BIOINFORMATICS

Vol. 20 no. 1 2004, pages 51–57
DOI: 10.1093/bioinformatics/btg372

Light-weight integration of molecular biological databases

Stephan Philippi

Department of Computer Science, University of Koblenz, P.O. Box 201 106, 56016 Koblenz, Germany

Received on March 11, 2003; revised on June 26, 2003; accepted on July 8, 2003

ABSTRACT

Motivation: Due to the increasing number of molecular biological databases and the exponential growth of their contents, database integration is an important topic of research in bioinformatics. Existing approaches in this area have in common that considerable efforts are needed to provide integrated access to heterogeneous data sources.

Results: This article describes the `LIMBO` architecture as a light-weight approach to molecular biological database integration. By building systems upon this architecture, the efforts needed for database integration can be significantly lowered.

Availability: As an illustration of the principle usefulness of the underlying ideas, a prototypical implementation based upon the `LIMBO` architecture is described. This implementation is exclusively based on freely available open source components like the `PostgreSQL` database management system and the `BioRuby` project. Additional files and modified components are available upon request from the author.

Contact: philippi@uni-koblenz.de

INTRODUCTION

The publicly available molecular biological knowledge is at present scattered over several hundred internet accessible databases (Baxevanis, 2003). Due to the growth in the number of databases and their contents, it is getting more and more unlikely that complex biological questions can be answered by consulting only a single database. Consequently, the integration of heterogeneous molecular biological databases is one of the most important areas of research in bioinformatics (Stevens *et al*., 2001). Database integration in the life sciences faces a variety of problems, which roughly fit into the four categories: technical, political, syntactical and semantical.

From a syntactical point of view, a severe problem is that flatfiles are still the de-facto standard for the exchange of molecular biological data. In order to be able to integrate available data on the basis of flatfiles, their implicit data structures need to be reconstructed. In addition, parsers have to be built to insert flatfile contents into a target database. More technical problems arise if a database is not available as flatfile, but only via dynamically generated HTML pages. In this case, a mediation approach using wrappers is needed (Wiederhold, 1992), which takes even more efforts to realize. Technical and syntactical problems could be easily avoided if database providers would grant read access via standardized database interfaces like JDBC and ODBC. This way, the reconstruction of data structures and the development of wrappers and parsers would be unnecessary. Thus, database integration could be much more efficient. Unfortunately, only few providers allow direct access to their databases via standardized interfaces. Notable exceptions in this context are DDBJ (Miyazaki *et al*., 2003), EMBL (Wang *et al*., 2002) and NCBI (Wheeler *et al*., 2000), to which access is provided by means of web services, SOAP and XML (World Wide Web Consortium, 2003, www.w3c.org). As current database management systems (DBMS) and their interfaces support many security features, access restrictions as described above actually have political reasons that are beyond the scope of this article. In order to be able to store and/or to provide access to integrated data sources via a DBMS, a target data structure needs to be developed. In addition to the already described problems arising from this task, there are also semantical difficulties, as the same data is often described by different terms amongst databases (e.g. Davidson *et al*., 1995 and Schulze-Kremer, 2002). Consequently, semantically existing relations between entries in different databases are not always easy to identify.

According to Karp (1995), approaches for the integration of biological databases are based on hypertext navigation, data warehouses, multi-database query languages and federated database technology. Due to the number of existing databases, each is usually linked only to relatively few others (Williams, 1997). The usefulness of this approach for purposes other than the support of interactive data browsing is therefore limited. To allow for 'bulk' queries, more sophisticated means for data integration are needed (Davidson *et al*., 1995). Data warehouse-based approaches for data integration locally mirror data sources to provide integrated access. The main advantage of a data warehouse is that queries can be answered efficiently in comparison to non-mirroring approaches. However, in addition to the obvious need for sufficient resources, a drawback of data warehouse-based approaches is that mirrored data sets have to be actively kept

up to date in order to account for changes in the original data sources. If a data warehouse is based on a DBMS, considerable efforts are needed to update the internal schema for data storage, as the structures of source databases are changing over time (Critchlow *et al*., 2000). A prominent example for a data warehouse approach not based on a DBMS is SRS (Etzold *et al*., 1996), which uses full text indexing methods on flatfiles for efficient querying. Approaches that do not suffer from the problem of unrecognized changes in the original data sources are multi-database query languages and federated databases. Multi-database query languages like CPL (Buneman *et al*., 1995) and OPM (Chen and Markowitz, 1995) allow one to create queries explicitly targeted towards different databases. Federated database approaches take this idea further in that an integrated schema is defined against which queries are posed (e.g. Haas *et al*., 2001). This way, the user does not need to know where the data is stored. Making use of a wrapper/mediator architecture (Wiederhold, 1992), source databases transparently deliver up to date results in such a setting. A problem with approaches in this area is that they are not efficient if data needs to be parsed from HTML pages and/or if the user poses a query which, due to restricted interfaces, cannot be directly passed to the source databases.

The approaches described up to now do not account for the above described semantical problems of data integration. In order to address issues of semantical heterogeneity, additional efforts based on ontologies are needed. Ontologies describe concepts of a particular application domain together with relations between them, like 'is-a' and 'part-of' (Noy and McGuinness, 2001). If attributes and/or entries of databases to be integrated are linked to concepts of an ontology, semantical heterogeneities can be overcome by answering queries independently of the terms used in source databases for specific concepts [e.g. TAMBIS (Stevens *et al*., 2000) and SEMEDA (Köhler *et al*., 2003)].

The different approaches to database integration described above have one thing in common, the integration task itself is very demanding and time-consuming. Manual structure extraction from flatfiles, development of flatfile parsers and HTML wrappers as well as the definition and maintenance of database schemata needed to store and/or access databases are common tasks, which have to be carried out for each data source to provide integrated access. Taking the vast number of databases and the fact that they are dynamically changing their structure over time into account, large scale database integration takes a huge amount of effort.

Starting from this perspective, the remainder of this article describes the LIMBO architecture as a more light-weight approach to molecular biological database integration. The next section introduces the principles of this architecture followed by the illustration of its practical use with a prototypical implementation. Finally, the introduced proposal is discussed and perspectives on future work are given.

THE LIMBO ARCHITECTURE

In order to make the underlying guidelines for the LIMBO development as a light-weight approach to database integration explicit, some basic requirements are briefly discussed in the following (see also Karp, 1995). From our point of view, an approach to database integration should be:

- generally applicable, i.e. arbitrary data sources should be covered;
- relatively easy to use, i.e. integrating data sources should be as easy as possible;
- able to answer queries with reasonably up to date data;
- flexible, i.e. changes of schema or format of data sources should be easy to cope with;
- powerful with respect to querying, i.e. a standardized query language like SQL or OQL should be supported and
- efficient, i.e. queries should be answered without lengthy delays.

Due to the distributed nature of data sources accessible via the internet, requirements like efficiency and query answering with current data cannot be equally fulfilled. However, the above requirements usually have different priorities leading to different solutions for specific application scenarios. As query efficiency is more important than query answering with up to date data in our application context, the approach described here is based on the data warehouse idea. A high level view of the LIMBO architecture is given in Figure 1.

The main component of the architecture is the central data warehouse used to store the contents of the databases to provide integrated access. From a technical point of view this warehouse can be built with any standard relational or object-oriented DBMS. The data inserted into this central data store can have their origin in different kinds of sources, like flatfiles or SQL/OQL accessible databases. Once the data are available in the central data warehouse, uniform access via a standardized query language is provided. The contents of the central data warehouse can then be made available for interactive querying using e.g. a standard three tier approach consisting of the central data store as back-end, a web server capable to dynamically create HTML pages as the middle tier and web browsers as light-weight clients. In addition to data access via web browsers, the query language of the underlying DBMS can be used directly to access data on different integration levels. Further benefits of this architecture are that a system can be scaled independently on different tiers and also a framework is provided for the extension with e.g. additional data analysis features. Another integration scenario is the set up of an application specific database centered around a particular problem (also referred to as *data mart*). In this case, data often needs to be integrated more tightly, i.e. not

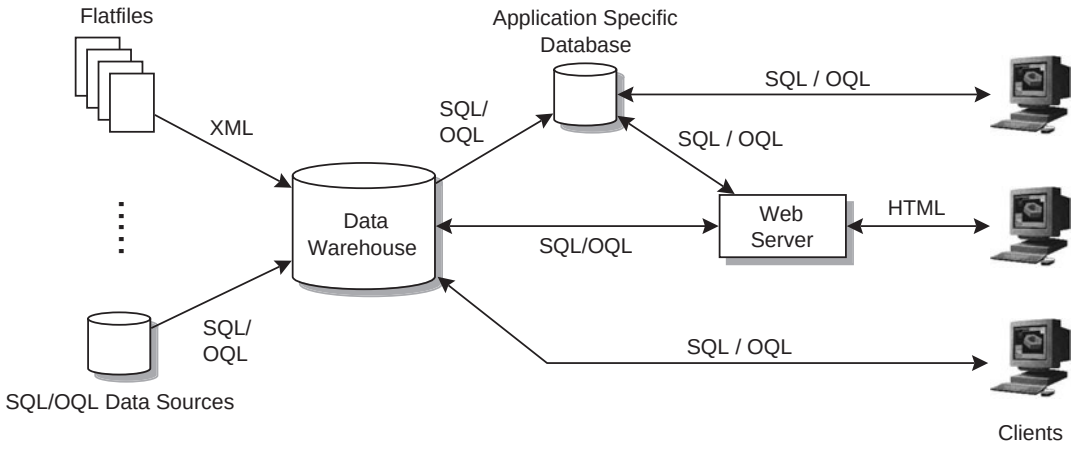

Fig. 1. Overview of the LIMBO Architecture.

only a uniform access to different data sources is needed, but actually the merging of data items from different sources as well as the integration with newly created data. Due to the availability of a standardized query language for data access in the underlying DBMS of the LIMBO architecture, the tight integration of molecular biological databases is supported too.

After this architectural overview, data conversion, data integration and data export as the main tasks of a system built upon this architecture are described in more detail.

Data conversion

As stated above, flatfiles are still the de-facto standard for the exchange of molecular biological data. Most of the publicly available databases, therefore, offer in one way or the other the opportunity to download either the whole database or the results of a query as flatfile. As flatfiles cannot be imported directly into a database to provide structured access to data items, they first have to be converted into a format that is better suited for this purpose. The principle idea for data conversion in the LIMBO architecture is to transform flatfiles into an XML format (World Wide Web Consortium, 2003) for further processing. First of all, a document type definition (DTD) is needed, which reflects the principle structure of flatfiles from a specific data source in a general way. In detail, flatfiles are described by a DTD with the hierarchical ordering of data items, their nesting structure as well as optional and mandatory properties. Due to the regular structure of flatfiles and the stored data items, the development of a DTD for flatfiles from a specific data source is mostly straight forward. If a parser as a basis for the XML conversion of flatfiles from a specific data source is available, the DTD can be used to validate the document by means of widely available XML parsers. This feature is especially useful in the context of an automated strategy for updating the central data warehouse, as changes in the structure of the data source can be automatically detected this way. If a converted flatfile is successfully validated, the XML parser is further used as a basis for the insertion of data items from the XML document into the data warehouse. The structure needed in a database to insert this data into is described next.

Data integration

Traditionally, DBMS-based approaches for database integration demand the development of a target database schema to store and/or access the data, integrated access is to be provided too. Due to the complexity of molecular biological data, this is a demanding task, especially if many different data sources have to be integrated. A solution to this problem could be a generally agreed upon data structure, which is capable of storing all kinds of biological data. As Markowitz and Ritter (1995) and others pointed out, such a structure is unlikely to be developed. Consequently, a target schema for data integration is not fixed once developed. Instead, changes to the structure of data sources have to be accounted for during the lifetime of a system providing integrated data access. As the development and maintenance of a target integration structure requires considerable effort (Davidson *et al*., 1995), the question arises as to what are the actual benefits of such a structure? Obviously, a database schema is needed to host the data, integrated access is to be provided to. Still the question remains as to what use complex database structures have for molecular biological database integration from the point of view of the biologist using such a system? In fact, the usefulness of such structures is limited for two reasons. First of all, biologists do not want and actually should not have to learn complex data structures consisting of several hundreds of tables and their relationships to be able to use a system providing access to integrated data sources. But even if biologists have detailed knowledge of such a structure, the usefulness of this knowledge is limited, as it is not always clear in which table/attribute specific information is stored. According to van Helden *et al*. (2000), the same kind of biological information

is not always available in a single attribute within existing databases, but in different ones. Out of this, the classical way of using database structures is not suitable for molecular biological database integration. In fact, a query would be needed which searches all attributes in all tables to be able to retrieve all relevant data items on the basis of given keywords. As standardized languages like SQL and OQL are focused on the structure of databases, such a query cannot be posed in a general way. Instead a complex query has to be built, which explicitly contains all tables and attributes to be searched. Not only is such a query subject to modifications if the underlying database structure changes, also it can not be easily optimized for efficient querying answering. Systems like SRS (Etzold *et al*., 1996), DBGET/LinkDB (Fujibuchi *et al*., 1998) and SIR (Ramu, 2001) account for this situation in that a full text index is used to provide integrated access to the underlying data sources. The advantage of this approach is that there is no need for detailed knowledge of data structures. However, as these systems are not making use of database technology and do not provide standardized access interfaces like SQL or OQL to integrated data, an alternative solution is described in the rest of the section.

Due to the efforts needed to develop fine grained target structures for molecular biological database integration on the one hand and their only limited use on the other, a different approach to this problem is favored in the LIMBO architecture. In detail, a generalized database structure is introduced in the following, which consists of very few tables, but is capable of storing arbitrary data items without loss of information. In fact, only two tables are needed in the basic set up—their structures are given in Figure 2. Whereas the DATA table stores tuples containing name and value of an attribute of a source data item, the RELATIONS table is used to store relations between tuples of the first one. If, for instance, data items of a flatfile from a specific data source store information about proteins and contain an entry ‘Sequence Data’ which itself consist of sub-entries ‘Length’ and ‘Sequence’, the XML converted flatfile reflects this hierarchical structure. During data insertion this hierarchical ordering is used to store the given information in the generalized data structure as illustrated with an entry from an imaginary protein database in Figure 2. The DATA table contains four tuples, each consisting of a primary key identifier and attributes, which store name and value of example data entries. The first tuple represents the top level of the hierarchical structure of the data item. If an entry in the XML converted flatfile does not have a value, but serves as a nesting construct for hierarchical data ordering, the ‘value’ attribute is set to ‘NULL’. The example protein data item hierarchically contains ‘Sequence Data’, which itself contains ‘Length’ and ‘Sequence’ information. The hierarchical relations between these tuples are given in the RELATIONS table. This table contains three entries, each consisting of two foreign keys which reference source and target identifiers of related tuples in the DATA table. An additional table as given

ID	Attribute	Value
1	Protein	NULL
2	Sequence Data	NULL
3	Length	’474’
4	Sequence	’MYQTL...

a) DATA

From	To
1	2
2	3
2	4

b) RELATIONS

Fig. 2. Basic database structure.

ID	Data Source	Release
1	Swiss-Prot	40.43
2	TrEMBL	22.12
3	TransFac Public	5.0

c) METADATA

Fig. 3. Extension of basic database structure.

in Figure 3 may be added to the schema in order to store meta data about the integrated data sources. In this case, the ‘value’ attribute of the top level tuple of a data item refers to an entry in the METADATA table. This way, the origin of each data item can be made transparent to the user, if needed.

The setting described up to now supports full text queries over integrated data sources. If more sophisticated queries are to be answered, the DATA table needs to be extended with a single attribute for each additional data type to cover. In case, e.g. proteins with a certain length have to be filtered, an attribute ‘int_value’ of type INTEGER extends the DATA table. This attribute not only stores protein length information, but in fact all kinds of integer data from the integrated data sources.

The advantages of this generalized structure for data storage are manyfold. First of all, no structure needs to be developed to provide integrated access to heterogeneous data sources. As this is usually one of the most demanding and time-consuming tasks, the efforts needed for database integration are considerably lowered. Furthermore, changes of the structure of data sources do not affect the target data structure—the only difference in this case is that newly inserted data items are related in a different way within the same generalized structure. Consequently, no administrative efforts to maintain the target database structure are needed. However, the DTD describing the flatfile structure as well as the XML converter are affected and have to be modified accordingly to reflect the changes in the structure of the data source. From a technical point of view, querying the generalized structure for search terms is straightforward, as only a single table has to be searched. Due to the use of only few tables for data storage, only few indices have

to be set up to be able to answer queries efficiently. Since many of the data entries in molecular biological databases consist of free text, like descriptions and comments, efficient querying also has to be based on full text indexing features, which are available for almost every DBMS. The advantages from a user's point of view are that there is no need to know about (potentially changing) complex data structures and also different kinds of interfaces are available to access the integrated data sources.

Data export

Once the data are accessible in an integrated way as described above, different kinds of export interfaces are needed. Due to the use of a DBMS for data storage, a standardized query language like SQL or OQL provides uniform access to the integrated data sources. A data item from the original flatfile can be reconstructed by retrieving all elements (transitively) related to a specific root tuple. For efficiency reasons, an additional 'root' attribute, which stores the root element of each attribute/value pair in the original flatfile, augments the described structure of the `DATA` table. This way, hierarchically nested data structures are not leading to nested queries for tree processing. Instead, all sub-data items of a flatfile entry can be retrieved with a less complex and more efficient query by means of this additional attribute.

In order to set up an application specific database, rules for the merging of data items from different sources have to be defined. If these rules are available, the set up of a data mart is straightforward using SQL or OQL. The target database structure in this context is not bound to be the same as the one used within the central data warehouse. As application specific databases are often used to integrate in-house research data with publicly available knowledge, an extended generalized or even a completely unrelated structure may be used.

In contrast to the machine level interface to integrated data sources, an interactive query interface on top of it is needed to provide the opportunity for biologists to pose queries without having to know SQL or OQL. Such an interactive query interface ideally provides search capabilities like a general purpose web search engine to retrieve relevant entries from integrated data sources. Due to the hierarchical structure of flatfile data entries and their storage in a generalized structure in the data warehouse, also a generalized front end for the display of these data items can be easily built with a template for the visualization of hierarchically ordered attribute/value pairs.

PROTOTYPICAL IMPLEMENTATION

As a proof of concept of the underlying ideas of the introduced architecture for light-weight integration of molecular biological databases, this section describes a prototypical implementation based on freely available open source components. In detail, the `PostgreSQL` (PostgreSQL Project, 2003, www.postgresql.com) object-relational DBMS hosts the central data warehouse and an application specific database. A freely available web server is used to dynamically generate HTML pages. Due to the fact that data entries in molecular biological databases are often free textual descriptions, full text indexing methods have to be applied. For the `PostgreSQL` DBMS several full text indexing add-ons are available. In our prototypical implementation we opted for the `FullTextIndex` module, which is part of the `PostgreSQL` distribution. If a data item is inserted into the data warehouse, this module scans the contents of the 'value' attribute and inserts the tokens found into a specific index table. In order to prevent the indexing of tokens that do not carry a meaning by themselves, like articles, a list of tokens not to be indexed is included into the indexing module. As we also had to prevent indexing of sequences and other information not suited for keyword-based querying, the delivered full text index module was modified to fit our needs.

The example integration scenario described below is the set up of an application specific database storing data about mammalian transcription factors. Relevant source databases chosen for this purpose are `Swiss-Prot`/`TrEBML` (Bairoch and Apweiler, 2000) and the public version of `TransFac` (Wingender *et al*., 2000). These databases are freely available as flatfiles and thus converted into an XML format in the first processing step. The converters built for these purposes are partially based on the `BioRuby` open source project (BioRuby, 2003, www.bioruby.org). As a result of this conversion, flatfiles are available in a more structured format. Figure 4 gives an impression of the outcome from this processing step for parts of `Swiss-Prot`, specifically the `GATA4` transcription factor.

In the next step an XML parser is used to validate the XML representations of the flatfiles and to insert attribute/value pairs into the generalized data warehouse structure as described in the last section. With the central data warehouse set up this way, uniform access to the source databases can already be provided (Fig. 1). As an application specific database containing only mammalian transcription factors is needed in the example scenario, a data mart has to be created. For our in-house research project `TransFac` entries are slightly better suited than entries from `Swiss-Prot` and `TrEMBL`. Out of this, the project data mart was first populated with all the mammalian transcription factors available in the 'factor' table of `TransFac`. In the second step, data entries from `Swiss-Prot` and `TrEMBL` not available in `TransFac` completed the set up. Even if this construction of the data mart only needed basic filtering rules and no merging of data entries from different sources, the availability of an SQL interface to the data warehouse in principle supports arbitrarily complex filtering and merging scenarios. The prototypical implementation of the `LIMBO` architecture was finally completed with an HTML-based interface, which allows access to the data mart in a user-friendly manner.

```
<Protein>
 <Protein_Name>GAT4</Protein_Name>
 <Species>HUMAN</Species>
 <Data_Class>STANDARD</Data_Class>
 <Molecule_Type>PRT</Molecule_Type>
 <Sequence_Length>442 AA</Sequence_Length>
 <Accession_Number>
  <Primary>P43694</Primary>
 </Accession_Number>
 <Dates>
  <Created>
   <Date>01-NOV-1995</Date>
   <Comment>Rel. 32, Created</Comment>
  </Created>
  <Sequence_Update>
   <Date>01-NOV-1995</Date>
   <Comment>Rel. 32, Last sequence update
   </Comment>
 </Sequence_Update>
 <Annotation_Update>
  <Date>16-OCT-2001</Date>
  <Comment>Rel. 40, Last annotation update
  </Comment>
 </Annotation_Update>
</Dates>
<Description> .....
```

Fig. 4. Part of a XML flatfile representation.

Due to the simplicity of the introduced architecture, the prototype consisting of data warehouse and application specific database as described above was realized in a relative short period of time. Owing to the use of a generalized schema for data storage in the data warehouse and the data mart, explicit development of an application specific integrated database schema was not necessary. As this strategy saved a lot of conceptual work, most of the efforts left to realize the prototype had to be spent to develop XML converters for the source flatfiles. In the light of the current trend to provide XML-based access to molecular biological databases [e.g. DDBJ (Miyazaki *et al*., 2003) and EMBL (Wang *et al*., 2002)], the development of flatfile converters might not be necessary any longer in the not too far future. In consequence, there are realistic prospects to further lower the efforts needed for data integration based on the introduced architecture.

DISCUSSION AND FUTURE WORK

In this article, an architecture for the light-weight integration of molecular biological databases was presented. In comparison to the existing work in this area, the efforts needed to integrate heterogeneous data sources are lowered considerably. Due to the use of a standard DBMS and a generalized target database structure, most of the requirements for database integration introduced above are fulfilled by implementations of the LIMBO architecture. Especially problems with constantly evolving data warehouse schemata due to dynamically changing structures of source databases can be overcome this way. As a proof of concept, a prototypical implementation of the architecture has been developed that is exclusively based on freely available open source components. In an example scenario the data warehouse was set up with Swiss-Prot, TrEMBL and parts of the public version of TransFac. Based on these data sources an application specific database was created, which stores mammalian transcription factors to support an in-house research project.

Even if the experiences in setting up the described environment are encouraging, questions about technical limitations naturally arise. This is especially true, as the chosen structure for data storage in the LIMBO architecture leads to very many tuples in the tables of a system. Actually, the DATA and RELATIONS tables of the central data warehouse in the above described setting with three integrated data sources contain several million tuples. Even if these are huge numbers, they are not a problem from a technical point of view, because current DBMSs are only limited in the amount of tuples per table by the resources of the underlying hardware. DBMS features to run and access databases on distributed clusters transparently and access databases on distributed clusters as well as to split large tables into physical partitions indicate that the presented approach can also be used for more sizeable integration scenarios than the one described above.

The work presented in this article is an ongoing project, which will be extended in future versions with an ontology layer to also allow for the semantical integration of molecular biological databases. In addition, an automated update strategy for the central data warehouse will be developed as well as a tool for semi-automated structure extraction from flatfiles.

ACKNOWLEDGEMENTS

The author would like to thank the head of the scientific staff council of the Bielefeld University Gerlinde Günther-Boemke for showing others their limits. Further acknowledgements go to Jacob Köhler, Dion Whitehead and the peer reviewers for commenting an earlier version of this article.

REFERENCES

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

Baxevanis,A. (2003). The molecular biology database collection: (2003) update. *Nucleic Acids Res.*, **31**, 1–12.

BioRuby. (2003) Project Documentation.

Buneman,P., Davidson,S.B., Hart,K., Overton,C. and Wong,L., (1995) (A data transformation system for biological data sources).

Proceedings of the International Conference on Very Large Databases (VLDB), Zürich, Switzerland, 158–159.

Chen,I.A. and Markowitz,V.M. (1995) An overview of the Object-Protocol Model (OPM) and OPM data management tools. *Inform. Sys.*, **20**, 393–418.

Critchlow,T., Fidelis,K. Ganesh,M., Musick,R. and Slezak,T. (2000) DataFoundry: information management for scientific data. *IEEE Trans. Inform. Tech. Biomed.*, **4**, 52–57.

Davidson,S., Overton,C. and Buneman,P. (1995) Challenges in integrating biological data sources. *J. Comput. Bio.*, **2**, 557–572.

Etzold,T., Ulyanow,A. and Argos,P. (1996) SRS: information retrieval system for molecular biology data banks. *Methods Enzymol.*, **226**, 114–128.

Fujibuchi,W., Goto,S., Migimatsu,H., Uchiyama,I., Ogiwara,A., Akiyama,Y. and Kanehisa,M. (1998) DBGET/LinkDB: an integrated database retrieval system. *Proceedings of the Pacific Symposium on Biocomputing*, 681–692.

Haas,L.M.,. Schwarz,P.M., Kodali,P., Kotlar,E., Rice,J.E. and Swope,W.C. (2001) DiscoveryLink: a system for integrated access to life sciences data sources. *IBM Sys. J.*, **40**, 489–511.

Karp,P. (1995) A strategy for database interoperation. *J. Comput. Biol.*, **2**, 573–586.

Köhler,J., Philippi,S. and Lange,M. (2003) SEMEDA—ontology based integration of biological databases. *Bioinformatics* **19**, *in press*.

Markowitz,V.M. and Ritter,O. (1995) Characterizing heterogeneous molecular biological database systems. *J. Comput. Biol.*, **2**, 547–556.

Miyazaki,S., Sugawara,H., Gojobori,T. and Tateno,Y. (2003) DNA Data Bank of Japan (DDBJ) in XML. *Nucleic Acids Res.*, **31**, 13–16.

Noy,N. and McGuinness,D.L. (2001) Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Medical Informatics Report SMI-2001-0880*, Stanford University.

PostgreSQL Project (2003) Documentation.

Ramu,C. (2001) SIR: a simple indexing and retrieval system for biological flat file databases. *Bioinformatics*, **17**, 756–758.

Schulze-Kremer,S. (2002) Ontologies for molecular biology and bioinformatics'. *In Silico Biol.*, **2**.

Stevens,R., Baker,P., Bechhofer,S., Ng,G., Jacoby,A. Paton,N.W., Goble,C.A. and Brass,A. (2000) TAMBIS: transparent access to multiple bioinformatics information sources. *Bioinformatics*, **16**, 184–186.

Stevens,R., Goble,C.A., Baker,P. and Brass,A. (2001) A classification of tasks in bioinformatics. *Bioinformatics*, **17**, 180–188.

van Helden,J., Naim,A., Mancuso,R., Eldridge,M. Wernisch,L., Gilbert,D. and Wodak,S.J. (2000) Representing and analysing molecular and cellular function using the computer. *Biol. Chem.*, **381**, 921–935.

Wang,L., Riethoven,J.-J. and Robinson,A. (2002) XEMBL: distributing EMBL data in XML format. *Bioinformatics*, **18**, 1147–1148.

Wheeler,D.L., Chappey,C., Lash,A.E., Leipe,D.D., Madden,T.L., Schuler,G.D. Tatusova,T.A. and Rapp.B.A. (2000) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.*, **28**, 10–14.

Wiederhold,G. (1992) Mediators in the architecture of future information systems. *IEEE Comp.*, **25**, 38–49.

Williams,N. (1997) Bioinformatics: how to get databases talking the same language. *Science*, **275**, 301–302.

Wingender,E., Chen,X., Hehl,R., Karas,H., Liebich,I., Matys,V., Meinhardt,T., Prüß,M., Reuter,I. and Schacherer,F. (2000) TRANSFAC: an integrated system for gene expression regulation. *Nucleic Acids Res.*, **28**, 316–319.

World Wide Web Consortium (2003) Standard Documentations.