BIOINFORMATICS

Vol. 17 no. 5 2001
Pages 468–478

BioMolQuest: integrated database-based retrieval of protein structural and functional information

*Yury V. Bukhman and Jeffrey Skolnick**

Laboratory of Computational Genomics, Donald Danforth Plant Science Center, 893 N. Warson Rd, St Louis, MO 63141, USA

Received on July 12, 2000; revised on November 13, 2000; accepted on January 22, 2001

ABSTRACT

Motivation: Information about a particular protein or protein family is usually distributed among multiple databases and often in more than one entry in each database. Retrieval and organization of this information can be a laborious task. This task is complicated even further by the existence of alternative terms for the same concept.

Results: The PDB, SWISS-PROT, ENZYME, and CATH databases have been imported into a combined relational database, BioMolQuest. A powerful search engine has been built using this database as a back end. The search engine achieves significant improvements in query performance by automatically utilizing cross-references between the legacy databases. The results of the queries are presented in an organized, hierarchical way.

Availability: http://bioinformatics.danforthcenter.org/yury/public/home.html. Unavailable for commercial users.

Contact: skolnick@danforthcenter.org

INTRODUCTION

The number of proteins whose structures have been experimentally determined is increasing at a fast pace. This growing abundance of structural information makes it possible to study the complex relationships between macromolecular structure and biological function. Proteins that have a similar fold often have a similar function (Creighton, 1993). However, the relationship between structure and function is not necessarily straightforward. There are cases when two proteins with similar folds have different functions or proteins having similar functions adopt different folds (Creighton, 1993). In particular, there is strong evidence that the function of an enzyme may be determined by a local structural motif rather than the global fold (Fetrow and Skolnick, 1998). A well-known example of this are serine proteases. Bacterial and eukaryotic serine proteases have similar active sites but very different global structures (Branden and Tooze, 1991). Further elucidation of the relationships between protein structure and function remains one of the most important problems in biology (Thornton *et al.*, 1999). The exploration of this problem will benefit from advances in the area of bioinformatics (Kanehisa, 2000), which should provide for the convenient retrieval of structural data in conjunction with related information about biological function and the classification of protein folds.

The primary source of protein structure information is the Protein Data Bank (PDB) (Berman *et al.*, 2000), where the atomic coordinates of the solved protein structures are deposited. While the PDB also contains some functional annotation at various levels of description, unfortunately there are several problems, which make its use somewhat problematic. First, the annotations are provided by authors of the PDB entries, and are therefore often somewhat inconsistent. This problem is exacerbated by the fact that there exist multiple, alternative names for many proteins. The Data Uniformity project under way at RCSB may alleviate this problem (see http://www.rcsb.org/pdb/latest_news.html#Uniformityplan). However, at the time of writing, the project has not been completed. Second, there are often many structures of the same protein that differ in crystallization conditions, point mutations, and other details. Each of these structures is listed as a separate PDB entry. It is not immediately clear from the result of a query whether different hits represent the same protein or different ones. Finally, it is often of interest to retrieve the structural classification of the proteins in addition to the functional annotation. There are several distinct systems of structural classification (Murzin *et al.*, 1995; Holm and Sander, 1996; Orengo *et al.*, 1997), and there exists a separate database for each of them. Thus a scientist faced with a list of PDB entries as a result of a keyword query generally does not know how many distinct proteins are recovered, whether some structures of interest are missing due to the use of alternative protein names in the annotation, or what the fold types of the proteins that were found are. Answering these questions requires additional analysis and necessitates the use of other web sites.

It would be beneficial to integrate experimental information about biomolecular structures contained in the PDB with more consistent sources of functional

*To whom correspondence should be addressed.

annotation and structural classification. In the contemporary world of bioinformatics, the information about structure, classification, and the biological function of each protein or family of proteins is spread over many database entries in several databases, forming a web of data in which it is all too easy to get lost. In order to help researchers access information efficiently, a good retrieval system is necessary which should ideally perform two tasks:

- search several databases simultaneously;
- output an organized summary of the information that is currently available for the protein of interest.

A few powerful search engines, most notably SRS (Etzold *et al.*, 1996) and DBGet/LinkDB (Fujibuchi *et al.*, 1998), have been developed to address the first of these tasks. They allow the user to search multiple databases simultaneously. Moreover, one can cross-link results of a search to other databases to retrieve more information on proteins of interest. However, the result of an SRS or DBGet/LinkDB query is generally a flat list of database entries, leaving one wondering how many distinct proteins or protein families have been retrieved, what are the groups of entries describing the same protein, or whether some entries of interest have been missed due to alternative annotation terms.

Both SRS and DBGet/LinkDB use original biological databases in their flat file format and operate by creating and searching indexes of these files. This approach has allowed the development of very robust and extensible systems capable of handling dozens of different databases. However, the ability of these systems to perform complex queries on the data is somewhat limited. A more comprehensive solution to the data integration problem is to import the databases of interest into a data warehouse, that is, a single large database which may utilize, for example, a relational database management system. The databases imported into the warehouse are usually referred to as 'legacy' databases. Cross-references between legacy database entries can be used to assemble all of the available information about a given protein. This information can be queried, retrieved, and organized as a whole, which allows both flexible querying capabilities and the generation of informative reports. The user can be relieved of a significant part of the data analysis and assembly work. The developers can take advantage of industry-standard database technologies, a wide choice of open source and proprietary database management systems, and the SQL query language.

Although data warehousing is used widely in industry (see e.g. Anahory and Murray, 1997), its application in academic science has been problematic. The main limitations of large data warehouses are their complexity and high cost. That is probably why they are often considered impractical for systems that aim to provide access to a large number of diverse biological datasets (Fujibuchi *et al.*, 1998). Nevertheless, smaller systems serving communities of users interested in a particular research area may still be feasible and useful. The rapid progress of open source database management systems that are free for non-commercial use makes it even more tempting to explore the advantages that moderate-scale data warehouses may have to offer.

One data warehouse type system in structural biology is 3DinSight (An *et al.*, 1998), which includes the PDB along with the SWISS-PROT and PIR sequence databases (Barker *et al.*, 2000) as well as information about functional sequence motifs (PROSITE) (Hofmann *et al.*, 1999), enzymes, reactions, and metabolites (LIGAND) (Goto *et al.*, 2000), etc. A certain degree of data integration and concerted data retrieval from different sources is achieved. In particular, one can run SQL queries on a relational database including PDB, SWISS-PROT, PIR, PROSITE, and the PMD mutant database (Kawabata *et al.*, 1999). 3DinSight provides a form-driven interface as well.

Another interesting data warehousing project is InterPro (Apweiler *et al.*, 2000), which integrates data from several databases of protein domains and sequence motifs. The entries from those databases are merged, and new InterPro entries are created which contain information derived from the original databases. This thorough approach to data warehousing can add a lot of value to the resource, but it is very labor-intensive.

We have attempted to create our own warehouse of information that may be of interest to those researchers who study relationships between the three-dimensional structure and biological function of proteins. The legacy databases that we have initially integrated include PDB, SWISS-PROT, ENZYME, and CATH (Berman *et al.*, 2000; Bairoch and Apweiler, 2000; Bairoch, 2000; Orengo *et al.*, 1997). PDB is the primary source of biological structure information, SWISS-PROT and ENZYME provide a thorough and consistent functional annotation, and CATH provides information about folding domains and their classification. Other databases of interest to the structural biology community, as well as information about protein structures predicted by our own group, are to be added in the very near future. 3DinSight does not include CATH or any equivalent, but it does have other data sets and tools that we do not provide so far, thus our two services may in part complement each other.

A distinctive feature of our service is a powerful search engine that allows users to search all of our legacy databases at once from a simple query form and produces its output as a logically organized report. The search engine automatically, by default, utilizes inter-database cross-references to ensure a thorough retrieval of all

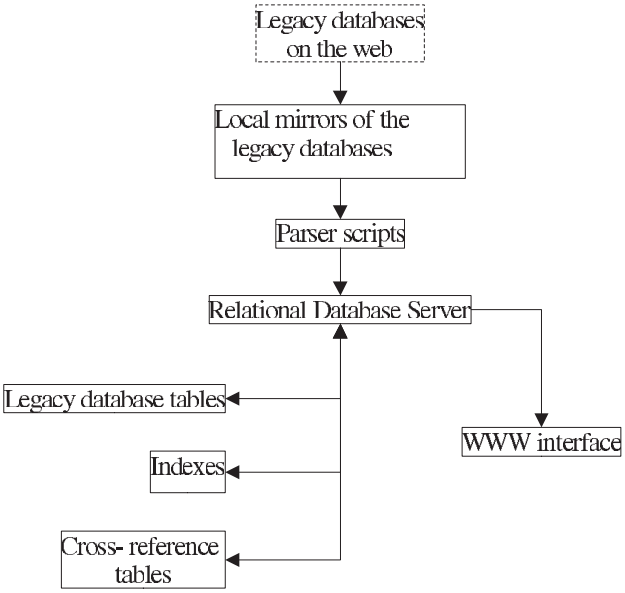

Fig. 1. BioMolQuest system overview. Components are represented by boxes, information flow by arrows.

information available about the protein or group of proteins of interest to the user. The cross-referencing can be turned off when it is inappropriate. Below, we show that the strategy of using cross-references may lead to a significant enhancement of the query performance. We hope that this project will provide a valuable service to the scientific community, serve as a platform for further development, and contribute to the ongoing search for better ways to organize and retrieve biological information.

DATABASE DEVELOPMENT AND ADMINISTRATION

System overview

The components of BioMolQuest are shown schematically in Figure 1. Most of the software was built using the Perl scripting language (Wall *et al.*, 1996). We also use the MySQL relational database management system, which is known to be fast, reliable, capable of handling large data sets, and free for non-commercial use (DuBois, 2000). Perl has a broad community of users who have developed a multitude of modules implementing object classes that can be plugged into Perl programs. We use DBI and related modules for communication between our Perl programs and the MySQL database server, the CGI module to create HTML output, and modules derived from DBIx::TextIndex for creating and searching indexes of textual database fields (see http://www.perl.com/CPAN-local/modules/index.html).

The legacy databases are downloaded via ftp using the Mirror package (http://www.sunsite.org.uk/packages/mirror/). The legacy database records are then imported into a relational database. The resulting database consists mostly of text fields, which are indexed to allow faster searching. The entries of the legacy databases are cross-referenced to each other. A separate Perl program provides the BioMolQuest search engine and a web interface.

Data organization

A legacy database entry usually consists of a set of records listed sequentially in a text file. The records contain specific kinds of information, for instance biochemical classification, organism, literature references, etc. We map each entry to a row in a relational database table, and the records to the columns. Some records contain lists or tables: these are often mapped to separate relational tables cross-referenced to the entry table by the entry identifier. Thus each legacy database gives rise to a corresponding set of tables in the warehouse.

In order to allow query across multiple databases, we maintain a special set of tables cross-referencing entries from different legacy databases to each other. The tables list primary identifiers from one database and corresponding primary identifiers from the other. These tables can be used in SQL joins to retrieve information from two databases about the same protein or simply as a ready, comprehensive source of cross-references for a search engine.

Data warehouses often integrate data more tightly by actually merging legacy database entries into new entities. This approach may add value to the data and speed up database queries, but it is much more labor-intensive. Besides, it largely makes sense when entries from different legacy databases represent the same kind of object, such as sales of the same product or sequence and various properties of the same protein chain. In our case, however, ENZYME entries represent classes of proteins, SWISS-PROT entries represent individual distinct chains, PDB entries represent structures that may contain more than one distinct chain, etc. Keeping these clearly different types of objects separate while maintaining cross-references between them seems the most simple and natural design approach.

Parsing database files

We use custom Perl scripts to parse database files, whereby each particular database field is usually processed by a subroutine. Depending on the nature of the field, its entire contents or the information extracted from it can be put into a scalar variable, array, or hash. These variables are mapped to the corresponding relational database fields. Perl's support for various nested lists, particularly arrays of hashes, makes it easy to create data structures that fit the original data in a natural way.

The format of database files is not always consistent. This is particularly true of the PDB, where the official format is not always strictly observed, and the format specification itself has been changing over time. While efforts are under way at the RCSB to correct this problem (see http://www.rcsb.org/pdb/latest_news.html#Uniformityplan), it has not been completely rectified at the time of writing. We try to adjust our parsers to the most common format variations. For example, PDB files have COMPND records where the information about biological molecules making up the structure is presented. In the contemporary PDB format, the COMPND record is subdivided into sub-records specifying the name of each macromolecule, corresponding chains, Enzyme Classification (EC) numbers, etc. These sub-records are marked by special identifiers, called 'tokens'. In many older PDB files, the tokens are missing, and the COMPND record contains simply the name of the macromolecule, sometimes followed by an EC number. Our PDB parser takes this situation into account. In those cases where the tokens are absent, it assumes that the entire COMPND record represents the name of a single molecule and possibly an EC number. It attempts to extract the latter using an appropriate regular expression. It also assumes, perhaps somewhat riskily, that all of the chains belong to the molecule named in COMPND. To make matters even more interesting, sometimes the COMPND record is altogether absent, in which case the parser assumes that the name of the molecule is contained in the HEADER record.

Of course, it is virtually impossible, as well as dangerous, to try to account for all possible deviations from a database's format specification. The fallback strategy is to assign NULL values to those variables that are not reported in the expected format. For example, the resolution of PDB structures obtained by diffraction methods is supposed to be reported in REMARK 2 record, in columns 23–27, as a real(5.2) number. Our parser also accepts the real(5.1) format, as resolution is often reported that way. No other resolution formats or places in the file where it might be reported were accepted at the time of writing. If a protein structure has been solved by a diffraction method, one might expect resolution to be reported. At the time of writing, there were 7552 PDB entries that contained the word 'diffraction' in the specification of their experimental technique. Only 12 of them had their resolution set to NULL by our parser. 11 of those did actually have resolution reported in one form or another, so some loss of information did occur, but it was not very significant.

Indexing text fields

Much of the annotation contained in the molecular biology databases is in the form of English text. Direct searches of relational database fields containing textual data are usually inefficient. The data need to be indexed, and the searches need to be run on those indexes. At the time of writing, the MySQL database management system did not provide an adequate text indexing and searching capability. Fortunately, this problem was addressed by a few Perl modules. We used one of them, DBIx::TextIndex, developed by Daniel Koch and freely available at the CPAN archive of Perl modules (see http://www.perl.com/CPAN-local/modules/by-module/DBIx/DBIx-TextIndex-0.04.readme).

In order to better suit the needs of BioMolQuest, we have modified DBIx::TextIndex rather extensively in the following ways:

- split the module in two: one for building indexes, and a separate one for searches;
- changed the parser subroutine to ensure better handling of complex chemical names;
- changed SQL searches and related operations to allow word completion in the queries;
- dropped calculations of result scores.

The last modification makes the code considerably simpler and faster, albeit less sophisticated. The BioMolQuest searches are done on the per-database-field basis. The scoring of results is usually unimportant, because most of these fields do not contain more than a few words. Besides, the results are still output in a logical order, which follows from the order in which the fields are searched, e.g. hits in the headers come before hits in the comments.

Cross-referencing legacy database entries

In order to search and retrieve all the information about a particular protein, one must be able to cross-reference entries from the legacy databases to each other. As discussed above in Section Data organization, we maintain special tables of cross-references for different pairs of the legacy databases. These tables list primary identifiers from one database and corresponding primary identifiers from the other. The tables are rebuilt from scratch each time the legacy databases are reloaded. The cross-reference information is systematically extracted from each of the legacy databases, calculated where necessary, and merged to make sure that each set of cross-references is as complete as possible.

SWISS-PROT entries have a special record (DR) cross-referencing them to a number of other databases. PDB entries usually have DBREF records cross-referencing their chains to sequence databases, most often to SWISS-PROT. In cases where other sequence databases are listed in the DBREF records, it is sometimes possible to derive a cross-reference to SWISS-PROT. For example, if both

a PDB entry and a SWISS-PROT entry refer to the same PIR entry, they likely contain information about the same protein. Unfortunately, this device cannot be used with cross-references to the EMBL/GenBank nucleotide sequence database (Benson *et al.*, 2000; Baker *et al.*, 2000), because many of its entries encompass multiple Open Reading Frames.

Both the PDB and SWISS-PROT can be cross-referenced to ENZYME using EC numbers often listed in their entries. Additionally, lists of SWISS-PROT entries are found in ENZYME records. PDB entries in which EC numbers are omitted are cross-referenced to ENZYME using information in the corresponding SWISS-PROT entries.

Sometimes, direct cross-references between two databases are not available. For example, there are no cross-references between SWISS-PROT and CATH. However in a data warehouse environment, one can reconstruct this missing information using simple queries. CATH domain identifiers contain the PDB chain designation. PDB chains are, in turn, cross-referenced to SWISS-PROT. Using these two pieces of information, a CATH/SWISS-PROT cross-referencing table, albeit a somewhat incomplete one, can easily be created.

Many PDB structures are of macromolecular complexes that contain a set of different protein chains. Fortunately, both DBREF records cross-referencing PDB to sequence databases and a significant part of the annotation, i.e. COMPND and SOURCE records, refer to individual chains or lists thereof. We try to keep track of cross-references between PDB and other databases on the level of individual chains whenever possible. Thus, cross-referencing tables that include PDB list both PDB IDs and chain identifiers, represented as PDB ID plus the chain letter.

Keeping the warehouse up to date

In order to build our warehouse, we mirror the legacy database ftp archives to a local archive disk, parse the files into the MySQL database, index the textual data, and rebuild the cross-reference tables. All of these operations are performed in a fully automatic mode by Perl scripts. A master Perl script invokes each of these scripts in an appropriate order. If any of the scripts fails, the execution of subsequent steps is aborted, and an e-mail message is sent to the database administrator. The administrator must then fix the problem before the update is resumed. The master script is run as a cron job once a week, which is the update frequency of PDB and SWISS-PROT.

The database exists in two copies: an internal copy and a public copy, kept on a separate machine, which runs a public web server. The scripts that load data into the database operate on the internal copy. After an update is completed, the public copy of the database is quickly replaced by the internal copy, thereby ensuring that the update-related disruption in the operation of the web server is kept to a minimum.

THE BIOMOLQUEST SEARCH ENGINE

User interface

The search engine has a simple query form similar to those of web search engines (not shown). The user can choose what to search for from a pull-down menu. The default choice of 'Any keywords' lets the user search virtually the entire textual annotation from all databases. Other choices include SWISS-PROT or PDB identifier, EC number, biomolecule name, organism, heterogen in a PDB entry, resolution of a PDB entry, and length of a chain in the PDB. Most of these search fields correspond to several fields in the underlying database. Comparison operators >, <, =, >=, <= are used in resolution or chain length queries. If there are two or more query words, the user can use a radio button menu to make the search engine match all query words, any of the words, or the entire phrase. The 'Advanced search string' option allows the user to use an expression of his or her own design.

Results of the BioMolQuest search for 'ribonuclease T1' with default options are listed in Table 1. The search results are generally organized as a set of trees. The root of each tree is a 'hit' which was found in one of the legacy databases, for example, by keywords. A hit may represent an individual protein chain, as in SWISS-PROT and many PDB entries, or a group of protein chains, such as a class of enzymes in the ENZYME database, a class of structural domains in CATH, or a complex of several co-crystallized distinct chains in the PDB. A hit is followed by an associated SWISS-PROT entry or entries, if they are found. SWISS-PROT strives to be a non-redundant database of protein chains (Bairoch and Apweiler, 2000), so we assume that each SWISS-PROT entry corresponds to a distinct chain. This chain can have one or more structural domains and its structure may have been reported in one or more of the PDB entries. The folds of these domains (from CATH) and the PDB entries are reported for each SWISS-PROT chain. Since this search engine is intended mostly for scientists interested in structural biology, the hits and SWISS-PROT chains that do not have any PDB entries associated with them are not reported by default. This behavior can be changed at the advanced query page (not shown).

The search results are displayed as HTML tables, with relationships between hits and the corresponding SWISS-PROT, CATH, and PDB entries clearly indicated. The results are hyperlinked to the legacy database web sites and the particular legacy database entries.

Table 1. Results of the BioMolQuest search for 'ribonuclease T1'

Hit	SWISS-PROT	PDB					
		1rn4	1rga	1rls	1gsp	3hoh	7gsp
		1rnt	1rgc	1rpf	1rhl	4bir	1fys
	RNT1_ASPOR	2rnt	1rgk	1rpg	1ygw	4bu4	1fzu
		3rnt	1rgl	1rph	2bir	4gsp	1g02
	GUANYL-SPECIFIC	4rnt	1rn1	1bu4	2bu4	4hoh	
	RIBONUCLEASE T1	5rnt	2aad	1b2m	2gsp	5bir	
	PRECURSOR	6rnt	2aae	1bir	2hoh	5bu4	
		7rnt	1lra	1bvi	3bir	5gsp	
ENZYME:		8rnt	1trp	1ch0	3bu4	5hoh	
3.1.27.3		9rnt	1trq	1det	3gsp	6gsp	
Ribonuclease T1	RNMS_ASPSA						
	GUANYL-SPECIFIC	1rds	1rms				
	RIBONUCLEASE MS						
	RNSA_STRAU	1sar	1gmq	1rge	1rgh		
	GUANYL-SPECIFIC	2sar	1gmr	1rgf	1rsn		
	RIBONUCLEASE SA	1gmp	1ay7	1rgg			
	RNF1_GIBFU	1fus	1rck				
	GUANYL-SPECIFIC	1fut	1rcl				
	RIBONUCLEASE F1						
		1b20	1b2s	1b2z			
	N/A	1b21	1b2u	1b3s			
		1b27	1b2x	1box			
PDB: 1aqz	RNMG_ASPRE						
Ribotoxin	RIBONUCLEASE	1aqz					
	MITOGILLIN PRECURSOR						

The query was run on December 14, 2000. CATH domains are not shown.

Search mechanism

The outline of the searching algorithm. A user query is applied to each database column index in the following general order: ENZYME classes, fields of ENZYME entries, fields of SWISS-PROT entries, fields of PDB entries, and CATH domain classes. The idea behind this order of queries is to go from classes of proteins described in ENZYME to individual non-redundant protein chains contained in SWISS-PROT, to structures of these chains contained in the PDB, and finally to domain descriptions. Each time the query finds a database hit along this path, the search engine uses cross-referencing tables to retrieve pointers to all information about the protein chain or chains covered by the hit. The outline of the search algorithm is shown below:

1. Search each database field in the order indicated above
    2. For each hit:
        3. If found previously (e.g. by cross-reference to a previous hit), then reject
        4. Else use cross-reference tables to do the following:
            5. Retrieve all related SWISS-PROT entries
            6. For each SWISS-PROT entry:
                7. Retrieve CATH domains
                8. Retrieve PDB structures
            9. Retrieve related PDB entries not found in 8
        10. End if
    11. Next hit
12. Next field

Careful use of cross-references. As we shall see in the examples below, the use of cross-references provides a more powerful mechanism of data retrieval than a simple search of individual database entries. However, cross-references must be constructed and used with care in order to avoid spurious search results. One danger of misusing cross-references lies in the fact that different database entries of the same protein chain may in fact be different in ways that are important to the user. For example, if a user is looking for structures of a protein in a complex with a certain ligand, he/she may not want to retrieve all other structures of the same protein in which the ligand of interest is absent. For that reason, cross-referencing is

automatically turned off if the user searches for properties that may be unique to each PDB structure. The current set of search field choices for which cross-referencing is off includes heterogen, resolution, and chain length. For other types of queries, *the user always has an option to turn cross-referencing off* by using the advanced query form (not shown) and setting 'Retrieve molecules related to database hits' to 'none'.

Another danger arises when cross-references are used to go from the PDB to SWISS-PROT and, further, to other PDB entries to retrieve all instances of a protein found by PDB annotation. An increasing number of PDB entries contain structures of macromolecular complexes that contain more than one protein. However, the user is not necessarily interested in all of the structures of all the components. For example, many proteins are co-crystallized with antibodies. The user may want to retrieve all structures of the protein of interest, but not those of the immunoglobulin. For this reason, we keep cross-references between the PDB and other databases at the chain level, rather than at the PDB entry level, whenever possible. If a hit of a PDB entry has been narrowed down to a set of chains, only cross-references involving those chains are used for further information retrieval.

Complex queries by means of repeat searches. Our searches are carried out on the per database field basis. This often allows the search engine to avoid hitting false positives; if all of the search terms are in the same database field, they are likely to relate to the same concept. However, sometimes a user may be interested in running complex queries incorporating more than one database field. For example, if one wishes to find all ribonucleases in *E.coli*, one must generally look for 'ribonuclease' in the molecule definition fields and for 'Escherichia' and 'coli' in source organism fields. In the current implementation of our user interface, this kind of inquiry can be run in 2 steps: one can find all ribonucleases and then limit the result set to entries whose organism is *E.coli*. This implementation is the price for having a simple, one-entry query form. Of course, it would be possible to create a form with 2 or 3 entries combined by Boolean operators if there is a subsequent popular demand for such a utility.

In a repeat search, as described above, we also make use of cross-references to produce a meaningful result. The results of the first search can be represented as an array of trees, with each tree having three levels (see Table 1). The database entry initially matched by the search is the root of the tree, the second level is populated by related SWISS-PROT entries, and the third level is populated by CATH and PDB entries related to each SWISS-PROT entry. A few more PDB entries that are found to be related to the root, but not to any of the SWISS-PROT entries, may also reside at the second level. When a second search is run over the result set, each database entry found by it is checked against each tree in the result set. A node of a tree is marked as validated if the new hit is to the same database entry. The validation is transferred down the tree to database entries that may be considered instances of the validated entity, i.e. from the ENZYME entry to related SWISS-PROT entries representing proteins that are instances of the enzyme class, and from a SWISS-PROT entry to related PDB entries containing structures of the SWISS-PROT protein. The validation is also transferred up the tree from child to parent, thereby preserving the branch on which the validated node is located. However, validation is not transferred to siblings of a validated node. For example, if one seeks to limit one's result set by the resolution of the PDB structures, then the validation of a PDB structure that satisfies new query conditions should not be transferred to other structures of the same protein that do not. The nodes that are left unvalidated at the end of the search are deleted from the result set.

Example search

Suppose we have searched for 'ribonuclease T1' with the default settings, i.e. we are searching entire annotation in all databases, using cross-references, and matching both query words. Ribonuclease T1 is the name of the enzyme EC 3.1.27.3. This enzyme is a ribonuclease that performs 'two-stage endonucleolytic cleavage to 3′-phosphomononucleotides and 3′-phosphooligonucleotides ending in G-P with 2′, 3′-cyclic phosphate intermediates' (see http://www.expasy.cbr.nrc.ca/cgi-bin/enzyme-search-ec). Ribonuclease T1 is found in *Aspergillus oryzae*. Similar enzymes are found in several other fungi species as well as in some bacteria. Bacterial enzymes are apparently unrelated to eukaryotic enzymes by sequence, but are similar in structure, specificity, and mechanism of action (Sevcik *et al.*, 1990). Importantly, the ribonuclease T1 enzymes found in different species, even the ones closely related to the *A.oryzae*, are often called by different names: i.e. ribonuclease MS, ribonuclease F1, etc.

The results of this search are summarized in Table 1 (CATH domains are not shown). In addition to RNT1_ASPOR, the SWISS-PROT entry for the *A.oryzae* enzyme, the search engine also retrieves RNMS_ASPSA, RNSA_STRAU, and RNF1_GIBFU, each with a corresponding set of PDB entries. These are all enzymes of the class 3.1.27.3. They are found in spite of the name differences mentioned above. The search engine also retrieves nine additional PDB entries. These are structures of barnase, a bacterial T1-like ribonuclease, and an additional structure RNSA_STRAU, which has not been picked up by cross-references to SWISS-PROT. Even though RNBR_BACAM, the SWISS-PROT entry

for barnase, and the 3.1.27.3 entry in ENZYME are not cross-referenced in either database, these PDB entries have still been found. The reason for that is that EC 3.1.27.3 is mentioned in their PDB molecule definitions. An additional PDB structure, found through its PDB annotation, is 1aqz, a ribotoxin related to ribonuclease T1 (Kao and Davies, 1995).

Altogether, the search engine finds 81 ribonuclease T1 entries in PDB. Four of these can be considered false positives. There are three PDB entries that are actually structures of a different ribonuclease (ribonuclease A), which were mistakenly cross-referenced to RNT1_ASPOR. 1aqz is not exactly a ribonuclease T1 enzyme, either, although it is related; whether this should be considered a true or false positive may depend on the purposes of the user.

Most of the PDB entries are shown to be the structures of four SWISS-PROT proteins. A user who is interested in ribonuclease T1 from *A.oryzae* only will have no difficulty separating structures of interest from all the others because it is clearly shown in the search result pages with which SWISS-PROT entry each PDB structure is associated.

Query performance

The search engines and search mechanisms. We have compared the performance of some simple keyword queries on our search engine to some other non-proprietary search engines available. The performance of the queries have been evaluated in terms of precision and recall of the retrieval of PDB entries. The enhancement of the retrieval of PDB entries and related information by keywords is the main goal of BioMolQuest in its present form. Other systems, such as SRS servers and DBGet, provide more broad functionality.

The search engines to which we have compared BioMolQuest to are RCSBs SearchLite at http://www.rcsb.org/pdb/searchlite.html, SRS server at EBI at http://www.srs.ebi.ac.uk/, and the SRS server at the Indiana University at http://www.iubio.bio.indiana.edu/srs6bin/cgi-bin/wgetz?-page+top. We also included in the comparison searches through InterPro at the EBI SRS server. The search procedures were as follows:

- At BioMolQuest and RCSB, simply enter the query and hit the Search button.
- At EBI SRS, do a two-step search to utilize cross-references:
  - search SWALL (a more complete SWISS-PROT) + enzyme databases (LENZYME, BRENDA, ENZYME, EMP, UENZYME) + PDB + PDBFinder, or search InterPro
  - cross-link all the results to PDB and PDBFinder
- At Indiana SRS, do a two-step search:
  - search SWISS-PROT + SWISSNEW + ENZYME + LENZYME + PDB
  - cross-link all the results to PDB

Query performance measurement and ideal sets. The performance of the queries was measured in terms of precision, defined as the ratio of the number of true positives to the total number of hits, recall, defined as the ratio of the number of true positives to the number of entries in an ideal set, and response time.

The ideal set is the set of all PDB entries known to be relevant to a query. In order to be able to construct an ideal set, one has to clarify the exact meaning of the query. For example, when searching for 'ribonuclease *and* T1', does one mean 'ribonuclease T1' the enzyme class 3.1.27.3, or the particular 'ribonuclease T1' of *A.oryzae*? We have generally taken the position of preferring broader definitions of true positives to more narrow ones, since having to weed out a few less interesting hits is usually less damaging to the users than not finding what they were looking for. Our queries and the corresponding definitions of true positives are listed here:

- ribonuclease *and* T1: an enzyme catalyzing two-stage endonucleolytic cleavage to 3′-phosphomononucleotides and 3′-phosphooligonucleotides ending in G-P with 2′, 3′-cyclic phosphate intermediates (some enzymes commonly included in this class, such as ribonuclease Ms of *Aspergillus saitoi,* may catalyze cleavage at nucleotides other than G, but they still have a preference for the G's; these enzymes are considered true positives);
- glutamine *and* amidotransferase: an enzyme catalyzing the transfer of NH_2 from Gln to some other molecule, thereby producing Glu, or the reverse reaction; includes components of multi-chain complexes where one chain may hydrolyze Gln and another chain may attach NH_2 to a different molecule;
- sodium *and* channel *and* inhibitor: a peptide inhibiting the normal function of a sodium channel.

The ideal sets were constructed using the following procedure:

- Run query on each of the search engines,
  - assign true and false positives based on the annotation of the hits;
  - combine all true positives into a single list.
- Run BLAST on a true positive,
  - examine the annotation of the BLAST hits which are not already on the list of true positives to see if any new true positives have been found;

Table 2. Comparison of search engine performance

Search engine	True positives	False positives	False negatives	Precision	Recall	Average time[a] (s)	Minimal time[a] (s)
			Ribonuclease and *T1*				
BioMolQuest	77	4	27	0.95	0.74	10	10
SRS EBI[b]	70	6	34	0.92	0.67	64	55
SRS Indiana[b]	64	3	40	0.96	0.62	33	32
RCSB	59	10	45	0.86	0.57	30	18
			Glutamine and *amidotransferase*				
BioMolQuest	22	0	0	1.00	1.00	13	13
SRS EBI[b]	22	0	0	1.00	1.00	56	50
SRS Indiana[b]	21	0	1	0.95	0.95	30	29
InterPro/SRS[b,c]	14	6	8	0.70	0.64	21	9
RCSB	18	0	4	1.00	0.82	8	5
			Sodium and *channel* and *inhibitor*				
BioMolQuest	31	0	4	1.00	0.89	10	9
SRS EBI[b]	29	3	6	0.91	0.83	76	64
SRS Indiana[b]	29	0	6	1.00	0.83	37	36
RCSB	11	0	24	1.00	0.31	6	4

The queries were used to retrieve PDB entries as described in Section Query performance.
The queries have been run on December 14, 2000. The exact results will obviously change with time as the databases constantly grow and change.
[a] Six measurements done on December 19–20, 2000 between 9 a.m. and 10 p.m. US Central time. The clock was stopped when a result page began to load.
[b] Sum of three steps, as discussed in Section Query performance measurement and ideal sets.
[c]EBI SRS server; no results obtained for the other two queries; InterPro does not cross-link to PDB on the Indiana SRS server.

- if any of the known true positives have not been found by the BLAST search, use one of them as a query for another BLAST;
- repeat until all of the known true positives have been found by at least one BLAST search.

- Use literature references to confirm each true positive,
  - use references from SWISS-PROT entries to confirm them, then run BLAST of the SWISS-PROT sequence against PDB to confirm corresponding PDB entries; use e-value of 10^{-10} *and* sequence identity $\geqslant 90\%$ as the cut-off (somewhat less stringent cut-offs had to be used for some short-chain sodium channel inhibitor structures which were missing a few residues)
  - if there are no relevant references in SWISS-PROT, search PubMed for protein names
  - examine review papers (Sevcik *et al.*, 1990; Possani *et al.*, 1999; Norton, 1991).

Response times were measured with a timer. The timer was started when a query was submitted and stopped when a search result page began to load. For SRS servers, we added up the times of the three operations involved in a search using cross-references. The first operation is to run the initial query; the second operation is to tell the SRS server that we wish to cross-link the entire result set; the third operation is to specify that the cross-link should be to the PDB, upon which the server finds the target PDB entries. Each of these steps involves submitting a form to the server and waiting for it to respond. Response times reported in Table 2 are the averages of six measurements made at various times of the day between 9 a.m. and 10 p.m. US Central time over 2 days. The measurements have been carried out at the Becker Medical Library on the Washington University campus to make sure that the BioMolQuest data have to travel through a network like the data from the other search engines, as network delays are likely to affect the response times.

The results. Table 2 shows the comparison of the search results by the BioMolQuest and other search engines. BioMolQuest and SRS tend to have better recall than RCSB while maintaining comparable precision. InterPro does not have an entry for ribonuclease T1, and the entries for sodium channel inhibitor fail to cross-reference to PDB. InterPro also seems to fail to cross-reference to PDB at the Indiana SRS server. The search of InterPro for glutamine amidotransferase with subsequent cross-reference to PDB at the EBI server performs relatively poorly.

The main reason for the better recalls of BioMolQuest and the SRS servers compared to the RCSB is the use of

database cross-references to retrieve all instances of the same protein or a class of proteins in all of the legacy databases. The use of cross-references creates the effect of amplifying the annotation content: it is sufficient to find the search terms in one of the database instances of a protein or a protein class to retrieve all or most of the other instances. This allows the search engine to find database entries where annotation is incomplete or alternative terms are used. While performance of SRS is close to that of BioMolQuest, the user must possess a certain degree of sophistication to use it the way we did. Casual users would probably just use an SRS server to search the PDB directly, which would not give them any advantage over using the RCSB web site. Simplicity of the user interface is an important factor in web design. While scientists are generally more sophisticated than average web users, most of them would rather spend their energies on their research topics than on the subtleties of database querying techniques.

Table 2 also shows the time it takes BioMolQuest and other servers to deliver results of a query. The times reported for the SRS servers are the sums of three operations, as discussed above. One should not overinterpret the exact times we report, as they are almost certainly affected by factors that have nothing to do with the search engine design, such as network delays and differences in the server hardware and loads. We report the minimal observed times along with the averages, as they reflect the server performance under light load and/or minimal delays in the network. It is obvious that the BioMolQuest response times are comparable to those of the other search engines for queries of moderate size. BioMolQuest does slow down considerably for larger queries. For example, if one searches for 'transcription *and* factor', the search engine has to go through about 5000 SWISS-PROT entries, finding 356 PDB entries in the end. The average BioMolQuest response time to this query is 59 s. RCSB seems to slow down for large queries as well (42 s average for 'transcription *and* factor'), while SRS servers are not affected.

DISCUSSION

We have implemented a search engine based on a relational database of annotations imported from PDB, SWISS-PROT, ENZYME, and CATH (Berman *et al.*, 2000; Bairoch and Apweiler, 2000; Bairoch, 2000; Orengo *et al.*, 1997). This search engine allows for more powerful annotation searches than is possible with any of the individual legacy databases. Information from the legacy databases is integrated using inter-database cross-references provided in the legacy database entries or inferred when necessary and/or possible. Automatic use of these cross-references by the search engine significantly improves recall of keyword queries without any additional effort on the part of the user. At the same time, application of the queries to individual database fields, rather than entire entries, ensures good precision. The output of the queries is presented in a logical order that clarifies the relationships among the retrieved database entries. A complex query is possible through repeat searches that limit the results of the previous searches. This search engine can be used as a convenient gateway to the legacy databases, particularly the PDB.

BioMolQuest should be considered a work in progress. Its current weaknesses include sluggish response times for large queries and the lack of quality control of cross-references. The first problem will be addressed by better software design, possibly including parallelization of the queries. The cross-references shall be verified by sequence identity. Other directions of BioMolQuest development include integrating additional resources into the service, such as metabolic pathway information and protein structures predicted by our group.

ACKNOWLEDGEMENTS

This research was supported in part by NIH grant No. GM-48835 of the Division of General Medical Sciences.

REFERENCES

An,J., Nakama,T., Kubota,Y. and Sarai,A. (1998) 3DinSight: an integrated relational database and search tool for structure, function and property of biomolecules. *Bioinformatics*, **14**, 188–195.

Anahory,S. and Murray,D. (1997) *Data Warehousing in the Real World*. Addison-Wesley, Longman, UK.

Apweiler,R., Attwood,T.K., Bairoch,A., Bateman,A., Birney,E., Bucher,P., Codani,J.-J., Corpet,F., Croning,M.D.R., Durbin,R., Etzold,T., Fleischmann,W., Gouzy,J., Hermjakob,H., Jonassen,I., Kahn,D., Kanapin,A., Schneider,R., Servant,F. and Zdobnov,E. (2000) InterPro—an integrated documentation resource for protein families, domains and functional sites. *CCP11 Newsletter,* **10**.

Bairoch,A. (2000) The ENZYME database in 2000. *Nucleic Acids Res.*, **28**, 304–305.

Bairoch,A. and Apweiler,R. (2000) The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.*, **28**, 45–48.

Baker,W., van den Broek,A., Camon,E., Hingamp,P., Sterk,P., Stoesser,G. and Tuli,M.A. (2000) The EMBL nucleotide sequence database. *Nucleic Acids Res.*, **28**, 19–23.

Barker,W.C., Garavelli,J.S., Huang,H., McGarvey,P.B., Orcutt,B., Srinivasarao,G.Y., Xiao,C., Yeh,L.S., Ledley,R.S., Janda,J.F., Pfeiffer,F., Mewes,H.W., Tsugita,A. and Wu,C. (2000) The Protein Information Resource (PIR). *Nucleic Acids Res.*, **28**, 41–44.

Benson,D.A., Karsch-Mizrachi,I., Lipman,D.J., Ostell,J., Rapp,B.A. and Wheeler,D.L. (2000) GenBank. *Nucleic Acids Res.*, **28**, 15–18.

Berman,H.M., Westbrook,J., Feng,Z., Gilliland,G., Bhat,T.N., Weissig,H., Shindyalov,I.N. and Bourne,P.E. (2000) The Protein Data Bank. *Nucleic Acids Res.*, **28**, 235–242.

Branden,C. and Tooze,J. (1991) *Introduction to Protein Structure*. Garland, New York.

Creighton,T.E. (1993) *Proteins. Structures and Molecular Properties*. Freeman, New York.

DuBois,P. (2000) *MySQL*. New Riders, Indianapolis.

Etzold,T., Ulyanov,A. and Argos,P. (1996) SRS: information retrieval system for molecular biology data banks. *Meth. Enzymol.*, **266**, 114–128.

Fetrow,J.S. and Skolnick,J. (1998) Method for prediction of protein function from sequence using the sequence-to-structure-to-function paradigm with application to glutaredoxins/thioredoxins and T1 ribonucleases. *J. Mol. Biol.*, **281**, 949–968.

Fujibuchi,W., Goto,S., Migimatsu,H., Uchiyama,I., Ogiwara,A., Akiyama,Y. and Kanehisa,M. (1998) DBGET/LinkDB: an integrated database retrieval system. *Pac. Symp. Biocomput.*, **3**, 681–692.

Goto,S., Nishioka,T. and Kanehisa,M. (2000) LIGAND: chemical database of enzyme reactions. *Nucleic Acids Res.*, **28**, 380–382.

Hofmann,K., Bucher,P., Falquet,L. and Bairoch,A. (1999) The PROSITE database, its status in 1999. *Nucleic Acids Res.*, **27**, 215–219.

Holm,L. and Sander,C. (1996) Mapping the protein universe. *Science*, **273**, 595–602.

Kanehisa,M. (2000) *Post-Genome Informatics*. Oxford University Press, Oxford.

Kao,R. and Davies,J. (1995) Fungal ribotoxins: a family of naturally engineered targeted toxins? *Biochem. Cell Biol.*, **73**, 1151–1159.

Kawabata,T., Ota,M. and Nishikawa,K. (1999) The Protein Mutant Database. *Nucleic Acids Res.*, **27**, 355–357.

Murzin,A.G., Brenner,S.E., Hubbard,T. and Chothia,C. (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, **247**, 536–540.

Norton,R.S. (1991) Structure and structure-function relationships of sea anemone proteins that interact with the sodium channel. *Toxicon*, **29**, 1051–1084.

Orengo,C.A., Michie,A.D., Jones,S., Jones,D.T., Swindells,M.B. and Thornton,J.M. (1997) CATH—a hierarchic classification of protein domain structures. *Structure*, **5**, 1093–1108.

Possani,L.D., Becerril,B., Delepierre,M. and Tytgat,J. (1999) Scorpion toxins specific for Na^+-channels. *Eur. J. Biochem.*, **264**, 287–300.

Sevcik,J., Sanishvili,R.G., Pavlovsky,A.G. and Polyakov,K.M. (1990) Comparison of active sites of some microbial ribonucleases: structural basis for guanylic specificity. *Trends Biochem. Sci.*, **15**, 158–162.

Thornton,J.M., Orengo,C.A., Todd,A.E. and Pearl,F.M. (1999) Protein folds, functions and evolution. *J. Mol. Biol.*, **293**, 333–342.

Wall,L., Christiansen,T. and Schwartz,R.L. (1996) *Programming Perl*. O'Reilly, Sebastopol, CA.