing – all data elements are labeled.
- Informationally complete - the context of the data is in the same document as the data itself. The context for the data in a RDBMS is often not intuitive from the table structure and must be looked up in the metadata repository.

These advantages are especially useful to the biosciences. The main problem with bio-databases is the quality, not the quantity, of data. Traditional RDBMS systems are very good at handling large amounts of data. What they still struggle with is incomplete or complex data. XML information modeling solves this problem with its extensibility and flexibility. An RNAML document can contain multiple sequence elements, for example. If the pre-cursor RNA has not been established yet, the document can only have a mature sequence element. No structure has to be reserved in the document for the pre-cursor sequence, as in a relational design.

Extensibility is a great advantage to a field where the restructuring of ideas is quite common. This new paradigm demanded a new structure for modeling information. Unlike the case of a relational schema built from an ER diagram, where the new entities and relationships would cause a re-write of the diagram and a major schema change in any resultant relational databases, using XML new information can be handled in a more flexible manner. In general there are two ways to handle it, namely, to add new elements into the existing XML schema, or to design a new document definition. In the case of the latter, XSLT can be used to translate data from the old format to the new one.

XML modeling makes use of four different features of XML structure:

- Data elements – actual data that needs a context
- Tags – supply the context and scope of the data
- Attributes – provide metadata and interpretation instructions for the entire tag
- Hierarchy – structures the relationships among data elements

These four features must be used correctly to create a good model. As with entity-relationship modeling, information modeling with XML comes with its own rules and guidelines on what a "good" model is. These rules impact the structure of the document, which in turn impact the design and performance of any database based on it. When analyzing the document to determine the quality of the design, the first step is to inspect the data elements to ensure they are truly data in need of a context, not metadata. The tags defining the data should clearly define the context and apply to all data and tags in its scope. One of the most common mistakes in XML Information Modeling (XIM) is using attributes to store data. To remedy this, one should inspect each attribute to make sure it corresponds to the scope of the tag, and does not include data.

Once the document has been analyzed at the individual tag level, or in small groupings, we must look at the "paths" as a whole, to make sure we are capturing everything correctly, without duplication or extraneous levels. One way to do this is to create a context listing, or flattening, of the document by taking each data element and writing the complete tag path. These sentences can be re-written in English to ensure that each path makes sense and is contextually complete. We should not have to go to a different path to complete the context for a data element.

A scan of the data elements verifies that each of them is a piece of data, un-interpretable without some context, satisfying the first rule of XIM. Next we look at each context sentence, making sure the context accurately and complete identifies the data element and that the attributes are correct (metadata, not data, and apply to the entire scope of the tag).

The adjusted RNAML DTD provides both a structure and a context to store miRNA data. But having a document structure is not enough. We need a way to manage and query the data contained within the documents. There are at least two possible ways to store a collection of XML documents: transfer the elements into tables in a RDBMS, or use a database designed specifically for storing whole XML documents.

3. Developing eXist for NXD development

We have chosen eXist as our native XML database. eXist (http://exist-db.org) is an open source native XML database. While eXist deploys with a GUI for data administration, data management, and ad hoc queries (XPath or XQuery) eXist also offers a wide array of application programming interfaces with the database such as direct HTTP access from a browser, XML-RPC, SOAP, WebDAV, and support for the XML:DB Java API. XML:DB has been chosen.

eXist models an XML document as a variation of a complete tree. A complete n-ary tree is where every node at every depth has n children. However, eXist relaxed this requirment so that the number of children nodes a node can have depends on the maximum number of children a sibling node has. The maximum number of children nodes for a given level is called the size of that level. If a node has less children than the size, "spare" empty children nodes are inserted. Each node is then labeled in document order The size of the node is recomputed for every level dynamically – the overhead of this is outweighed by the benefits conferred by not forcing each node to have the same number of children. Since this follows the DOM model, even the data is treated as a separate node – a leaf node.

A benefit of this numbering scheme over others that have been proposed is the capability of being able to calculate any "relative" to the selected node dynamically. This saves space, as pointers do not have to be stored, and processing time, as tree-traversal does not have to take place. (size_i = size of the level node_i occurs in)

$\text{Parent}(\text{node}_i) = [\ (\text{node}_i - 2)/\text{size}_i] + 1$

$\text{First Child}(\text{node}_i) = [\text{size}_i\ (\text{node}_i - 1)] + 2$

To access these numbered nodes, the miRNA collection uses a system of three indices. These indices minimize the amount of actual node accesses that must take place.

- Dom.dbx: the main storage area of nodes in the eXist architecture, it associates the internal unique node ids to the actual nodes.
- Elements.dbx: an index consisting of element and attribute names as a key. Each key points to an array that contains the node identifier of each node corresponding to that key. Using this index, a query can come up with a list of all needed node ide, then use Dom.dbx to fetch the actual nodes.
- Words.dbx: an inverted word look up index used to take advantage of the fact that XML documents can be unstructured data. This is a powerful capability not fully realized yet by RDBMS. This index can be tailored to ignore sections of the document.

Each index is an index for the whole *collection* of documents. This collection-coverage is more efficient when searching across multiple documents, a common usage of XML DBs.
. Data of XML format in RNAML syntax is used to store the miRNA source data in a native XMLDB. After creating an ICD, we have developed application programs to map from the source files to a DOM document following the RNAML format. Once a sequence's data was in RNAML format, the documents are stored in a collection called miRNA using the XML:DB API. Once that was done, all that was left was to create and tweak the indices. The major burden of design with native XML databases occurs during the DTD design.

4 Implementation and comparative studies

We have constructed miRNA databases based on native XML and two other modeling approaches, namely, E-R based relational and XML shredded relational. After the implementation of these three approaches, a number of queries were further conducted to compare the performance of these different models. For the same original source data, we measured several factors such as overall storage size, speed for query answering, etc., for each implementation. Based on our experiments, we conclude that a native XML database does the best job at providing geneticists a flexible and extensible storage solution for sequence data. We have also obtained other useful observations as well. Since efficient query processing and data mining tasks depend on effective data modeling, experience and lessons learned from this research shed important insight for our future research.

References

1. Meier, W. (2003) eXist: An Open Source Native XML Database. *Web, Web-Services, and Database Systems. NODE 2002 web and Database Related Workshops.* 2593.
2. Waugh, A., Gendron, P., Altman, R., Brown, J., Case, D., Gautheret, D., Harvey, S., Leontis, N. Westbrook, J., Westhof, E., Zuker, M., Major, F. (2002) RNAML: A Standard Syntax for exchanging RNA information. *RNA*, **8,** 707-717.