INTERACT: an object oriented protein-protein interaction database

[1]Karen Eilbeck, [1]Andy Brass, [2]Norman Paton and [3]Charlie Hodgman.

[1]School of Biological Sciences and [2]Department of Computer Science,
University of Manchester, Oxford Road, M13 9PT, UK
[3]GlaxoWellcome Research and Development, Gunnels Wood Road, Stevenage, SG1 2NY, UK

Abstract

Motivation: Protein-protein interactions provide vital information concerning the function of proteins, complexes and networks. Currently there is no widely accepted repository of this interaction information. Our aim is to provide a single database with the necessary architecture to fully store, query and analyse interaction data.
Results: An object oriented database has been created which provides scientists with a resource for examining existing protein-protein interactions and inferring possible interactions from the data stored. It also provides a basis for examining networks of interacting proteins, via analysis of the data stored. The database contains over a thousand interactions.
Contact: k.eilbeck@stud.man.ac.uk

Keywords: Java, object oriented database, protein-protein interaction, OQL.

Introduction

Protein-protein interactions are intrinsic to every cellular process. They form the basis of phenomena such as DNA replication and transcription, metabolism, signal transduction, and cell cycle control. Understanding the role of a protein within a cell relies on discovering the biological context in which it performs its tasks, so knowing the interactions it makes is vital. Genome sequencing projects have identified many novel proteins, with unknown function, giving impetus to create a resource that makes available for analysis as much information as possible on protein-protein interactions.

Proteins do not act in isolation, but form complexes, networks, and often protein machines (Alberts, 1992) in order to efficiently carry out cellular processes. The ultimate goal of studying protein interactions is to produce protein linkage maps for entire species. The first linkage map of an entire genome was created using the Yeast 2 Hybrid method on Bacteriophage T7 (Bartels et al, 1996). Twenty-two interactions were discovered between the 53 expressed proteins. Linkage maps of larger, more complex genomes can be expected in the future, as rapid experimental procedures such as the Yeast 2 hybrid method are generating vast amounts of interaction data. The information is published, but not stored electronically in a manner suitable for the complexity of the data, as there is no ideal repository. [1]Systematic cataloguing of protein-protein interactions was proposed by Lander (1996), as one of ten vital post genome sequencing strategies.

It is now apparent that the purpose of some proteins is to become a scaffold that holds other proteins in the correct orientation. This has been demonstrated by Cohen et al, 1998, with the discovery of IKAP, a protein which is responsible for the assembly and regulation of kinase complexes. Such proteins are likely to have many interactions, and may reveal a great deal of information about the processes in which they are involved. Protein-protein interactions are also increasingly important to drug discovery, as highlighted recently with the creation of a non-peptidyl mimic of G-CSF (Tian et al, 1998). The activation of the JAK family of protein kinases is triggered by homodimerization of the G-CSF receptors after the binding of G-CSF. The drug mimics the protein-protein interaction, but can be administered orally unlike the original hormone.

Information relating to protein interactions is currently spread through many databases and is not structured in a readily analyzable form. Existing systems have not developed at a satisfactory rate to keep up with the massive influx of interaction data. For example sequence databases such as SWISS-PROT have interaction information hidden in the free text annotation associated with function. This format does not provide a suitable context for analysis and querying.

Other databases have taken a more structured approach to storing interaction information. For example, the Yeast Protein Database (Hodges et al, 1998) has catalogued every expressed protein in *Saccharomyces cerevisiae*. For each protein it has the capability to list the interactions, associations and multi-protein complexes it makes. This database can be queried via properties of the protein, but details about the interactions can only be retrieved through the specific proteins involved. The MIPS database (Mewes et al, 1998) is similar to YPD in the information that it contains, but it stores binary protein interactions in a set of tables that can only be queried by

[1] Copyright © 1999, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

scrolling through to find the gene name of interest. This site also stores a list of 230 multi-protein complexes found in yeast. A prototype database Cyspid (Panzer et al, 1997) exists for the storage of cytoskeletal protein interaction data. Cyspid explicitly represents biological relationships between the interacting proteins in a relational database. The database is navigated via an HTML interface. There are also databases such as Ecocyc (Karp, 1997) which have stored the known metabolic pathway information including enzymes and reactions. This type of database indirectly stores protein protein interactions. However, none of the existing databases have sought to directly answer protein protein interaction questions. They have behaved as repositories for interactions whilst delivering another function. There is an obvious lack of a purpose-built database for protein interactions for both retrieval and analysis of data.

Object oriented database technology provides a means to fully accommodate and query the data associated with protein interactions. Real life biological entities and the relationships they make with each other are inherently complex, as characterized by Barry (1996).The data lacks unique identification and has numerous many to many relationships. For example; a protein may contain many motifs and a motif may occur in many proteins. The modeling facilities, including inheritance and explicit relationships, allow intuitive representations of biological concepts.

Biological data has many of the characteristics for which object oriented databases were initially proposed. Object databases describe data using a collection of classes. Classes define the attributes, relationships and operations of the objects that are their instances. Classes can be arranged in inheritance hierarchies, whereby the properties of a parent class are inherited by its children. The direct support for relationship constructs in the data model, the provision of inheritance, and the close association of programs with data means that applications associated with rich information models and complex analytical processing are well suited for use with such databases.

Although classifying object oriented databases is not an exact science, it is probably fair to say that there are now two principal categories of object database. Object relational databases (Stonebraker, 1996) are extensions to the relational model to include object types and collection types as attributes of tuples. Certain of the features of object relational databases are likely to make them more suitable than their relational predecessors for handling a range of common biological types (e.g. sequences). The other principal category of object database builds on the ODMG industry standard (Cattell, 1997). ODMG compliant systems are not descended from the relational model, and can be seen as providing leaner and more focused support for object modelling in databases. We anticipate that there will be considerable use of systems from both of these categories with biological data. INTERACT makes use of an ODMG compliant database.

A number of other projects that use object oriented databases are described in the literature. An object oriented database for protein structure data is described in Gray (1990), where the emphasis is on supporting ad-hoc queries using a declarative query language. More recently an object oriented database for transcription factor interactions, namely protein-DNA interactions has been described (Ghosh, 1999). This database can be manually interrogated using a query language or via a 'canned query' interface with pre-defined queries.

This paper describes a protein-protein interaction database that stores both experimental data on interactions and derived results from bioinformatics analysis. As well as proposing a software infrastructure for managing, querying and presenting interaction data, it provides an example of the use of the emerging Object Database Management Group (ODMG) (Cattell et al, 1997) standard for object databases in a biological application.

Design

A user requirements analysis was carried out that consisted of both a questionnaire and interviews with scientists directly involved in protein–protein interaction experimentation (academic and industrial). The scientists were asked how they would expect to begin a query with the database, and most commonly they chose to use the name, accession or sequence of a protein of interest, while others wanted to start with the gene. They were then asked what questions they would like to query the database with. A comprehensive list was returned, a sample of which is given in Figure 1. The information ascertained was used to identify a set of use-cases, which are textual descriptions of discrete user goals for the system. These use-cases played an important role in the design of the schema of the database, by indicating some of the information that would be needed in addition to the basic interaction data.

The Unified Modeling Language (UML) (Fowler, M. & Scott, K, 1997) was used throughout the design process, to model the database and provide structure to communication with domain experts. A Class Diagram (Figure 2) for the database was developed to describe the information required to support the use-cases. The model is based on the assumption that an *Interaction* consists of two interacting *Proteins*. An *Interaction* may be described by many *Experiments*, which indicates the method and conditions the interaction was discovered under.

It became immediately apparent from the response of the scientific community that supplementary data needed to be stored with the experimental data, to enhance the range of queries and analyses that can be supported. The schema was refined by adding: *Homologue*, *Motif* and *PDB* classes. The *Homologue* class stores relevant homologue/analogue information relating to the protein of interest. It is derived from an ungapped BLAST (Altschul et al, 1990) result of the original protein, using the SWISS-PROT database, with an expect value of 0.01 and using the BLOSUM62 matrix. *Homologue* thus contains the score and probability as attributes, and has a relationship with both the original *Protein* and the homologue *Protein*. A *Protein* has up to 10 *Homologues* derived from the 10 top scoring hits of the BLAST report. The purpose of this class is to provide a built in answer to the user defined query "Are there any homologues or analogues to the proteins involved in this interaction"?

What part of the protein is involved in the interaction?
Does it interact with itself?
What proteins interact with proteins of a given motif?
What are the key references of this interaction?
What is the location of the interaction?
What other molecules does it interact with?
What type of interaction is it?
What pathway is it involved in?
How strong is the interaction?
Which residues are key?
Are there any homologues/analogues?
Does a crystal structure exist?

Figure 1. Questions people would ask a Protein-protein interaction database, if available, as ascertained during the user requirements analysis.

The class *Motif* has been designed to store information from the popular motif database, PROSITE (Bairoch et al, 1997). This class has the following attributes: *identifier* and *name* to store the PROSITE accession number and name, *type* for kind of motif information (pattern or matrix), and *pattern* for the string to store the actual motif. The scientist can therefore query the database with a known PROSITE motif either by name or accession, or browse the motif information for a given protein, as suggested by the use-case analysis.

The *Protein* class has other essential relationships. It has a collection of *Keywords*, a collection of *PDB*s, and a collection of *Genes*. The protein is associated with a collection of genes rather than a single gene to account for the nature of sequence databases where overlapping coverage in the sequence databases and therefore redundancy is common (Altschul et al, 1994). A *Protein* also belongs to a *GeneFamily*, whereby homologous proteins are grouped together regardless of species. A *Complex* class has also been modelled as a collection of *GeneFamilies* involved and a collection of *GenericInteractions*. The *GenericInteraction* class also refers to the specific instances of *Interaction* that it was derived from.
Another important feature of the design is the way in which inheritance has been used to model references. All of the classes that may have a bibliographic reference inherit from the parent class *ReferencedThing*, which has a many to many relationship with *Reference*. All of the child classes therefore have that relationship. *ReferencedThing* is an abstract class – it is not meaningful to create *ReferencedThing* objects directly, so all instances are direct instances of its children classes.

Implementation

The database has been implemented using the commercial object database POET 5.1, which follows the Object Database Management Group (ODMG) standard. The ODMG standard has been developed jointly by the principal object database vendors, and is being adopted incrementally by products. The standard specifies:

1. *An object data model* that allows definition of classes with attributes and relationships, in conjunction with a range of collection types (sets, lists, bags and dictionaries) and an inheritance mechanism. The ODMG model can be seen as a conventional object model, but one that is shared by several products.

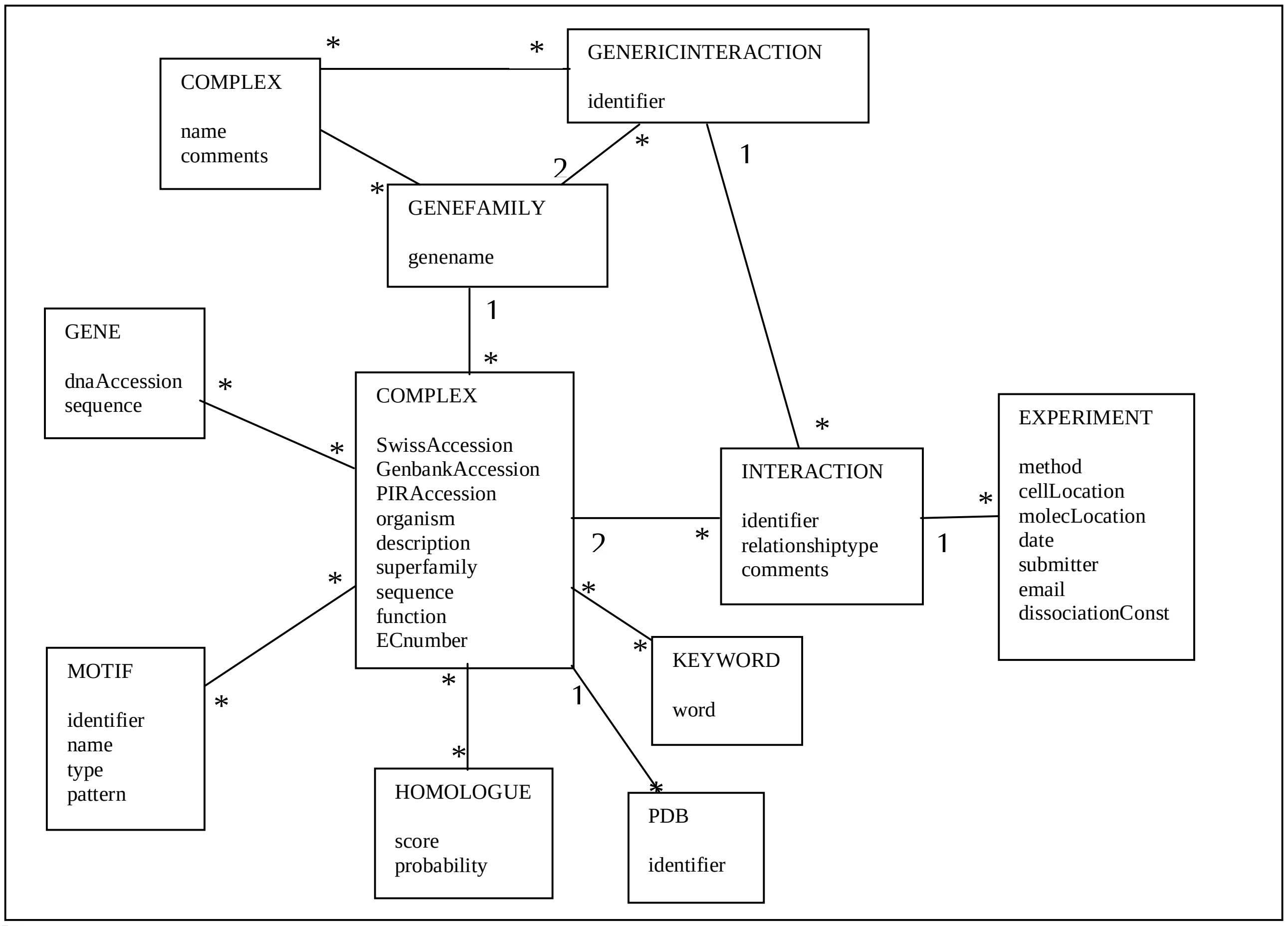
2. *A query language*, OQL, that has a syntax and computational power comparable to that of SQL, but which acts over the richer ODMG model.

3. *A collection of language bindings*, for Java, C++ and Smalltalk. A language binding allows an existing object oriented programming language to access database data (more or less) as if the database consisted of classes defined in the programming language. It also allows the programming language to query the database using OQL.

In the protein-protein interaction database, the UML schema is mapped to a collection of classes in the ODMG Java binding. For example the class *Protein* is declared as:

```
public class Protein extends ReferencedThing{
    // Attributes
    String superfamily;
    String organism;
    String Swissacc;
    String GenBankacc;
    String description;
    String comments;
    String sequence;
    // Relationships
    SetOfObject homologues;
    SetOfObject motifs ;
    SetOfObject interactions;
    SetOfObject keywords;
    Gene dna;
   …}  .
```

(a)



(b)

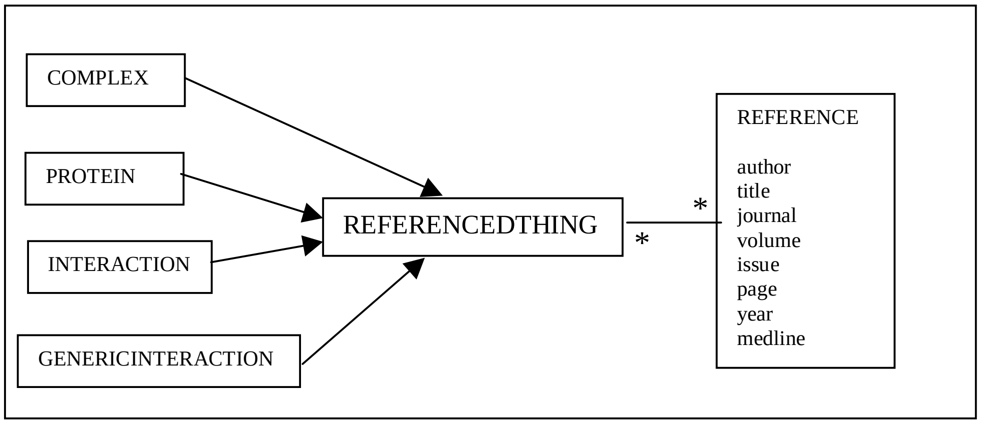


Figure 2. Class Diagram in UML of the INTERACT database. Classes are depicted as rectangles with lists of attributes. Relationships are shown as connecting lines with the cardinality of the relationship given at either end, i.e. two proteins are involved in an interaction and many interactions may involve the same protein. Part (a) presents the bulk of the classes involved in the database, whereas (b) describes the relationship between *Reference* and and the abstract class *ReferencedThing.* The classes *Protein*, *Interaction*, *Complex* and *GenericInteraction* inherit from the class *ReferencedThing, as shown by the arrows.*

The mapping from the UML schema to Java classes is straightforward. Each UML class is represented as a Java class. Attributes of the UML classes map onto attributes of the Java class. Relationships in the UML diagram map onto a pair of attributes, one on each of the related classes, each of which represents one end of the relationship. Instances of *Protein* are created in Java programs, and are stored in the database for future access by queries and programs

Loading the data

Currently, protein-protein interaction data is gathered from three locations:

- The interactions tables provided by MIPS.
- Scientist submission from a WWW form
- Submission from published literature.

Specific interaction information is automatically retrieved from the MIPS database. The supplementary information provided at this site, such as experimental method and references, is also retrieved where available. A suite of programs has been developed to retrieve the interaction data, supplement it with value added information from other sources, and populate the database (Figure 3). For each binary interaction entry in the MIPS interaction tables, the program interrogates the external sources. A BLAST search is performed to find the top ten best hits, which become instances of the class *Homologue*. SWISS-PROT is searched for much of the subsidiary information relating to a protein, such as its keywords and accession numbers, as well as for the links to PROSITE and MEDLINE. PROSITE is accessed for relevant motif information, and MEDLINE is searched for references.

The external bioinformatics sources used to provide supplementary data for the database have been "wrapped" in Java (Figure 3) to allow the analysis to be performed cleanly from within a Java program using the *HTTPUrlConnection* class. Wrapping the sources in Java means that the load program benefits from a consistent view of both the data sources and the database being populated – both look like Java objects. In addition, the Java wrappings will make it easier to cope with changes to the interfaces of sources, and should allow the database to adapt easily to use CORBA wrapped sources when these become available.

Querying the database

The INTERACT database has been developed principally to support the querying and analysis of interaction data. There are three principal ways in which queries or analyses can be expressed against the database:

1. *Canned queries*. The use cases identified during the requirements analysis phase have been used to direct the development of a form-based interface in Java. The top level window in this interface consists of a scrolling list of questions. Selecting a question initiates an additional request to the user for parameters for the question, subsequent to which the question is evaluated, and the user is presented with the results of the query in a form-based browser.

2. *Interactive queries*. The Object Query Language (OQL) can be used to express declarative requests for information from the database. For example, the following OQL query finds all of the proteins interacting with proteins containing the PROSITE motif *PS00188*.

```
SELECT p
FROM p in ProteinExtent, i in p.interactions,
     p1 in i.proteins, m in p1.motifs
WHERE m.identifier = "PS00188" AND p != p1
```

The result of this query is the protein *p*, where: *p* is in the collection of instances of the class protein (this is known as *ProteinExtent*); *i* is one of the *interactions* in which *p* participates; *p1* is the other protein involved in the interaction *i*; *m* is one of the motifs in *p1*; and the identifier of *m* is *PS00188*.

For readers familiar with SQL, the basic structure of an OQL query should be familiar. The *SELECT* clause indicates what appears in the result of the query, the *FROM* clause indicates what collections are used to answer the query, and the *WHERE* clause imposes some restrictions on the values of interest. The principal difference from SQL is that this query contains no value-based joins – the relationships between objects are explored by naming relationship attributes in the *FROM* clause.

```
DEFINE yeast AS
SELECT p
FROM p IN ProteinExtent, h in p.homologues,
p1 in h.originalProtein
WHERE  p1.organism = "SACCH*" AND h.intScore
> 1000;
    SELECT *
    FROM p IN yeast
    WHERE    p.organism    =    "HOMO*"    AND
p.description =<CI> "*dna*"
```

This second example is a nested query that finds the collection of proteins that are human homologues of the Yeast proteins with a BLAST score of above a threshold of 1000, and have the word 'DNA' in their description. It demonstrates the used of the DEFINE statement to nest the query, and a case insensitive qualifier used whilst comparing strings.

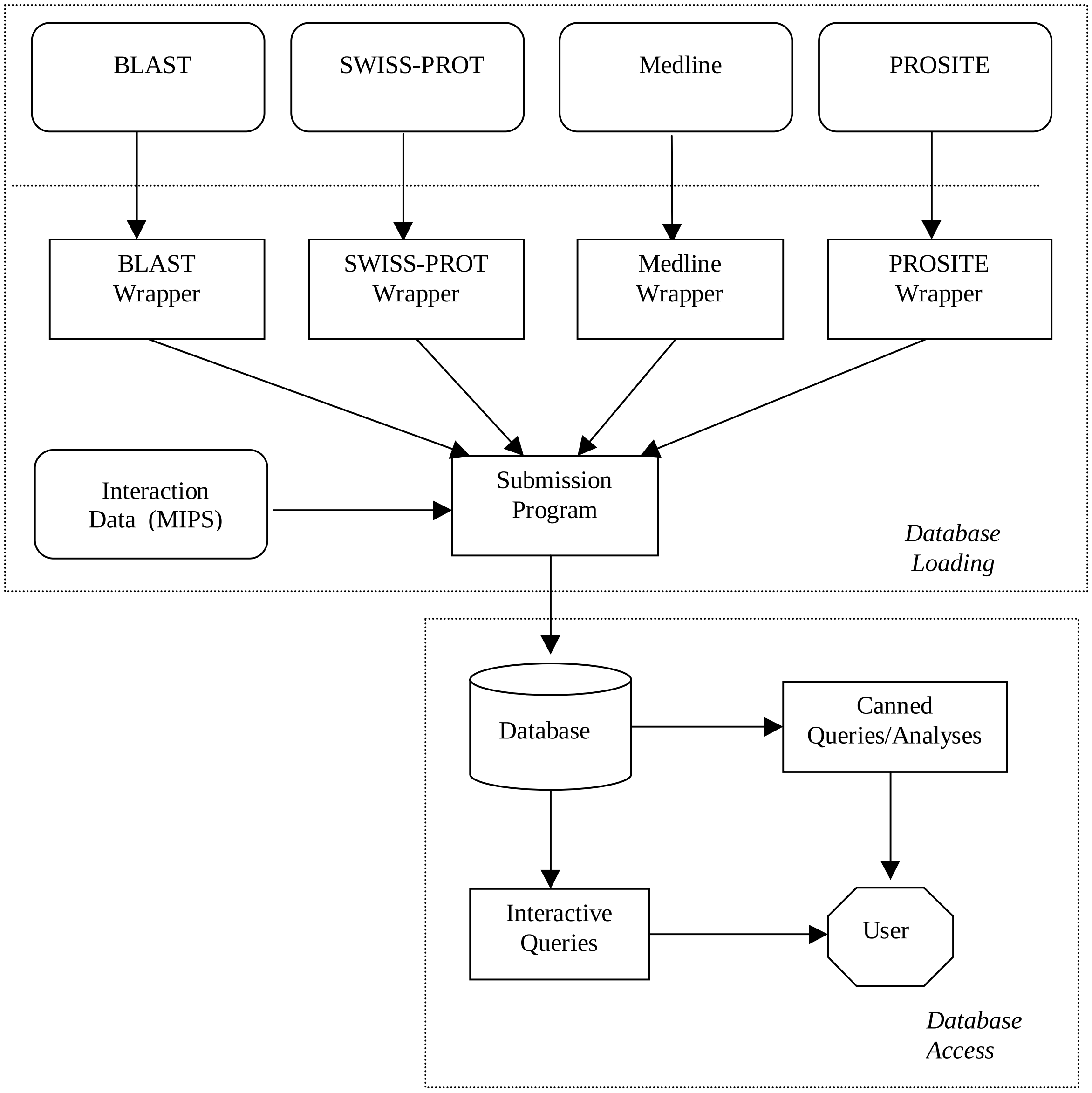


Figure 3.
An overview of the architecture of the submission system and user access to the database.

Analysis programs.

As the database can be accessed directly from Java using a much cleaner interface than is generally available for accessing relational databases from Java, INTERACT provides a convenient setting for the development of more complex analysis programs.

As an example of analysis, a clustering program further analyzes the interactions reported for yeast to obtain networks of interacting proteins. An adjacency matrix is created of the interactions, where each axis contains all of the yeast proteins in the database. A depth first search is performed over the matrix, where interacting neighbours of a protein are found sequentially, and then interacting neighbours are searched for in turn. This method revealed 92 networks, the largest of which contained Actin and has 232 members. An example of a smaller cluster of proteins involved in exocytosis is displayed graphically in Figure 4. The use of *graph theory* methods can also be extended to detect any pathways of interaction between any two seemingly non interacting proteins.

The database has proven to be a valuable tool for determining putative interactions in species other than yeast. An analysis program compares all of the proteins involved in yeast interactions to other genomes, using

BLAST to determine analogous interacting partners. The interactions have thus been mirrored in *C. elegans*, *S. Pombe* and human, where cross species analogies can be drawn. Potentially the most useful comparison is that with *C. elegans* as it is also a fully sequenced genome (C.elegans Sequencing Consortiun, 1998). It has been shown by Chervitz et al(1998) that comparison of protein sequences between yeast and the nematode revealed a one to one mapping of core functional proteins, suggesting yeast to be an excellent model organism for such study. For interaction data, it can therefore be assumed that if an anaologue of a yeast interaction is found in *C. elegans* it is most likely to be part of a fundamental eukaryotic process. This resource can be found on the INTERACT web site:
http://bioinf.man.ac.uk

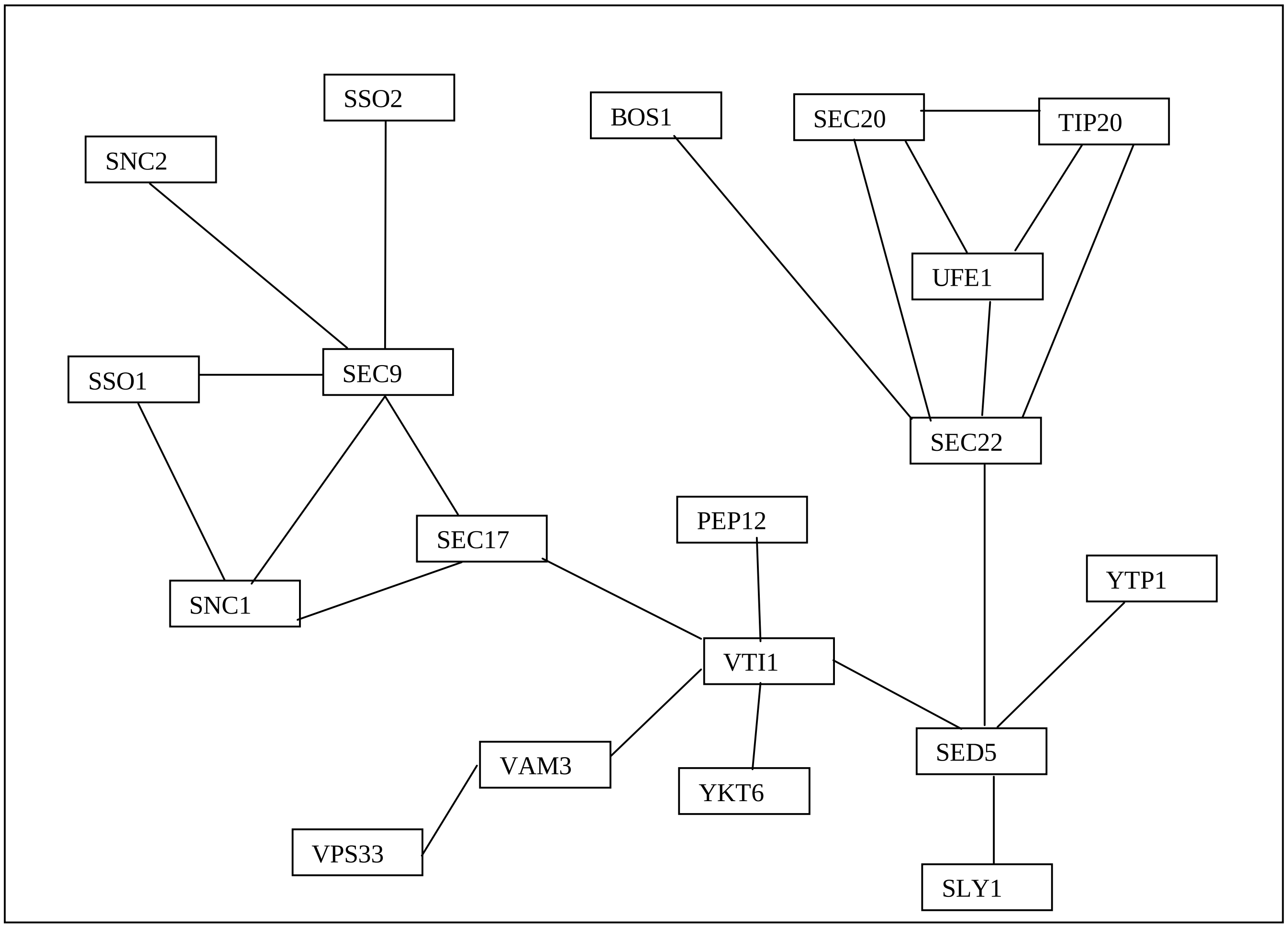


Figure 4.
A graphic representation of a network of proteins involved in secretion. The left hand side of the diagram displays proteins mainly involved in vesicular transport, whereas the right side is involved in targetting and protein transport between the endoplasmic reticulum and the golgi.

Conclusions

An object oriented database has been built for storing data on protein-protein interactions and multi-protein complexes. It is an advance over previous methods of interaction storage because of the richness of the data contained in the database and the comprehensive querying and analysis facilities provided.

Each protein involved in an interaction is supplemented with information derived from other databases such as PROSITE and PBB. Every interacting protein undergoes a BLAST search and the top ten homologues with a expectation of less than 0.01 are retained. They serve to add value to the data, whereby analogies can be drawn from the information contained. It can be postulated that homologues of the interacting protein may also make analogous interactions. This is particularly useful for cross species analysis, and as the majority of interactions currently stored belong to yeast, analogies can be drawn to other eukaryotes.

The database allows many experiments to be catalogued for each interaction. This is important, as a single result is not always conclusive, and may be confirmed by further experimentation. A yeast two hybrid result plus an affinity chromatography result is more believable that the former alone due to the risk of false positives.

The web-based browser provides biologists with the opportunity to view data via a structured route through the database. It allows many of the questions identified in the user requirements analysis to be answered through straightforward interactive dialogues. The canned query interface allows pre-written complex queries to be asked of the data. These methods make the data held readily accessible to the scientific community.

The database provides extensive facilities for querying and analysing the data in the database. OQL queries can be expressed through an interactive interface, or invoked from Java applications, which view the database as consisting of a collection of Java classes. Such analyses may predict new networks, pathways or complexes. Currently no other store of protein interactions is so closely integrated with such comprehensive analysis facilities.

Acknowledgement

This work is funded by the BBSRC and GlaxoWellcome through a CASE studentship.

References

Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D. 1990.Basic Local Alignment Search Tool. Journal of Molecular Biology, 215: 403-410.

Altschul, S., Boguski, M., Gish, W., Wootton, J. 1994. Issues in searching molecular sequence databases. Nature Genetics, 6:119-129

Alberts, B., Maike-Lye, R. 1992. Unscrambling the puzzle of biological machines. Cell, 68, 415-420.

Bairoch, A, Bucher, P, Hofmann, K. 1997. The PROSITE database, its status in 1997. Nucleic Acids Research, Vol.25, No.1, pp.217-221

Bartels, P.L., Roeklein, J.A., Sengupta, D., Fields, S. 1996 A protein linkage map of Escherichia coli bacteriophage – T7. Nature Genetics 12: 72-77.

The C. elegans Sequencing Consortium.1998. Genome sequence of the nematode C. elegans: a platform for investigating biology. The C. elegans Sequencing Consortium. Science;282(5396):2012-8

Cattell, R., et al, 1997. The Object Database Standard: ODMG2.0 .Morgan Kaufmann.

Chervitz SA, Aravind L, Sherlock G, Ball CA, Koonin EV, Dwight SS, Harris MA, Dolinski K, Mohr S, Smith T, Weng S, Cherry JM, Botstein D 1998. Comparison of the complete protein sets of worm and yeast: orthology and divergence. Science;282(5396):2022-8

Cohen, L., Henzel, W., Baeuerle, P. 1998. IKAP is a scaffold protein complex of the IkB kinase complex. Nature 395: 292-296.

Fowler M. & Scott K. 1997 UML Distilled. Addison Wesley.

Ghosh, D.1999. Object oriented Transcription Factors Database(ooTFD). Nucleic Acids Research, 27:315-317.

Gray, P.M.D., Paton, N.W., Kemp, G.J.L. and Fothergill, J.E. 1990. An Object-Oriented Database for Protein Structure Analysis. Protein Engineering, Vol 4, No 3, 235-243.

Hodges, P., Payne, W. E. and Garrels, J. 1998. The Yeast Protein Database (YPD): a curated proteome database for Saccharomyces cerevisiae. Nucleic Acids research, 26: 68-72.

Karp, P., Riley, M., Paley, S., Pellegrini-Toole, A. 1997. Ecocyc: Electronic Encyclopedia of E. coli genes and metabolism. Nucleic Acids Research. 25 (1)

Lander, S. 1996. The new genomics: gobal views of biology. Science 274:536-539.

Mewes HW, Hani J, Pfeiffer F, Frishman D 1998. MIPS: a database for protein sequences and complete genomes. Nucleic Acids Research 26: 33-37.

Panzer, S., Cooley, L. & Miller, P. 1997. Using explicitly represented biological relationships for database navigation and searching via the World-Wide Web. CABIOS 13: 281-290.

POET Java SDK Programmer's Guide 1997. POET Software Corporation, San Mateo and POET Software GmbH, Hamburg.

M. Stonebraker and D. Moore1996. Object-Relational Databases - The Next Great Wave, Morgan-Kaufmann,

Tian, S., Lam, P. 1998.A small non peptidyl mimic of Granulocyte Colony Stimulating Factor. Science 281:257 – 259.