World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:8, 2009

An Algebra for Protein Structure Data

Yanchao Wang, and Rajshekhar Sunderraman

***Abstract*—**This paper presents an algebraic approach to optimize queries in domain-specific database management system for protein structure data. The approach involves the introduction of several protein structure specific algebraic operators to query the complex data stored in an object-oriented database system. The Protein Algebra provides an extensible set of high-level Genomic Data Types and Protein Data Types along with a comprehensive collection of appropriate genomic and protein functions. The paper also presents a query translator that converts high-level query specifications in algebra into low-level query specifications in Protein-QL, a query language designed to query protein structure data. The query transformation process uses a Protein Ontology that serves the purpose of a dictionary.

***Keywords*—**Domain-Specific Data Management, Protein Algebra, Protein Ontology, Protein Structure Data.

I. Introduction

IN the past decade, protein data has been growing rapidly due to more and more advanced experimental techniques. The flood of protein data, their high heterogeneity, their multi-structure, multi-format, multi-access method, mismatch of low level treatment and high level nature and the complexity make it much more important and challenging in biology [3]. Therefore, the problem how to efficiently store, retrieve, analyze and modify protein data is becoming an important issue for most protein scientists and computer scientists. In order to solve this problem, a Domain Specific Object Oriented DataBase Management System (DSOODBMS) is designed to manipulate Protein Data. In this DSOODBMS, Protein Query Language (Protein-QL) and Protein Object-Oriented DataBase (Protein-OODB) are provided to deal with the queries in protein domain which can be easily extended into other biological domains. In this application system, two ways are designed to match Protein-QL to Protein-OODB. One is to directly interpret Protein-QL syntax to Protein-OODB, the other uses Protein Algebra Architecture to connect them that can optimize the queries which is very important for complex queries and large dataset to provide better performance for protein data management.

In this paper, an architecture called Protein Algebra architecture is described, which connects Protein-QL and Protein-OODB and optimizes protein data queries. It has three components, Protein Ontology, Protein Algebra and Protein Wrapper. The Protein Algebra provides an extensible set of high-level genomics data types (GDTs) (e.g., genome, gene, chromosome, protein) and protein data types (PDTs) (e.g. primary, secondary, tertiary, protein) together with a comprehensive collection of appropriate genomic functions (e.g., translate, transcribe, decode) and protein functions (e.g., sequence, getPrimary, nearestNeighbour), it also provides genomics and protein operations to deal with protein domain specific object queries. Protein Ontology which is designed as a dictionary is used to map Protein Algebra to Protein-QL. Protein Wrapper connects Protein Algebra and Protein-OODB which makes Protein Algebra independent of Protein-OODB.

The rest of this paper is organized as follows: section II describes the current architecture for protein domain specific object oriented database management system. Section III presents protein algebra architecture. Related work is discussed in section IV. The conclusion and future work will be shown in section V.

II. Current Architecture of Protein Domain Specific Object-Oriented Database Management System

The overall architecture of protein domain specific object-oriented database management system for protein structure data called Protein-OODBMS is a three-layer architecture that consists of the following components: a client API, Middleware (including a RMI server, a query language for protein structures (Protein-QL), and an object-oriented database for protein structures (Protein-OODB)), and Data Server layer.

The current architecture of protein domain specific object-oriented database management system (DSOODBMS) in Fig. 1 illustrates the details of Protein-OODBMS. This system extends the object-oriented database (OODB) system by adding two additional layers Protein-QL and Protein-OODB above OODB, it is designed specifically for protein domain, but it is a first step in building a general Bio-OODBMS for biological applications.

The clients can use this system to send domain specific requests and manage the database. The Client writes simple domain specific queries according to Protein-QL and sends them to the Server. The Server receives the queries and communicates with Protein-QL and checks the grammar of queries according to the syntax of Protein-QL. Then system converts queries into EYEDB queries. Finally EYEDB sends the results back to the server. This system provides clients convenient access and is easily mastered [2].

Yanchao Wang is with The IRMACS Centre, Simon Fraser University, 8888 University Drive, Burnaby, BC V5T 1S6, Canada (corresponding author to provide phone: 778-782-7078; fax: 778-782-7065; e-mail: yanchao_wang@yahoo.com).

Rajshekhar Sunderraman is with Computer Science Department, Georgia State University, 34 Peachtree Street, Atlanta, GA 30303 USA (e-mail: raj@cs.gsu.edu).

The following part shows detail of this new architecture of DSOODBMS for protein structure data:

1. Client API can have multiple types such as Java Client Application, PQL Plus Client, Data Browser, Visualization and PDB Expert shown in Fig. 1.
   - Java Client API [2] can easily be viewed and mastered by any user who knows requested parameter without much computer background. Clients can be able to formulate Protein-QL queries and have them sent to the server for execution. This API also provide help functions to help the clients send queries, display results in a domain friendly manner.
   - PQL Plus Client [2] is much like SQL Plus interaction that allows clients to send protein-QL queries directly to Protein-QL without any java code.
   - Data Browser [2] provides clients to view protein data in PDB format or object format.
   - Visualization Client [2] is like Rasmol tool that allows clients to view protein data structure and functions. It also supports linking the protein to RCSB PDB to get 3D view.
   - PDB Experts [6] also can be considered as Client API which is designed to provide the possible ways to curate data and improve the PDB (most protein data are stored in PDB format) data quality. PDB Experts input PDB file from here, system deals with the PDB data clean such as data identification, data errors, data redundancy, data ambiguity, data heterogeneity, data consistency, conflict data and obsolete data [5].
2. Server/Listener provides the basic services of the system. The users should be able to know the types of parameters, results and functions from Server/Listener which hides the details of implementation.
   - Protein-QL is designed as domain specific high-level query language and be able to provide convenience for users to store, retrieve, and modify data. It defines some basic operations such as SELECT, INSERT, DELETE, UPDATE that can be executed on basic data types as well as on protein data types. Protein-QL defines a list of queries in protein terms (such as nearest neighbors, subparts of protein and so on) which enables domain scientists to query information in their own language without much syntactical restriction. For example, the query `sequence(proteinName)` should return the sequence of protein named "proteinName" shown in Fig. 2.
   - Protein-OODB can provide possible method to solve some protein data sources' problems and is used to connect Protein-QL and OODB which makes Middleware independent of the underlying OODB.

   In addition, in order to simplify the queries in Protein-QL, the protein, primary, secondary and tertiary protein structures are defined as internal data

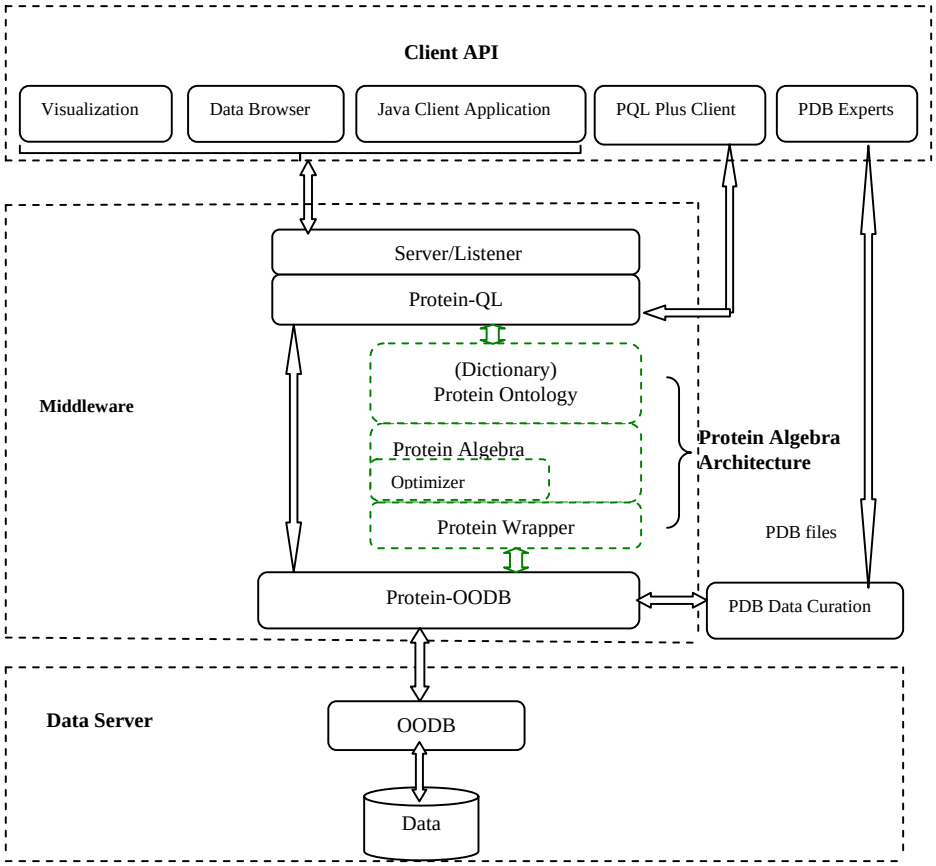

Fig. 1 The architecture of DSOODBMS for protein structure data

types such that domain scientists can easily formulate complex requests for data without much of a learning curve.

- Mapping Protein-QL to Protein-OODB can be finished in two ways, one is direct mapping which was done in the system and the other is to use Protein Algebra Architecture mapping. Since direct mapping does not provide any query optimization, the Protein Algebra Architecture mapping is designed with query optimization to provide better performance on protein data management.

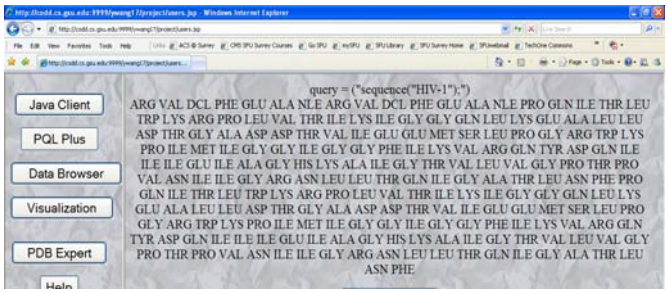

Fig. 2 The result of sequence("HIV-1")

3. In Data Server layer, OODB will provide users basic operations. EYEDB is used as the underlying OODB. Data storage will hold the protein data.

III. Protein Algebra Architecture

The formalization of the Protein Algebra is as follows:

```
sorts
  GDTs/PDTs | normal data types (such as
  string, int…)
ops
  genomics/protein operators
```

Genomics operators can be translate, transcribe and splice. And protein operators in [2] such as sequence, getPrimary and so on which have already been implemented in the Protein-OODBMS. By using this format, the Protein Algebra can easily query on genomics and protein domain data. For example:

```
sorts
  Protein, Sequence
ops
  sequence: Protein → Sequence
```

This simple algebra contains two PDTs for Protein and amino acid Sequence of protein primary structure as well as one operator sequence which returns its amino acid Sequence for a given Protein. This algebra can be syntactically expressed as `sequence(Protein)`.

A. Protein Algebra Data Types and Operations

Protein Algebra supports an extensible set of high-level genomics data types (GDTs) [1] (e.g. genome, gene, protein) and provides *protein data types* (PDTs) and collection of appropriate genomics and *protein operations or functions* such as transcribe, splice, sequence, getPrimary, nearestNeighbour and so on.

Firstly, this algebra designed new sorts called Protein Data Types (PDTs) (e.g. Protein, Primary, Secondary, Tertiary) and ops (protein operations, e.g. sequence, getPrimary, noOfChain, nearestNeighbor) which have same format as sorts and operators (ops) of GenAlg [1]. The following shows some examples of sorts and ops:

```
i. sorts
      Protein, Sequence, length
   ops
      sequence: Protein  → Sequence
      lengthOfSequence: Sequence→length
ii. sorts
      Protein, Primary
     ops
      getPrimary: Protein → Primary
iii. sorts
      Protein, ChainNumber
     ops
      noOfChain:  Protein → ChainNumber
iv.  sorts
      Protein, nearestNeighbour
     ops
      nearestNeighbour3D:Protein→Protein
```

Secondly, the algebra extends Genomics Data Types (GDTs) and Genomics Operations of GenAlg [1] to protein's structures in order to get more detailed information by providing more sorts and ops on protein data types (PDT) and protein operations. For example, it can extend sorts and ops of GenAlg as follows:

```
sorts
  Gene, PrimarymRNA, mRNA, Protein,
  Primary, Sequence
ops
  transcribe: Gene → PrimarymRNA
  splice: PrimarymRNA → mRNA
  translate: mRNA → Protein
  getPrimary: Protein → Primary
  sequence:  Primary → Sequence
```

Thirdly, the algebra can return multiple types in the same queries instead of single returned type result of GenAlg as following examples shown.

Example 1. Gets type and function of protein "HIV-1". ("HIV-1" is an abbreviation of "HIV-1 Protease")

```
sorts
   Protein, String union String
ops
   getTypes: Protein  →  String
   union
   getFunctions:  Protein  →  String
```

Type and function can be returned in one query even though they are strings in different formats.

Example 2. Gets sequence and secondary structure of protein "HIV-1".

```
sorts
   Protein, Sequence union Secondary
ops
   sequence:     Protein  → Sequence
   union
   getSecondary: Protein  → Secondary
```

In the Protein-OODBMS, Sequence is string and secondary is an object.

Finally, it can have some conditions or constraints on sort and ops which are very important for queries.

Example 3. Gets sub-sequence from position 0 to position 50 of protein.

```
sorts
  Protein, subSequence
ops
  subSequence: Protein→subSequence(0, 50)
```

Example 4. Gets sequence of protein which has the same sub-sequence as protein "HIV-1".

```
sorts
  Protein, Sequence
ops
  sequence : Protein(location
  (Protein.proteinName,
  subSequence("HIV-1",5,20))>=0)→Sequence
```

B. Protein Algebra Optimization

The Protein Algebra provides query optimization for large database and complex queries to provide system much better performance. The basic idea of the optimization is as follows: Suppose that the query contains several constraints, Protein Algebra checks them starting from the most inner one, it will stop query if present condition does not pass the checking, which saves time and optimizes queries. The following examples 5 and 6 illustrate how Protein Algebra optimizes the complex queries by using optimizer inside of Protein Algebra.

Example 5. Gets sequence of protein which the length is greater than the length of protein "HIV-1" and has the same sub-sequence as protein "HIV-1".

```
sorts
   Protein, Sequence
ops
  sequence:
  Protein(lengthOfSequence(Protein.
  proteinName)>lengthOfSequence("HIV-1"),
  location(Protein.proteinName,
  subSequence("HIV-1",5,20))>=0)→Sequence
```

Protein Algebra decides whether checking will go through next condition depending on the first condition `lengthOfSequence(Protein.proteinName) > lengthOfSequence("HIV-1")` is true or not. Example 6 is similar as this one.

Example 6. Gets type of protein which has the same number of helix as protein "HIV-1" and the number of chain is greater than protein "HIV-1".

```
sorts
  Protein, String
ops
  getTypes:
Protein(noOfHelix(Protein.proteinName)==
  noOfHelix("HIV-1"),
  noOfChain(Protein.proteinName)>
  noOfChain("HIV-1"))  →  String
```

C. Protein Ontology

Ontology is a controlled vocabulary to describe the functions, process and components for specific domains and used by people, databases, and applications to share domain information [9]. In the computer world, ontology is known as a machine-readable vocabulary that is specified with enough precision to allow differing terms to be precisely related [9]. Ontology enables users to share data, reuse and analyze domain data, especially for complicated biological data. But due to different goals and/or shortcomings of existing ontologies, this paper designed an ontology called Protein Ontology to resolve syntactic, terminological and semantic differences that are induced by multiple protein data sources. The Protein Ontology is capable of defining and identifying genomics and protein data objects, data operations and terminologies. It is also able to solve the problems of identical protein information represented differently in different data sources and same name used in the distinct concepts in different research to remove protein data ambiguity, incompatibility and inconsistency.

The Protein Ontology is designed as a dictionary to map Protein-QL [2] to Protein Algebra. The following examples show how it works.

Example 7. Gets primary structure of protein "HIV-1".

Protein-QL query is:

```
(Protein.primary)(Protein.proteinName="HIV
-1");
```

Protein Ontology will map it into Protein Algebra as follows:

```
sorts
  Protein, Primary
ops
  getPrimary: Protein  →  Primary
```

Example 8. Gets type and function of protein "HIV-1".

Protein-QL query is:

```
(Protein.types,Protein.functions)(Protein.
proteinName= "HIV-1");
```

It will be translated by Protein Ontology into Protein Algebra as follows:

```
sorts
  Protein, String union String
ops
  getTypes: Protein → String
  union
  getFunctions: Protein → String
```

Example 9. Gets sequence of protein which the length is greater than the length of protein "HIV-1" and has the same sub-sequence as protein "HIV-1".

Protein-QL query is as follows:

```
(sequence(Protein.proteinName))(lengthOfSe
quence(Protein. proteinName)
 >lengthOfSequence("HIV-1"),  location
(Protein.proteinName,subSequence("HIV-
1",5,20))>=0)
```

It should be mapped to Protein Algebra as follows:

```
sorts
  Protein, Sequence
ops
sequence:Protein(lengthOfSequence(Protein.
proteinName)>lengthOfSequence("HIV-1"),
location(Protein.proteinName,
subSequence("HIV-1", 5, 20))>=0)→Sequence
```

The Protein Ontology has two important goals. The first one is to identify the objects in genomics and protein domains.

The second one is to interpret Protein-QL queries to Protein Algebra and remove the data ambiguity, incompatibility and inconsistency by defining genomics and protein domain specific terminologies to describe the syntax and semantics.

D. Protein Wrapper

The Protein Wrapper capsulate the knowledge of Protein-OODB except for providing a pathway from Protein Algebra to Protein-OODB, which makes the Protein Algebra independent of underlying database. Thus the users only need to recode the Protein Wrapper without changing Protein Algebra if Protein Algebra is integrated into other data sources. In addition, Protein Wrapper can interpret Protein Algebra with query optimization to Protein-OODB.

Example 10. Gets primary structure of protein "HIV-1".

```
sorts
  Protein, Primary
ops
  getPrimary: Protein → Primary
```

This algebra contains two PDTs -- Protein and Primary as well as one operator getPrimary. It is translated into Protein-OODB as follows:

```
select p.Primary from Protein p where
p.proteinName="HIV-1";
```

Example 11. Gets type and function of protein "HIV-1".

```
sorts
  Protein, String union String
ops
  getTypes:       Protein → String
  union
  getFunctions:   Protein → String
```

This algebra can be translated into Protein-OODB as follows:

```
select p.types, p.functions from Protein p
where p.proteinName="HIV-1";
```

Examples 10 and 11 show a general format that Protein Wrapper translates Protein Algebra to Protein-OODB. The following examples 12 and 13 will illustrate how Protein Wrapper interprets Protein Algebra optimization queries to Protein-OODB queries. In these two examples, Protein Wrapper will interpret and at the same time check the conditions. It will continue to check next condition if present one is satisfied. Otherwise Protein Wrapper will stop translation and send result back to Protein Algebra.

Example 12. Gets sequence of protein which the length is greater than the length of protein "HIV-1" and has the same sub-sequence as protein "HIV-1".

```
sorts
  Protein, Sequence
ops
sequence:Protein(lengthOfSequence(Protein.
proteinName)>lengthOfSequence("HIV-1"),
location(Protein.proteinName,
subSequence("HIV-1", 5, 20))>=0)→Sequence
```

This query can be translated into Protein-OODB as follows:

```
select  sequence(p.proteinName)
from    (select     p.proteinName
         from      Protein p
    where(lengthOfSequence(p.proteinName)>
              lengthOfSequence("HIV-1"))
where   location(p.proteinName,
        subSequence("HIV-1",5,20)>=0);
```

If the condition `lengthOfSequence(p.proteinName) > lengthOfSequence("HIV-1")` is true, then the translation will go through following conditions. Otherwise the translation will stop for this protein and start next translation for another protein. Example 13 has similar syntax.

Example 13. Gets type of protein which has the same number of helix as the one of protein "HIV-1" and the number of chain is greater than protein "HIV-1".

```
sorts
  Protein, String
ops
getTypes:Protein(noOfHelix(Protein.protein
Name)==noOfHelix("HIV-1"),
noOfChain(Protein.proteinName)>
noOfChain("HIV-1"))→ String
```

This algebra will be translated into Protein-OODB as follows:

```
select  p.types
from    (select  p
         from   Protein p
         where (noOfHelix(p.proteinName)>
                noOfHelix("HIV-1"))
where   noOfChain(p.proteinName)>
        noOfChain("HIV-1");
```

Protein Wrapper translates algebra according to the order of Protein Algebra constraints without losing any optimization of Protein Algebra. And the detail of translation from Protein Algebra to Protein-OODB will be done automatically by the system to make this design easily be understood.

IV. Related Work

In [7], authors talk about PO ontology algebra that allows multiple diverse sources stored in the protein ontology for future information retrieval. The PO approach provides semantic relationships among multiple sources. This approach allows the users to exploit protein information from different sources, which makes protein data sources integration more scalable.

PRONTO [8] constructs a protein ontology that mines the literature and the data sources. It only represents relationship among protein literatures and does not formalize knowledge about protein process.

In [4], authors introduce a system called Periscope/SQ, which is based on an extension of relational algebra. They define new physical operators and make use of the effective optimization for selectivity estimation of string pattern matching of complex sequence queries.

Genomics Algebra (GenAlg) [1] proposes an approach to expressing complex genomics operations through Genomics Algebra. That approach builds a completely new expressive algebra to present biological operations such as transcribe, translate etc. But there is a still further need to extract protein data information from Genomics Algebra based on data types and operations. Therefore, the algebra in this paper creates new operators and sorts based on GenAlg [1], and apply them

into the protein DSOODBMS to map Protein-QL to Protein-OODB such that it can take advantage of OODB and algebraic optimization to make queries easier and faster.

V. Conclusion and Future Work

The paper presents an algebra architecture that is protein domain specific and provides query optimization. It is three-component architecture, Protein Ontology, Protein Algebra and Protein Wrapper. Protein ontology as a dictionary maps Protein-QL queries to Protein Algebra queries. Protein Algebra extends Genomics Algebra to protein domain and optimizes queries. Protein Wrapper is designed to connect Protein Algebra and Protein-OODB and makes Protein Algebra independent of Protein-OODB.

For the future work, the goal is to make all the mappings and translation in Protein-DSOODBMS be automatically done by using suitable algorithms so that users can easily use the system without any difficult learning. In addition, the Domain Specific Object Oriented DataBase Management System (DSOODBMS) is presently implemented in protein domain, it will be extended to other biological domains such as DNA, RNA and so on including adding other algebras such as DNA algebra, RNA algebra into the Bio-OODBMS. It is also a plan to extend the Protein Domain Specific OODB Management System (Protein-DSOODBMS)
to provide wider services shown in Fig 3 by formulating Protein-OODB into XML format such that the system not only allows users to input and output XML queries, but also provides a few databases for users to choose for their queries which makes it independent of underlying database, therefore users request protein data in object oriented format, but data can be stored in multiple formats such as OODB (such as EYEDB), relational DB (such as MySQL), XML DB or other data storages.

References

[1] J. Hammer and M. Schneider, "The GenAlg project: developing a new integrating data model, language, and tool for managing and querying genomic information," *ACM SIGMOD*, vol. 33, pp. 45-50.

[2] Y. Wang, R. Sunderraman, and P. Phoungphol, "A high level programming environment for protein structure data," *2007 International Symposium on Bioinformatics Research and Applications (ISBRA 2007)*, pp. 215-226.

[3] J. Hammer and M. Schneider, "Genomics Algebra: A new, integrating data model, language, and tool for processing and querying genomic information," *First Biennial Conference on Innovative Data Systems Research*, pp. 176-187.

[4] S. Tata, W. Lang, and J.M. Patel, "Periscope/SQ: interactive exploration of biological sequence databases," *Proceedings of the 33rd international conference on Very large databases, VLDB '07*, 007, pp. 1406-1409.

[5] Y. Wang and R. Sunderraman, "PDB data curation," Engineering in Medicine and Biology Society, 2006. *EMBS '06. 28th Annual International Conference of the IEEE*, 2006, pp. 4221 - 4224.

[6] Y. Wang and R. Sunderraman, "Database management system for protein structure data," *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering*, pp.526-531, 2008.

[7] A.S. Sidhu, T.D. Dillon, and E. Chang, "Ontology algebra for composition of protein data sources," *IEEE 2007*, pp.144-140

[8] I. Mani, Z. Hu, and W. Hu, "PRONTO: a large-scale machine-induced protein ontology," *2nd Standards and Ontologies for Functional Genomics Conference (SOFG 2004)*, UK.

[9] http://www.alphaworks.ibm.com/contentnr/semanticsfaqs

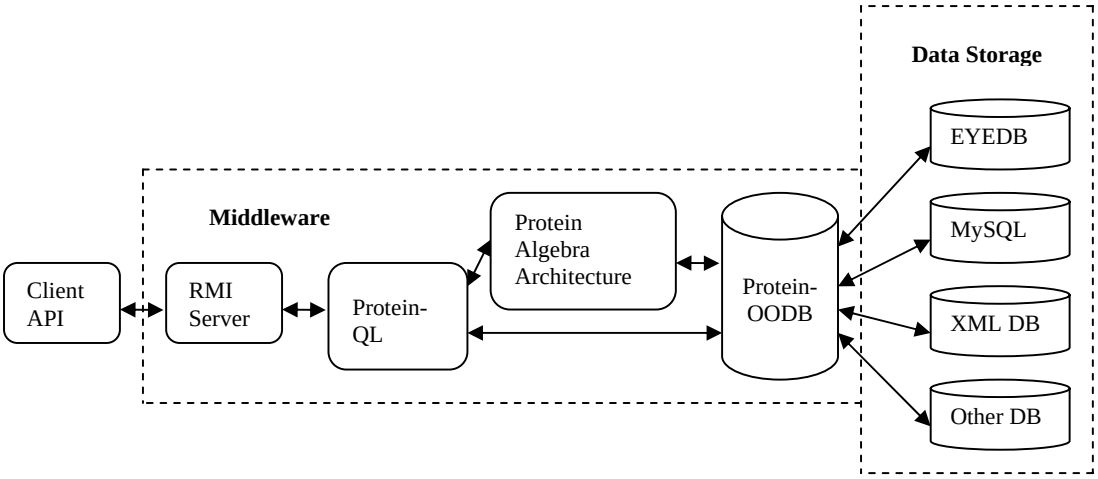

Fig. 3 The architecture of DBMS for protein structure data