The Use of Object–Oriented Datamodels for Biomolecular Databases

Karl Aberer

GMD-IPSI, Dolivostr. 15, 64293 Darmstadt, Germany
E-mail: aberer@darmstadt.gmd.de

1 Introduction

New biochemical and computational methods have led to an explosion in the available biomolecular data. Thus the management of large databases has become one of the important applications of computational tools in biomolecular research. From a data management viewpoint the main difficulties in managing biomolecular data are related only to a minor degree to the quantity of available data. Databases of much larger sizes are common in many application areas. The challenge is to deal with the complexity of the data with respect to the data structures, integrity constraints and algorithms. Thus conventional data management approaches, like file–based data management or relational database management systems, are not able to deal adequately with the typical requirements of biomolecular data management in general, despite of the fact that they are frequentely used to cover certain applications of the area. Thus the interest has risen in using object–oriented data models and data base systems for biomolecular data management. Object–oriented data models were especially developed to satisfy requirements from application areas where data and operations are intrinsically complex, like engineering, science or document management.

This paper reflects some experiences on the use of object–oriented data models for biomolecular databases, that were gained during the work on the Docking–D project[1], where object–oriented database technology is used to build up an integrated database for the support of drug design. We want to elucidate different aspects, on what are the advantages for using object–oriented data models in biomolecular data management, how object–oriented database management systems can be used and what are the limitations of the current state in the technology. The content of this paper is organized as follows. In Section 2 we will discuss the important types of biomolecular databases and the properties of the data they contain. In Section 3 we will review the basic concepts of object–oriented data models. Then we will be in the position to discuss in Section 4 some of the reasons why object–oriented data models are adequate to be used with biomolecular data. In Section 5 important aspects of data management and in particular the state–of–the–art of object–oriented database management systems will be presented. Finally, in Section 6, we study different approaches, that were taken for biomolecular data management.

1. Docking–D is part of the German national joint project RELIWE, a research consortium for the ”Computation and Prediction of Receptor-Ligand Interaction”, in which EMBL Heidelberg and GMD participate as research institutions and BASF, Ludwigshafen and E. Merck, Darmstadt, participate as industrial partners. This project is supported by the German ”Bundesministerium für Bildung und Wissenschaft” BMBW, grant number 01IB302E.

2 Biomolecular Databases

Current research efforts in molecular biology are directed to completely uncover the genomes of whole organisms and to understand the mechanisms of how an organism works at a molecular level. Both research directions produce different types of databases. For genome research sequence and mapping databases play the major role while for the research on biological mechanisms protein and small molecule databases are of importance. However, the different classes of databases are strongly interconnected as the information encoded in the genome eventually produces the basic functional units of biology. For overviews on biomolecular databases and elementary notions of molecular biology, see e.g. [24].

Sequence Databases. They are in the center of interest in molecular biology as the complete information on any organism is encoded as sequential information in its genome, by means of a four letter alphabet [11]. For example the DNA sequence of the humane genome alone contains about 3 billion base pairs. Therefore it is clear that huge datasets of sequence information are produced in biomolecular research. Genes are the basic functional unit of heredity and encode the information, that is needed to encode a specific protein sequence. About 10 % of the human genome consists of genes, which accounts to about 100.000 genes. Proteins are the basic functional unit of any organism. The transmission of the DNA information to proteins is performed by messenger RNA. Protein sequences consist in general of twenty different amino acids. In the translation process three base pairs of a DNA sequence are translated to one amino acid of a protein sequence. One can distinguish databases on nucleotide sequences and on protein sequences. Examples of nucleotide sequence databases are the EMBL data library with 2 million bases from 180.000 sequence entries in 1994 [40], the DDBJ Data Bank of Japan and GenBank of NCBI [12]. Prominent examples of protein sequence databases are PIR, which contained 1994 about 67.000 entries with 19 million residues [7], and SWISS–PROT with about 40.000 entries in 1994 [5]. A sample SWISS–PROT entry is given in Figure 1. It is important to note that the different sequence databases continuously exchange and update their data from each other. In particular protein sequence databases are updated from nucleotide sequence databases by translating nucleotide to protein sequences.

Mapping Databases. The genome of an organism is organized in chromosomes, e.g. the human genome contains 24 chromosomes. A genome map describes the order of genes and other characteristic markers, and the spacing between them on each chromosome. Such maps are produced at different granularities, corresponding to different mapping techniques, like genetic linkage maps or different types of physical maps [24]. Identifying characteristic fragments of chromosomes and order their locations on the respective chromosome is an important step in uncovering the human genome. The ultimate goal is to determine the base sequences of the characteristic fragments and establishing in this way the connection between the information on nucleotide sequences and their locations on the chromosomes. One example of a mapping database is GDB [15] which is based on a relational database system. A sample entry is given in Figure 2.

Protein Structure Databases. Protein sequences determine the three–dimensional structure of proteins in principle uniquely. The three–dimensional structure of a protein is the key element for its function. As there exists no tractable way to compute the three–dimensional structure from the sequence information, different methods must be employed for protein structure elucidation. The protein structure information is currently rapidly growing. It is important to remark, that the structure information on a single protein already leads to a considerably large dataset. The central repository

for protein structure data is PDB [8], which currently contains about 3000 entries. A sample PDB entry is given in Figure 3. Some techniques for uncovering the three–dimensional structure of proteins identify patterns on the protein sequences, so–called motifs, that are related to structural features. PROSITE [6] provides a library of such motifs.

Small Molecule Databases. The function of many proteins is to interact in highly specific ways with small (organic) molecules. In this context the proteins are called receptors, while the small molecules are called ligands. The investigation of the interaction of receptors and ligands is particularly important for pharmacology. As the three–dimensional structure of many pharmacological relevant proteins becomes available, an analysis of their interaction with small molecules becomes possible. Small molecule databases play by now an important role in pharmaceutical companies, which store in proprietary databases huge numbers of potential drugs in specialized, commercial database management systems. These systems support fast search on the topological structure of molecules. One public available small molecule database is the Cambridge structure database with about 100.000 entries obtained by crystallographic structure analysis.

We have characterized only some of the most important types of databases that are relevant for molecular biology. Many other databases cover specialized aspects of biomolecular research. For example, with regard to proteins other databases contain information on artificial protein mutations or protein alignments, which relate proteins that are similar due to the evolutionary process. Much information is still not prepared as structured data, but only available in the literature. Therefore bibliographic databases, like Medline [33], have also to be considered as relevant information resource in biomolecular research.

3 Object–oriented Data Models

As a basis for the forthcoming discussion we want to shortly review the basic concepts of object–oriented data models. Although there is no common agreement on a formal object–oriented data model, there exist basic concepts, that are considered to be characteristic for an object–oriented data model. We will present the concepts in the way they are primarily used in the context of object–oriented database systems [4], which sometimes slightly differ from those used in object-oriented programming languages. A detailled discussion of such differences can be found e.g. in [22].

Object Identity. The central abstraction in object–oriented data models is the notion of object. An object is characterized by three features, namely its object identity, its state and its behavior. The state is characterized by a set of attributes and the behavior by a set of methods, that are applicable to the object. The object identifier is independent of the state of the object, i.e. the identity of an object does not change when the value of its state changes. The concept of immutable object identifiers is in object–oriented data models an important consistency constraint and forms the basis for a generic referencing mechanisms.

Classes and Types. Objects that share common properties are described by classes. Object of a class share the same behavior. In object–oriented database systems often classes are used additionally as containers for objects, i.e. they correspond to a set of objects that are explicitely instantiated as members of the class and are stored in the database. This set of objects is called the class extension. Unfortunately there exists quite a confusion and disagreement on the notions of classes and types, which are often used interchangeably in different systems. If a distinction is to be made, the notion of type

relates to objects' interface and behavior only and is closer related to the concept of abstract data types in programming languages, while the notion of classes comprises also aspects related to the implementation of the objects, i.e. the structure of the objects' state and the methods' implementation.

Complex Objects. The attributes that make up the state of an object can be of a primitive data type, like integer, string, floating point number etc., can contain references to other objects, or can be of a complex data type built up by type constructors, like set, tuple, list, array, dictionary etc. Using the set type constructor one can define multi–valued attributes and. This allows to model mapping cardinality constraints between classes, like 1:1, 1:n or n:m relationships with attributes containing single object references and object reference sets. Other type constructors, e.g. for ordered data types, are offered by many object–oriented data models and are in particularly useful for advanced application domains, like scientific applications.

Encapsulation. The notion of encapsulation derives from the concept of abstract data types in programming languages. It comes from the need to distinguish between the specification and the implementation of an operation. The objects have a publicly accessible interface with the specification of the operations that can be applied to an object and a hidden implementation. For the implementation of methods typically a general purpose programming language is available. In this way a controlled access to objects is established and consistency constraints can be ensured. There is a distinction between two extremes of encapsulation, namely programming language encapsulation and database encapsulation. With programming language encapsulation the state of the objects is only accessible by methods of the object's interface, while with database encapsulation (part of) the state of the object is visible, and only the implementation bodies of methods are encapsulated. This enables for example direct access to properties for ad–hoc queries.

Inheritance. In order to reuse common features of classes most object–oriented data models support some sort of inheritance mechanism. With inheritance a class possesses besides the own attributes and methods also those of its superclasses. We distinguish data models that support inheritance from several superclasses, so–called multiple inheritance, and those that support only inheritance from one superclass, so–called single inheritance. The graph defined by the inheritance relationship between classes forms a hierarchy, i.e. it is acyclic. According to the information that is inherited one can distinguish three kinds of inheritance hierarchies in object–oriented data models. In a subtype hierarchy only the interface information is inherited, in an implementation hierarchy additionally the method implementations are inherited, and in a classification hierarchy the class extensions of the classes taking part in the hierarchy are in a set inclusion relationship. In many object–oriented systems the different types of inheritance are coupled.

Polymorphism. Polymorphism is defined as the usage of the same method name with different method implementations. One can distinguish ad–hoc polymorphism and universal polymorphism [13]. With ad–hoc polymorphism the same method name is used in different otherwise unrelated classes. Universal polymorphism occurs when classes share the same method name by means of a class relationship, for example inheritance. When a system cannot detect the correct implementation of a polymorphic method at compile time, it has to be chosen in run–time. One calls this mechanism late binding.

4 Object–Oriented Data Base Modelling in Biomolecular Databases

Data modelling in the biomolecular application domain requires flexible and expressive data models, because of the many complex concepts that are interconnected in various ways and the extensive usage of particular data types in scientific applications. In the following we discuss some typical requirements that occur in data modelling for biomolecular data and how they are covered by using object–oriented data modelling concepts. Of course this list must remain incomplete. For example many general aspects related to modular design or reuse will not be discussed [38], as they are not particularly related to characteristics of biomolecular data modelling and management.

Complex Structures. In biomolecular applications data is often organized as complex graphs. For an illustration of this, see the conceptual schema of an integrated protein–ligand database given in Figure 4. In this schema, for example, a protein consists of several chains, which in turn consist of a sequence of residues, which consist of single atoms. The object–oriented data model allows to express such relationships directly by using references. For comparison, in the relational model one needs to define primary and foreign keys, often artificially, and to store relationships between objects in separate tables. Thus the object–oriented data model allows to express the complex structures more directly. The consistency constraints on the identifiers, like uniqueness and immutability, are maintained by the system.

Semantic Constraints. A simple 1:n relationship, as provided by a relational or structurally object–oriented data model, is in principal sufficient to represent aggregate objects, like proteins, structurally correct. However, additional semantic constraints are defined with such a part–of relationship. For example, a single atom with coordinates in 3D space makes little sense without the context of the protein or molecule it belongs to. Thus the existence of the part object, the atom, depends on the existence of the whole object, the molecule. In this case we have the existence dependency constraint. On the other hand, for a protein–ligand complex, it makes perfectly sense, that both part objects, the receptor protein and the ligand, exist independently of the complex. This example shows, that although both situations can be modelled in the same way structurally, the constraints that are imposed on a relationship can be substantially different. Other constraints that might be considered in a part–of relationship are cardinality constraints or sharability of parts. The support of arbitrary constraints of this kind, for example those imposed on relationships, is one of the key features of object–oriented data models. They allow to maintain these constraints by encapsulating the objects, and ensuring that object creation, manipulation and deletion methods do not violate the consistency constraints. Some frequent types of constraints, as for example those indicated for the part–of relationship, are supported by many object–oriented database systems in a predefined way, similarly as relational database systems support basic constraints of the relational data model, like unique primary keys. However by means of the type definition mechanisms, also non–predefined application–specific consistency constraints can be ensured by an object–oriented database system, while in relational systems this responsibility is delegated to the applications.

Data Types for Scientific Applications. Different specialized data types play a key role for biomolecular data. In particular, ordered data types are needed to represent nucleotide or protein sequences or genome mapping data. It is one of the major drawbacks of the relational data model, that it inheretly does not support ordered data types. Object–oriented data models support ordered types in different ways. Either they provide built–in ordered data types as type constructors, like lists, arrays

or dictionaries. Alternatively ordered data types are defined by using the abstract data type definition mechanism. The same technique can also be used to define other relevant data types, like spatial data types needed to represent 3–dimensional structure information on proteins. Using the encapsulation mechanism it is possible to implement efficient data structures and to hide them through an operational interface. It is also possible to delegate expensive operations and complex algorithms, like sequence alignment or search algorithms, to existing external programs.

Derived Data. Derived data plays an important role in biomolecular data management. From the raw biomolecular data, different kinds of derived information can be computed on the fly. A simple example is the atom–atom distance in a molecule, that can be derived from the stored atom coordinates. Object–oriented data models allow to provide this derived data by means of methods. For comparison, in a relational database the derived data has either to be stored separately or each application has to implement the corresponding algorithms. Of course also in an object–oriented database the derived data can be precomputed and stored redundantly, which makes sense if the computation of this data is expensive. In this case by encapsulation the consistency between the original and derived data can be maintained, by providing appropriate update methods and hiding the object state.

5 Data management with Object-Oriented Database Systems

Data access. One of the main factors for the success of relational database systems is the provision of a declarative query language [23]. This allows the user to access databases in an ad–hoc manner. Query optimization and evaluation techniques relieve the user from dealing with the technical details of accessing a relational database efficiently. Of course such access mechanism are also desirable for biomolecular databases, as particularly in a research environment database access has to be flexible. The more flexible access to existing databases is, the more likely it is that analyzing and combining this data supports the creation of new insights in the research field. Many of the current system lack such flexible data access capabilities. In record–oriented file–based systems the users are either restricted to a predefined set of application programs or have to program the access functionality themselves. Although relational database systems provide flexible access mechanisms, the relational data model does not support a direct modelling of biomolecular data, as discussed earlier, and thus understanding and accessing database contents may become difficult.

The much richer set of data modelling primitives of object–oriented database systems makes the design and implementation of a declarative query language a more difficult task. Thus, most of the existing object–oriented database systems provide querying capabilities, that can be characterized as a combination of navigational and set–oriented access. Evaluation of conditions on objects or the values of attributes, paths and methods can be performed by set iteration. However, the user has still to decide on many operational aspects, like the strategy of class traversal. Finding an optimal strategy, which by the way is often equivalent to finding a feasible strategy, can be an intricate task in complex databases.

In contrast, fully declarative query languages for object–oriented databases allow to specify the information need, without any indication of how the data is obtained operationally. At the same time these languages allow to exploit some of the typical features of object–oriented data models, like

navigation along paths or the invocation of methods. Different query languages were proposed for that purpose [9], however the efficient processing of such queries is still an active research area. In particular the exploitation of the method semantics is a critical aspect to this regard [2].

Another critical issue, when accessing a database, is the appropriate communication of the structure of the data, i.e. the database schema, and the access mechanisms to the non-expert user. Since object–oriented data models enable a more direct representation of complex conceptual models [38], the communication of the database schema to the user is substantially simplified in complex applications. In combination with a simple–to–use ad–hoc query language, access to a database should then be possible in a relatively intuitive manner. Additional mechanism supporting the user in the process query formulation and evaluation may be envisaged [1].

Database Integration. There exist over hundred publicly available biomolecular databases and many more local ones. The value of each of these databases can be much higher if an integrated access to these databases would be possible [39][24]. By detecting regularities or inconsistencies or by combining complementing data, a much deeper understanding of the biomolecular domain can be gained.

Object-oriented database systems are particularly attractive to be used for database integration. Object-oriented data models provide data modelling primitives to establish relationships between classes, like part–of, generalization/specialization or association. These relationships can be used to integrate heterogeneous database schemas [25][30]. For example, by means of the generalization relationship, two classes, corresponding to data on the same aspect, e.g. nucleotide sequences, in two different databases, can be generalized to one common class in an integrated schema. Appropriately defined methods allow for example to resolve ambiguities and inconsistencies for the generalized class or to encapsulate the access to external databases on a physical level.

Current efforts are directed to standardize the interfaces to object–oriented systems [41], and thus supporting the integrated usage of heterogeneous systems. These standardization efforts are also a step towards simplifying the interaction between object-oriented databases and algorithmic tools based on object-oriented programming languages.

In order to progress in the important area of the integration of biomolecular databases, standardization efforts for biomolecular data are needed not only on a technical level, but also on a semantic level. For the same reasons why object-oriented data models are adequate to be used for biomolecular databases, they appear also to be a good choice for defining standards for the exchange and integration of biomolecular data. For a common understanding of the structure and semantics of biomolecular data, not only the data formats and structures need to be standardized, but also the semantics of the operations on this data.

Future Directions in Object–oriented Database Management Systems. Although the advantages of object–oriented database management are obvious, current limitations of the systems have to be considered. Lacking or not yet established standards or formal models on object–oriented data models and systems complicate the usage of the systems and the data exchange. The stability and performance of the systems, in particular in a multi-user and distributed environment needs to be further evaluated. Systems lack of database management facilities that are common in relational database management systems [28], e.g. declarative query facilities, views, or authorization mechanisms. However, as systems become more wide-spread and mature, many of these limitations can be expected to rapidly disappear.

With regard to management of biomolecular data, some particular requirements occur, that are covered by existing systems rarely, be it object–oriented or relational. They are more likely to be found in special purpose systems and applications developed for molecular biology. A particularly important requirement is the evolution of database schemas in populated databases, as the models evolve over time in biomolecular research. It would be desirable to provide dedicated support for characteristic properties of scientific data, like incompleteness, inconsistency, errors, valid alternatives or imprecision. In a research environment different users interact with the databases in different ways. To support this concepts for workflow management or versioning are needed. Security aspects play an important role in particular in an industrial environment. One can expect that due to the flexibility of object–oriented database systems, it will be much more likely, that such extended functionalities can be provided within these, than in conventional database systems.

6 Sample approaches

Standards for data exchange and integration

It has become clear from the previous sections that object–oriented data models provide many useful concepts that ease the modelling of biomolecular data. So it is not far fetched to use object–oriented data models or data models that adhere to some of the object–oriented concepts for defining standardizations for exchange and integration of biomolecular databases. We mention two approaches which follow fairly different philosophies.

SDDL. [17] SDDL provides a logical data model for sequence databases. It was developed in the context of the PIR [7] protein sequence database. With SDDL the syntax *and* semantics of the model are defined. In this way general aspects of the semantics needed for modelling of sequence databases are provided. SDDL consists of predefined types and type constructors, with specified operational semantics, which can be used to define schemas for sequence databases. In this way SDDL provides a general data model that reflect the characteristics of sequence data, but does not fix a particular database schema.

GenInfo. [36] GenInfo uses the standardized formal specification language ASN.1 to define a comprehensive data model for genetic data. It provides in ASN.1 syntax a global schema for genetic databases, and is currently also extended to protein databases. The data model adheres to some structural object–oriented data modelling concepts but does not define operational semantics.

Libraries for accessing file-based databases

With the library approach an object–oriented programming language is used for implementing applications that access biomolecular databases. Therefore only the data structures resident in main memory are represented according to the object–oriented data model. No particular support is given for the persistent storage of objects and multi–user access.

PDBLib. [14] PDBLib is implemented in the object–oriented programming language C++ and provides memory resident data structures that can be derived from PDB database entries. In this way it provides by means of a C++ class library an abstract interface to PDB. The classes of the library are

divided into four different groups. So-called intrinsic classes describe the macromolecular structures of protein chains, residues and atoms. Extensible classes provide a layer that separates the implementation details of intrinsic classes from the other parts of the library and the user. Iterative classes model sets of molecular objects and allow iterating and filtering over them. I/O classes are used to load molecular structures from files. PDBLib is the basis of a set of tools called PDBTool, that also provides a biology-oriented query language and a molecular browser to access the objects represented in PDB.

Dedicated biomolecular database systems

There exist several dedicated database systems that were developed for the management of biomolecular data. Most of them exhibit some structurally, object–oriented features.

AceDB. [16] AceDB is a special purpose object–oriented database system which was originally designed to meet the needs of the C. elegans mapping and sequencing project. The data model supports classes and methods, but does not support inheritance. The objects can be grouped in a hierarchical way. As key features the data model allows to dynamically change the database schema by adding new attributes, the possibility to attach freely searchable textual annotations everywhere, and particular support to efficiently handle null values. A number of graphical interface tools support the access to the database system. One of the tools supports set–oriented, navigational access to the database. AceDB is also used as a front–end for the Integrated Genome Database (IGD) [37], which integrates existing heterogeneous genome databases and is implemented on top of a relational DBMS.

Object-oriented database systems

These systems support the object-oriented data model also at the persistent storage level, i.e. the object–oriented data model is used to represent data stored on secondary storage and the access to the persistently stored objects is made (in principle) transparent for the database system applications. The functionality of such a system can cover a whole spectrum, ranging from efficient access mechanisms and buffering strategies, over support for multi-user access, to efficient processing of declarative queries. Considerable differences exist in the data models provided.

P/FDM. [20], [26] In P/FDM a rigorous data modelling approach is conducted to represent information, which is essentially derived from PDB, and to support a flexible retrieval mechanism on this information. The system is implemented in PROLOG and uses a functional object–oriented data model for the database schema. In the functional object–oriented data model classes can be accessed only by means of functions, properties correspond there to so-called stored functions. The distinction between stored and derived data is considered as a purely physical issue. The main focus of P/FDM is on enabling a declarative access to the database either by using the logic programming language PROLOG or by using the functional query language DAPLEX. Simple optimizations of DAPLEX queries are performed, like reordering the classes involved in a query and identifying operations that can be delegated to external (UNIX) scripts performing fast search operations on files. This system illustrates some of the advantages object–oriented data modelling together with a declarative query language and support for navigational access can offer: an intuitive logical data model of the database, navigation through object networks, the ability to represent derived data in the

data model, the combination of complex calculations with data retrieval and the easy extensibility of the database schema.

MapBase. [18] MapBase is a system to support the experimental workflow in a laboratory for constructing genome maps. As the application is in a multi–user environment it obviously calls for using a full–fledged databases management system. The choice fell on ObjectStore [32], which uses C++ as data definition and manipulation language. ObjectStore was considered to be one of the object–oriented database management systems, that offer good performance. The flexibility of C++ as data model allowed for a straightforward implementation of the necessary data structures and application programs. During project evolution however some problems occurred; C++ is designed as a programming language, and thus provides many low-level programming mechanisms. This makes it difficult to use and error prone for database schema design. For example for a single concept, like object references, several different mechanisms are available, or the programmer is responsible avoiding problems like memory leaks. The lack of a well-defined logical data model makes it difficult to communicate the schema to the users. Support for schema evolution is missing. The query facilities did not match the requirements, as only access to attributes, but not to methods, is supported. For these and other reasons MapBase has been developed as an intermediate layer between ObjectStore and the applications. It provides a simple special–purpose query language and a logging mechanism. The logging mechanism supports redo logs for recovery and schema evolution. As additionally multi–user access to ObjectStore proved to be problematic due to hot spots, the MapBase layer also handles this functionality by a simple serialization mechanism for atomic MapBase operations.

Docking–D.[3] Docking–D is a prototype for storing, retrieving and updating data on proteins and small molecules and is used to obtain new insights into the interaction of proteins and ligands by analyzing existing experimental data. The data comes from heterogeneous flat–file data bases, like PDB, SWISS–PROT, the PDM mutant database [35] etc., but also contains algorithmically or manually prepared additional data and corrections to the existing data. A methodology was developed to integrate all this data in one object–oriented database with an integrated schema, see Figure 4. For the implementation the object–oriented DBMS prototype VODAK [30] developed at GMD–IPSI is used. The VODAK data model provides all standard features of object–oriented data models, and additionally provides mechanisms that allow to introduce new application–specific data modelling primitives, for example class relationships [29]. The DBMS has a powerful SQL–like query language with a query optimization module, that performs algebraic optimization [2]. Application–specific optimization rules allow to exploit the particular semantics of methods defined in the database schema. The system also supports multi–user access fully by means of the open–nested transaction concept [34].

In the following table we summarize some properties of the systems discussed. Note that the information is given according to the literature and does not necessarily cover the latest state of the different systems. The database sizes are given for published applications and are used to indicate what order of magnitude the typical databases are.

	PDBLib	P/FDM	MapBase	AceDB	Docking−D
data model	C++	FDM	C++	AceDB	VML
support for persistent storage	no	yes	yes	yes	yes
query language	navigational graphical	functional	navigational	navigational graphical	SQL−like
query optimization	no	yes	no	no	yes
multi−user support	no	no	serialization of atomic commands	no	open−nested transaction management
approx. database sizes reported	n/a	60MB	50 MB	40MB	500 MB
support for schema evolution	n/a	no	yes	yes	no

7 Conclusions

To summarize, we have analyzed why the object–oriented data model is appropriate to be used for data modelling in biomolecular databases. They allow a structurally adequate representation of the data and can capture the operational semantics of applications. An analysis of existing biomolecular databases has shown, that for each fundamental class dedicated data types are required, that cannot be easily supported for example in relational database systems. The numerous, heterogeneous biomolecular databases make database integration techniques necessary, for which object–oriented database systems are well suited. Different examples of concrete systems have shown, how the object–oriented data model is used at different levels of processing, namely data exchange, data integration, data storage and data retrieval. Appropriate processing capabilities, as they are for example offered by object–oriented database management systems, are definitely needed. Despite of the progress made in the field, the situation is still not always satisfying. Only close interaction between experts from the fields of molecular biology and computer science can lead to substantial progress to cover the particular data management requirements in molecular biology.

Standardization efforts in the field of biomolecular databases would be extremely valuable, as the number of heterogeneous databases is rapidly growing in the area. An adequate approach to define such a standard, would be based on the object–oriented data model. As in computer science standards for object–oriented data models are evolving in parallel (e.g. OMG, ODMG, OQL [27]), such developments should be considered for the definition of standards for biomolecular data. In this way it will be easier to concentrate the standardization efforts on the particular application-specific aspects.

8 References

[1] Aberer, K., Klas, W., Furtado, A.L.
Designing a User-Oriented Query Modification Facility in Object-Oriented Database Systems
Proceedings of CAiSE*94, Utrecht, June 1994.

[2] Aberer, K., Fischer, G.
Semantic Query Optimization for Methods in Object-Oriented Database Systems
Proceedings of the 11th IEEE International Conference on Data Engineering, Taipei, Taiwan, March 6th–10th, 1995.

[3] Aberer, K.:
Demand–Driven Database Integration for Biomolecular Applications,
in [24].

[4] Atkinson, M., Bancilhon, F. DeWitt, D., Dittrich, K., Maier, D., Zdonik, S.
The object–oriented Database System Manifesto,
in Proc. 1st Int. Conf. on Deductive and Object–oriented Databases, Kyoto, Elsevier, 1989.

[5] Bairoch A., Boeckmann B.:
”The SWISS–PROT Protein Sequence Data Bank”,
Nucleic Acids Res., 19 (Sequences Suppl.), 1991, pp. 2247 – 2249

[6] Bairoch, A.:
PROSITE: a dictionary of sites and patterns in proteins,
Nucleic Acids Res. 20, suppl., 2013–2018, 1992.

[7] Barker W.C., George D.G., Hunt L.T., Garavelli J.S.:
”The PIR protein sequence database”,
Nucleic Acids Res., 19 (Sequences Suppl.), 1991, pp. 2231 – 2236

[8] Bernstein F.C., Koetzle T.F., Williams G.J.B., Meyer E.F. Jr., Brice M.D., Rodgers J.R., Kennard O., Shimanouchi T., Tasuni T.:
”The Protein Data Bank: a computer based archival file for macromolecular structures”, J. Mol. Biol. 112, 1977, pp. 535 – 542

[9] Bertino E., Negri M., Pelagatti G., Sbattella L.:
”Object–Oriented Query Languages: The Notion and the Issues”,
IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 3, June 1992, Pages 223–237

[10] Bertino, E., Martino, L.:
Object-Oriented Database Systems,
Addison-Wesley, 1994.

[11] Boguski M.S., Ostell J., States D.J.:
”Molecular sequence data–bases and their uses”,
in ”Protein Engineering” eds. Rees,Sternberg,Wetzel, 1992, pp. 57 – 88

[12] Burks C., et al.:
”GenBank”
Nucleic Acids Res., 19 (Sequences Suppl.), 1991, pp. 2221 – 2225

[13] Cardelli, L., Wegner, P.
On understanding types, data abstraction and polymorphism,
ACM Computing Surveys 17(4), p471–522, 1985.

[14] Chang W., Shindyalov I.N., Pu C., Bourne P.E.:
“Design and Application of PDBlib, A C++ Macromolecular Class Library”,
CABIOS, 1994.

[15] Cuticchia, A., Fassman, K.H., Kingsbury, D.T., Robbins R.J., Pearson P.L.,
The GDB (TM) Human Genome Data Base Anno 1993,
Nucleic Acids Research, Vol. 21 No. 13: 3003–3006, 1993.

[16] R. Durbin, J. Thierry–Mieg,
The ACEDB Genome Database,
WWW page, http://probe.nalusda.gov:8000/acedocs/dkfz.html.

[17] George, D.G., Orcutt, B.B., Mewes, H.W., Tsugita, A.,
An Object-oriented Sequence Database Definition Language (SDDL),
Protein Seq. Data Anal, 5, p 357–399, 1993.

[18] Goodman N., Rozen S., Stein L.:
“Building a Laboratory Information System around a C++–Based Object–Oriented DBMS”,
Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994

[19] Goodman N.:
An Object-Oriented DBMS War Story: Developing a Genome Mapping Database in C++
in [27], pp. 216 – 237.

[20] Gray P.M.D., Paton N.W., Kemp G.J.L., Fothergill J.E.:
”An object–oriented database for protein structure analysis”,
Protein Engineering, 1990. 3(4), pp. 235 – 243

[21] Hachem N.I., Qiu K., GennertM.,Ward M.:
”Managing Derived Data in the Gaea Scientific DBMS”,
Proceedings of the 19th VLDB Conference, Dublin, Ireland, 1993, pp. 1 – 12

[22] Heuer, A.
Objekt-orientierte Datenbanken,
Addison-Wesley, 1992

[23] Jarke M., Koch J.:
”Query Optimization in Database Systems”,
ACM Computing Surveys, Vol. 16, No. 2, June 1984, Pages 111–151

[24] Karp, P. (Ed.),
Final report on the 1994 Meeting on Interconnection of Molecular Biology Databases,
WWW page http://www.ai.sri.com/people/pkarp/mimbd.html, Stanford University, August 9–12, 1994.

[25] Kaul M., Drosten K., Neuhold E.J.:
"ViewSystem: Integrating Heterogenous Information Bases by Object–Oriented Views",
Proceedings of the sixth International Conference on Data Engineering, Los Angeles, California, February, 1990

[26] Kemp G.J.L., Jiao Z., Gray P.M.D., Fothergill J.E.:
"Combining Computation with Database Access in Biomolecular Computing",
Proc. of ADB 94, 1994.

[27] W. Kim (Ed.),
Modern Database Systems,
ACM Press, 1995.

[28] W. Kim,
Object–Oriented Database Systems: Promises, Reality and Future,
in [27].

[29] Klas, W., Aberer, K., Neuhold, E.J.
Object-Oriented Modelling for Hypermedia Systems Using the VODAK Modelling Language (VML)
A. Dogac, T. Özsu, A. Biliris, T. Sellis (Eds.) Advances in Object-Oriented Database Systems: NATO ASI Series F, June 1994. Springer-Verlag 1994.

[30] Klas, W., Fankhauser, P., Muth, P., Rakow, T., Neuhold, E.J.
Database Integration using the Open Object-Oriented Database System VODAK
Bukresh O., Elmagarmid A.K. (Eds.), Object Oriented Multidatabase Systems, Prentice Hall, 1994.

[31] Klas, W., Fischer, G., Aberer, K..
Integrating Relational and Object-oriented Database Systems Using a Metclass Concept
Journal of Systems Integration, Kluwer Academic Publishers, Vol. 4, No. 4, 1994. (revised version of 93–9)

[32] Lamb, C., Landis, G., et al.,
The ObjectStore Database System,
Comm. of the ACM, Vol. 34, No. 11, p 32–39, 1991.

[33] **Medline**, access via Entrez on WWW page http://atlas.nlm.nih.gov:5700/Entrez/version1/entreztopmed.html

[34] Muth, P., Rakow, T., Weikum, G., Brössler, P., Hasse, C.
Semantic Concurrency Control in Object-Oriented Database Systems
Proceedings of IEEE Ninth International Conference on Data Engineering, Vienna, Austria, April 19-23, 1993, pp 233–242. Los Alamitos: IEEE 1993

[35] Nishikawa,K., Ishino,S., Takenaka,H., Norioka,N., Hirai,T., Yao,T. and Seto,Y.
Constructing a protein mutant database
Protein Engng, 7, 733, 1994.

[36] Ostell, J.
Programmers reference: NCBI software development kit,
NCBI, Building 38A, 8600 Rockville Pike, Bethesda, MD 20894, USA.

[37] Ritter O., Kocab P., Senger M., Wolf D., Suhai S.:
“Prototype Implementation of the Integrated Genomic Database”,
Computers and Biomdeical Research, 27, pp. 97–115, 1994

[38] Rumbaugh, J., Blaha, M., et al.
Object–oriented Modelling and Design,
Engelwood Cliffs, N.J., Prentice Hall, 1993.

[39] Sillince M., Sillince J.A.A.:
”Sequence and structure databanks in molecular biology: the reasons for intergration”,
Journal of Documentation, vol. 49, no. 1, March 1993, pp. 1 – 28

[40] Stoehr P.J., Cameron G.N.:
”The EMBL data library”,
Nucleic Acids Res., 19 (Sequences Suppl.), 1991, pp. 2227 – 2230

[41] Soley, R.M., Kent, W.,
The OMG Object Model,
in [27].

9 Figures

```
ID   TRFE_HORSE     STANDARD;      PRT;   706 AA.
AC   P27425;
DT   01-AUG-1992 (REL. 23, CREATED)
DT   01-AUG-1992 (REL. 23, LAST SEQUENCE UPDATE)
DT   01-JUL-1993 (REL. 26, LAST ANNOTATION UPDATE)
DE   SEROTRANSFERRIN PRECURSOR (SIDEROPHILIN) (BETA-1-METAL BINDING
DE   GLOBULIN).
OS   EQUUS CABALLUS (HORSE).
OC   EUKARYOTA; METAZOA; CHORDATA; VERTEBRATA; TETRAPODA; MAMMALIA;
OC   EUTHERIA; PERISSODACTYLA.
RN   [1]
RP   SEQUENCE FROM N.A.
RM   93277958
RA   CARPENTER M.A., BROAD T.E.;
RL   BIOCHIM. BIOPHYS. ACTA 1173:230-232(1993).
CC   -!- SIMILARITY: STRONG INTERNAL HOMOLOGY BETWEEN THE 2 DUPLICATED
CC       DOMAINS OF THE PROTEIN.
DR   EMBL; M69020; ECTFRA.
DR   PROSITE; PS00205; TRANSFERRIN_1.
KW   IRON TRANSPORT; GLYCOPROTEIN; METAL-BINDING; DUPLICATION; SIGNAL.
FT   SIGNAL        1     19       BY SIMILARITY.
FT   CHAIN        20    706       TRANSFERRIN.
FT   DISULFID     26     64       BY SIMILARITY.
FT   DISULFID     36     55       BY SIMILARITY.
FT   DISULFID    134    215       BY SIMILARITY.
FT   METAL        79     79       IRON 1 (BY SIMILARITY).
FT   METAL       111    111       IRON 1 (BY SIMILARITY).
FT   BINDING     140    140       ANION (POTENTIAL).
FT   BINDING     480    480       ANION (POTENTIAL).
FT   CARBOHYD    515    515       POTENTIAL.
SQ   SEQUENCE   706 AA;  78094 MW;  2475866 CN;
     MRLAIRALLA CAVLGLCLAE QTVRWCTVSN HEVSKCASFR DSMKSIVPAP PLVACVKRTS
     YLECIKAIAD NEADAVTLDA GLVFEAGLSP YNLKPVVAEF YGSKTEPQTH YYAVAVVKKN
     SNFQLNQLQG KKSCHTGLGR SAGWNIPIGL LYWQLPEPRE SLQKAVSNFF AGSCVPCADR
     TAVPNLCQLC VGKGTDKCAC SNHEPYFGYS GAFKCLADGA GDVAFVKHST VLENLPQEAD
     RDEYQLLCRD NTRKSVDEYK DCYLASIPSH AVVARSVDGK EDLIWGLLNQ AQEHFGTEKS
     KDFHLFSSPH GKDLLFKDSA LGFLRIPPAM DTWLYLGYEY VTAIRNLRED IRPEVPKDEC
     KKVKWCAIGH HEKVKCDEWS VNSGGNIECE SAQSTEDCIA KIVKGEADAM SLDGGFIYIA
     GKCGLVPVLA ENYETRSGSA CVDTPEEGYH AVAVVKSSSD PDLTWNSLKG KKSCHTGVDR
     TAGWNIPMGL LYSEIKHCEF DKFFREGCAP GYRRNSTLCN LCIGSASGPG RECEPNNHER
     YYGYTGAFRC LVEKGDVAFV KHQTVEQNTD GRNPDDWAKD LKSENFKLLC PDGTRKSVTE
     FKSCYLARAP NHAVVSRKEK AACVCQELHN QQASYGKNGS HCPDKFCLFQ SATKDLLFRD
     DTQCLANLQP TTTYKTYLGE KYLTAVANLR QCSTSRLLEA CTFHRV
```

Figure 1: A sample SWISS–PROT entry. The amino acid sequence is given in the field with label SQ. References to other databases are given in the fields labeled with DR. In the field AC the unique accession number is stored.

```
GDB ID: G00-120-621     Symbol: G6PD              Type: Gene
Aliases: G6PD1
Locus Name: glucose-6-phosphate dehydrogenase
Cyto Location: Xq28                               Ref. Marker: No
Assign. Modes: Annealing of Homologous DNA-DNA or DNA-RNA Sequences;
               Linkage/Family Studies; Neighbor Analysis (part of large
               fragment studies); Analysis Based on DNA Digested with One or
               More Restriction Endonucleases; Somatic Cell Hybrids
MIM: 305900
EC: 1.1.1.49
EST:
PIR:
DNAseq: M12996,M19866,M21248,M24470,M26748,M26749,M26750,M35604,M57554,
        M57560,M57561,M57563,M61689,M61690,M61691,M61692,M61693,M61694,
        M61695,M61696,M61697,M61698,M61699,M61700,M61701,M61702,M65225,
        M65226,M65227,M65228,M65229,M65230,M65231,M65232,M65233,M65234,
         X03674,X06489,X13387,X14520,X14521,X53815,X55448
Annotation: null
Created: Sep 19 1989     Last Modified: Feb 14 1995

              Allele                                              Max
Locus         Set    Type  Probe/Enzyme                         Het   GDB ID
-------------------------------------------------------------------------------
G6PD          Aa     RS    pKSB / FokI                          0.50  G00-057-823
G6PD          Ba     RS    pKSB / PvuII                         0.38  G00-057-824
G6PD          Ca     DINUC G6PD.PCR1.1/G6PD.PCR1.2              0.39  G00-060-413
G6PD          Da     RS    G6PD.L/G6PD.R / BspHI                0.49  G00-061-247
G6PD          Ea     PT    G6PD.G.7/G6PD.I / SacI/ScaI          0.42  G00-061-249
G6PD          Fa     PT    G6PD.F/G6PD.Md / BclI                0.36  G00-061-257
G6PD         Fb     PT    G6PD.PCR2.1/G6PD.PCR2.2,G6PD.AS+ 0.28  G00-061-703
G6PD          Ga     RS    p2.1 / PstI                          0.34  G00-061-706
G6PD          Bb     RS    pGD-T-5B / PvuII                     0.48  G00-061-707

Primary Author  Citation                                          Year GDB ID
-------------------------------------------------------------------------------
Freije, D       Am J Hum Genet 51:66-80                        ** 1992 G00-038-997
Korn, B         Hum Mol Genet 1:235-242                           1992 G00-039-245
Maestrini, E    Hum Mol Genet 1:275-80                            1992 G00-039-252
Pan, Y          Nature Genet 2:103-6                              1992 G00-037-676
Van den Ouwela+ Hum Mol Genet 1:269-73                            1992 G00-039-153
Van den Ouwela+ Nature Genet 2:99-102                             1992 G00-037-674
Chen, EY        Genomics 10:792-800                            ** 1991 G00-029-162
Schlessinger, D Genomics 11:783-93                                1991 G00-029-236
Abidi, FE       Genomics 7:363-76                                 1990 G00-016-860
Beutler, E      Cell 62:7-9                                       1990 G00-023-178
Djabali, M      Genomics 7:587-93                                 1990 G00-023-255

...
```

Figure 2: A sample GDB entry.

```
COMPND    CRAMBIN                                                        1CRN   4
HEADER    PLANT SEED PROTEIN                      30-APR-81   1CRN       1CRND  1
COMPND    CRAMBIN                                                        1CRN   4
SOURCE    ABYSSINIAN CABBAGE (CRAMBE ABYSSINICA) SEED                    1CRN   5
AUTHOR    W.A.HENDRICKSON,M.M.TEETER                                     1CRN   6
REVDAT   2   03-DEC-81 1CRNA   1       SHEET                            1CRNB  2
REVDAT   1   28-JUL-81 1CRN    0                                        1CRNB  3
REMARK   1                                                             1CRN   7
REMARK   1 REFERENCE 1                                                 1CRNC  2
REMARK   1  AUTH   M.M.TEETER                                          1CRNC  3
REMARK   1  TITL   WATER STRUCTURE OF A HYDROPHOBIC PROTEIN AT ATOMIC   1CRNC  4
REMARK   1  TITL 2 RESOLUTION. PENTAGON RINGS OF WATER MOLECULES IN     1CRNC  5
REMARK   1  TITL 3 CRYSTALS OF CRAMBIN                                 1CRNC  6
REMARK   1  REF    PROC.NAT.ACAD.SCI.USA         V.  81  6014 1984      1CRNC  7
SEQRES   1     46  THR THR CYS CYS PRO SER ILE VAL ALA ARG SER ASN PHE  1CRN  51
SEQRES   2     46  ASN VAL CYS ARG LEU PRO GLY THR PRO GLU ALA ILE CYS  1CRN  52
SEQRES   3     46  ALA THR TYR THR GLY CYS ILE ILE ILE PRO GLY ALA THR  1CRN  53
SEQRES   4     46  CYS PRO GLY ASP TYR ALA ASN                          1CRN  54
HELIX    1  H1 ILE      7  PRO     19  1 3/10 CONFORMATION RES 17,19    1CRN  55
HELIX    2  H2 GLU     23  THR     30  1 DISTORTED 3/10 AT RES 30       1CRN  56
SHEET    1  S1 2 THR     1  CYS     4  0                                1CRNA  4
SHEET    2  S1 2 CYS    32  ILE    35 -1                                1CRN  58
TURN     1  T1 PRO    41  TYR    44                                     1CRN  59
SSBOND   1 CYS      3    CYS     40                                     1CRN  60
SSBOND   2 CYS      4    CYS     32                                     1CRN  61
SSBOND   3 CYS     16    CYS     26                                     1CRN  62
CRYST1   40.960   18.650   22.520  90.00  90.77  90.00 P 21          2  1CRN  63
ORIGX1      1.000000  0.000000  0.000000        0.00000                 1CRN  64
ORIGX2      0.000000  1.000000  0.000000        0.00000                 1CRN  65
ORIGX3      0.000000  0.000000  1.000000        0.00000                 1CRN  66
SCALE1       .024414  0.000000  -.000328        0.00000                 1CRN  67
SCALE2      0.000000   .053619  0.000000        0.00000                 1CRN  68
SCALE3      0.000000  0.000000   .044409        0.00000                 1CRN  69
ATOM      1  N   THR     1      17.047  14.099   3.625  1.00 13.79      1CRN  70
ATOM      2  CA  THR     1      16.967  12.784   4.338  1.00 10.80      1CRN  71
ATOM      3  C   THR     1      15.685  12.755   5.133  1.00  9.19      1CRN  72
ATOM      4  O   THR     1      15.268  13.825   5.594  1.00  9.85      1CRN  73
ATOM      5  CB  THR     1      18.170  12.703   5.337  1.00 13.02      1CRN  74
ATOM      6  OG1 THR     1      19.334  12.829   4.463  1.00 15.06      1CRN  75
ATOM      7  CG2 THR     1      18.150  11.546   6.304  1.00 14.23      1CRN  76
ATOM      8  N   THR     2      15.115  11.555   5.265  1.00  7.81      1CRN  77
ATOM      9  CA  THR     2      13.856  11.469   6.066  1.00  8.31      1CRN  78
ATOM     10  C   THR     2      14.164  10.785   7.379  1.00  5.80      1CRN  79
ATOM     11  O   THR     2      14.993   9.862   7.443  1.00  6.94      1CRN  80
ATOM     12  CB  THR     2      12.732  10.711   5.261  1.00 10.32      1CRN  81
ATOM     13  OG1 THR     2      13.308   9.439   4.926  1.00 12.81      1CRN  82
ATOM     14  CG2 THR     2      12.484  11.442   3.895  1.00 11.90      1CRN  83
ATOM     15  N   CYS     3      13.488  11.241   8.417  1.00  5.24      1CRN  84

....
```

Figure 3: A sample PDB entry. In the fields labelled with SEQRES the residue sequence of the protein is given. The atom coordinates are listed in the fields with label ATOM.

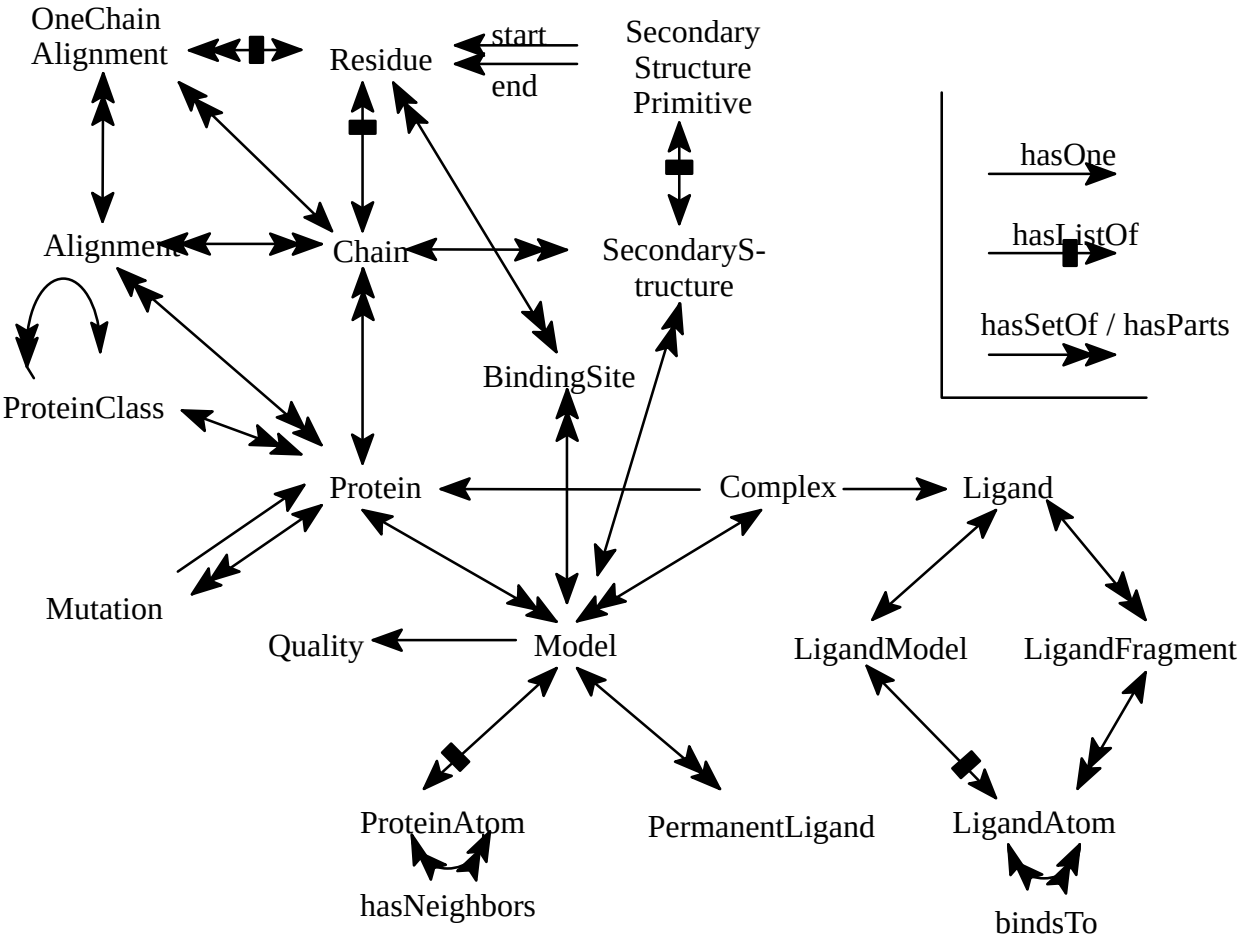

Figure 4: The Docking–D conceptual schema. The schema covers data gathered from different databases on proteins and small molecules. It reflects only the important relationships between classes, and can be considered as a road–map to the database schema. The hasOne arrows indicate relationships between classes, where an object of one class references exactly one object of another class. The hasListOf arrows indicate relationships, where an object of one class references a set of objects of another class, that is ordered. The hasSetOf arrows indicate relationships, where an object of one class references an unordered set of objects of another class. The set–valued relationships can correspond to different semantic aspects. For example, a protein has several models, as in general there is no agreement on the correct model. On the other hand, for a protein many different mutations may be known, such that in this case the set–valued relationship corresponds to a physical fact and not to the lack of knowledge.