BIOINFORMATICS

Vol. 18 no. 1 2002
Pages 130–139

The Binding Database: data management and interface design

Xi Chen [1], Yuhmei Lin [1], Ming Liu [1,2] and Michael K. Gilson [1,*]

[1]Center for Advanced Research in Biotechnology, University of Maryland Biotechnology Institute, 9600 Gudelsky Drive, Rockville, MD 20850, USA

Received on May 23, 2001; revised on August 10, 2001; accepted on August 20, 2001

ABSTRACT

Motivation: The large and growing body of experimental data on biomolecular binding is of enormous value in developing a deeper understanding of molecular biology, in developing new therapeutics, and in various molecular design applications. However, most of these data are found only in the published literature and are therefore difficult to access and use. No existing public database has focused on measured binding affinities and has provided query capabilities that include chemical structure and sequence homology searches.

Methods & results: We have created Binding DataBase (BindingDB), a public, web-accessible database of measured binding affinities. BindingDB is based upon a relational data specification for describing binding measurements via Isothermal Titration Calorimetry (ITC) and enzyme inhibition. A corresponding XML Document Type Definition (DTD) is used to create and parse intermediate files during the on-line deposition process and will also be used for data interchange, including collection of data from other sources. The on-line query interface, which is constructed with Java Servlet technology, supports standard SQL queries as well as searches for molecules by chemical structure and sequence homology. The on-line deposition interface uses Java Server Pages and JavaBean objects to generate dynamic HTML and to store intermediate results. The resulting data resource provides a range of functionality with brisk response-times, and lends itself well to continued development and enhancement.

Availability: The BindingDB is publicly available at http://www.bindingdb.org. The database schema and DTD file are available at http://www.bindingdb.org/bind/entity_report.html, and http://www.bindingdb.org/bind/deposition/xml/BindingDB.xml, respectively.

Contact: gilson@umbi.umd.edu

*To whom correspondence should be addressed.

[2]Present address: NovaScreen, 7170 Standard Drive, Hanover, MD 21076, USA.

INTRODUCTION

The noncovalent association of molecules is of fundamental importance in biology. Indeed, almost every life process at the molecular level involves noncovalent association at some level of specificity. Examples include the action of hormones at their receptors, the distinctions between self and nonself molecules made by the immune systems, and the specificity of catalysis by enzymes. Similarly, many medications act by binding and modulating the activity of targeted biomolecules. Indeed, the design of such targeted molecules is of substantial commercial importance in the pharmaceutical industry, and molecular recognition is of increasing interest in chemistry because of its relation to catalysis, host–guest chemistry, and self-assembling systems. As a consequence, the scientific literature contains a large and growing body of experimental data on what molecules bind each other and how strongly they bind. These data are of great value in molecular design applications, and also as a basis for developing a deeper understanding of the physical chemistry of noncovalent binding. However, binding measurements are currently difficult to find and use because of the limitations of paper publication. Even on-line literature databases offer no easy way to discover, say, all binding measurements involving some molecule of interest.

For these reasons, a workshop in 1997 considered the possibility of developing a scientific database focused on binding thermodynamics. Participants included representatives from academic, industrial, and government laboratories. The resulting report (http://www.bindingdb.org/update/workshop_rep1b.html) strongly endorsed the development of a binding database and proposed the following guidelines:

- The database should include binding affinities for a variety of molecules, including biopolymers such as proteins, nucleic acids and carbohydrates, as well as small organic molecules such as drug candidates and synthetic host and guest molecules.
- To the extent possible, experimental details should be

included to support sophisticated interpretation and evaluation of the data.

- A wide range of queries should be supported, including SQL queries based on textual and numerical criteria, and nonSQL queries such as chemical substructure and BLAST sequence searches.
- The database should be a public resource accessible via the WWW.
- Every effort should be made to promote direct deposition of new binding data by experimentalists.

Meeting these goals posed a number of technical challenges, many of which also are faced by developers of other scientific databases. The Binding DataBase (BindingDB) project (Chen *et al.*, 2001) has addressed these challenges and is now in operation at www.bindingdb.org. The present paper describes the database and explains the technical approaches used to implement it. In particular, the following issues are addressed.

Data management

Today's biomolecular databases use various data management approaches, including structured flat files (STAR/mmCIF; Westbrook and Bourne (2000); ASN.1; Steedman (1993); Benson *et al.* (1999), and XML-based methods), relational DataBase Management Systems (DBMS'; Hutsman *et al.*, 1991; Kumar *et al.*, 2000) object-relational DBMSs (Zhu and Zhang, 1999; Carazo and Stelzer, 1999) and object-oriented DBMSs (Aberer, 1994; Nakata *et al.*, 1999). These approaches are defined and discussed elsewhere (Achard *et al.*, 2001). The present paper explains the rationale for and implementation of BindingDB's data management system, which combines a relational DBMS with flat files for sequence data and XML files for data interchange.

Data organization and content

Once a DBMS has been chosen, the next step is to determine exactly what data will be stored and how they will be organized. The content is determined primarily by the scientific goals of the database and, in the case of BindingDB, was defined in large part by the scientists who attended the 1997 workshop. The organization of the data is important because, among other things, it determines how easily the database can be expanded and amended in the future. The content of BindingDB is organized into a relational data dictionary and an XML Document Type Definition (DTD).

Query and data retrieval

Perhaps the chief reason for establishing a database is to facilitate the retrieval of the specific data that each user needs. Ideally, a variety of query methods must be supported. For example, users may wish to search for binding reactions by molecule name, protein sequence, chemical substructure, affinity, or combinations of the above. The present paper describes the techniques used to support these and other types of query in the BindingDB.

Data deposition

The collection of data plays a vital role in a database but is rarely an easy task. A basic goal is to collect as much data as possible, and accordingly to make the deposition methods easy to access and use. In addition, the requirement that the data be of high quality leads to requirements for validation software and human curation. On the other hand, budget constraints motivate maximal automation of the deposition procedures, and any specific realization must to some extent be a compromise among these competing goals. For example, a client–server application can be made easy to use and can provide relatively strong assurances of data integrity, but it can be difficult to distribute the software and keep it up to date. On the other hand, online deposition via the WWW maximizes accessibility, but web technology lacks the flexibility that a stand-alone client can deliver. Which method is preferred will depend upon the demands of the individual project. The BindingDB has tried both approaches and has settled for now on web deposition as the preferred method because gathering data is critical at this stage of the project and web-accessibility permits the broadest possible dissemination of the deposition tools.

Usability

A user-friendly interface is essential if a database is to be of value to the scientific community. The user-interface of the BindingDB was designed according to the principles of human–computer interactions, following a staged process. The initial stage was establishment of guidelines and prototyping. Feedback was then gathered from a small group of potential users and revisions made accordingly. A broader user survey was then done and more detailed revisions were made. Details of this process can be found in our previous paper and are not repeated here (Chen *et al.*, 2001).

DATA MANAGEMENT

BindingDB currently runs on a 600 MHz dual-processor Windows NT server with 256 MB RAM and uses a commercial relational DBMS, Oracle Enterprise Edition Release 8.1.6, as its central data management system. The relational model avoids data redundancy, which is relevant here because data attributes are often repeated for multiple binding measurements. For example, binding affinities might be measured for two molecules over a range of temperatures but with no other parameters varying. In

addition, a commercial DBMS immediately provides a number of valuable features, such as efficient query procedures, scalability, and useful database administration tools. There are also an array of training materials and a pool of expert users. These attributes make the database easier to develop and maintain. However, we have tried to minimize our use of features unique to Oracle in order to minimize the difficulty of porting to a different DBMS should the need arise.

BindingDB's relational DBMS is supplemented by two sets of flat files. First, protein and nucleic acid sequences are stored not only in the DBMS but also in external files that can be searched with BLAST (Altschul *et al.*, 1997). Each sequence in an external file contains a relational ID that is used to link back to the appropriate data in the relational DBMS, allowing integration of BLAST with SQL queries (see Integration of NonSQL Query Functions). Second, BindingDB's WWW deposition interface generates an intermediate XML file that is later parsed into the relational DBMS; the XML files are also stored on disk to provide a parallel view and an archive of the data. Each XML file contains a coherent data 'entry' and provides an unfragmented view of the data. In contrast, the data in the relational database are scattered across multiple tables and linked only by keys. The XML files are thus more appropriate for data exchange and also represent a human-readable backup for the relational database. Although XML files can be searched, the BindingDB does not use them as the basis for data queries because the performance would be much slower than in the relational DBMS.

DATA ORGANIZATION AND CONTENT

The fundamental view of the data is that of an experiment in which two reactants are dissolved in a reaction solution and studied by an experimental technique with a specified instrument at some pH, temperature and pressure. The resulting raw data are interpreted in terms of a reaction model—e.g. $A + B \leftrightarrow AB$—and processed quantitatively with a data fitting method to yield a binding affinity and other results characterizing the binding reaction. Related experiments that are deposited together are grouped into a single data entry, which is associated with one or more citations. The BindingDB describes these data in a relational data dictionary and in an XML DTD, as now described.

Relational data dictionary

The current data dictionary comprises 35 database tables, as illustrated in the Entity-Relation (ER) diagram in Figure 1 and defined in detail at http://www.bindingdb.org/bind/entity_report.html. Data in these tables are organized into Entries, each of which contains one or more sets of binding measurements, along with literature citations and database administrative information. An Entry might include studies of a single binding reaction as a function of temperature and pH, or of a given protein's interactions with a collection of small-molecule ligands. The ENTRY table contains an Entry ID that appears as a foreign key in every entry-specific table. It also includes a descriptive title of the entry, the deposition date, the measurement technique, and information on the entrant, the person entering the data.

Although the relational model dictates a normalized representation of the data and their relationships (Codd, 1972; Date, 1995), full normalization does not necessarily produce the most efficient database system. A relational database application should balance the goals that commonly invoked queries be executed rapidly and that the data remain consistent through minimization of data redundancy. Thus, although the BindingDB initially aimed for complete data normalization, some denormalization was later introduced for the sake of efficiency.

The tables other than the Entry table can be grouped into four categories: experimental results and details, reactants, reaction solution, and references. These categories are now discussed in more detail.

Experimental results and details. The results tables are at the core of the data dictionary because they contain the final results along with links to all the methods, conditions, etc. that pertain to these results. Because the BindingDB aims to include significant experimental detail, each measurement technique requires special attention and a unique results table. To date, two techniques have been included. The first, Isothermal Titration Calorimetry (ITC), was selected because it provides a relatively complete description of binding thermodynamics and therefore yields a good data foundation for other measurement techniques. The second technique, enzyme inhibition, was selected because it is used for a relatively large number of measurements, including many that are of pharmacologic interest. The results tables are also specific to the reaction model. Thus, separate tables are included for ITC data interpreted with the binding models $A + B \leftrightarrow AB$ and $2A + B \leftrightarrow A_2B$. We initially explored dictionary structures that could accommodate all possible reaction models, but the resulting dictionaries were complex, hard to query, and still incomplete, whereas the current approach is straightforward to implement, query, and extend. The dictionary currently contains three results tables: ITC_RESULT_A_B_AB, ITC_RESULT_2A_B_A2B, and KI_RESULT, for ITC results with the two reaction models listed above, and for enzyme inhibition with the $A + B \leftrightarrow AB$ reaction model, respectively.

The ITC results tables accommodate the range of commonly measured thermodynamic data generated by

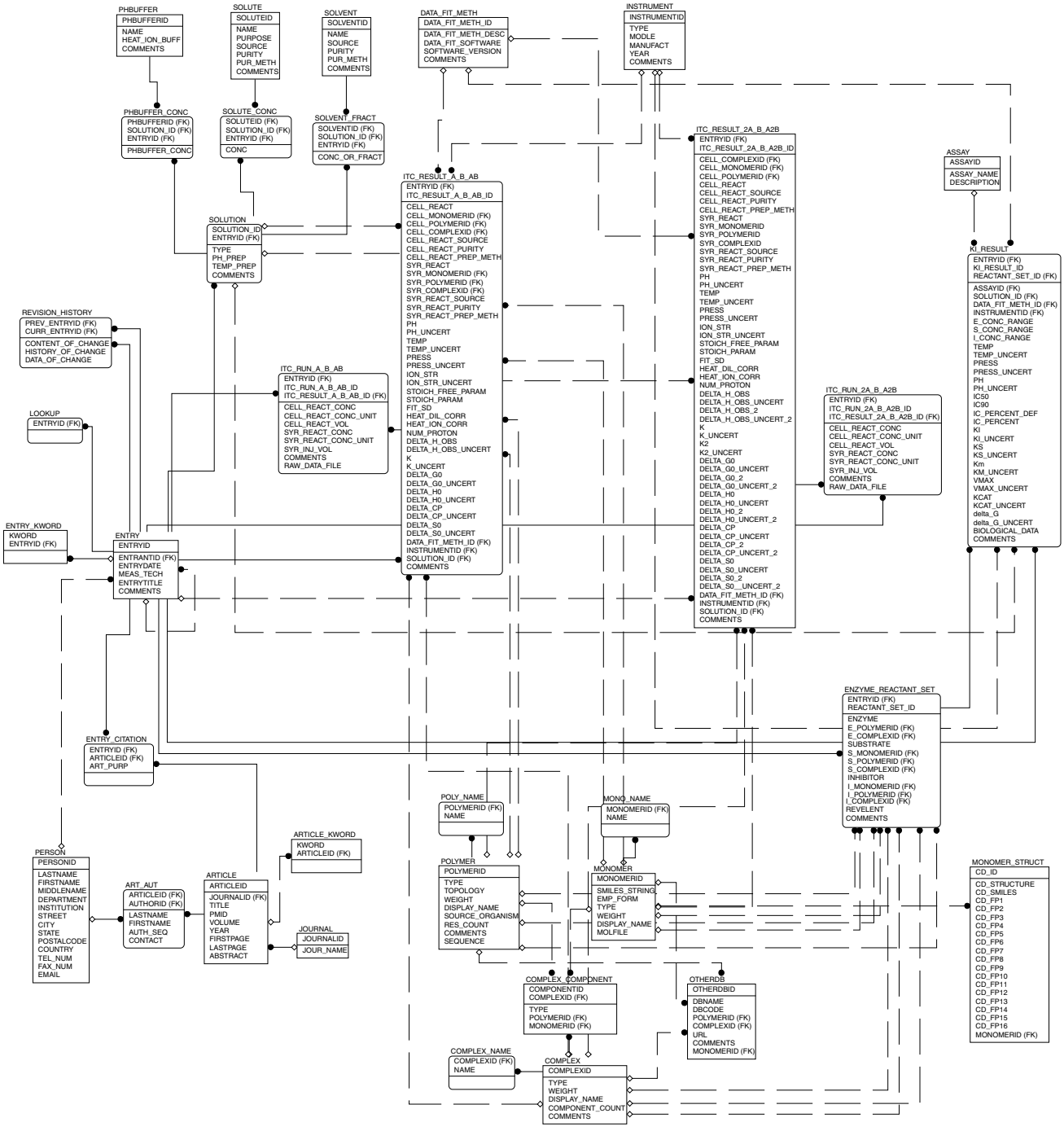

Fig. 1. Entity-relationship diagram of the relational BindingDB. See http://www.bindingdb.org/bind/entity_report.html for details.

ITC experiments; i.e. the standard changes in free energy, enthalpy, entropy and heat capacity, and the equilibrium constant. They also include attributes specific to the technique, such as flags indicating whether the data have been corrected for the heat of dilution and heat of buffer ionization, and whether the reaction stoichiometry is a free parameter and, if so, its value. Foreign keys from the INSTRUMENT and DATA_FIT_METH tables link to details regarding the instrument and data fitting method used to generate a set of results. One ITC thermodynamic

result is usually generated from a series of calorimetry runs, and a separate table, ITC_RUN_A_B_AB, is provided to store information about each separate run. This table includes the attribute RAW_DATA_FILE for experimentalists willing to upload the raw data for archival and possible reanalysis by others; however, the details of this attribute are not yet fully defined, since different instrument/software combinations yield different data files.

The enzyme inhibition results table KI_RESULT allows enzyme-inhibitor affinity to be specified in several different ways, including K_i, IC_{50} and IC_{90}. Other IC values are accommodated by allowing the user to specify a percent inhibition, such as 80%, in IC_PERCENT_DEF and then to list the IC_{80} in IC_PERCENT. The table also accommodates other enzymatic data that may have come out of the experiment, such as K_s, K_m, K_{cat}, and V_{max}. In keeping with practice in the literature, concentration of enzyme, substrate and inhibitor are described in ranges instead of single values. An additional ASSAY table allows for free-text description of the enzyme-inhibition assay method. The BIOLOGICAL_DATA attribute is set to 'yes' when the study that yielded the binding data also provides biological data from cell culture, for example. As for ITC results, instrument and data fitting method information is provided via foreign keys to the INSTRUMENT and DATA_FIT_METH tables.

Each reactant is identified in the results through one of three foreign keys from the MONOMER, POLYMER and COMPLEX tables; which one is used depends upon the type of the reactant. The reactant tables define the compounds studied, while the results tables add information, such as supplier and purity, regarding the actual materials used in an experiment. The reactant IDs are brought directly into the ITC result tables, but are brought into a separate table, ENZYME_REACTANT_SET for enzyme-inhibition measurements to avoid extensive repetition of experimental conditions for the large volume of data that may be generated by high-throughput studies of one receptor with multiple ligands. The following section describes the treatment of reactants in more detail.

Reactants. The chemical entities involved in binding reactions are defined in the reactant tables, while details of the materials actually used, such as supplier and purity, are given in the results tables. The BindingDB is designed to accept binding data for a variety of molecules, and different data structures are appropriate to different types of molecule. Thus, as shown in Figure 1, each reactant is classified and stored as a 'monomer', a 'polymer', or a 'complex'. A monomer is a small organic molecule, such as tyrosine or penicillin. Monomers are identified not only by name but also by chemical structure. Thus, the MONOMER table includes a SMILES string (Weininger, 1988) and an MDL Molfile (http://www.mdli.com/downloads/literature/ctfile.pdf) as attributes to enable chemical structure queries (see section Integration of nonSQL Query Functions). Proteins and polynucleotides are stored as 'polymers' in BindingDB, and are identified by name and sequence. The polymer sequences in the relational database are exported into a FASTA file daily in order to enable BLAST searching, as described in Integration of NonSQL Query Functions. An assembly of monomers and/or polymers is stored in BindingDB as a 'complex'. The COMPLEX table contains no monomer or polymer information, but only general information for each complex, such as type and molecular weight. The monomer and polymer constituents of a complex are registered in the COMPLEX_COMPONENT table, leaving the detailed description of each monomer and polymer to be stored in the MONOMER and POLYMER tables. This treatment of molecular complexes imposes the computational cost of an additional level of query, but it avoids data redundancy between monomers and polymers and the components of complexes. It is not too expensive computationally because the DBMS is efficient and the number of table joins is still low.

Multiple names, or synonyms, for molecules are accommodated with the MONO_NAME, POLY_NAME and COMPLEX_NAME tables, which allow any number of names to be stored for each reactant. For convenience, however, a DISPLAY_NAME attribute is included in the MONOMER, POLYMER and COMPLEX tables; this is the name that will be displayed in query results. The DISPLAY_NAME attribute creates a slight de-normalization to the molecule tables since it repeats information in the synonym tables. However, this denormalization is convenient because it allows identification of a frequently used attribute without an extra join operation.

When additional data for a molecule exist in one or more other databases, references to these data can be stored in the OTHERDB table. This includes the other database's name, the identifier of this molecule in that database, the name of the molecule, and a URL link to that molecule. We plan to establish dynamic links to other databases in the future.

Reaction solution. The reaction solution is the medium in which the binding reaction occurs. A solution is viewed as consisting of a base solvent that may be a mixture (e.g. ethanol/water), and that may contain multiple solutes (e.g. NaCl) and a pH buffer. The reactants are not considered to be part of the reaction solution. Solutions are described with seven tables; the main SOLVENT, SOLUTE and PHBUFFER are linked to the SOLUTION table via associative (many-to-many) tables that specify the concentration of each solution component in each solution. This simple, generic data representation of

solution information of a reaction works well for the current ITC and KI techniques and can be readily extended if necessary.

References. References appropriate to each data entry are stored in the references group of tables. The ARTICLE and ENTRY tables are associated via the ENTRY_CITATION table. The PERSON table, which stores information both about article authors and about data entrants, is associated with the ARTICLE table via the ART_AUTH table. To facilitate access to publications, the ARTICLE table includes an attribute for the PubMed reference ID. The current policy of the BindingDB is that data may be deposited if and only if the detailed *method* used to generate them is published in a refereed journal. Therefore, it is possible to include unpublished data based upon a published method. In such cases, 'BindingDB' is used as the reference's journal name and a BindingDB article ID is assigned. Currently the experiment methods are limited to the well established ITC and enzyme inhibition methods.

XML data description

The BindingDB uses XML files for data transfer and validation during the deposition process and as a human-readable backup for the relational DBMS, where each XML file contains one complete data entry. XML is widely accepted in industry and in the scientific community, and its use is supported by a number of readily available tools. The use of XML by the BindingDB is thus expected to facilitate future exchanges of data with other biomolecular and chemical databases. This section briefly describes the BindingDB's current DTD and its relationship with the relational data model. The Data Deposition Methods section describes how XML files are used in the deposition process.

The literature provides few guidelines for how to define an XML DTD that is compatible with a complex relational database. The DTD must include the same data attributes as the relational data model, but the organization of the attributes must be different because XML files are organized hierarchically, so table relations cannot be directly transferred into the DTD. Here, the conversion from relational tables to XML is based upon a description of the hierarchical DTD as a tree of nodes, where the terminal (leaf) nodes are nonkey attributes of the data tables, and foreign key relations correspond to branches in the tree. Note that this approach causes the XML file to be completely denormalized in relational terms, because it requires reused data to be repeated with each use. In contrast, the relational data model reuses data by forming multiple links from a single occurrence of the reused item. Figure 2 illustrates this approach with the top levels of the DTD's data hierarchy for the BindingDB. The root element of an ITC data entry includes data elements ('leaves') such as the entry date and entry keywords, along with nodes leading to lower levels of the hierarchy with information on the entrant, literature citations and results. The results node in turn has a number of data elements as well as nodes with links to yet lower levels with further details about the reactants, solution, instrument, etc. The full XML definition can be accessed at http://www.bindingdb.org/bind/deposition/xml/BindingDB.xml.

It is not a coincidence that the hierarchy of this XML file is similar to that of the data deposition interface (see section User interface for data deposition). Use of a similar hierarchy facilitates the transfer of data collected via the deposition interface into XML files, while the connections between the DTD and the relational dictionary allow transfer from XML files to the relational DBMS.

QUERY AND DATA RETRIEVAL

Integration of nonSQL query functions

The BindingDB provides two query techniques that are outside the capabilities of pure SQL: chemical substructure/similarity search and sequence homology search. It is possible to provide such capabilities within an object-relational database management systems by creation of new data types and methods; e.g. Relational Extenders (IBM DB2), Cartridges (Oracle) and DataBlades (Informix). However, this approach is still not straightforward (http://www.daylight.com/products/daycart.html) and the methods are specific to each commercial database. In order to minimize the dependence of BindingDB upon the features of any specific commercial DBMS, we have implemented these special queries with external middleware.

Sequence homology. BindingDB uses BLAST to allow users to find reactions involving proteins and nucleic acids homologous to a user-specified protein or nucleic acid. The search returns a modified BLAST report that contains links to binding data associated with each homologous sequence, allowing binding reactions to be accessed and reviewed. This functionality is provided as follows.

Protein and nucleic acid sequences are archived in the POLYMER table of the main DBMS. These sequences are exported to external flat files containing protein and nucleic acid sequences in FASTA format. The description line of each sequence flat files includes the DISPLAY_NAME attribute of the molecule, and a general identifier (gnl) ID containing the table ID for the matching entry in the POLYMER table of BindingDB. The FASTA files are regenerated daily from data in BindingDB and preprocessed with the program Formatdb from the BLAST2.0 executables available at NCBI (ftp://ncbi.nlm.nih.gov/blast/executables/README.bls). Java Servlets (http://www.javasoft.com/docs/books/tutorial/servlets/) in

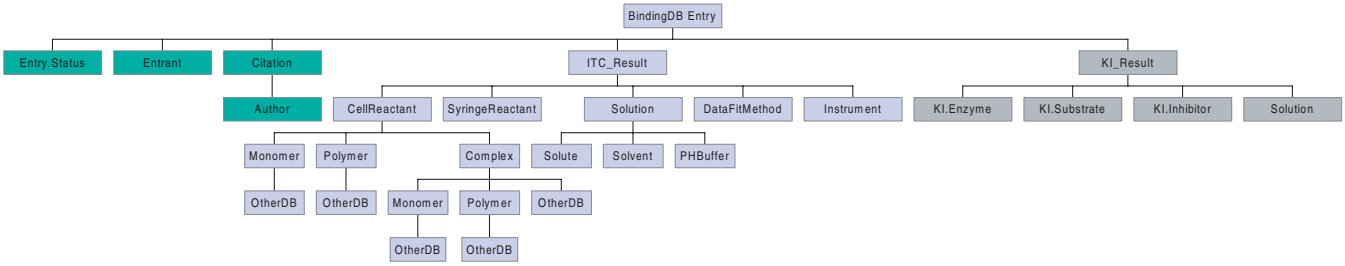

Fig. 2. Diagram of part of the DTD for a BindingDB XML file. Key nodes and only a few leaves (data attributes) are shown. Green: general information regarding the entry. Blue: information regarding an ITC measurement. Grey: information regarding an enzyme inhibition measurement. The full DTD is much more extensive; it includes, for example, Solute, Solvent and PHBuffer under KI.Result, and includes Monomer, Polymer, and Complex information under SyringeReactant, KI.Substrate and KI.Inhibitor. The full DTD is available at http://www.bindingdb.org/bind/deposition/xml/BindingDB.xml.

the BindingDB web-server use the Java Runtime class to control the execution of Blastall and return the results in HTML format with links from each polymer sequence to the appropriate binding reactions in BindingDB.

Chemical structure, substructure and similarity. BindingDB uses JChem1.5 (ChemAxon, Hungary; Csizmadia, 2000) to provide chemical search capabilities for small molecules: monomers alone and monomers in complexes. When monomer information is deposited—either via a chemical draw applet or by pasting an MDL Molfile into the deposition forms—JChem generates chemical fingerprints (http://www.jchem.com/doc/admin/GenerFP.html) that are stored in the MONOMER_STRUCT table in the relational database (see Figure 1). The number of fingerprint attributes—currently 16—can be adjusted to tune accuracy versus speed. A user can run a chemical search by either drawing a target structure with a chemical draw applet or by pasting an MDL Molfile into the query form. When the query is run, the fingerprint of this molecule is generated and searched with SQL against those in the MONOMER_STRUCT table. Potential hits are then retrieved and tested with a nonSQL algorithm. Confirmed hits are linked to binding reactions through the MONOMERID attribute in the MONOMER_STRUCT table and are returned in a display page with links to the binding measurements that involve each compound.

Linking directly to BindingDB from other applications on the WWW

Hyperlinks can be made directly to binding measurements in BindingDB from other web-sites or databases. For example, the URL http://www.bindingdb.org/servlet1/dbsearch/Summary?entryid=161&itc_result_a_b_ab_id=11&energyterm=kJ/mole provides direct access to Entry 161, Result 11 in the table of ITC studies of reactions of the form A + B $\leftrightarrow$ AB (i.e. ITC_RESULT_A_B_AB). The link connects to the Reaction Details page of the specified reaction.

DATA DEPOSITION METHODS

It is envisioned that data will be deposited into the BindingDB through at least three routes: literature extraction and deposition by on-site BindingDB staff; deposition of new data at or near publication time by off-site experimentalists not otherwise associated with the BindingDB; and high-throughput data uploads from computer-controlled instrumentation. The data deposition software is designed to support all three modalities.

BindingDB's first data deposition interface was a client–server tool for local use, developed with Oracle Forms v6.0. Data are entered via a series of forms and then saved into an intermediate database that contains the same data attributes as the main database, but has denormalized tables in order to simplify the connection to the Forms client. Data in the intermediate database are then transferred to the main database via a PL/SQL script. This client–server approach has several advantages. First, the Forms software provides tools to facilitate the development process. Second, the resulting software enforces table relations during deposition, thus minimizing concerns about data integrity. Finally, because the client connects directly to the intermediate database, it can reduce the work of deposition by allowing the user to view and reuse existing data. However, this approach also poses serious limitations. First, the Forms software is somewhat inflexible and thus does not allow the user-interface to be highly tailored for the application. More importantly, the client must be installed on the user's machine, making it difficult to distribute and to keep updated. Although Oracle Forms can generate a Java version of the client that should, in principle, be highly portable, we found that the resulting application executed very slowly and did not work perfectly on all targeted platforms.

We have therefore developed a Web-based deposition

method that is compatible with widely used browsers. In this method, data deposited via the Web are used to generate an XML file (see section XML Data Description) that is then parsed into the main database. It is expected that ultimately all BindingDB data will be deposited via this XML format, which will thus serve as a final common pathway for data deposited via forms on the WWW, from automated instrumentation, and potentially from customized forms created by other research groups and database developers. For example, we are currently collaborating with a manufacturer of microcalorimeters to facilitate data deposition from their proprietary data analysis and management software.

User interface for data deposition

Each entry in BindingDB has about 400 attributes that can be entered by a depositor, though many are not mandatory. It is a nontrivial task to design a good interface to collect so many data efficiently. This section briefly describes the user-interface, and then elaborates on design features that enhance usability.

The main interface for data deposition consists of two browser frames. The left-hand frame always displays the list of data categories, and the right-hand frame displays the data entry form appropriate to the selected category. Each data categories on the left is hyperlinked to the appropriate data form on the right. The categories are: Entry, Entrant, Citation, Syringe Reactant, Cell Reactant, Solutions, Data Fitting Method, Instruments and Results. The top-down ordering of the links in the left-hand frame is intuitively reasonable. More importantly, following this order ensures that any data required for a given category has already been entered in a previous category. However, the user is also permitted to jump among categories so long as data integrity would not be violated. For example, the user is not able to specify a molecular complex until at least one Small Molecule or Polymer has been entered, because a Complex must be built from Small Molecule and/or Polymer constituents. Similarly, links to Results are inactive until mandatory information on Reactants, Solutions, etc. have been entered and saved. The user can check the completeness of an entry by clicking 'Check and Submit'. The resulting report highlights which mandatory data are missing. If none are missing, the user can complete the submission process with one more click.

It is important that the user be free to begin a deposition, exit from the web-site, and then resume the deposition process without losing data. This capability is supported through by assignment of a unique deposition ID to each entry—completed or in progress—and storage of incomplete entries in the DBMS as JavaBeans (see section Implementation of WWW deposition method), along with the appropriate deposition IDs.

Even after submission of an Entry is complete, data can still be revised by the entrant (depositor). Thus, data that are marked 'Hold' can be revised via the 'Continue Deposition' link on the home page, while data that have been submitted without a hold can be revised via the 'Revise Existing Entry' link. In the latter case, the user is provided with a duplicate of the existing entry but with a new ID. The user can then modify the entry, add a description of the reason for the revision, and resubmit the data. The old entry is retained but marked as obsolete to prevent it from being returned on routine queries.

It is worth pointing out several features that are included to promote usability and avoid errors during the submission process. First, the left–right category–detail interface design gives the user an understanding of the overall structure of the data and a task-oriented workflow, while allowing free navigation among categories and protecting against violations of data integrity. Second, drop-down menus of standard items have been provided where possible, such as for journal names. Different scientific units are allowed, notably kcal/mole versus kJ/mol; the software automatically converts into consistent internal units for storage. Because experiments are often repeated with only minor changes in conditions, a 'Replicate Result' option is provided to copy an existing Result record into a new one, to be modified and saved. Finally, data entered in one category that are used in another category are made available in pull-down menus for convenience and to ensure the formation of correct relational links. For example, all of the reactants entered in the ligand, polymer and complex categories are automatically placed into pull-down menus in the results form. If the user references a ligand, say, in the results form, and later deletes the ligand that was used, a warning message is generated.

Implementation of WWW deposition method

The overall structure of the web-accessible data deposition method is as follows. On-line forms in standard HTML are generated and updated dynamically with JavaServer Pages technology (JSP: http://java.sun.com/products/jsp/). Data entered by the user but not yet forming a complete entry are configured as JavaBeans objects (see two paragraphs below) and stored as binary objects in the Oracle DBMS. The depositor can recall, view, edit and complete these data using the JSP forms. When an entry is complete, the data in the JavaBeans are used to generate an XML file following the DTD outlined above. The Oracle XML parser and the DTD are used to check for completeness of mandatory data items, and XML files passing this validation stage are then transferred by the parser into the relational BindingDB database. A facility is also provided to move data from an XML file into JavaBeans in the DBMS so that data uploaded as XML files can be reviewed and edited as needed via the on-line forms.

JSP technology was selected as the method for generating dynamic forms because it runs on the server side and can generate standard HTML pages, thus minimizing download times and demands on the technical capabilities of the user's browser. Javascript code is used to warn the user of simple errors during data entry, such as entering alphabetic characters in a numeric field. More complex errors are handled by the following mechanism. The JSP code can format and display the data in each form displayed in the right-hand frame in one of three modes: 'new', 'retry' and 'success'. When the user first accesses a deposition page, it is shown in 'new' mode, with all data fields blank. After some data have been entered and saved, accessing the form causes the data stored in the corresponding JavaBeans object to be retrieved, and the form is displayed in 'retry' mode with the appropriate fields filled out. The user is then able to modify and complete the form. After all the data in a form are complete, the user clicks the Save button, which activates the validation methods of the appropriate JavaBeans object. If the data pass the test, the JSP code presents the data in the 'success' format which does not include any form fields but is instead a plain report of the final data. (However, leaving and then returning to the page also allows the user to further edit data that have already been validated.) If the data fail validation, the page is presented again in 'retry' format, with annotations indicating the problem items. This design architecture is used for every form page and facilitates maintenance and modification of the pages and validation criteria.

As noted above, data entries are stored in the form of JavaBeans objects (http://java.sun.com/products/javabeans/) in the DBMS. Each form page—e.g. entrant—is stored in one row of one database table, along with the deposition ID of the entry. (This table is not included in Figure 1 because it is only loosely tied to the main database tables.) JavaBeans—Java classes coded with predefined patterns for their property and method information—are useful for several reasons. Thus, data attributes can be saved and retrieved through their methods into and from property attributes and, as noted above, validation methods are included in each JavaBean object. Also, the introspection feature of JavaBeans automatically matches the names of the bean properties with the names of the form input elements. This feature alone is extremely helpful since each deposition page is a complex form containing a significant number of input elements.

Two steps are involved in the transfer of a validated data set in the form of JavaBeans objects into the main database. First, the data in the JavaBeans are used to construct an XML file according to the DTD described above. The creation of the XML file is a recursive process. It starts at the top node, but all lower level nodes must be completed before the current node is completed. The deepest recursion has five levels: result/cell reactant/complex/monomer/otherDB. Second, the XML file is parsed into the relational database, as follows. Given an XML file, the database's ENTRY table is queried to identify whether the file contains a new entry, based upon deposition date, title, measurement technique and comments. If this is a new entry, a new row is added to the ENTRY table and a new entry ID is assigned. The subsequent deposition sequence follows relation rules since the data integrity is rigorously reinforced by the relational database. The transfer of data from XML into the DBMS is greatly simplified by the XPath utility of the Oracle XML parser. The parser accepts an XPath expression and returns all nodes or elements that match it. For example, the command 'selectNodes('BindingDB_ITC/Citation[Citation.Title=Binding Studies of Lectins']/@Author')' will return all authors in the paper entitled ''Binding Studies of Lectins'. This result can then be deposited into the PERSON table of the database.

CONCLUSIONS AND PLANS

The BindingDB, now in operation, provides quick response times and an intuitive and flexible interface, according to user surveys. Continued development, built upon the techniques described here, is proceeding smoothly. The methods thus appear to be effective and may be of interest for use in other scientific databases. We invite further comments from users regarding all aspects of the interface as well as the data specification. It is hoped that such input will lead over time not only to a widely used database, but also to a stable and generally accepted data specification, presumably in the form of an XML DTD, that will promote exchange and access to binding data.

We also invite the deposition of new data. The relevant policies are: (1) data are acceptable if the materials and methods are published in a refereed journal; (2) freely deposited data will remain publicly accessible via the BindingDB so long as it is in operation. Ideally, experimentalists will deposit their new data in BindingDB concurrently with publication, so that most binding data become accessible on the web. In the future, we plan to address the issue of data quality further by establishing a mechanism for data checking and curation. Also, it may be useful to allow users to attach annotations to data in BindingDB, subject to the requirement that the annotations themselves be explained and supported in a refereed publication.

It will also be important to integrate BindingDB further with other biomolecular databases. For example, links can be made to different descriptions of binding reactions in other molecular interaction databases like ProNIT, DIP, BIND, and Relibase. Such links can take the form of database IDs or URLs stored in BindingDB

or—perhaps preferably—dynamic queries performed in related databases at run-time.

Finally, as BindingDB grows, it may be necessary to optimize performance to minimize response times. This can be done by indexing the commonly searched attributes, such as reactant name, author name and keywords, and by distributing the processing of queries across several computers. Network latencies can be minimized by the use of geographically dispersed mirror sites.

ACKNOWLEDGEMENTS

This project is supported by NSF grant DBI-9808318 and benefitted from early support by the National Institute of Standards and Technology through the Standard Reference Data and Systems Integration for Manufacturing Applications programs. We thank Drs P.Bourne, H.Berman, K.Breslauer, M.Doyle, N.Hodge, D.Pilch, E.Plum, J.Rumble, F.Schwarz, J.Westbrook, and W.Windsor for many helpful discussions. We also thank ChemAxon for facilitating access to the JChem software.

REFERENCES

Aberer,K. (1994) The use of object-oriented data models in biomolecular databases. *Conference on Object-oriented Computing in the Natural Sciences*. Heidelberg, Germany, pp. 3–13.

Achard,F., Vaysseix,G. and Barillot,E. (2001) XML, bioinformatics and data integration. *Bioinformatics*, **17**, 115–125.

Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Benson,D.A., Boguski,M.S., Lipman,D.J., Ostell,J., Ouellettee,B.F., Rapp,B.A. and Wheeler,D.L. (1999) GenBank. *Nucleic Acids Res.*, **27**, 12–17.

Cattell,R.G. (1996) *The Object Database Standard: ODMG-93*. Morgan Kaufmann, San Francisco, CA.

Carazo,J.M. and Stelzer,E.H. (1999) The BioImage Database Project: organizing multidimensional biological images in an object-relational database. *J. Struct Biol.*, **125**, 97–102.

Chen,X., Liu,M. and Gilson,M. (2001) Binding DB: a web-accessible molecular recognition database. *Com. Chem. High T. Scr.*, in press.

Codd,E.F. (1972) Further normalization of the data base relational model. In Rustin,R. (ed.), *Data Base Systems*. Prentice-Hall, Englewood Cliffs, NJ, pp. 33–64.

Csizmadia,F. (2000) JChem: Java applets and modules supporting chemical database handling from web browsers. *J. Chem. Inf. Comput. Sci.*, **40**, 323–324.

Date,C.J. (1995) *An Introduction to Database Systems*. Addison-Wesley, Reading, MA.

Hutsman,M., Richelle,J. and Wodak,S.J. (1991) SESAM: a relational database for structure and sequence of macromolecule. *Proteins*, **11**, 59–76.

Kumar,A., Cheung,K.H., Ross-Macdonald,P., Coelho,P.S., Miller,P. and Snyder,M. (2000) TRIPLES: a database of gene function in *Saccharomyces cerevisiae*. *Nucleic Acids Res.*, **28**, 81–84.

Nakata,K., Takai,T. and Kaminuma,T. (1999) Development of the Receptor Database (RDB): application to the endocrine disruptor problem. *Bioinformatics*, **15**, 544–552.

Steedman,D. (1993) *ASN.1 The Tutorial and Reference. Technology Appraisals*. Twickenham, UK.

Weininger,D. (1988) SMILES: a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, **28**, 31–36.

Westbrook,J.D. and Bourne,P.E. (2000) STAR/mmCIF: an ontology for macromolecular structure. *Bioinformatics*, **16**, 159–168.

Zhu,J. and Zhang,M.Q. (1999) SCPD: a promoter database of the yeast *Saccharomyces cerevisiae*. *Bioinformatics*, **15**, 607–611.