THE UNIVERSITY OF BRITISH COLUMBIA

SQPATH:
A Combined SQL-XPATH Query System for RNAML Data

Instructor Dr. V.S. Lakshmanan

Chita	Christian	95365995	cchita@cs.ub.ca
Patel	Rashesh	70615026	rashesh_patel25@yahoo.ca
Yang	Jinmei	27014026	jinmeiyang@hotmail.com

Wednesday, Apr. 27th, 2004

Technical Report Number TR-2004-13

Abstract:

RNA secondary structure prediction has become a major bioinformatics research area, since it could be inferred that all functions of a single-stranded RNA are influenced by its secondary structure [29]. Progress in this field has been hindered, among other things, by the lack of a unified repository for RNA informatics data exchange, and by the lack of a standardized file format. We propose to advance the cause for such a centralized RNA database, and to look at what would be the fastest query approach, should one exist: to store the indexes in a relational table, and use SQL to narrow the set of potential answers to only the matching files, prior to performing XPATH on the RNAML file itself, or to store the indexes at the highest (i.e. first) level of an XML file, and use XPATH exclusively. We have found that storing the indexes in a relational table and using both SQL and XPATH is faster by at least one order of magnitude than storing the indexes at the 1st level of an XML file and using XPATH only. Furthermore, the discrepancy between the speeds of the two query methods increases with the number of files. We describe system we have build to test our hypothesis, our testing procedure and results, and explore avenues that will allow us to generalize our results to other XML databases.

1. Problem Presentation

On April 14, 2003, The International Human Genome Sequencing Consortium announced the successful completion of the Human Genome Project – the sequencing of the 3 billion DNA letters in the human genome [1]. This development has sparked a renewed interest in RNA-Informatics – field of Information Science whose main objective is to design software tools to compute RNA sequence alignments, and to predict RNA secondary and 3-D structures [2], [3]. This exponential increase in RNA-Informatics' activities resulted in a proportional increase in the amount of RNA-data that was generated, stored, and utilized as input to various software programs.

Unfortunately, for a significant period of time, this RNA-data did not follow a unified format: programs will read input files and produce output files that were unique to the respective programs, thus creating significant obstacles in the way of exchanging this data between programs or research groups [11]. To answer this need, a group of RNA scientists met in 1998 and 1999 ([11]) and proposed the introduction of an unified syntax, based on the XML markup language: the RNAML syntax.

Nonetheless, despite its potential for facilitating the exchange of RNA-data, the research community is slow to adopt this standard. We propose to further the cause of RNAML adoption by exposing the merits of a centralized RNAML database. In particular, we assume the existence of such a database (arguments in its favour will be introduced in Section 2: Motivation), and propose to look at what would be the fastest method to query such a database. This research question is sparked by the specific, yet various needs researchers in the field have with respect to the information they require form the RNAML data file.

We have purposely targeted the needs of a specific RNA informatics research group: the BETA laboratory at the University of British Columbia [4]. In so doing, we have inferred a subset of RNA molecule attributes that constitute the terms most likely to be queried while using our proposed RNAML database. Consequently, our project would attempt to answer the following question:

Proposed Research Question:
Given the benefits for the RNA informatics research community at large of maintaining an RNAML data base, what would be the fastest query method: **store the indexes in a relational table, and use SQL to narrow the set of potential answers to only the matching files, prior to performing XPATH on the RNAML file itself,** ***OR*** **store the indexes at the highest (i.e. first) level of an XML file, and use XPATH exclusively *?***

Results:
Storing the indexes in a relational table and using both SQL and XPATH is faster by at least one order of magnitude than storing the indexes at the 1st level of an XML file and using XPATH only. Furthermore, the discrepancy between the speeds of the two query methods increases with the number of files.

2. Motivation

Ribonucleic acid (RNA) is one of the two types of nucleic acids found in living organisms (the other one being DNA – deoxyribonucleic acid). From a high level perspective, DNA's only role is the storage of genetic information [5], whereas RNA is further divided in several types, fulfilling several functions, as follows [6] (Please see Table 1 bellow):

RNA type	*Function*
Ribosomal RNA (rRNA)	forms complexes with proteins to create ribosomes, the site of protein synthesis within the cytoplasm of the cell
Messenger RNA (mRNA)	carries the information recorded in DNA from the nucleus to the cytoplasm of the cell
Small nuclear RNA (snRNA)	involved in pre-mRNA splicing
Heterogenous nuclear RNA (hnRNA)	is the primary transcript from the eukaryotic enzyme, RNA polymerase II. hnRNA is the precursor of all mRNA often called "pre-mRNA", prior to the removal of introns
Small nucleolar RNA (snoRNA)	found in the cell's nucleolus where it processes and methylates rRNA
Transfer RNA (tRNA)	carries amino acids to nascent polypeptide chains synthesised on the ribosomes
Table 1:	RNA types and corresponding funtions

2.1. Terminology:

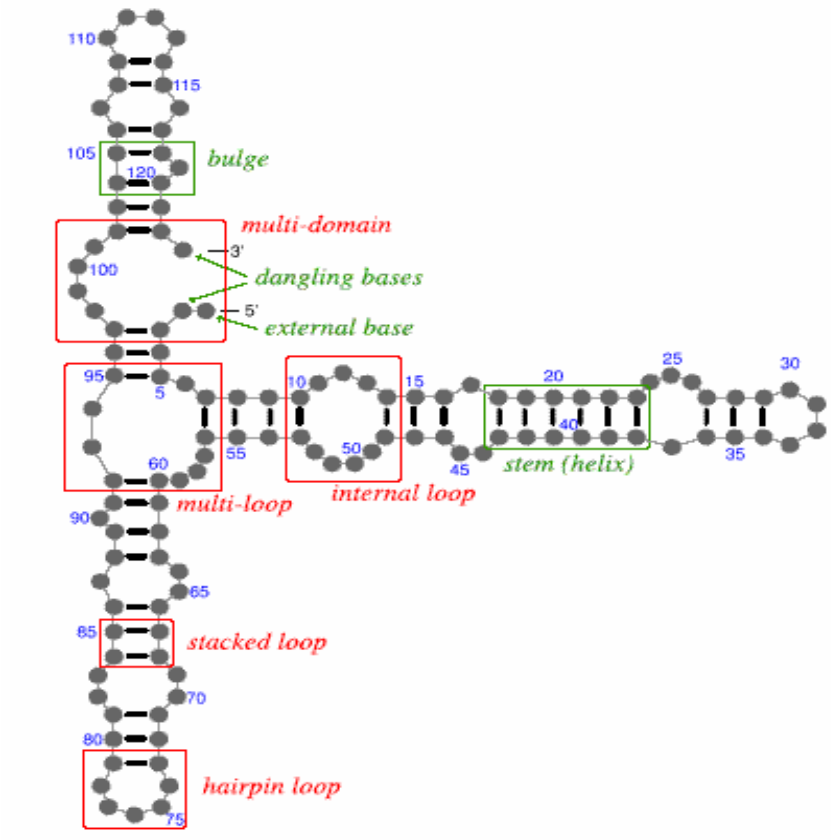
2.1.1. RNA composition

DNA and RNA molecules (also known as nucleic acids) are composed of sequences of four types of nucleotides or bases: Adenine (A), Cytosine (C), Guanine (G) and Thymine (T) for DNA or Uracil (U) for RNA. Furthermore, RNA molecules are continuous single strands that when folded back on themselves form secondary structures consisting of double stranded segments and other complex tertiary structures [7]. The double stranded sections are held together by hydrogen bonds between the bases.

2.1.2. Elementary Structure Composition:

Figure 1 bellow introduces the main components of the RNA elementary structure composition [8].

Figure 1: Example of a pseudoknot-free secondary structure containing all elementary structures [8].



2.2. Importance of RNA Secondary Structure

The structures formed by the folding of RNA segments in a living cell have been shown to be important for regulatory, catalytic, or structural roles. Others have been shown not to participate in certain specific roles [7]. In addition, an improvement in the RNA secondary structure prediction will undoubtedly led to a better understanding of life at the micro-molecular level. In turn, this is likely to trigger a domino effect: an increase in understanding of *k* concepts at the *l* sublevels will lead to a better understanding of *m* concepts at the *(l+1)* levels. This in turn, could trigger a better understanding of the currently unknown virus activity lethal mechanisms.

2.3. RNA Informatics

Several algorithms for predicting the secondary structure formed by folding RNA-segments have been proposed: some exploit the minimum free-energy concept [10], while others use statistical methods (quasi-Monte-Carlo, genetic algorithms, stochastic grammars) [7]. Fact is, using biophysics for the prediction work is tedious, time-consuming, and error-prone – hence, the prediction work-load has been shifted to the bioinformatics field.

2.3.1. RNAML motivation

With several prediction methods currently researched, we can easily imagine that there exist a large number of software programs manipulating and exchanging the RNA data of interest. We can thus infer two types of problems [11]. First, would be the pipelining of data among the programs ran within the same research group. That is, it would be a too strong inference to be made to assume that all programs belonging to a single group will be using the same format for their input files, and this is especially a problem since several programs require pipelining their execution. At a higher level of abstraction, this problem is even more pertinent with respect to exchanging data among different research groups, especially when the groups reside in different countries.

As pointed out in [11], there exist several initiatives to provide a standardized language for exchanging biological data. Namely, the BIOML (BIOpolymer Markup Language), the BSML (Bioinformatic Sequence Markup Language), and the CORBA Bio effort. However, none of these initiatives deals with RNA specific data, as their syntax has no provisions for RNA structure information.

Henceforth, a group of RNA scientists met in 1998 and 1999 ([11]), and advanced a proposal for the adoption by the scientific community of a standardized file format and syntax for representing RNA structure information. To facilitate the exchange of such data, the skeleton for this format was to follow the design principles behind the Extensible Markup Language 1.0 (XML [12]), hence the name for the new format – RNAML. This would provide, among other things ([11]), the advantage of allowing for extension of the syntax without breaking programs understanding only the prior version(s).

2.3.2. RNAML centralized DB motivation

As already mentioned in Section 2.2, several research groups around the world are working on the secondary structure prediction problem. Given that this problem presents several angles, groups tend to specialize in solving one particular aspect of it. Furthermore, there are several groups specializing in the collection of RNA structure data only. This has unfortunately led to each group maintaining and augmenting a core repository of files tailored to serve the needs of the respective research group. In particular, groups specialize either in the study of a specific RNA type, and/or in work with a specific file format. For example, [13] is specialized in *rRNA*, and in addition, the data is in diagram format only. [14, 15] are specialized in *tRNA*, and [15] is in diagram format only, whereas [16, 17] are specialized in *snoRNA* only.

We would like to specifically emphasize the inconvenience a researcher working on the secondary structure prediction problem would experience with the input data in diagram format. This implies that the researcher in question has to *manually reconstruct* the sequence from the diagram – a tedious and highly error prone process. In addition, this operation has to be performed for every RNA strand of interest, and some may reach an 10^2, and even 10^3 order of magnitude in sequence length. Alternatively, if all groups were to use a unified format, namely RNAML, the RNA informatics work will see a significant increase in the streamlining of data processing. This is especially true since, as explained in [11], the RNAML file can be appended with the information relevant to all research groups, and each group will only have to parse the information of local interest.

In addition, the authors of this paper would like to advance the case for the instantiation of a centralized RNAML database repository, with 24/7 access unbounded by any fees, membership, or university affiliation. We would see this as beneficial especially in the light of the following fact: currently, not all the information a certain group possesses is readily made available to the community at large. One of the reasons behind this would be that the process is time consuming, and requires time and resources without necessarily providing a benefit to the initiator. We would argue that with a central repository, groups will only check-in their specific subset of files, but will have access to all the files all the other groups have contributed. Once specific groups no longer have to worry about maintaining their local repository, and in addition will have access to the pool of files from every other group around the world – or at least from North-America, we expect them to become active participants in the growth of the central repository.

3. Related Work

From our research, we can infer that the current efforts seem to gravitate around the problem of SQL-XML Translation. Namely, most initiatives seem to target the problems of how to store XML data in a relational table (in other words, how to "SQL-ize" XML data), and how to translate and SQL query in an XML contra part and vice versa. In particular, we have seen this trend in both the industry and the academia, as outlined in the following two subsections.

3.1. Current Approaches

3.1.1. Industry

One such initiative ([18]), offers to the paying public a course where

> […] you'll learn how to use standard XML-based techniques to query and modify data, and how to bulk-load large XML documents into SQL Server tables. The course combines detailed, example-based instruction on techniques with clear explanations of the underlying concepts. ([18])

The class will teach, among other things, how to use FOR XML clause to return data from an SQL server as XML, or how to create XML Views that will expose SQL Server data.

We see this minor example as an instantiation of the current industry trend to provide *proprietary* extensions for using XML with relational databases [19]. Unarguably, this will also mean little, if any interoperability between two systems written by different companies. In addition, we can see the industry's emphasis on the implementation of one, or both of the following query approaches: SQL/XML or XQuery. The latter builds on the fact that it provides additional functionality in those cases where native XML programming, or XML views of the relational data are desired, whereas the former builds on the familiarity that SQL programmers already have with the language [19]. We note here that SQL/XML is entirely different from Microsoft's SQLXML; it is an extension of SQL part of ANSI/ISO SQL 2003 ([19, 20]).

We see this as a concretization of the general industry effort to provide means for enabling XML applications to use relational data, and for transforming the results of SQL queries into XML data.

3.1.2. Academia

In the academia, we have observed the same direction; namely, the transformation of SQL queries into ones that can be applied to XML data.

In particular, [21] addresses the problem of efficiently constructing materialized XML views of relational databases. In particular, this approach utilizes a middle-ware system for the XML view specification, for sending SQL queries to the relational database, and integrating the resulting tuple with XML tags.

The work in [22] focuses on the problem of efficiently structuring and tagging data from one or more tables as a hierarchical XML document. In particular, the paper suggests the use of scalar engines and aggregate functions to construct complex XML documents inside the relational engine. Next, [23] proposes to focus on the efficiency of the SQL queries generated by the XML-Query-to-SQL-Query translation process.

Finally, [24] suggests a tool that will take as input an OoAnalysis using UML, which will be stored as XML data, and finally generate SQL statements to create a relational database. From this relational schema, an SQL-to-XML mapping will be derived and, when used with SQL Server 2000, will allow for querying of data using XPATH rather than SQL, and the output of data in XML format.

4. Approach and Methodology

What we suggest is as follows. We do not see a need for combining the two worlds: SQL and XML; at least not at the level of combining queries. Our approach is to simplify as much as possible any given

approach that will correctly answer our starting point design goal:

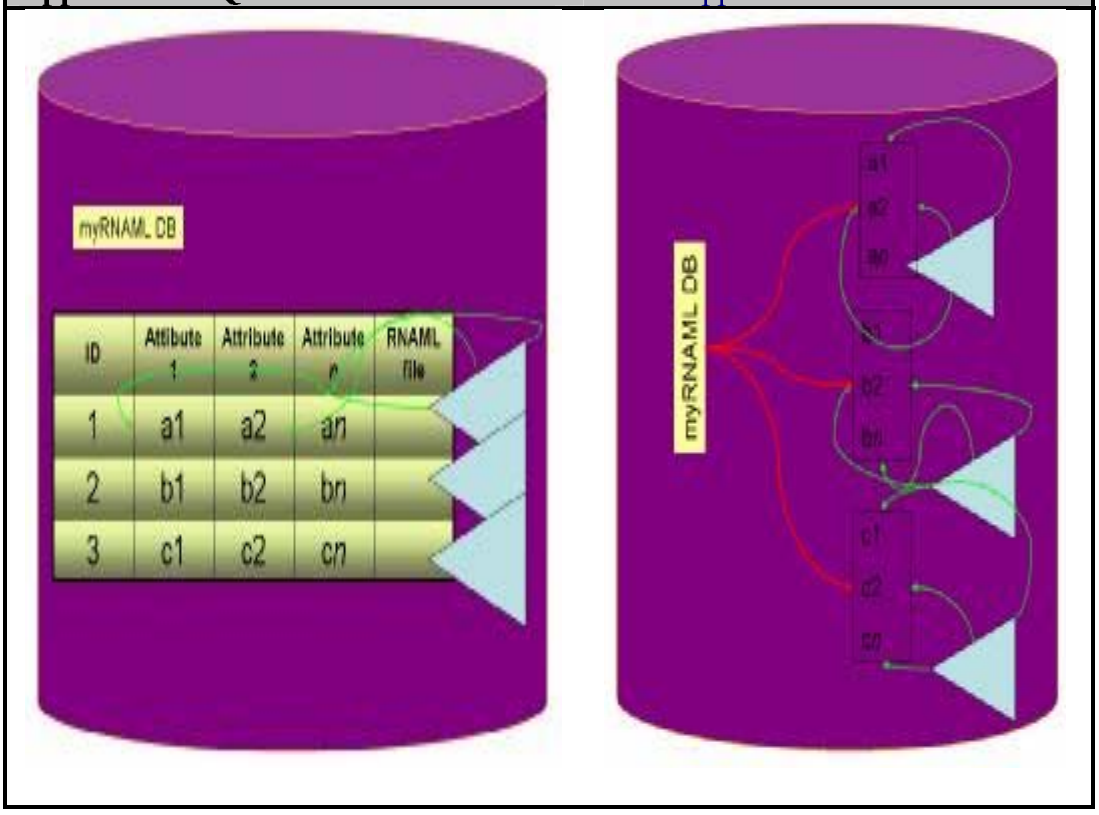
> Given an RNAML database with potentially thousands of files of various sizes, what would be the most efficient way to return a subset of files matching the user-input query parameters?

Given our internal knowledge of the most likely query terms ([25]), we propose to implement, analyze, and compare the execution speed of the following two approaches:

1) A database where the indices are stored in a relational table, and that will primarily use SQL to query this table. Should some, or all of the query terms reside exclusively in the RNAML file(s), then XPATH will be performed as needed.
2) Extract the same indices from each RNAML file, but store them at the first level of a separate XML file containing, in addition, the RNAML files themselves.

Schematically, what we suggest is as follows (please see Figure 2):

Figure 2:	*DB design approach*
Approach 1: SQL indices	Approach 2: XML indices



4.1. Software Used

We decided to use *Java* as the programming language for our implementation. This was mainly due to its cross-platform abilities, and to its GUI building facilities. With respect to the database server, we decided to use *Oracle9i Release 2 (9.2).*

This decision was taken as follows. Our main option would have been using *mySQL* on the Department's network. However, since we intended our project to be a performance measurement and comparison, we were looking for a product that will provide built-in capabilities for both SQL and XPATH. We were confident *Oracle* will deliver the desired functionality, but were unable to infer how we would insure the same within *mySQL.*

In addition, we were determined on using the latest *Oracle* version, because it affords XML schema validation – functionality not present in earlier versions. This latter requirement was the reason why we did not use the version of *Oracle* installed on the Department's servers – we opted for downloading the latest *Oracle* version on a personal machine (downloading and installing software is not permitted on the Department's machines).

4.2. Hardware Used

As mentioned in Section 4.1, we have used personal computers for implementing our project.
The Operating System was *Windows XP Professional*, and the H/W characteristics are as follows (please see Table 2 bellow):

Table 2:	*Hardware Configuration*
CPU:	Intel Pentium M processor 1500 MHz
Bus Speed:	400 MHz
Motherboard:	IBM 2722GCU
System:	IBM 2722GCU
BIOS:	IBM 1PET50WW (1.18)
Memory:	512 MB

4.3. Dataset

As mentioned in Section 2.3.2., an RNAML database does not currently exist; it is one of our project's secondary objectives to push for the instantiation of such a centralized repository. Henceforth, we resolved to the generation of synthetic data.

4.3.1. Generation

The generation of all RNAML files in our input databases was (human generated) random, in the following manner.

1) The initial RNA molecule is randomly produced via a call to a C++ executable file: *simfold* [8].

Simfold represents the software implementation of a known algorithm [26], and its accuracy was shown ([27]) to be 73 % -- accuracy is measured as the number of correctly predicted base pairs over the total number o base pairs in the experimentally determined structure.

2) A Perl script will utilize *simfold* to generate the predicted secondary structure of the initial RNA molecule. The parameters required are as follows:

Usage: genRNAML.pl <min_len> <max_len> <RNAML_filename> <id>
<min_len>: the minimum length of the sequence (between 1 and 3000)
<max_len>: the maximum length of the sequence (between 1 and 3000)
<RNAML_filename>: the filename of the RNAML file
<id>: the molecule ID, i.e. the primary key in the relational table

We have varied the size and the sequence of our generated files by using three additional Perl scripts that will call the script mentioned in "2" above with a fixed lower bound, and a random upper bound – for the sequence length parameter. Thus, a sample shell output for the generation of files (please see Section 4.3.2 Composition for further details regarding the database composition) will look as follows (please see Table 3 bellow):

Table 3: Small DB generation shell output

```
[okanagan]$ perl -w genSmallDB.pl
low 50 hi 96
low 50 hi 186
low 50 hi 161
low 50 hi 70
low 50 hi 93
low 50 hi 186
low 50 hi 143
low 50 hi 93
low 50 hi 159
low 50 hi 114
low 50 hi 188
low 50 hi 84
low 50 hi 58
low 50 hi 101
low 50 hi 103
low 50 hi 162
low 50 hi 79
low 50 hi 147
low 50 hi 189
…
…
```

The generation of the other two database types followed a procedure similar to the one outlined in Table 3, except for the values of the low and high bounds.

4.3.2. Composition

In order to diversify and solidify our conclusions, we have decided upon the generation of three main database types to be used as input for our implementation:

1) *Large DB*: this database contains 22 RNAML files, with sizes ranging from 231 KB – the smallest, to 247 KB – the largest, for a total size (on disk) of 5.19 MB.

2) *Misc DB*: this database contains 287 RNAML files, for a total size (on disk) of 6.26 MB, with sizes as follows:
   a. 260 small files, with sizes ranging from 2 KB – the smallest, to 11 KB – the largest.
   b. 20 medium-sized files, with sizes ranging from 116 KB – the smallest, to 139 KB – the largest.
   c. 7 large files, with sizes ranging from 234 KB – the smallest, to 250 KB – the largest.

3) *Small DB*: this database contains 900 RNAML files, with sizes ranging from 3 KB – the smallest, to 17 KB – the largest, for a total size (on disk) of 9.05 MB.

Please see Appendix A for a sample of the final XML file obtained from two RNAML files, together with their respective indices.

4.3.3. Total DB Size

We have limited the size of our databases to approximately *5-8 MB* each. This was not motivated by some voluntary implementation limitation from our part. Rather, it was a H/W limitation in the following sense: as mentioned in Section 4.2 Hardware Used, we only had 512 MB available memory on our test machine. Or, once we started the Oracle server, this process in itself will exhaust the entire available memory. Thus, increasing the size of the databases used would have led to significant paging, with no obvious benefit: given the significant discrepancy between the performance times of the two compared methods, we have no reasons to believe that an increased DB size will dramatically change this difference, all other variables held constant.

This was especially true since we were interested in studying the best-case scenario for both types (SQL and XML) of databases – the specific case where the entire data fits into the memory. The performance of the two approaches with disk resilient data was not within the scope of our project, because we did not want (H/W) disk performance to be an added factor in our comparison.

4.4. Query Methodology

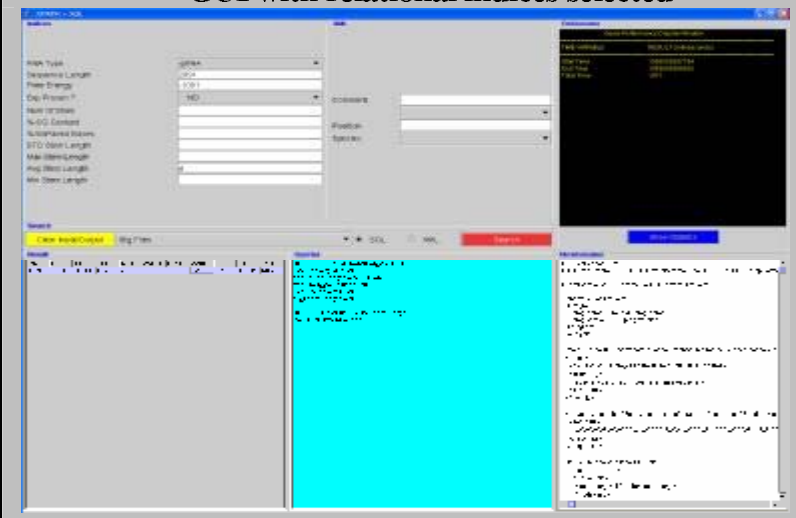
First, we would like to mention that from the user interaction point of view, the entire query process is taking place via a Java GUI interface; any communication with the Oracle server is transparent to the user (with the natural exception that the user must first manually connect the machine running our implementation with an Oracle server).

Our GUI will provide the user with the following functionality (please see Figures3, 4 bellow):

1) Ability to choose the input database size (small, mix, large), as outlined in Section 4.3.2
2) Ability to switch, for each database size, between the type with the indices stored in the relational table and the type with the indices extracted at the first level of an XML file
3) Ability to view the resulting query
4) Ability to observe the per-query results
5) Ability to observe aggregate results

Figure 3:

GUI with relational indices selected



GUI with 1st level XML indices selected

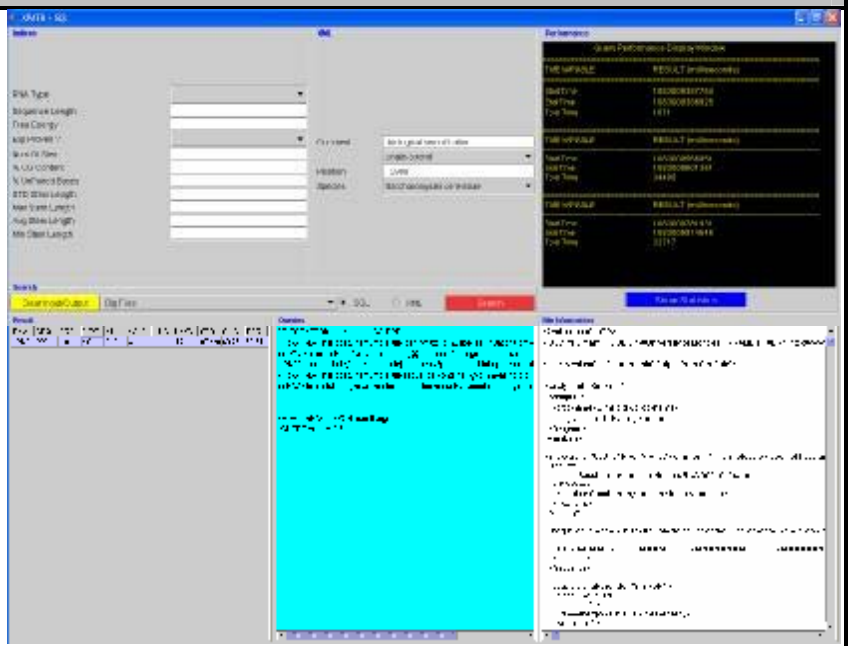
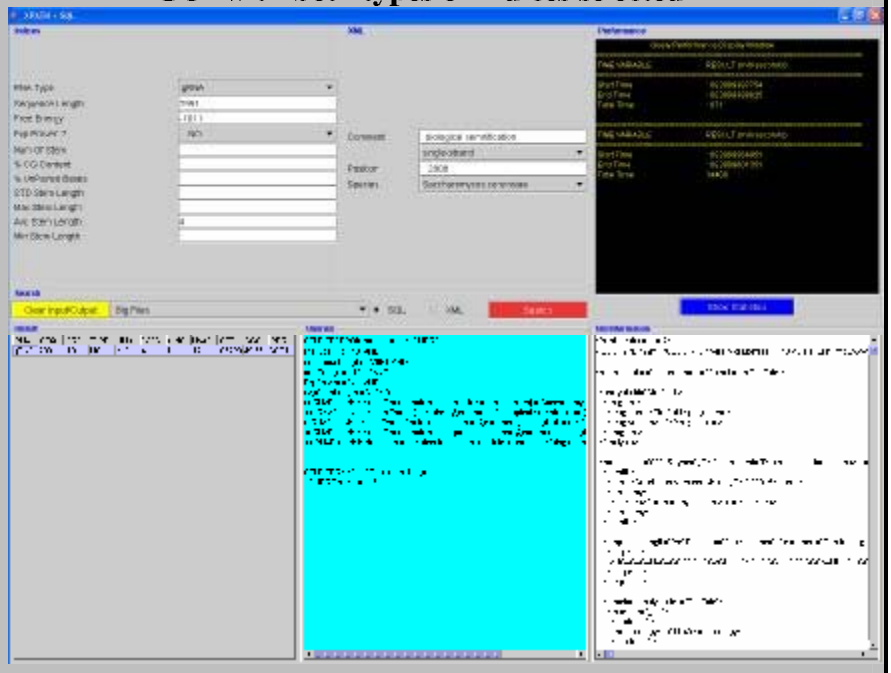
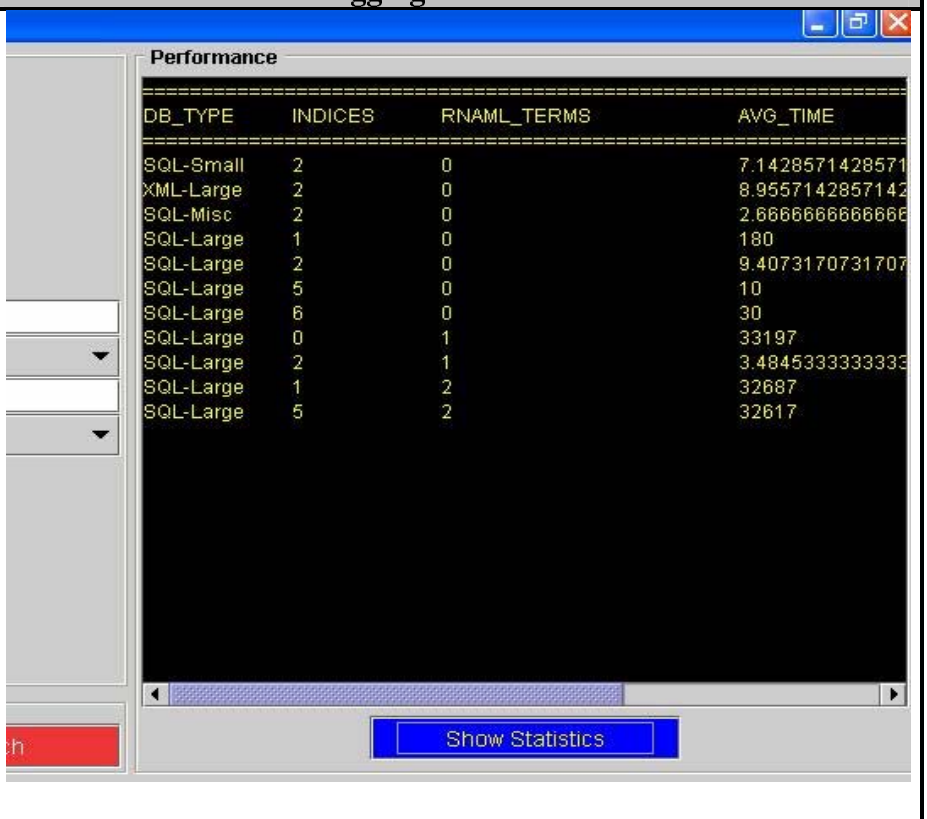


Figure 4:

GUI with both types of indices selected



Aggregate Results:



4.4.1. Indices stored in relational table

For the case when the extracted indices are residing in the relational table, the Oracle server handles the entire query process. In particular, the query steps are as follows:

1) An initial SQL query is performed, using as terms the values the user specified in the upper LHS panel of the GUI. If there are no additional values specified in the upper RHS panel, the query process ends here and the retrieved files, if any, are returned in the lower LHS panel (please see Table 4 bellow).

Table 4: sample query with indices only as terms

```
SELECT * FROM rnamlLarge WHERE
rnaType = 'gRNA' AND
sequenceLength = '2991' AND
freeEnergy = '-1011' AND
ExpProven = 'NO' AND
avgStemLength = '4'
```

2) If there are values present in the upper RHS panel, then an additional XPATH query becomes part of the SQL statement, and only those files matching the conditions specified by both queries are returned to the user, if any (please see Table 5 bellow).

Table 5: sample query with mixed terms: indices and RNAML-residing

```
SELECT * FROM rnamlLarge r  WHERE
rnaType = 'gRNA' AND
sequenceLength = '2991' AND
freeEnergy = '-1011' AND
ExpProven = 'NO' AND
avgStemLength = '4' AND
(r.RNAFile.existsNode('/rnaml/molecule/identity/taxonomy/species[.="Saccharomyces
cerevisiae"]')  = 1 ) AND
 (r.RNAFile.existsNode('/rnaml/molecule/structure/model/str-annotation/single-
strand//position[.="2908"]')  = 1  )
```

3) If there are values present only in the upper RHS panel (i.e. values not extracted from the original XML file), then a corresponding XPATH query is inserted in the body of a parent SELECT clause (please see Table 6 bellow).

Table 6: sample query with RNAML-residing terms

```
SELECT * FROM rnamlLarge r  WHERE
(r.RNAFile.existsNode('/rnaml/molecule/identity/taxonomy/species[.="Saccharomyces cerevisiae"]') = 1 ) AND
 (r.RNAFile.existsNode('/rnaml/molecule/structure/model/str-annotation/single-strand//position[.="2908"]') = 1  )
```

In all cases, the query itself is displayed for user examination in a dedicated GUI panel. Likewise, we provide functionality that affords the examination of any given file part of the result set.

4.4.2. Indices part of XML file – 1st level

In the case of XML-only databases, the query procedure is different, in the following manner. Due to the fact that Oracle stores XML files in a relational table, to perform any query on the stored file(s), a first SQL statement must be performed to extract the XML file.

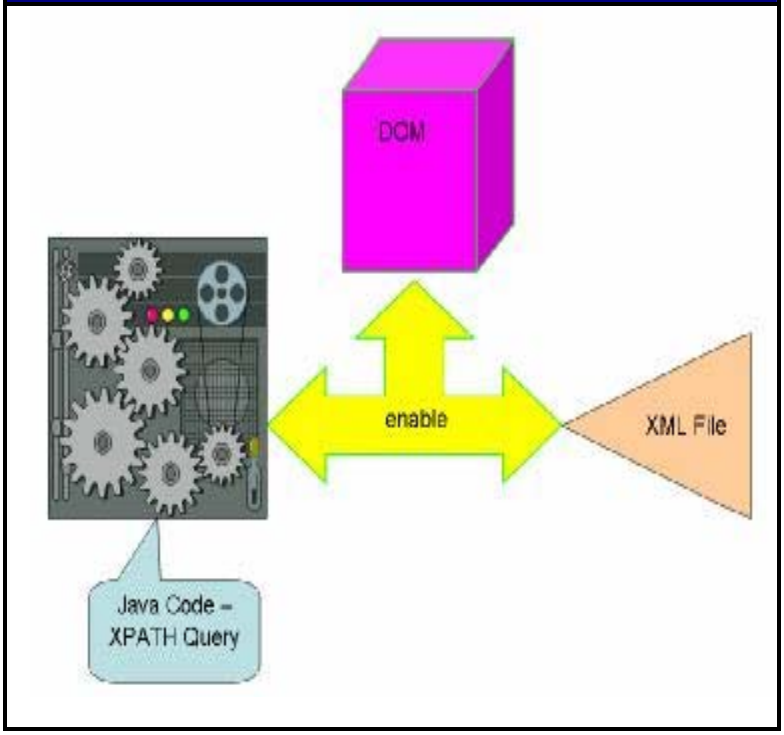
For example, in our case, we have only one XML file storing all the individual RNAML files and their indices. Thus, an initial SQL query would have to be performed, to enable further payload processing of the file. However, all statements of interest following the initial SELECT statement have to be performed on this initial statement's result. Or, the result of the subsequent statements (such as WHERE, GROUP BY -- see Table 7) is some section (i.e. a, or several molecules) part of the initial database file. Nevertheless, the result of the parent SQL query can only be a tuple (or several), or a given projection on the result tuple(s). This contradicts our goal of retrieving only specific sections of the result-tuple itself – SQL does not allow such an operation (please see Table 7 bellow).

Table 7: sample query performed on a single XML file stored in a relational table in Oracle

```
SELECT      * FROM myRNAMLDB
WHERE       cond1, cond2, …, cond n
GROUP BY    x, y, z
```

We have thus resolved to retrieve our RNAML database file from Oracle, and use DOM (Document Object Model, [28]) to perform all relevant XPATH queries. Schematically, DOM enables the following functionality (please see Figure 5). In particular, we used DOM to launch the Java code containing the XPATH queries of interest on our single XML file database.

Figure 5: schematic representation of XML query functionality made possible by DOM



Alternatively, we could have stored each XML database file in a regular directory in the computer's file system, and use DOM in the same manner. We decided against this for the following reasons:

1) Keeping our database files in Oracle will enable us to enforce and validate the XML schema.
2) To insure fair comparison with the database type having indices stored in a relational table: not using Oracle would have freed significant memory to be used in processing the XPATH queries.

Bellow, in Tables 8, 9, and 10 we provide the queries used to retrieve the same files as in Tables 4, 5, and 6 above; the only difference is that in the Tables bellow the input database is a unique XML file, with the indices of interest stored at the 1st level with each individual RNAML file.

Table 8: sample query with indices only as terms

```
SELECT indexedfile FROM largeDB
//largeFiles/*
self::node()[ RNATYPE = 'gRNA' and
SEQUENCELENGTH = '2991' and
FREEENERGY = '-1011' and
EXPPROVEN = 'NO' and
AVGSTEMLENGTH = '4' ]
```

Table 9: sample query with mixed terms: indices and RNAML-residing

```
select INDEXEDFILE from largeDB
//largeFiles/*
self::node()[ RNATYPE = 'gRNA' and
SEQUENCELENGTH = '2991' and
FREEENERGY = '-1011' and
EXPPROVEN = 'NO' and
AVGSTEMLENGTH = '4' ]
self::node()/rnaml/molecule/structure
/model/str-annotation/single-strand//position[ . = "2908"]
self::node()/rnaml/molecule/identity
/taxonomy/species[ . = "Saccharomyces cerevisiae"]
```

Table 10: sample query with RNAML-residing terms

```
select INDEXEDFILE from largeDB
//largeFiles/*
self::node()/rnaml/molecule/structure
/model/str-annotation/single-strand//position[ . = "2908"]
self::node()/rnaml/molecule/identity
/taxonomy/species[ . = "Saccharomyces cerevisiae"]
```

5. Results

In constructing our testing session, we have followed the following algorithm (please see Table 11):

Table 11: testing session algorithm

```
for each (DB) do {
 5 queries of of the following type:

        * type 1: SQL indices only:
                i) queries with _1_ term, "randomly" placed
                ii) queries with _5_ terms, "randomly" placed
                iii) queries with _11_ terms, "randomly" placed

        * type 2: RNAML indices only:
                i) queries with _1_ term, "randomly" placed
                ii) queries with _3_ terms, "randomly" placed

        * type 3: SQL + RNAML  indices:
                i) queries with _1_ SQL term, "randomly" placed AND
                                _3_ RNAML terms
                ii) queries with _5_ SQL terms, "randomly" placed
                                _3_ RNAML terms
                iii) queries with _11_ SQL terms, "randomly" placed AND
                                _3_ RNAML terms

                for each (group of 5) do {
                        2 known to return empty set as answer;
                        3 known to return some answer;
                }
  AND record the times (and the corresponding query for each)
}

Total:          40 queries
```

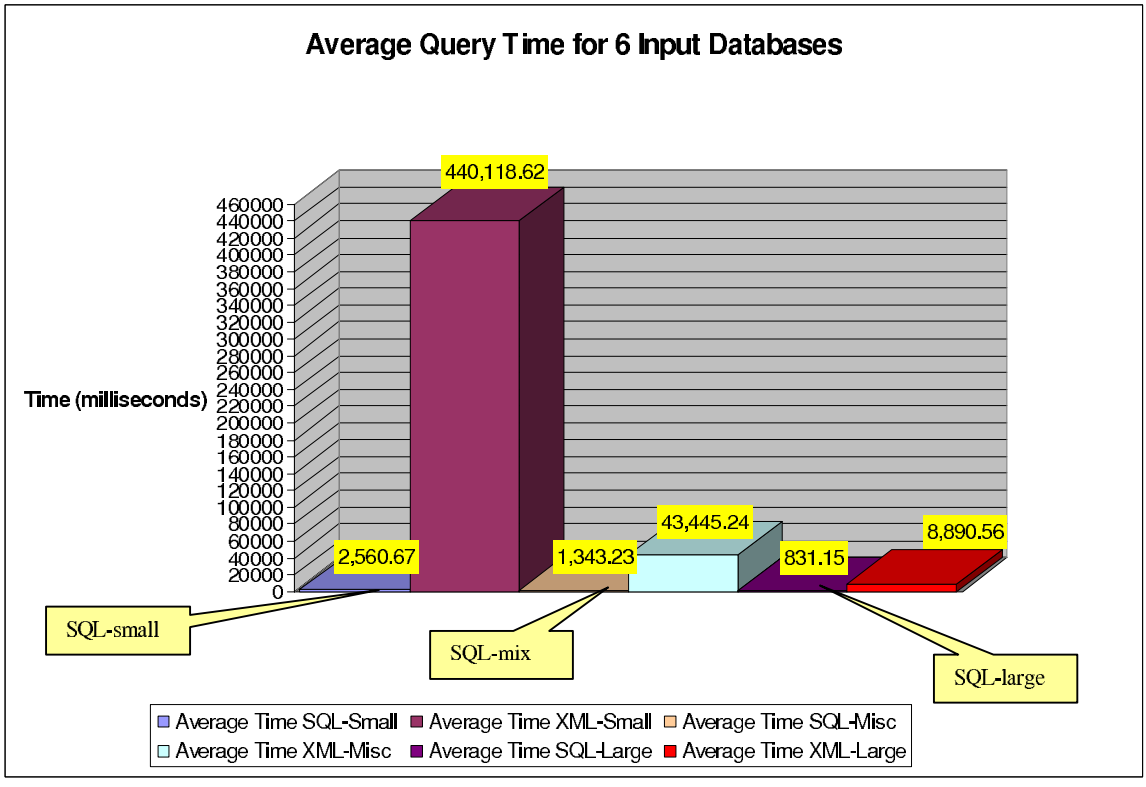
We present, in Table 12 and Figure 6, the aggregate results of the testing session conducted according to Figure 14 above. A detailed version, including which query terms have been used exactly, is included in Appendix B.

Table 12: aggregate testing results for the 6 input databases

DB_TYPE	SQL_INDICES	RNAML_TERMS	AVG_TIME (milliseconds)
SQL-Small	1	0	26
SQL-Small	5	0	10.2
SQL-Small	11	0	10.2
SQL-Small	0	1	1434.2
SQL-Small	0	2	2149
SQL-Small	1	2	4665
SQL-Small	5	2	5456
SQL-Small	11	2	5760.4
SQL-Small	0	3	3535
SQL-Misc	1	0	0
SQL-Misc	2	0	0
SQL-Misc	5	0	0
SQL-Misc	11	0	2
SQL-Misc	0	1	855
SQL-Misc	0	2	2854
SQL-Misc	1	2	2479.6
SQL-Misc	5	2	2776.75
SQL-Misc	6	2	2724
SQL-Misc	11	2	3034.2
SQL-Misc	0	3	50
SQL-Large	1	0	2
SQL-Large	5	0	6
SQL-Large	11	0	2
SQL-Large	0	1	8.88E+02
SQL-Large	0	2	1744
SQL-Large	1	2	1879
SQL-Large	5	2	1735
SQL-Large	11	2	1834.6
SQL-Large	0	3	100
SQL-Large	5	3	121
XML-Small	1	0	351955.8
XML-Small	5	0	181235
XML-Small	11	0	180902.4
XML-Small	0	1	9.66E+05
XML-Small	0	2	724146
XML-Small	1	2	348499.2
XML-Small	5	2	175316.2
XML-Small	11	2	175684.8
XML-Small	0	3	857653
XML-Misc	1	0	56171
XML-Misc	2	0	23664
XML-Misc	5	0	24925.8

XML-Misc	11	0	23568
XML-Misc	0	1	82997.2
XML-Misc	11	1	23343
XML-Misc	0	2	87663.25
XML-Misc	1	2	44568.4
XML-Misc	5	2	24617.75
XML-Misc	6	2	23824
XML-Misc	11	2	23281.5
XML-Misc	0	3	82719
XML-Large	1	0	13287.2
XML-Large	5	0	5854.6
XML-Large	11	0	5311.8
XML-Large	0	1	1.03E+04
XML-Large	0	2	14017
XML-Large	1	2	10076.8
XML-Large	5	2	6046
XML-Large	11	2	5313.6
XML-Large	0	3	12899
XML-Large	5	3	5778

Figure 6: graphic aggregate testing results for the 6 input databases



6. Contributions

The specific contributions of this paper are as follows:

1) We have suggested, and convincingly argued in favor of the instantiation of a central repository of RNAML files to be used and contributed to by the RNA-informatics research community (?)
2) We expose the fact that the current tools for querying databases are either:
    a. Relational data oriented
    b. XML data oriented
    c. Providing translations between the two

    We suggest the implementation of a system that will combine tools from both the relational and the XML domains, while letting each tool work on the data where it performs best. We argue that by extracting indices of interest from XML files, storing them in a relational table, and combining the use of SQL and XPATH we obtain better query performance and scalability (?)
3) We have conducted a comprehensive set of query combinations in the testing phase of our project, and showed that:
    a. Storing the indices in a relational table results in a query time faster by at least one order of magnitude as compared to extracting the indices at the first level of an XML file (?)
    b. This difference becomes overwhelming as the number of individual (RNAML) files in the database increases (?)

7. Approach Novelty

We do claim novelty with respect to our suggestion to instantiate a "Google of RNAML data": a freely available common repository where researchers will contribute and check-out RNAML files for use in their research activities. As mentioned in Section 2.3.1 RNAML Motivation, any given RNAML file can become exponentially complex following its journey among the various RNA informatics research groups; this will not hinder the activities of any particular group, as the XML syntax allows for the parsing of the exact information pertinent to a given research activity. This is in sharp contradiction with the current state of RNA data storage and exchange, as most groups use their own file format, and furthermore, do not always make their files available to other groups.

We do not believe that the concept of extracting indices of interest from an XML file and storing them in a relational table to improve query performance is novel, but we have not come across a similar system while we researched related work for our project.

8. Strengths and Limitations

We infer the following strengths and limitations for our system:

8.1. Strengths

- Our results can be extended to the design of other query systems for XML databases where most of the query terms are likely to be known in advance
- Uses the latest version of Oracle, allowing for XML Schema verification
- Shows that RNAML file format is a proper XML format, conforming to the storage and information retrieval typical for this data format
- Shows that XML is not the panacea of all database query issues
- Thorough testing session taking into account complex query-terms combinations
- Written in Java, and thus platform independent

8.2. Limitations

- Running the Oracle server on the same machine as the client dramatically reduces performance
- The Oracle server is a proprietary software product. This entails various dependency problems, and in addition, has poor XPATH support
- Oracle has poor XML support, in the sense that the XML file(s) is actually stored in a relational table, and thus requires a parent SELECT clause to gain access to the stored data

9. Future Work

For the future development of our project, we foresee the following steps:

1) Segregate the Oracle server from the client application, such that the processes dedicated to running Oracle no longer constitute a burden on the memory of the client machine
2) Generate databases of at least one order of magnitude larger (than our current 5 MB size), and verify whether our results still apply or if not, where and why do they differ
3) Migrate from Oracle to a product that will provide full XML support, such that we can launch XPATH queries directly on the stored file(s). Preferably, this product should be open source
4) Research the topic of what could improve query time for the case of those databases containing a significant number of files
5) Verify whether the use of XQUERY provides an improved query time for the case of the most complex queries – the ones using both indexed and non-indexed terms
6) Study how the lessons learned in this specific database environment can be extended to other types of XML databases where the query terms are likely to be known in advance

10. References

[1] http://www.genome.gov/11006929

[2] http://www-lbit.iro.umontreal.ca/RNA_Links/RNA.shtml

[3] http://www.uga.edu/RNA-Informatics/

[4] http://www.cs.ubc.ca/labs/beta/

[5] http://www.biores-irl.ie/biozone/genes3.html

[6] http://www.uq.edu.au/vdu/DNA_RNAfunction.htm

[7] Deogun S., Jitender., Donis, Ruben., Komina, Olga., Ma, Fangrui: RNA Secondary Structure Prediction with Simple Pseudoknots. APBC 2004, Dunedin, New Zealand. Conferences in Reasearch and Practice in Information Technology, Vol. 29.

[8] Andronescu, M.: Algorithms for predicting the secondary structure of pairs and combinatorial sets of nucleic acid strands. M. Sc. Thesis, The University of British Columbia.

[9] http://www.biochem.ucl.ac.uk/~shepherd/spred_tutorial/ss-intro.html

[10] http://www.santafe.edu/~pth/rna.html

[11] Waugh, Allison., Gendron, Patrick., Altman, Russ., Brown W., James., Case, David., Gautheret, Daniel., Harvey C., Stephen., Leontis, Neocles., Westbrook, John., Westhof, Eric., Zuker, Michael., Major, Francois. RNAML: A standard syntax for exchanging RNA information. RNA (2002), 8:707-717. Cambridge University Press.

[12] http://www.w3.org/TR/2000/REC-xml-20001006

[13] http://oberon.fvms.ugent.be:8080/rRNA/index.html

[14] http://rna.wustl.edu/tRNAdb/

[15] http://mamit-trna.u-strasbg.fr/2DStructures.html

[16] http://rna.wustl.edu/snoRNAdb/

[17] http://www.bio.umass.edu/biochem/rna-sequence/Yeast_snoRNA_Database/snoRNA_DataBase.html

[18] http://www.planetlearn.com/apsqlserandx.html

[19] http://www.stylusstudio.com/whitepapers/sqlxml.pdf

[20] http://builder.com.com/5100-6387-1044928.html

[21] Fernandez, Mary., Morishima, Atsuyuki., Suciu, Dan: Efficient Evaluation of XML Middleware Queries. ACM SIGMOD 2001 May 2124.

[22] Shanmugasundaram, Jayavel., Shekita, Eugene., Barr, Rimon., Carey, Michael.: Efficiently Publishing Relational Data as XML Documents

[23] Krsihnamurthy, Rajasekar., Kaushik, Raghav., Naughton F., Jeffrey.: Efficient XML-to-SQL Query Translation: Where to Add the Intelligence?

[24] Hayashi S. Larry., Hatton, John.: Combining UML, XML and relational databases technologies – the best of all worlds for robust linguistic databases

[25] Personal conversation with Beta Lab members involved in RNA secondary structure prediction research

[26] M. Zuker and P. Stiegler, Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information, Nucl. Acids. Res. (1981) 9: 133-148.

[27] D. H. Mathews, J. Sabina, M Zuker and D. H. Turner, Expanded Sequence Dependence of Thermodynamic Parameters Improves Prediction of RNA Secondary Structure, J. Mol. Biol. (1999) 288, 911-940

[28] Cagle, K., Gibbons, D., Hunter, D., Ozu, N., Pinnock, J., Spencer, P.: Beginning XML. Wrox Press Ltd., Birmingham UK, 2000

[29] Mironov, A., and Lebedev, V. F.: A kinetic model of RNA folding. *Systems*, 30 (1993) 49-56.