Biosequence Use Cases in MoBIoS SQL

Daniel P. Miranker, Willard J Briggs, Rui Mao, Shulin Ni and Weijia Xu
Department of Computer Sciences
University of Texas at Austin
{miranker, willard, rmao, shulin, xwj}@cs.utexas.edu [*]

Abstract

The sequencing and annotation of entire genomes has enriched the content of biological sequence databases such that new methods of sequence analysis, comparison and retrieval are being invented and rerun on an increasingly regular basis, generating new and more complete biological information. Examples include full genome comparisons and phylogenetic footprinting. Simple identification of homologous sequences based on BLAST searches is now just one option for querying the contents of a sequence database.

These developments underscore the need for more general methods of sequence data management and concomitant programming models that simplify biological discovery. MoBIoS, the Molecular Biological Information System, with mSQL, its set of SQL extensions, is such a system. MoBIoS supports two views of sequence data. Sequences are identified and stored based on long functional units (e.g. genes, proteins and chromosomes). Matching and analysis of sequences exploits distance-based methods comparing short-overlapping substrings. We show that a number of sequence analysis problems can thus be succinctly expressed as mSQL queries.

1 Introduction

MoBIoS, the Molecular Biological Information System (pronounced mobius), is a metric-space database management system targeting life-science data. Analogous to spatial databases which extend relational systems with index-structures and data types that support two and three-dimensional data and form the basis of geographic information systems (GISs), MoBIoS comprises built-in biological data types and index structures to support fast object storage retrieval based on the relative distance between objects determined by metric-distance functions (metrics) [MXM03, CNBYM01]. Figure 1 illustrates the MoBIoS platform. MoBIoS is built in Java on top of the McKoi open-source DBMS[McK]. McKoi includes a JDBC interface, allowing MoBIoS to integrate seamlessly with web-application tool stacks.

Work to date includes development or identification of effective metric-models of biological similarity for peptide sequences (mPAM), mass-spectrometer signatures and 3-d electrostatic models of proteins and respective applications [MXM03, XM04, ZBB].

[*]This research was supported by grants from the Texas Higher Education Coordinating Board and the National Science Foundation contract DBI-0241180, IIS-0325116, EIA-0121680, EF-0331453

In the case of sequences, more database machinery is needed than with atomic representations of mass-spectrometer signatures or 3-D protein models. In the latter two cases distance-based range queries and join queries are precisely analogous to spatial databases, except that absolute position in Euclidean space is replaced with relative distance determined by a metric. This is not to say that the community's understanding of query processing and index support of metric-space databases is at all mature. Competing indexing methods continue to be refined [Bri95, MXSM03, STMO03]. There is still no clear winner. Metric-space joins have barely been addressed [WS90, DGZ03].

A management challenge for biological sequences is that a biologist's view of a sequence is different than the computational view. Identification of biological sequences comprises long functional units (e.g. genes, proteins and chromosomes). Excluding the Smith-Waterman algorithm as an important exception, most comparative sequence analysis algorithms are structured such that they first break sequences into short overlapping substrings. Further processing compares substrings and may ultimately reassemble them into longer units. Thus far these algorithms rarely consider additional substructure; for example, the location of introns, exons, and transcription factor binding sites. These functional subunits are enumerated by name and position in the Genbank features table. As a group, these are referred to as sequence annotations.

Our speculation is that the granularity of sequence management in Genbank and related systems is largely responsible for the disassociation of annotation from sequence comparison. In common practice, a set of sequences is retrieved from Genbank by virtue of common annotations and/or BLAST-based similarity. The set of sequences are culled by further analysis of sequence content. Additional inspection or filtering of those sequences based on annotations requires ad-hoc scripts to map the resulting sequences back to their Genbank entries. Thus, we claim that there is ample motivation to integrate sequence analysis with database query engines and enable optimized query plans to interleave primary structure (sequence) and functional comparisons.

In addition to metric-distance based access paths, MoBIoS includes syntactic, logical and physical database extensions to manage biological sequences. The primary syntactic extension is called a *sequenceview*. *Sequenceviews* enable SQL programmers to specify that, in addition to storing and retrieving biological sequences as long functional units, a sequence may also be operated upon as a set of overlapping q-grams. Furthermore, users may specify one of a number of built-in metrics for comparing the similarity of q-grams, or they may specify their own in a manner similar to writing a stored procedure. Three logical operators make *sequenceviews* possible, *createfragments()*, *groupfragments()* and *merge()*. The corresponding physical operators and supporting structures are discussed elsewhere [BLM$^+$03].

In this paper we discuss our query language and illustrate its use to capture a number of sequence analysis protocols emerging in bioinformatics. The data model specified in Figure 2 will serve as the basis for each of the examples.

The first two tables are used to store DNA and protein sequences, where the column 'Seq' holds the sequences. The sequence id, SID, serves as a foreign key to a table of sequence annotations. Per the vernacular of the area this is called the feature table. Rooted in ASN.1, the semi-structured foundation of Genbank, a feature is a substring of a sequence, denoted by the offset of the first and last sequence element and labeled with one of a moderated list of tags [BKML$^+$02].

2 mSQL

mSQL is the name we have chosen for our data type and operator extensions to the SQL standard. mSQL introduces data types to manage sequence data, mass spectral data, and secondary and tertiary protein data. The primitive data types introduced to handle sequence data are called DNA, RNA, and Peptide, all of which are subtypes of a generic Sequence data type. In general, a Sequence can be thought of as a string with two important differences, stemming from the biological nature of sequences. First, the alphabet is limited to a certain set of characters depending upon the type of sequence, i.e. ACTG for DNA sequences. Second, we must

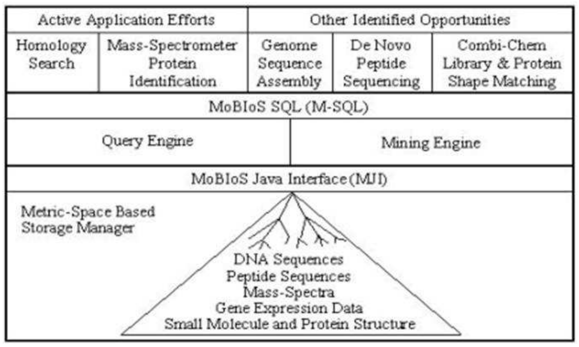

Active Application Efforts
Other Identified Opportunities
Homology Search
Mass-Spectrometer Protein Identification
Genome Sequence Assembly
De Novo Peptide Sequencing
Combi-Chem Library & Protein Shape Matching
MoBIoS SQL (M-SQL)
Query Engine
Mining Engine
MoBIoS Java Interface (MJI)
Metric-Space Based Storage Manager
DNA Sequences
Peptide Sequences
Mass-Spectra
Gene Expression Data
Small Molecule and Protein Structure

Figure 1: Architecture of the MoBIoS Platform

```
DNA_Sequence(SID, Organism, seq)
Protein_Sequence(SID, Organism, seq)
Feature_table(SID, FID, tag, start, end)
```

Figure 2: Tables and Attributes of the Example Schema

```
Select SID, Organism, Createfragments(seq, 3)
From DNA_Sequence;
```

Figure 3: *Createfragments()* Query

```
Select *
From Createfragments(DNA_Sequence.seq, 3);
```

Figure 4: *Createfragments()* Query

introduce the *revcomp()* operator to compute the reverse complement of DNA and RNA sequences; applied to ordinary strings, *revcomp()* would have no logical meaning.

mSQL also introduces two new SQL-level operators to convert sequence information between its two logical perspectives: *createfragments()* and *merge()*. These operators are rather similar to the unnest and nest operators popular in the extended-relational algebras of the early to mid-80s [JS82]. Their differences lie in the pre-processing and post-processing steps necessary for each to logically view sequences as sets of overlapping subsequences.

Createfragments() is a two-step operation. The first step takes a sequence of characters and a fragment length as input and returns a set of 2-tuples. The first attribute of each tuple is the offset from the original sequence, and the second attribute is the fragment. Each of these 2-tuples is an instance of an additional internal data type, SubSequence. SubSequence contains these two fields, offset and fragment, as well as an operator to obtain the length of the sequence. The vast majority of operations on sequence data will be performed using these SubSequences. After the first step of *createfragments()*, the set is unnested to yield the final usable result.

For example, assume that the DNA_Sequence table described above is populated with the following two rows: {R1, Rice, ACAA}, {R2, Rice, ACTCA}. The query in Figure 3 would yield the result shown in Table 1. This set is then unnested, yielding the final result in Table 2

Table 1: Intermediate Results of *Createfragments()* Operation

SID	Organism	Createfragments(Sequence, 3)
R1	Rice	[[0, ACA], [1, CAA]]
R2	Rice	[[0, ACT], [1, CTC], [2, TCA]]

Table 2: Final Results of *Createfragments()* Operation

SID	Organism	Createfragments(Sequence, 3)
R1	Rice	[0, ACA]
R1	Rice	[1, CAA]
R2	Rice	[0, ACT]
R2	Rice	[1, CTC]
R2	Rice	[2, TCA]

Note that these fragments are not guaranteed to be in sequential order. In the implementation of mSQL, the two steps of *createfragments()* have been combined into one, with the syntax demonstrated in Figure 4.

This yields the same results as shown in Table 2.

The *merge()* operation is nearly the reverse of *createfragments()*. An additional step is also needed at the beginning, due to the fact that the results of a query may not necessarily be in sequential order. For this step, we have overloaded the standard SQL GROUP BY operator to order these fragments by their offsets and then separate them into groups. Two fragments are considered to be in the same group if the difference between their offsets is less than the length of the fragments. The second step is the nest operation, which is applied individually to each of these groups. The final step is the actual merge, which merges each set of tuples back into one larger sequence. The sequences are ordered by offset and then the overlapping sections are removed, yielding one long SubSequence. The offset from the first fragment of the set is maintained as the offset from the original sequence for the entire subsequence.

The *merge()* operation can occur in either a one-dimensional or a two-dimensional case. We have just described the one-dimensional case, which assumes that all of the fragments are from the same parent sequence. The two-dimensional case is used on the results of a metric join. In this case the results are first grouped by the fragments from the first sequence, then by the fragments from the second, with the additional rule that two fragments must be from the same parent sequence to be in the same group. The nest and the merge are performed as usual.

It is not feasible or necessary to materialize the results of the *createfragments()* operation. With a fragment length of q and a sequence length of n, materializing *createfragments()* would require storing an additional q(n-q+1)-n characters. For this we introduce the concept of the *sequenceview*, analogous to SQL's view, which materializes the results of *createfragments()* as a secondary metric space index. Implementation details are discussed elsewhere [BLM$^+$03]. *Sequenceviews* can be used in the same manner as standard SQL views, without the same space or time overhead. In this way indices can be pre-built offline, speeding online queries.

3 Application Examples

3.1 Electronic PCR

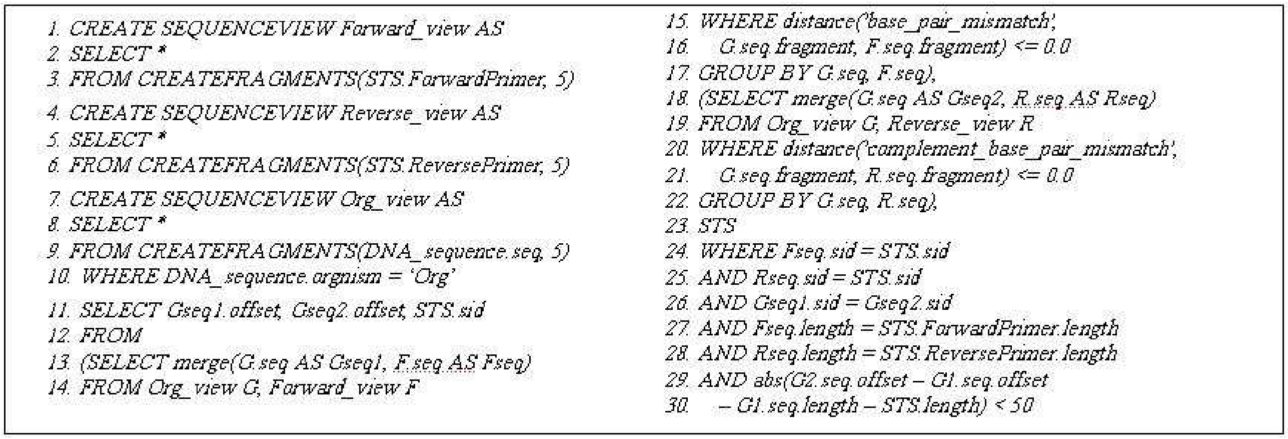
A sequence-tagged site consists of a pair of primers which can uniquely identify a site in the genome. Electronic PCR is used to computationally find sequence-tagged sites (STSs) in DNA sequences by searching for subsequences that closely match the PCR primers and have the correct order, orientation, and spacing that they could plausibly prime the amplification of a PCR product of the correct molecular weight [Sch97].

In-lieu of a procedural utility program, the Electronic PCR problem can be solved as an mSQL query (Figure 5). For brevity, we introduce some simplifications, i.e., we have not checked for the possibility of matching reverse complements. We have coded the problem as described below.

- Create *sequenceviews* for forward and reverse primers in a STS table; create a *sequenceview* for the genome of an organism. *(lines 1-3; lines 4-6; lines 5-9)*
- Utilize the metric-space index to find matching fragments of primers and genome sequences. Find pairs of merged fragments that match forward and reverse primers with the following conditions:
  - The primers are fully matched. *(lines 15-16; lines 20-21)*
  - The two genome fragments come from the same sequence. The two primers belong to the same STS. *(lines 24-26; lines 27-28)*
  - The spacing between the two genome fragments is within 50 bases of the length of the PCR product. *(lines 29-30)*

3.2 Conserved Primer Pair Discovery

To help solve the question as to whether evolution is adequately modeled by bifurcating trees, or if/when network models are critical, we used MoBIoS to determine a candidate set of PCR primers that would enable biologists to sample, amplify and sequence the DNA of any flowering plant in a large number of places. The query, which we hand-compiled, involves joining *sequenceviews* of the Rice and Arabidopsis genomes in search of shared substrings that fulfill the electronic PCR properties. The number of nucleotides, and therefore the number of logical rows, is in excess of half a billion. Our current implementation comprises an indexed nested-loop join,



```
CREATE SEQUENCEVIEW Forward_view AS
SELECT *
FROM CREATEFRAGMENTS(STS.ForwardPrimer, 5)
CREATE SEQUENCEVIEW Reverse_view AS
SELECT *
FROM CREATEFRAGMENTS(STS.ReversePrimer, 5)
CREATE SEQUENCEVIEW Org_view AS
SELECT *
FROM CREATEFRAGMENTS(DNA_sequence.seq, 5)
WHERE DNA_sequence.orgnism = 'Org'
SELECT Gseq1.offset, Gseq2.offset, STS.sid
FROM
(SELECT merge(G.seq AS Gseq1, F.seq AS Fseq)
FROM Org_view G, Forward_view F
WHERE distance('base_pair_mismatch',
    G.seq.fragment, F.seq.fragment) <= 0.0
GROUP BY G.seq, F.seq),
(SELECT merge(G.seq AS Gseq2, R.seq AS Rseq)
FROM Org_view G, Reverse_view R
WHERE distance('complement_base_pair_mismatch',
    G.seq.fragment, R.seq.fragment) <= 0.0
GROUP BY G.seq, R.seq),
STS
WHERE Fseq.sid = STS.sid
AND Rseq.sid = STS.sid
AND Gseq1.sid = Gseq2.sid
AND Fseq.length = STS.ForwardPrimer.length
AND Rseq.length = STS.ReversePrimer.length
AND abs(G2.seq.offset – G1.seq.offset
    – G1.seq.length – STS.length) < 50
```

Figure 5: Electronic PCR

which is $O(nlogn)$. We solved the problem in less than 2 days using 4 processors of a Sun 6800. The results are currently being validated in a wet lab [XBP+04]. This type of computation is usually the province of very-large clusters, running parallel copies of BLAST as the inner loop of an $O(n^2)$ solution. Please see Xu et al for the mSQL code for this query [XBP+04].

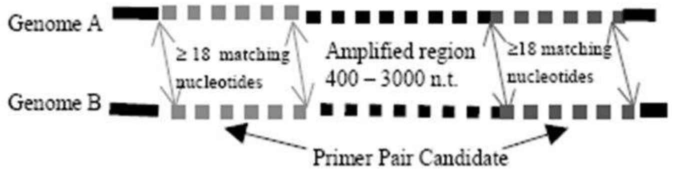
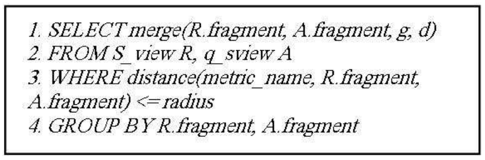


Figure 6: Finding Conserved Primer Pairs



```
SELECT merge(R.fragment, A.fragment, g, d)
FROM S_view R, q_sview A
WHERE distance(metric_name, R.fragment,
A.fragment) <= radius
GROUP BY R.fragment, A.fragment
```

Figure 7: Homology Search

3.3 Homology Search

The mSQL query to solve the homology search problem is illustrated in Figure 7. BLAST-like matching of hot-spots is accomplished by a metric-space join. A two-dimensional merge operator merges the matching q-grams [BLM+03, XMMW03]. An optional gap function, g and distance threshold, d, enables hot-spot extension.

3.4 All-way Genomic Conservation

The availability of whole genome sequence data makes it possible to discover conserved features across multiple organisms. Ultraconserved elements are sequence segments that are absolutely conserved (100% identity with no deletion or insertion) between orthologous regions of a number of genomes. See Figure 8. A recent computational study of the human, rat, and mouse genomes has found that ultraconserved elements play important roles in RNA processing, transcription regulation and development [BPM+04].

Again, for brevity and simplicity, we don't limit fragment matching to orthologous regions in the mSQL query for finding ultraconserved elements from three genomes. We first create *sequenceviews* with fragments of length 200 from each genome. Thus the minimum length of an ultraconserved element is 200. The query is formulated as in Figure 10.

Notice in lines 12 and 14, the join stipulates exact matches per the past work and the ability of the software. However, in MoBIoS we could easily repeat the study for varying amounts of sequence divergence by increasing the join distance.

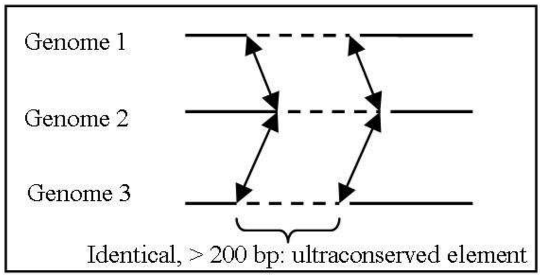


Figure 8: Ultraconserved Element

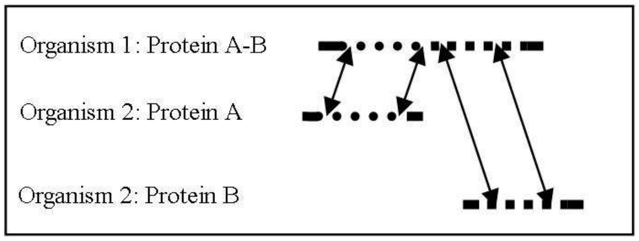
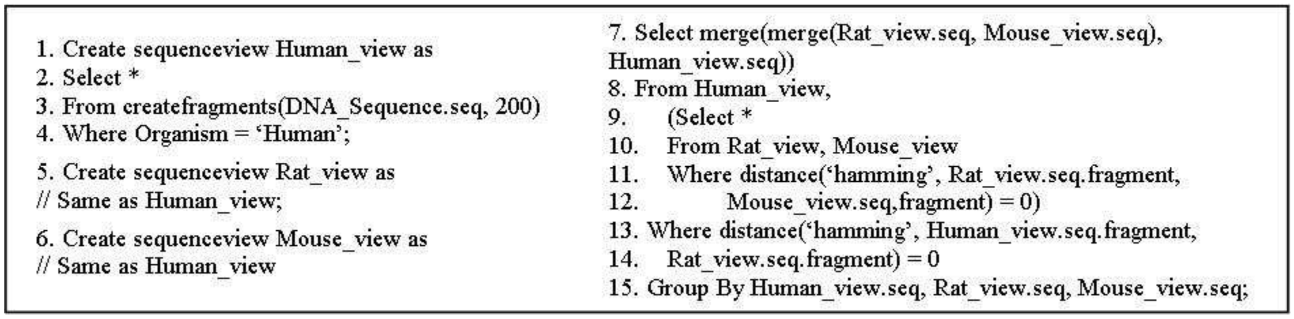


Figure 9: Rosetta Stone Protein Search

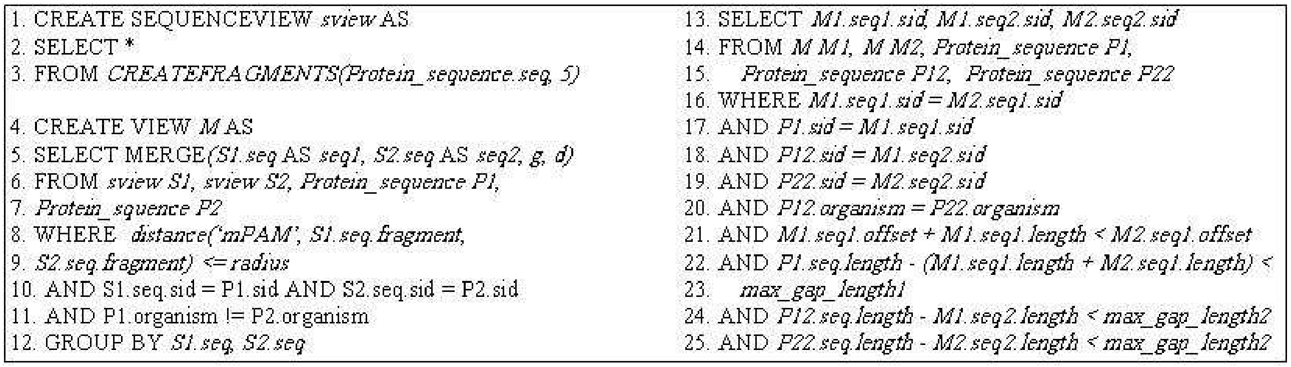


```
1. Create sequenceview Human_view as
2. Select *
3. From createfragments(DNA_Sequence.seq, 200)
4. Where Organism = 'Human';
5. Create sequenceview Rat_view as
// Same as Human_view;
6. Create sequenceview Mouse_view as
// Same as Human_view
7. Select merge(merge(Rat_view.seq, Mouse_view.seq),
Human_view.seq))
8. From Human_view,
9.      (Select *
10.     From Rat_view, Mouse_view
11.     Where distance('hamming', Rat_view.seq.fragment,
12.           Mouse_view.seq,fragment) = 0)
13. Where distance('hamming', Human_view.seq.fragment,
14.     Rat_view.seq.fragment) = 0
15. Group By Human_view.seq, Rat_view.seq, Mouse_view.seq;
```

Figure 10: Three-way Genome Comparison

3.5 Rosetta Stone

It has been observed that if two proteins A and B in one organism are both homologous to a single protein A-B in another organism (See Figure 9), there is a good chance that A and B interact or share related biological functions [MPN+99]. Such a protein A-B is termed a 'Rosetta Stone' protein. Sequence alignment methods can be used to find if two proteins in one organism have non-overlapping alignments on a single protein in another organism. The following mSQL query is to obtain the sequences IDs of these protein triplets (Figure 11).



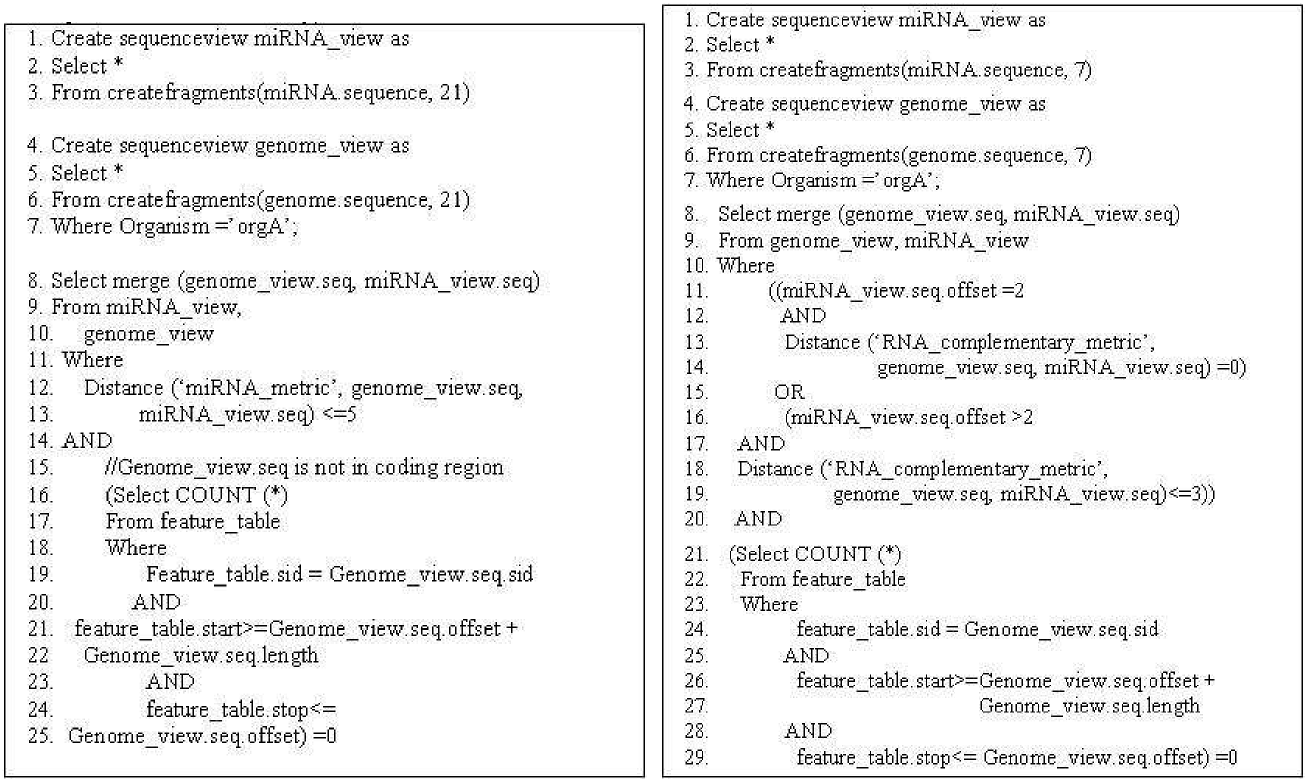
```
1. CREATE SEQUENCEVIEW sview AS
2. SELECT *
3. FROM CREATEFRAGMENTS(Protein_sequence.seq, 5)

4. CREATE VIEW M AS
5. SELECT MERGE(S1.seq AS seq1, S2.seq AS seq2, g, d)
6. FROM sview S1, sview S2, Protein_sequence P1,
7. Protein_squence P2
8. WHERE  distance('mPAM', S1.seq.fragment,
9. S2.seq.fragment) <= radius
10. AND S1.seq.sid = P1.sid AND S2.seq.sid = P2.sid
11. AND P1.organism != P2.organism
12. GROUP BY S1.seq, S2.seq

13. SELECT M1.seq1.sid, M1.seq2.sid, M2.seq2.sid
14. FROM M M1, M M2, Protein_sequence P1,
15.     Protein_sequence P12,  Protein_sequence P22
16. WHERE M1.seq1.sid = M2.seq1.sid
17. AND P1.sid = M1.seq1.sid
18. AND P12.sid = M1.seq2.sid
19. AND P22.sid = M2.seq2.sid
20. AND P12.organism = P22.organism
21. AND M1.seq1.offset + M1.seq1.length < M2.seq1.offset
22. AND P1.seq.length - (M1.seq1.length + M2.seq1.length) <
23.     max_gap_length1
24. AND P12.seq.length - M1.seq2.length < max_gap_length2
25. AND P22.seq.length - M2.seq2.length < max_gap_length2
```

Figure 11: Rosetta Stone Query

The proteins come from multiple organisms. There are three steps in our query.

- First, create a single *sequenceview* for protein sequences of all organisms. *(lines 4-12)*
- Use the metric space index to make local alignments between pairs of proteins from any two distinct organisms. The results are pairs of merged matching protein fragments. Create a view from the results. *(lines 4-12)*

- Produce the Rosetta Stone protein triplets using the following conditions:
  - Two proteins from one organism are aligned to a single protein from another organism without overlap. *(lines 16-20)*
  - The difference between the length of a protein and the total length of its aligned fragments must be less than a given constant. *(lines 21-25)*



```
Create sequenceview miRNA_view as
Select *
From createfragments(miRNA.sequence, 21)

Create sequenceview genome_view as
Select *
From createfragments(genome.sequence, 21)
Where Organism =' orgA';

Select merge (genome_view.seq, miRNA_view.seq)
From miRNA_view,
   genome_view
Where
   Distance ('miRNA_metric', genome_view.seq,
        miRNA_view.seq) <=5
AND
     //Genome_view.seq is not in coding region
     (Select COUNT (*)
     From feature_table
     Where
         Feature_table.sid = Genome_view.seq.sid
       AND
  feature_table.start>=Genome_view.seq.offset +
   Genome_view.seq.length
         AND
         feature_table.stop<=
 Genome_view.seq.offset) =0
```

(a) MiRscan query

```
Create sequenceview miRNA_view as
Select *
From createfragments(miRNA.sequence, 7)
Create sequenceview genome_view as
Select *
From createfragments(genome.sequence, 7)
Where Organism =' orgA';
 Select merge (genome_view.seq, miRNA_view.seq)
 From genome_view, miRNA_view
Where
      ((miRNA_view.seq.offset =2
        AND
        Distance ('RNA_complementary_metric',
                  genome_view.seq, miRNA_view.seq) =0)
       OR
        (miRNA_view.seq.offset >2
  AND
  Distance ('RNA_complementary_metric',
            genome_view.seq, miRNA_view.seq)<=3))
  AND
 (Select COUNT (*)
  From feature_table
  Where
        feature_table.sid = Genome_view.seq.sid
      AND
        feature_table.start>=Genome_view.seq.offset +
                             Genome_view.seq.length
      AND
        feature_table.stop<= Genome_view.seq.offset) =0
```

(b) miRNA target site query

Figure 12: RNAi Queries

3.6 RNA Interference

RNA interference (RNAi) refers to the post-transcriptional gene silencing (PTGS) induced by the direct introduction of double stranded RNA. In the past few years, RNAi has become a popular tool in molecular biology to knock out genes in a variety of organisms [Gur00, HCH01].

MicroRNAs (miRNAs), an important class of interfering RNA, are endogenous RNAs that are about 22 nucleotides long. MiRscan is an miRNA gene prediction tool in which all experimentally verified miRNA genes were compared with a 21nt windows passing through each conserved stem loop of the genome sequence [LGY+03]. TargetScan is a tool that predicts target sites conserved across multiple genomes. The first step of the algorithm is to search a set of orthologous 3' UTR sequences from one organism for perfect Watson-Crick complementary matches to bases 2-8 (from the 5' end) of the miRNA, and then extend matches [LSJR+03]. We expect such searching processes can also be expressed in mSQL as shown in Figure 12.

In both queries, the *sequenceviews* for known miRNA genes and sequences of one genome are created with fragment lengths 21 and 7, respectively *(lines 1-3; lines 4-7)*. To find the miRNA candidate (Figure 12(a)), we use 'miRNA_metric' as the metric distance function to measure the closeness of two miRNA segments *(lines 12-13)*. In Figure 12(b), the 'RNA_complementary_metric' is the metric distance function used between the reverse complement of the first RNA fragment and the second RNA fragment *(lines 11-19)*. The purpose of lines 16-25 in Figure 12(a) and lines 22-31 in Figure 12(b) are to exclude fragments that are derived from the coding region. The results from the above queries are subject to further evaluations such as meeting the energy required for RNA folding.

4 Discussion and Conclusion

MoBIoS, and especially its mSQL component, remains a work in progress. While we have successfully found a solution to the Conserved Primer Pair problem using the MoBIoS platform, none of the above queries have yet been implemented at a SQL level. In presenting them it is our goal to show that the future of genomics research goes far beyond the homology search now possible with programs such as BLAST; that as new, interesting problems arise with greater and greater frequency, biologists need tools that are powerful enough to adapt quickly to changing demands; and finally, that these tools must be easy to use and rely on already established standards. MoBIoS with mSQL promises to address all of these concerns.

The above queries are meant to represent what will soon be possible with a cohesive biological database management system such as MoBIoS. We have demonstrated the feasibility of performing useful queries on sequence data within a database management system itself, offering an alternative to the chains of programs previously necessary to solve complex genomics problems. However, many questions remain unanswered. We have yet to address the issue of regular expressions in queries. We also have not focused our attention on how to handle secondary and tertiary structure information. Elements of other bioinformatics and string algebras are under consideration to support these goals [TP03].

References

[BKML+02] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, B. A. Rapp, and D. L. Wheeler. Genbank. *Nucleic Acids Res.*, 20(1):17–20, 2002.

[BLM+03] W. J Briggs, W. Liu, R. Mao, W. Xu, and D. P. Miranker. SQL extensions and database mechanisms for managing biosequences. Technical Report TR-04-05, The University of Texas at Austin, Department of Computer Sciences, December 2003.

[BPM+04] G. Bejerano, M. Pheasant, I. Makunin, S. Stephen, W.J. Kent, J.S. Mattick, and D. Haussler. Ultraconserved elements in the human genome. *Science*, 304(5675):1321–5, May 2004.

[Bri95] S. Brin. Near neighbor search in large metric spaces. In *Proc. 21st Conference on Very Large Database (VLDB'95)*, pages 574–584, 1995.

[CNBYM01] E. Chavez, G. Navarro, R. Baeza-Yates, and J.L. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[DGZ03] V. Dohnal, C. Gennaro, and P. Zezula. Similarity join in metric spaces using ed-index. In *Proc. of the14th International Conference Database and Expert Systems Applications (DEXA 2003), Volume 2736 of Lecture Notes in Computer Science*, pages 484–493. Springer, May 2003.

[Gur00] T. Guru. A silence that speaks volumes. *Nature*, 404:804–808, 2000.

[HCH01] S.M. Hammond, A.A. Caudy, and G.J. Hannon. Post-transcriptional gene silencing by double-stranded RNA. *Nature Rev Gen*, 2:110–119, 2001.

[JS82] B. Jaeschke and H. J. Schek. Remarks on the algebra of non first normal form relations. In *Proc. ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, 1982.

[LGY$^+$03] L. P. Lim, M. E. Glasner, S. Yekta, C. B. Burge, and D. P. Bartel. Vertebrate microRNA genes. *Science*, 299(5612):1540, Mar 2003.

[LSJR$^+$03] B. P. Lewis, I. Shih, M. W. Jones-Rhoades, D. P. Bartel, and C. B. Burge. Prediction of mammalian microRNA targets. *Cell*, 115:787–798, 2003.

[McK] http://www.mckoi.com.

[MPN$^+$99] E. M. Marcotte, M. Pellegrini, H. L. Ng, D. W. Rice, T. O. Yeates, and D. Eisenberg. Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285(5428):751–3, 1999.

[MXM03] D. P. Miranker, W. Xu, and R. Mao. Architecture and application of MoBIoS, a metric-space DBMS to support biological discovery. In *15th International Conference on Scientific and Statistical Database Management. (SSDBM03)*, pages 241–244, 2003.

[MXSM03] R. Mao, W. Xu, N. Singh, and D. P. Miranker. An assessment of a metric space database index to support sequence homology. In *Proc. of the 3rd IEEE Symposium on Bioinformatics and Bioengineering*, Washington D.C, March 2003.

[Sch97] G. D. Schuler. Sequence mapping by electronic PCR. *Genome Research*, 7(5):541–550, May 1997.

[STMO03] S. C. Sahinalp, M. Tasan, J. Macker, and Z. M. Özsoyoglu. Distance based indexing for string proximity search. In *Proc. of IEEE Data Engineering Conference, ICDE 2003*, pages 125–136, Banglore, India, 2003.

[TP03] S. Tata and J. Patel. PiQA: An algebra for querying protein data sets. In *15th International Conference on Scientific and Statistical Database Management. (SSDBM03)*, pages 141–151, 2003.

[WS90] T.L. Wang and D. Shasha. Query processing for distance metrics. In D. McLeod, R. Sacks-Davis, and H. Schek, editors, *Proc. of the 16th International Conference on Very Large Databases*, pages 602–613, Brisbane, Australia, August 1990.

[XBP$^+$04] W. Xu, W. J Briggs, J. Padolina, W. Liu, R. Timme, C. R. Linder, and D. P. Miranker. Using MoBIoS' scalable genome join to find conserved primer pair candidates between two genomes. *Bioinformatics*, 20:i355–i362, 2004.

[XM04] W. Xu and D. P. Miranker. A metric model of amino acid substitution. *Bioinformatics*, 20:1214–1221, 2004.

[XMMW03] W. Xu, D. P. Miranker, R. Mao, and S. Wang. Indexing protein sequences in metric space. Technical Report TR-04-06, The University of Texas at Austin, Department of Computer Sciences, October 2003.

[ZBB] Xiaoyu Zhang, Chandrajit L. Bajaj, and Nathan Baker. Fast matching of volumetric functions using multi-resolution dual contour trees. (submitted for publication).