

Konrad-Zuse-Zentrum
für Informationstechnik Berlin

ZIB

Takustraße 7
D-14195 Berlin-Dahlem
Germany

ALEXANDER STEIDINGER

Data Management for Processing Molecules

Data Management for Processing Molecules

Alexander Steidinger

Abstract

Molecular dynamics simulations of possible ligands for proteins yield large amounts of data in the form of trajectories which are further processed in order to find metastable conformations. These conformations can then be used for docking between ligand and protein.

Around this core computation procedure lots of other data have to be managed. It should also be possible for external users not involved in program development to perform computations.

As a paradigm for other fields where a similar constitution of program usage and data processing is found we present a software architecture for data generation, access and management.

Requirements for this system include: Ease of use, graphical user interface, persistent storage of data concerning molecules, users, programs, program parameters, metadata, and results. A mere storage in the file system would render a quick overview of data more or less impossible. On the other hand, storing large amounts of binary data in a database doesn't yield any advantage concerning speed of access. Therefore, a hybrid approach combining file system and database is appropriate.

The system should be easily extensible by inserting new applications which can be controlled and whose results can be collected and stored.

The software system described here consists of different components, the presentation layer (graphical user interface), the business logic, the persistence layer (relational database plus file system), and an interface to the compute cluster (batch system for parallel processing).

We will discuss the alternatives and take a closer look at the components.

MSC: 68P05, 68P10, 68P20, 68N15, 68U35

CR: D.2.11, E.1, H.2.4, H.3.2, H.5.2

Keywords: molecular structure, molecular comparison, fingerprints, software architecture, thin client, web interface, web server, relational database, batch system, XML

Contents

1	Introduction	2
1.1	Requirements	3
1.2	Alternatives and design decision	4
1.3	Employed software	5
2	Architecture	6
2.1	Overview	6
2.2	Components	7
2.2.1	Presentation layer	7
2.2.2	Business logic layer	11
2.2.3	Persistence layer	12
2.2.4	Data model	12
3	Improvements and Outlook	14
4	Acknowledgements	15

1 Introduction

As a research group for computational drug design we deal with the processing of molecular data in order to find molecules which could be candidates for new drugs. We try to find these molecules by first computing metastable conformations – molecule geometries with high probability and life time – which are then used for docking against proteins. The other aim is to find similar molecules to known drugs by comparison, either topologically or based on structure.

Several applications have been developed by different scientists which can be combined to obtain a workflow from preprocessing incomplete molecule files to computing docking results. In Fig. 1 we see a workflow example. In a preprocessing step molecule geometries are

process	data structure		format input	output	database storage
	in	out			
preprocessing	geometry	topology	PDB/SDF	ZMF	meta data
conformation analysis rigorous thermodynamical sampling	topology trajectory	trajectory trajectory	ZMF	ZMF	meta data, numerical values topology, fingerprints
filtering	topology	list	database	list	database
molecule comparison	trajectory fingerprints	matrix scored list	ZMF / database	ASCII	ASCII
docking	trajectory	complex	ZMF	ZMF	ASCII

Fig. 1: Workflow example: The processes in the left column can be combined to build a pipeline of data processing. The next two columns denote the data structures used for input or generated as output. Then the storage formats for input and output are given. The last column shows what data are stored in the database for the processes.

augmented e. g. by missing hydrogen atoms. The source files with geometric data stem from different molecule databases like PDB, Maybridge, Chembridge or NCI. They use the file

formats PDB [1] or SDF [2]. The results are stored in our own data format, the ZIB molecular format (ZMF), which has a human readable part at the beginning describing hierarchically what is stored as compact binary data at the end.

A ZMF file contains numerical, topological (bonds, bond angles, dihedral angles) and structural information (trajectories of cartesian coordinates from molecular dynamics (MD), Hybrid Monte Carlo (HMC) or other simulation methods). Bond lengths, bond angle values and dihedral angle values can be computed from cartesian coordinates and topological information.

We have developed a reader for this format which is flexible enough to read ZMF files even when the format is extended or data types are changed.

Next comes the conformation analysis which results in trajectories of molecule geometries stored in ZMF files. The applications Epos and Flow compute conformations using different approaches.

At this time numerical values like atom mass or charge are stored in the database together with topological properties like number of aromatic atoms or double bonds and structural information like fingerprints which incorporate information about the neighborhood of atoms.

To compute the thermodynamical weights of the conformations rigorous thermodynamical sampling is applied which results in ZMF output as well. There are three programs for doing rigorous sampling differing in the mathematical procedures: ZMFree, Jump and molRate.

To reduce the search space of molecules for later comparison a filtering according to numerical values like the Lipinski rule of five can be applied to the database yielding result lists of molecules that fall into the given range of parameters.

Molecule comparison is made by either comparing the 3-dimensional structure of molecules or by comparing fingerprints. *Hashed fingerprints* [3] consist in our case of bit sets of fixed a length where a bit set to one means that a certain property is present in a molecule. The mapping of topology to bit sets is not injective which means that different molecules can have the same bit set to one but if two bits in the same position differ, the molecules differ in this property. Thus, fingerprints are used to divide search space into sets of dissimilar molecules.

We have devised a second kind of fingerprints – steric fingerprints – which take into account the dihedral angles of molecules which we compute in the conformation analysis step.

The docking process proves how good the candidates match given protein molecules geometrically. A good docking indicates that the ligand molecule could influence the biochemical behavior of the protein.

To speed up computation we use a cluster of workstations for parallel processing. At the front end a batch system is employed with several queues where jobs are sent to by submitting batch files.

This workflow example shows that lots of data of different formats are generated and processed. The data generated by our applications are either stored in ZMF files or as ASCII files. The data types we deal with can be divided into integers like number of aromatic atoms, real values like mass or charge, topological information and 3D structural information.

1.1 Requirements

It has to be decided which part of the data should be left in the file system and what should be stored in the database.

Certain properties of molecules like mass, charge or number of aromatic atoms and some metadata are permanently used for management, filtering and search. It lends itself to store such unstructured data in a database. Atom coordinates of trajectories of binary data should

be left in files as there is no advantage concerning saved space or time to access data in the database.

Therefore, the hybrid approach of having data in the file system as well as in a database will suit best.

To enable a user not familiar with command line editing or even direct access to computation resources to execute applications a web interface is the method of choice for uniform remote access to computing resources. With a graphical user interface the user has the possibility to view molecules to judge the outcome of his computations qualitatively.

The aim of our system is to facilitate the use of all the applications, to help combine them to a workflow from data input over data generation to data comparison. It shall be made easy to select data to start with and to use the computation results for further experiments. The concept of a data pool helps the user to manage molecules he or she wants to perform processing on.

From the above-mentioned setting requirements are set up to meet the needs of users: It should be possible to

- access and filter data
- augment existing data
- insert new data even from remote machines
- search and retrieve data
- move, compress or bundle data
- visualize molecules or computation results
- extend the system, e. g. by attaching new applications
- submit jobs to the batch system for parallel processing
- control batch jobs
- start molecule processing from the desktop

Our system is a tool to fulfil these requirements.

1.2 Alternatives and design decision

There are alternatives to the architecture as well as to the used software.

A web based system with graphical user interface could either be realised by a web portal or by software tailored to the special needs of the users which are determined by the field of research. Web portals are used to integrate different resources and services and make them accessible under a common interface. An example is GridSphere [4], which is a framework for implementing web portals obeying the portlet specification [5], a definition how to establish code for portals to run in a portal server.

As we want to set up an architecture specialized for our needs but still extensible and changeable we don't need the overhead of functionality at this stage. If the need arises in the future to integrate our system into other computation environments we could promote the servlets to portlets as the two concepts have many similarities. The web server would then be replaced by a portlet server.

The graphical user interface could either be a fat client with an application running on each client machine which stores the state of the computation locally or a thin client like a web browser. A thin client has the advantage that only the presentation layer has to be present whereas the business logic is hidden from the user and is executed on the server side. Thus, if the business code changes the thin client need not be updated in contrast to a fat one.

As mentioned above a hybrid approach of data storage in the file system as well as in a database is appropriate. There are different types of database management systems (DBMS). The most flexible one concerning structured data is the object-oriented DBMS (OODBMS) as it can store objects of arbitrary complexity, but it lacks speed and there only exists a niche market for such databases today.

In consequence database manufacturers augmented their data by some structure which is why these DBMS are called object-relational databases (ORDBMS). The reality looks like this: The structure consists mainly of a types like date, enumeration, set and BLOB (binary large object). A BLOB can be used for storage if no suitable data type exists in the database. If one wants to access a data item of a BLOB it must be fetched from the database and unpacked. Saving big files or files containing binary data in a database has no advantages over leaving them in the file system.

If tabular data exists it is appropriate to store them in a relational database because access is fast. One important example in our setting is the storage of fingerprints as strings of fixed length. Then there is a second advantage for speed-up: Computations beyond those possible by using SQL (structured query language, the standard for definition, access and manipulation of data in RDBMS) can be performed by *stored procedures*, functions written in a language the DBMS can understand. They can be invoked within an SQL statement. In our case such a function can compare fingerprints by using bit operations. There is no need to fetch data from the database one at a time and make computations externally.

1.3 Employed software

There are different programming languages in use today to deal with the combination of graphical user interfaces and business logic like PHP, Perl or Java.

They differ in their programmatic approach: Java is purely object oriented. It was designed for building software projects whereas PHP and Perl were designed for the web and text processing, respectively. Code generation and debugging of Java programs is easier than ever by using integrated development environments (IDEs) like Eclipse [6] which is written in Java and in wide use.

The decision to use Java with its implications on the use of other software is not made by merely looking at the language capabilities but also by taking into account the possibility to combine existing software products like web servers and databases.

Java comes with a wide range of application programming interfaces (APIs) either from the developer of Java or from third party developers. It is an interpreted language but as we don't need high speed computation for the presentation of data and business logic it is sufficiently fast to reply to all user interactions.

Java packages we benefit from in our framework are the CDK [7] package for manipulating molecular data, the JavaView [8] package for visualizing molecules, and packages for processing XML data [9].

A web server is needed to supply dynamically generated content to the client's web browser. The Tomcat web server from the Apache Jakarta project [10] is fully implemented in Java. So, interaction between Java code in the business logic layer and the web server is straightforward.

The Apache Software Foundation [11] provides not only the Tomcat web server we use but also the most commonly used web server on the Internet, the Apache web server. In connection with the Apache server the use of languages like Perl or PHP is possible. To generate web pages dynamically these languages make use of the Common Gateway Interface [12], a simple interface for communication with information servers like web servers. This has the disadvantages of starting a process for each web request which consumes time and resources and having no common or persistent state of variables between two web requests, i. e. the session tracking has to be handled by the programmer.

Another approach is the use of modules for Perl and PHP within the Apache server but there is still the drawback of the languages themselves. An extension to this is the invention of Perl Active Server Pages [13] which facilitates the generation and management of web content. As we decided to use one programming language throughout our system we abandoned the idea to employ this extension.

It is possible to integrate the Tomcat servlet container into the Apache server. Each time the URL locates a Java Servlet [14] or JavaServer Page (JSP) [15] the Apache server delegates these to Tomcat. Servlets are Java classes containing Java code to print out HTML code. JSPs are files which should contain mainly HTML code where Java code is embedded. The main functionality is delegated to JavaBeans referenced inside the JSP. This facilitates to separate the representation part from the business logic part such that web site designers can concentrate on HTML and pretty printing while Java developers manage the functionality.

As a database management system we chose MySQL [16]. Starting with the aim to provide simple and fast access to data it evolved to a convenient system with more and more features we need for our framework. MySQL comes with a Java JDBC driver [17] for database access via the Java language.

All three products – Java language, Tomcat web server, and MySQL database – are free of charge. The quality of the products is guaranteed by a large user community which contributes to reduce errors.

2 Architecture

2.1 Overview

We now give an overview of the software components of our architecture. A rough sketch can be seen in Fig. 2. The architecture is divided horizontally as well as vertically.

On the client side we have a graphical web browser for the display of dynamically generated web pages. The user can interact with the system via input fields and click buttons. He or she can manipulate the graphical representation of molecules by mouse motions. With text-based browsers like Lynx it is also possible to start computations but they lack the possibility to judge results visually. Furthermore, they have difficulties in displaying tables. To get the full capabilities of the system the browser should be Java capable as there are Java applets implemented for graphical representation of molecular data. The reason for using applets is the possibility for the user to interact with the web content, e. g. by rotating, moving and scaling molecules. The applet runs as a process in the browser. No other software is needed on the client side.

On the remote side several servers work together to make the data flow possible.

HTML pages are generated by the Tomcat web server [10] using Java Servlets and mostly JavaServer Pages.

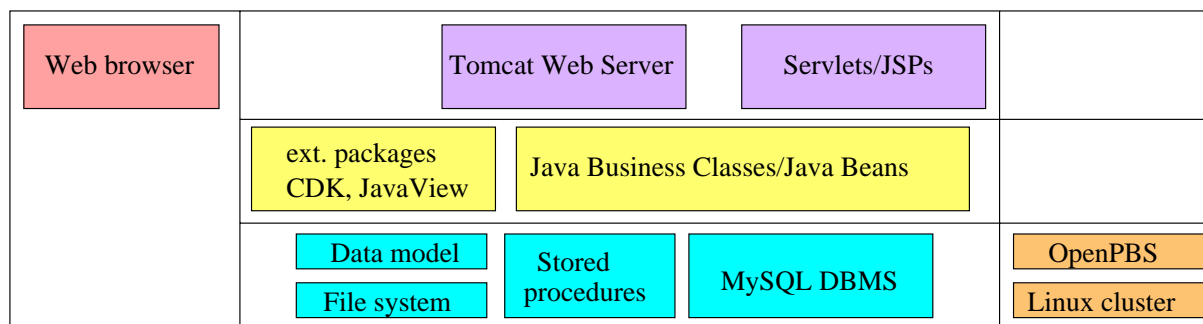


Fig. 2: Components of the software architecture, divided vertically into local client and remote server side, horizontally into presentation layer, business logic layer and a heterogeneous part at the bottom where persistent data either in the file system or in the database are accessed by the business layer or via the DBMS. External code of stored procedures is executed within queries to the DBMS. Batch jobs are submitted to the batch system of the Linux cluster as external processes started by the business classes.

The business logic is performed by JavaBeans associated with JSPs and additional Java classes. The bean classes contain code that should not appear within HTML code to keep the presentation and the logic apart. This layer has full access to system resources under the ID of the web server user. Packages external to the standard Java API are used for processing molecular data and displaying them (CDK and JavaView).

The persistence layer comprises the MySQL DBMS together with stored procedures written in C. The database is accessed via a JDBC driver delivered together with MySQL. Java enables an efficient SQL processing by generating so-called *prepared statements* that means SQL statements which are precompiled and can be reused with different query parameters. For small to middle range data requirements this DBMS suffices. There is no overhead for memory management like defining table spaces before generating tables. The database can be used nearly instantaneously and there is no need to employ a separate database administrator.

To speed up computation by parallel execution we use a Linux cluster consisting of 16 nodes with 2 processors per node. As an alternative, a Cray XD1 can be used. It has 6 nodes with 2 processors per node. Computation is triggered by submitting batch files to the OpenPBS batch system, which queues the jobs until enough processors are available. Users can choose between different queues. They differ in the amount of processors dispatched for a job and the maximal running time. If a computation has finished an email is sent to the user. More information on the hardware infrastructure is available at [18].

2.2 Components

2.2.1 Presentation layer

The user works with a Java enabled web browser to manage data and processing. User interaction is session-oriented through the login process. The session object unique to each user contains parameter values used during the session. If the browser accepts cookies they are used instead by the web server to store parameters.

It depends on the browser's capabilities and settings if a cookie persists for a session or longer. The maximal number of cookies a browser can store – at least 20 per unique host after the recommendation of the IETF [19] – could be too little to hold all the information generated during a session. This depends on how the web servers manages its cookies.

To circumvent this problem cookies can be disabled in the Tomcat web server. This way

it is assured that always session objects are used. This allows to easily check whether a user requesting a web page has logged in before. In Fig. 3 we see a part of the main page presented to the user after logging in with the description of the functionalities.

The user can choose between the following options

- overview of data, tarring, extracting, deleting, moving files
- file upload from the client’s machine to the remote server via the functionality of the web browser
- defining parameter input masks using XML
- computation: set parameters, choose queue, number of processors
- job control: viewing batch queues and killing erroneous jobs
- visualization of molecules with the JavaView package
- filtering of database according to molecule parameters
- comparison of molecules by means of fingerprints

The center of the page displays the applications from different developers which can be used together to form a complete workflow where the output of one program can serve as the input of the next one. The workflow ranges from data enrichment, data extraction and conformation analysis [20] to docking [21] and alignment [22, 23] of metastable conformations.

In detail, the current programs which can be applied to selected files are: Flexible and rigid docking, i. e. the molecules themselves are flexible or rigid during the docking process. Flexible alignment moves similar molecules until they are maximally aligned. If two molecules are docked together the quality of docking can be quantized by computing a score. After molecular dynamics simulation resulting in trajectories of molecule geometries the conformation analysis computes metastable conformations, i. e. geometries with high probability and long life time. In a preprocessing step, files in formats like SDF are read and augmented by missing data, e. g. hydrogen atoms. Flow is another program for computation of conformations, whereas ZIBgridfree does conformation analysis.

All the applications are executed in the batch system. To control the jobs a listing of the user’s and all other jobs is displayed. In case of a failure the user can kill his own jobs via a job ID.

Central to the applications is the data pool where the user selects the molecules he or she wants to process – viewing or running applications with them. It is individual to each user and persistent also between web sessions as it is realized as a hidden file in the users data path.

Filtering molecular data stored in the database is possible by giving ranges of values like mass, charge, number of aromatic atoms or number of H-bond donors. This process is fast as it uses query statements of the DBMS. It serves as a preselection of molecules whose properties lie in a certain range of values. The selected molecules can then be submitted to the applications for further processing.

The molecule editor MarvinSketch [24] can be used to click together a molecule which can then be stored in many different formats like SDF.

Molecules stored in ZMF files can be viewed in the browser. This has the advantage that no external software has to be installed to get a first impression of how molecules look like. In

The screenshot displays a software interface with three main sections:

- Upper left:** A vertical menu with options: "show remote", "upload", "manage rem", "move", "Data Pool", and "input data". Below the menu is a "Submit" button.
- Middle box:** A table titled "Main" with columns "ID", "Application", "Version", and "Information". Each row includes "Run", "Overview", and "Setup" buttons.

ID	Application	Version	Information	Run	Overview	Setup
1	Docking	1.0	Flexible and rigid docking tool	Run	Overview	Setup
2	MFAIalign		Flexible alignment	Run	Overview	Setup
3	pIScoring	1.1	Scores docked complex	Run	Overview	Setup
4	RigidDocking	1.3	Rigid docking process	Run	Overview	Setup
5	Epos	1.0	conformation analysis	Run	Overview	Setup
6	Epos	1.0	preprocessing	Run	Overview	Setup
7	Flow	1.0	computes conformations	Run	Overview	Setup
8	ZIBgridfree	1.0	conformation analysis	Run	Overview	Setup
- Right side:** A vertical list of links: "show jobs", "filter", "fingerprint", "molecule", and "editor".
- Bottom left:** A section titled "Data Pool" with a "Select all" checkbox and three checkboxes for "1FKBL.zmf", "1L87L.zmf", and "1L87L_out.zmf".
- Bottom right:** A link labeled "iface".

Fig. 3: Screenshot of part of the main window. **Upper left:** Management of remote data like file upload, tarring, extracting, moving and deleting files. The data pool contains the molecules chosen for computations. With the help of an XML file molecular data can be inserted into the database. **Middle box:** Different applications can be chosen to perform molecular computations or preprocessing after molecules have been selected. The setup deals with parameters for computation. **Right:** Molecules can be viewed using the JavaView package. Own jobs can be viewed and deleted if necessary. The database can be filtered according to numerical parameters. Fingerprints in the database can be compared with a given one using bit operations on the bit sets of the fingerprints resulting in a normalized numerical value called Tanimoto coefficient [3] by which the results can be ordered. **Bottom:** In the data pool one can select molecules for display, similarity search or running applications on.

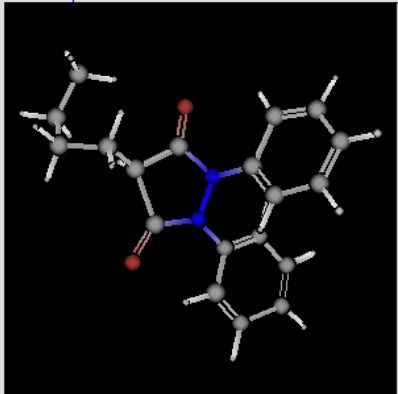
Visualizing Molecules

z	i	b
		c
		b

pubchem_phenylbutazone_out.zmf
59 Cluster

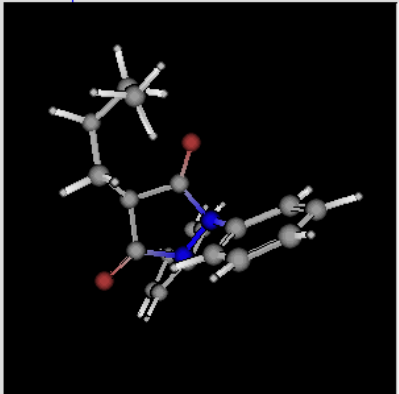
Conformation 5

timestep: 5



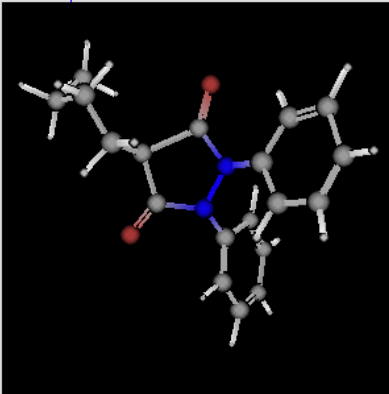
Conformation 6

timestep: 6



Conformation 7

timestep: 7



<< [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [11](#) >>

Some Keyboard shortcuts:

- Ctrl-p project panel: provides some display options
- w start autorotation
- q stop autorotation
- right mouse more options

Fig. 4: Example of visualization of phenylbutazone, a nonsteroidal anti-inflammatory drug and cyclooxygenase inhibitor: The given file contains a trajectory of molecule geometries, in this case the metastable conformations. Three of them are pictured. The user can scale and rotate the conformations to find out differences.

Fig. 4 you can see the visualization of phenylbutazone. The quality of conformation analysis can be judged roughly at a first glance.

Many different options can be chosen to manipulate molecules. The JavaView package makes it easy for developers to alter and extend the visual representation of objects, which is not constrained to molecules. Additionally, comparisons of molecular properties like fingerprints can be made visible e. g. as grey-scale matrix diagrams.

2.2.2 Business logic layer

The business logic is contained in JavaBean classes referenced from JavaServer Pages and in external classes. It provides access to the persistence layer by generating connections to the database and reading from and writing to the file system. The applications for molecule processing mentioned above are managed and surveyed by this business logic. The main functionality consists of

- logging into the system. After the user has logged in, a session ID is generated which is attached to a session object which is contained in each invoked web page. The session object represents the state of a session: it stores all attribute values generated throughout a session.
- managing the data pool. The data pool is persistent regardless of the current session. If the session ends either intentionally or due to a failure the data pool stays the same until it is changed during the next session. This is achieved by storage of the contents in a hidden file for each user in their main data path.
- storage of parameters. Mainly for the use of applications for molecular computations parameter values have to be set. In our case parameters are stored in files as input for applications which can automatically be deleted after they are read.
- start of queueing jobs into the batch system. The OpenPBS batch system [25] enables the use of several queues with different resource limits like computing time or maximal number of processors used.
- job control with the ability to kill one's own jobs. All jobs currently running are displayed to the user. He or she can recognize their own jobs by name. The job ID is unique and can be used to delete a running or queued job.
- reading special data formats like our ZMF file format. The ZMF reader traverses the hierarchical file structure given at the beginning of the file and identifies the byte code at the end. It is flexible enough to react to changes of the structure.
- processing molecular data with the CDK package. This Java package is used for storing molecular data in a specified format, e. g. mass, charge, element name, bonds or bond orders can be hierarchically accessed. The API allows the computation of *hashed fingerprints* and SMILES strings, i. e. the structure formulas [3]. A fingerprint in our setting is a bit set of a fixed length. The molecule is viewed as a graph with atoms as nodes and bonds as edges. A bit is set to one if a certain path up to a fixed length is present in the graph. Thus, equal subpaths of different molecules yield the same bits set to one. Therefore, this map is not injective and one cannot conclude from the same bit sequence that the molecules are similar. But one can conclude from different bit sequences that the molecules are dissimilar.

2.2.3 Persistence layer

The persistence layer has the purpose to permanently store data which is relevant over a long time span. Part of the data is stored in the file system like source files in the formats SDF, PDB, MOL2, and ZMF. Extracted data are stored in the database for filtering, searching and comparison as well as documentation of computations by using metadata.

The data stored in the database comprise the following:

- Data relevant to search and comparison: Topological data and numerical values like mass, charge, number of H-bond acceptors and donors, aromatic atoms and bonds and so on.
- For a fast and preliminary comparison of molecules, molecular fingerprints have shown to be an appropriate means. As these are bit sets, they can be efficiently stored as a series of bytes in the form of characters. Stored procedures written in C can read strings and perform comparison of fingerprints by comparing the characters of the strings. We use the *Tanimoto coefficient* [3] as a similarity measure. It can be computed fast and readily as its calculation involves only bit operations.
- User data is stored to control the access to the system and to assign computation data and results.
- XML files are used for the generation of HTML masks for computation parameters used by the applications. These are developed independently of this architecture for different purposes like alignment, docking or conformation analysis. Each developer generates the XML file for their application. The processing of XML files is achieved by using Java packages for reading and transforming them into HTML [26] [27].

2.2.4 Data model

For an efficient usage of database space and access to data the following concept for the data model has been established.

There are four different types of data that are stored in the database: (1) Meta data like location of files or user names, (2) numerical values of molecular data, (3) topological data like fingerprints and SMILES strings, (4) structural data like bond information and structural fingerprints using dihedral angles.

A filtering of the data under (2) yields molecules whose numerical attributes lie in a given range, e. g. mass, charge or counting numbers like number of H-bond donors or acceptors. The Lipinski rule of 5 gives a range of parameters for filtering out drug-like molecules [28]. This fast filtering process can serve as a front end for topological search and comparison of filtered molecules regarding the data under (3). The SMILES strings contain the topology in the form of structural formulas whereas the fingerprints incorporate neighborhood relations. The search in these data delivers molecules of similar topology which potentially have similar pharmacological properties.

A more time-consuming search can be performed using the structural information under (4). There, 3-dimensional data like bond angles and dihedral angles are used for comparison.

Part of the database tables can be seen in Fig. 5. The most important tables for storage of molecular information are *proteins*, *nonproteins*, *trajectories* and *topology* because they contain either molecular properties or information about how the molecular data was enriched

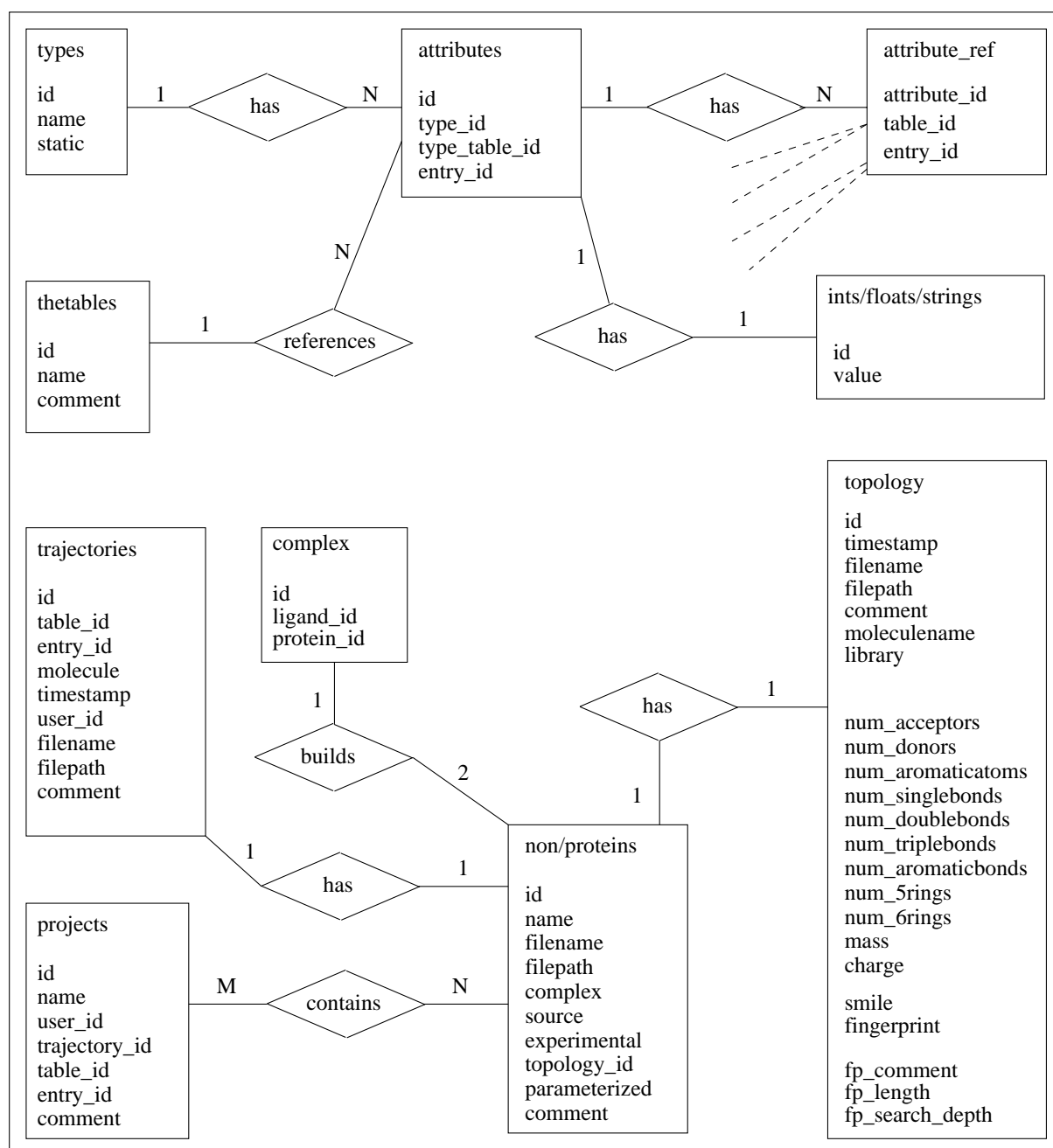


Fig. 5: Entity-relationship model for a part of the database schema. The diamonds represent relationships between the tables depicted as rectangular entities. The numbers at the lines denote the cardinalities of the relations, i. e. how many entities can take part in the relation. Not all relationships are shown to keep the picture simple. The main tables for molecular data are *protein*, *nonprotein*, *topology*, and *trajectories*. Mostly metadata are stored except for the topology table which contains numerical values for filtering molecules. Not shown are e. g. the client and steric fingerprint tables. The dashed lines in the upper right corner emanating from the *attribute_ref* table shall indicate that the attributes refer to several tables.

or preprocessed (parameterized), e.g. whether molecules exist as a complex or whether the data is experimentally determined.

All tables are noted in *thetables*. Values not belonging to static attributes of the main tables are stored in the *ints*, *floats* or *strings* tables. If a new attribute appears it is saved in the *attribute* table where the type is entered in *types* and standard attributes marked as static. The table in which an attribute appears as well as the location within this table is saved in *attribute.ref*. *proteins* and *nonproteins* contain information about preprocessed data. If a protein and a ligand form a complex this is noted in *complex*.

Important for the filtering of molecular data is the *topology* table. It contains numerical and topological values like mass or fingerprint. The fingerprints are generated as bit sets by viewing them as graphs with atoms as nodes and bonds as edges. From the paths of the graph random numbers are generated for setting the bits. The length of the fingerprint and the search depth within the graphs are stored together with the fingerprint.

The user can define projects for his computations. In the *projects* table it is noted which proteins and non-proteins belong to a given project. If *trajectories* are computed their location and affiliation to proteins or non-proteins can be stored.

3 Improvements and Outlook

The data model is divided into the file system part and the database part. Currently, users have to know the location of their data. In the future it is the aim to give the users a transparent view of the data such that they don't need to know its location.

In order to make the system available to a large number of users some improvements have to be made.

To prevent unwanted access to the system, the security mechanism using merely the session ID will not suffice. Only admitted users should be able to use the system. It has to be prevented that a user enters a web page without passing the login process. Thus, the session ID has to be checked for each web page. However, this is not sufficient as one could guess or copy the ID from somewhere. And if the user has cookies enabled in his web browser no session ID is generated. A more elaborated procedure uses so-called credentials, e.g. the user's password to generate a token with cryptographic algorithms which has a certain lifetime. This can be achieved by using the standard Java packages Java Cryptography Extension (JCE) [29] and Java Authentication and Authorization Service (JAAS) [30]. To differentiate between users concerning their access rights one could employ access control lists (ACLs) which use roles. Such roles can be defined for the Tomcat web server and would be stored in the database. An encrypted data exchange between web browser and server via the https protocol can be employed.

If many users take part in the system at the same time, the web server and the database must be prepared for it. The establishment of a database connection consumes time and resources. In contrast to opening and closing a connection for each user request connections can be reused with the help of a *connection pool*. The Tomcat web server can be configured for using such a pool.

It should be easily possible to replace our DBMS with another one as only standard SQL is used. The only thing to take into account are the stored procedures which can differ from database to database.

For the representation of and access to computation results we plan to use a simple XML data format which is then transformed to HTML. The check of XML files concerning syntax

only does not prevent the use of wrong datatypes for the attribute values or wrong tag names. In order to preclude such failures, XML Schema [31] could be used to define data types for each attribute. An XML validator can then find mistakes if an attribute name does not match the schema.

The email messages of the batch system which display only rudimentary status information could be enhanced by sending additional information via JavaMail.

To separate the web server from the business logic and database part, a remote interface to an RMI server (remote method invocation) [32] can be employed. The local methods of the servlets and JSP beans are only stubs (surrogates). The methods' code is executed within the RMI server at the remote site. This has the advantage of having different operating system user IDs for web server and RMI server and thus the users can have different access rights to system resources. Also, if the web server crashes the programs running in the RMI server are left undisturbed.

4 Acknowledgements

I wish to thank the following students for advancing this software architecture to the current state: Aysam Gürlür for contributing to the insertion and access to our applications in the web interface, Sebastian Moll for the setup of the table structure for storing arbitrary attributes, and Antje Wolf for her commitment to enhance our ZMF file reader, the implementation of the data pool and the inclusion of JavaView for making the visualization of molecules in the web site possible.

Furthermore, I want to mention that this work has been accomplished during my membership in the Berlin Center for Genome Based Bioinformatics [33].

References

- [1] Protein Data Bank. URL <http://www.rcsb.org/pdb>. 3
- [2] MDL CTfile formats. URL http://www.litlink.com/solutions/white_papers/ctfile_formats.jsp. 3
- [3] A. R. Leach. *Molecular Modelling, Principles and Applications*. Prentice Hall, 2001. 3, 9, 11, 12
- [4] GridSphere Portal Framework. URL <http://www.gridisphere.org/gridisphere/gridisphere>. 4
- [5] JSR-000168 Portlet Specification. URL <http://jcp.org/aboutJava/communityprocess/final/jsr168>. 4
- [6] Eclipse integrated development environment. URL <http://www.eclipse.org>. 5
- [7] The Chemistry Development Kit (CDK). URL http://almost.cubic.uni-koeln.de/cdk/cdk_top. 5
- [8] JavaView. URL <http://www.javaview.de>. 5
- [9] Extensible Markup Language XML. URL <http://www.w3.org/XML>. 5
- [10] Apache Jakarta Tomcat. URL <http://tomcat.apache.org>. 5, 6

- [11] Apache Software Foundation. URL <http://www.apache.org>. 6
- [12] The Common Gateway Interface. URL <http://hoohoo.ncsa.uiuc.edu/cgi>. 6
- [13] Apache Active Server Pages. URL <http://www.apache-asp.org>. 6
- [14] Java Servlets. URL <http://java.sun.com/products/servlet>. 6
- [15] JavaServer Pages. URL <http://java.sun.com/products/jsp>. 6
- [16] MySQL. URL <http://www.mysql.com>. 6
- [17] Java Database Connectivity JDBC. URL <http://java.sun.com/products/jdbc>. 6
- [18] BCB Cluster Complex at ZIB. URL <http://elfie.bcbio.de>. 7
- [19] HTTP State Management Mechanism. *Internet Engineering Task Force (IETF)*. URL <http://www.ietf.org/rfc/rfc2965.txt>. 7
- [20] M. Weber and H. Meyer. ZIBgridfree – adaptive conformation analysis with qualified support of transition states and thermodynamic weights. *ZIB-Report 05-17*. URL <http://www.zib.de/Publications/Reports/ZR-05-17.pdf>. 8
- [21] A. May, S. Eisenhardt, J. Schmidt-Ehrenberg, and F. Cordes. Rigid body docking for virtual screening. *ZIB-Report 03-47*. URL <http://www.zib.de/Publications/Reports/ZR-03-47.pdf>. 8
- [22] D. Baum. Multiple semi-flexible 3d superposition of drug-sized molecules. *ZIB-Report 04-52*. URL <http://www.zib.de/Publications/Reports/ZR-04-52.pdf>. 8
- [23] T. Baumeister and F. Cordes. A new model for the free energy of solvation and its application in protein ligand scoring. *ZIB-Report 04-51*. URL <http://www.zib.de/Publications/Reports/ZR-04-51.pdf>. 8
- [24] MarvinSketch – a molecule editor. URL <http://www.chemaxon.com/marvin>. 8
- [25] OpenPBS Personal Batch System. URL <http://www.openpbs.org>. 11
- [26] Xerces2 Java XML Parser. URL <http://xerces.apache.org/xerces2-j>. 12
- [27] Xalan XSLT processor. URL <http://xml.apache.org/xalan-j>. 12
- [28] C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Delivery Rev.*, 23(1-3), pages 3–25, 1997. 12
- [29] Java Cryptography Extension JCE. URL <http://java.sun.com/j2se/1.5.0/docs/guide/security/jce/JCERefGuide.html>. 14
- [30] Java Authentication and Authorization Service JAAS. URL <http://java.sun.com/j2se/1.5.0/docs/guide/security/jaas/JAASRefGuide.html>. 14
- [31] XML Schema. URL <http://www.w3.org/TR/2004/REC-xmlschema-0-20041028>. 15
- [32] Remote Method Invocation. URL <http://java.sun.com/docs/books/tutorial/rmi>. 15
- [33] Berlin Center for Genome Based Bioinformatics. URL <http://www.bcbio.de>. 15