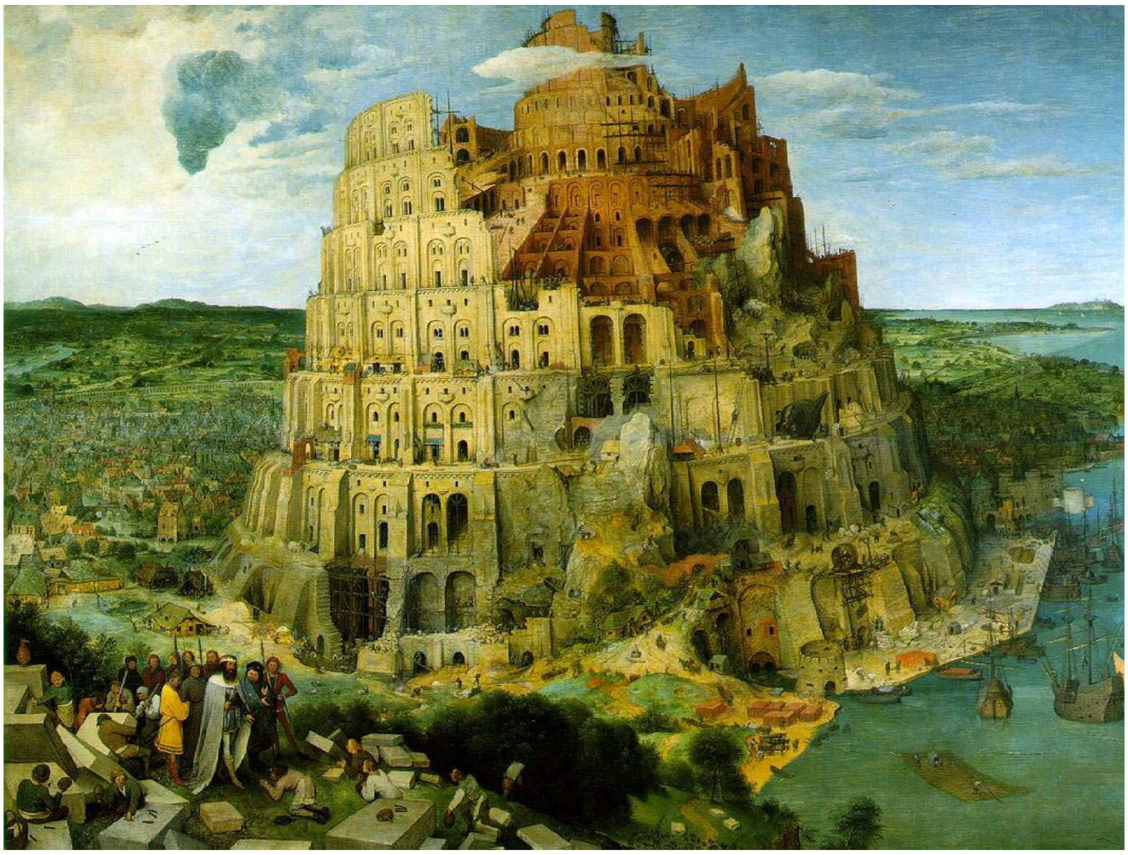
SEMEDA (Semantic Meta-Database): Ontology Based Semantic Integration of Biological Databases



Dissertation zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften

Eingereicht an der Technischen Fakultät der Universität Bielefeld von

Jacob Köhler

Dipl. Biol. Jacob Köhler

"SEMEDA (Semantic Meta-Database): Ontology Based Semantic Integration of Biological Databases"

PhD Thesis

Bielefeld 2003

Technischen Fakultät der Universität Bielefeld

AG Bioinformatik, Prof. Dr. Ralf Hofestädt

Gedruckt auf alterungsbeständigem Papier °° ISO 9706

Title page:

The Tower of Babel, 1563

Pieter Bruegel (about 1525-69), usually known as Pieter Bruegel the Elder

Oil on oak panel, 114 x 155 cm

Kunsthistorisches Museum Wien, Vienna

http://www.khm.at/system2E.html?/staticE/page437.html

The Book of Genesis tells of a great tower built by men not only from fear of a second Flood but above all "to make a name for themselves". Gods punishment was the Babylonian confusion of tongues, with men unable to understand each other, the result being that the tower was never finished.

Mein Dank

gilt Prof. Ralf Hofestädt für die Betreuung der Arbeit, die Freiheit die er dabei gewährte und dafür, dass er es mir ermöglichte die Arbeit an der Universität Bielefeld fertig zu stellen. Dr. habil. Steffen Schulze-Kremer (RZPD Berlin) führte mich in die Thematik der semantischen Datenbank Integration ein und unterstützte die Arbeit während der gesamten Zeit durch Diskussionen und Vorschläge bei der Umsetzung. Prof. Ipke Wachsmuth danke ich für das Interesse welches er der Thematik entgegenbrachte sowie für stimulierende Diskussionen. Dr. Christian Bauckhage danke ich dafür, dass er sich so kurzfristig bereiterklärte mit Interesse in der Prüfungskommission mitzuwirken.

Mein besonderer Dank gilt Matthias Lange in dem ich neben einem zuverlässigen stets hilfsbereiten Kollegen auch einen guten Freund gefunden habe, der mir von Anfang bis Ende der Arbeit mit Rat und Sachverstand zur Seite stand. Obwohl ich Dr. Stephan Philippi erst ein Jahr vor Fertigstellung der Dissertation kennerlernte, trug er durch Sachverstand und konstruktiver Kritik in der entscheidenden Phase der Arbeit sehr zu ihrem Gelingen bei. Meinem Kollegen Dion Whitehead danke ich für die sprachlichen und inhaltlichen Korrekturen und die alltägliche Englisch Nachhilfe.

Obwohl ich Werner Ceusters (Language & Computing, Belgien) nur einen Abend lang kennen gelernt habe, lernte ich viel von ihm, unter anderem die ontologisch-multilinguale Definition von Spargel. Prof. Martin Wahl (IfM Kiel) wird überrascht sein, dass ich ihm hier danke, doch half mir vieles was ich durch seine ausgezeichneten Betreuung meiner Diplomarbeit gelernt habe auch bei meiner Dissertation. Dr. Thomas Steger-Hartmann (Schering AG, Berlin) danke ich für sein Interesse und dafür, dass er mich in einigen wichtigen Fragen beriet.

Meinen Kollegen der Technischen Fakultät Bielefeld, insbesondere der AG Bioinformatik, danke ich für anregende Diskussionen und konstruktive Kritik. Den mittlerweile am IPK Gatersleben arbeitenden Kollegen danke ich für Rat, Support und die Tatsache, dass ich SEMEDA auf den dort befindlichen Systemen entwickeln konnte. Den Mitarbeitern des RZPDs Berlin danke ich für die Unterstützung bei der Anbindung ihrer Datenbank an SEMEDA. Als assoziiertes Mitglied des Graduierten Kollegs “Dynamics and Evolution of Cellular and Macromolecular Processes“ der Humboldt Universität Berlin hatte ich die Möglichkeit an Seminaren und Veranstaltungen des GKs teilzunehmen, und dadurch meinen fachlichen Horizont auch jenseits der Datenbank Integration zu erweitern.

Vielen unerwähnten Freunden, Kollegen und Studenten danke ich für die schöne Zeit und für das was ich im Laufe der Dissertation in Seminarräumen sowie bei Kaffee, Cocktails und bei vielen anderen Gelegenheiten gelernt habe.

Meinen Verwandten danke ich dafür, dass sie stets für mich da waren. Meiner Frau Regina danke ich für Geduld, ihre beständige alltägliche Unterstützung sowie für den Rat bei der Gestaltung der Nutzer Schnittstelle. Meinen Söhnen Milan und Louis danke ich dafür, dass es ihnen immer wieder aufs neue gelingt die Wichtigkeit der hier vorliegenden Arbeit zu relativieren.

Table of Content

Abbreviations

BDS	BioDataServer. The MARGBench component which provides SQL access to heterogeneous data sources
DAG	Directed Acyclic Graph
DBMS	Database Management System
GUI	Graphical User Interface
JDBC	Java Database Connectivity
JSP	Java Server Pages
OODBMS	Object Oriented Database Management Systems
RDBMS	Relational Database Management System
RDF	Resource Description Framework. The RDF specifications provide a lightweight ontology system to support the exchange of knowledge on the Web.
SEMEDA	Semantic Meta Database
SQL	Structured Query Language

1 Introduction

And the Lord said, 'Behold, they are one people, and they all have the same language. And this is what they began to do, and now nothing which they purpose to do will be impossible for them'

Genesis 11:6

Searching and integrating data from various sources is often a prerequisite for research in the field of Molecular Biology and Bioinformatics. Due to the obvious importance of data integration for the life science community (Stevens et al. 2001), a plethora of approaches for the integration of molecular biological databases exists (Davidson et al. 1995, Karp 1995, Jakobovits 1997, Freier et al. 2002b). Due to the fact that most 'databases' were started as flatfiles, the most common approach to database integration is based on indexed flatfiles, for example DBGET/LinkDB (Fujibuchi et al. 1998), SRS (Etzold et al. 1996) and SIR (Ramu 2001). Nowadays, most molecular biological databases are implemented on relational database management systems (RDBMS) that provide standard interfaces like JDBC and ODBC for data and metadata exchange. By using these interfaces, many technical problems of database integration can be overcome and semantic issues remain as described for example in (Kim and Seo 1991, Karp 1995, Williams 1997). As these problems still challenge current approaches to database integration, they are briefly summarized in the following:

1) The fact that different databases often use different words for the same things results in problems that can be overcome by using either controlled vocabularies or ontologies like the Gene Ontology (Ashburner et al. 2000, Gene-Ontology-Consortium 2001), EC numbers (International-Union-of-Biochemistry 1992), CAS Registry (Buntrock 2001) etc. However, a related problem in this area is that there are no tools available, which enable database owners to collaboratively edit and maintain controlled vocabularies or ontologies. Also, there is no systematic method to define which database uses which controlled vocabulary. Therefore, often 'uncontrolled vocabularies' and different controlled vocabularies are used across databases.

2) Attribute names are often not self-explanatory or misleading and equivalent attributes have different names in different databases. Whereas one database might for example use the attribute name 'ec_nr', another database might use 'id' for an attribute,

which also contains EC numbers. Out of this, attributes cannot be easily mapped between different databases.

3) Querying databases often requires knowledge about the content of its tables, e.g. if a table only contains data about one species or one enzyme group. About which mouse species does the mouse genome database 'http://www.informatics.jax.org/' contain data? The database schema does not contain an attribute 'organism'. Unless the user is a biologist who knows that mouse experiments are generally done with special strains of Mus Musculus, it is impossible to find species information via the database query forms on the web page. This is no problem, as long as the interface of the source database is used. As systems for database integration have their own query interface, tables as described above have to be semantically refined with additional information.

4) Due to the lack of a systematic linking mechanism between databases, even up to date integration systems such as SRS (Etzold et al. 1996) and KEGG (Kanehisa 1997a, b, Kanehisa et al. 2002) only link the 'most important' attributes. This is due to the fact that the number of existing molecular biological databases is too high to survey. Therefore, compared to the fact that at present more than 400 molecular biological databases exist (Discala et al. 2000, Baxevanis 2002), the degree of interlinking is low (Williams 1997).

These issues related to semantics become more significant when more than a few databases have to be integrated. For example the BioDataServer, a mediated database integration system that was developed within the MARGBench project (Freier et al. 1999, Freier et al. 2002a, Freier et al. 2002b), can at present be used to query about 15 different databases. Each database has between 2 and 250 database tables. Each database table has between 2 and 15 attributes. Thus the simple task to find all relevant database attributes that might be searched for the enzyme "amylase" becomes a challenge, because it requires that the user has in depth knowledge about all databases. This problem is even more eminent in existing database integration systems such as SRS. Typical SRS (Etzold et al. 1996, Etzold and Verde 1997, Zdobnov et al. 2002) installations integrate more than 100 databases, although SRS does not solve many of the issues related to semantic database heterogeneity.

Starting from this perspective, the aim of this thesis is the development of concepts suitable to solve most of the aforementioned problems of semantic database heterogeneity. To demonstrate the practical use of the presented ideas, the SEMEDA system is developed.

The work presented in this thesis is outlined in the following,. The state of the art in the relevant disciplines is introduced and reviewed in chapter 2. This includes on the one hand the current state of molecular biological databases, their heterogeneity and the integration of molecular biological databases. On the other hand the current usage of ontologies in general and with special regard to database integration is described.

The principles of semantic database integration as introduced in this thesis are new and suitable to be used also in other database integration systems, which have to deal with a high number of semantically heterogeneous databases. Therefore in Chapter 3 the newly introduced principles for ontology based semantic database integration are presented independent of their implementation.

Chapter 4 introduces the requirements for the implementation of a semantic database integration system (SEMEDA). Several general requirements for the integration of molecular biological systems from the scientific literature are discussed with regard to the feasibility of their implementation in general and in SEMEDA. In addition, the requirements specific to semantic database integration are introduced. In addition how the BioDataServer is used to overcome "technical" heterogeneity, so that SEMEDA only has to deal with semantic heterogeneity is analysed.

In chapter 5, an appropriate data structure for storing ontologies, database metadata and the semantic definitions as described in Chapter 3 is developed. Subsequently, it is discussed how this data structure can be edited and queried.

In Chapter 6, SEMEDAs software design, implementation and system architecture is given.

Chapter 7 describes the use of SEMEDA and its interfaces. The user interface SEMEDA-edit is used to collaboratively edit ontologies and to semantically define databases using ontologies. SEMEDA-query is the query interface that provides uniform access to heterogeneous databases. In addition, a set of procedures exists which can be used by external applications.

In order to use SEMEDA to semantically define databases, an appropriate ontology is needed. Although SEMEDA allows building ontologies from the scratch, due to the fact that generating ontologies is a labour intensive time-consuming task, it would be preferable to use an existing ontology. Therefore, in chapter 8 several ontologies were evaluated for their usability in SEMEDA. The intention was to find out if a suitable ontology can be found and imported or whether it is more appropriate to build a custom ontology for SEMEDA.

It turned out that the existing ontologies were not well suited for semantic database integration. In chapter 9 general and SEMEDA specific ontology design principles are introduced which were then followed to build a custom ontology for database integration. The structure of this custom ontology and some issues concerning its use for semantic database integration are explained.

In chapter 10, the practical use of SEMEDA is described by two examples. The first section of this chapter shows how SEMEDA supports the building of user schemata for the BioDataServer. The second section describes how the clone database of the RZPD Berlin (Deutsches Ressourcenzentrum für Genomforschung GmbH) is connected to SEMEDA and thus linked to the other databases.

In the discussion (chapter 11) SEMEDA is compared to existing database integration systems, especially other ontology based integration systems. It is further discussed how principles for semantic database integration apply to other database integration systems and how they might be implemented there. A database mirror is proposed to improve the overall performance of SEMEDA and the BioDataServer.

Whereas parts of this thesis were previously published in (Köhler et al. 2000, Köhler and Schulze-Kremer 2001), only the parts of the publications that represent the work of the author are incorporated in the thesis. Information about the MARGBench System (mainly parts of section 4.3) are compiled from personal communication (Matthias Lange, IPK Gatersleben, Germany, January 2002) and (Freier et al. 1999, Freier et al. 2002b).

SEMEDA is available at:

http://www-bm.ipk-gatersleben.de/semeda/

Login: semeda

Password: pw

2 State of the Art

This chapter gives an overview about the state of the art in research areas relevant to semantic integration of molecular biological databases. These are molecular biological databases, the different storage methods and DBMS of molecular biological databases, existing database integration systems and the use of ontologies.

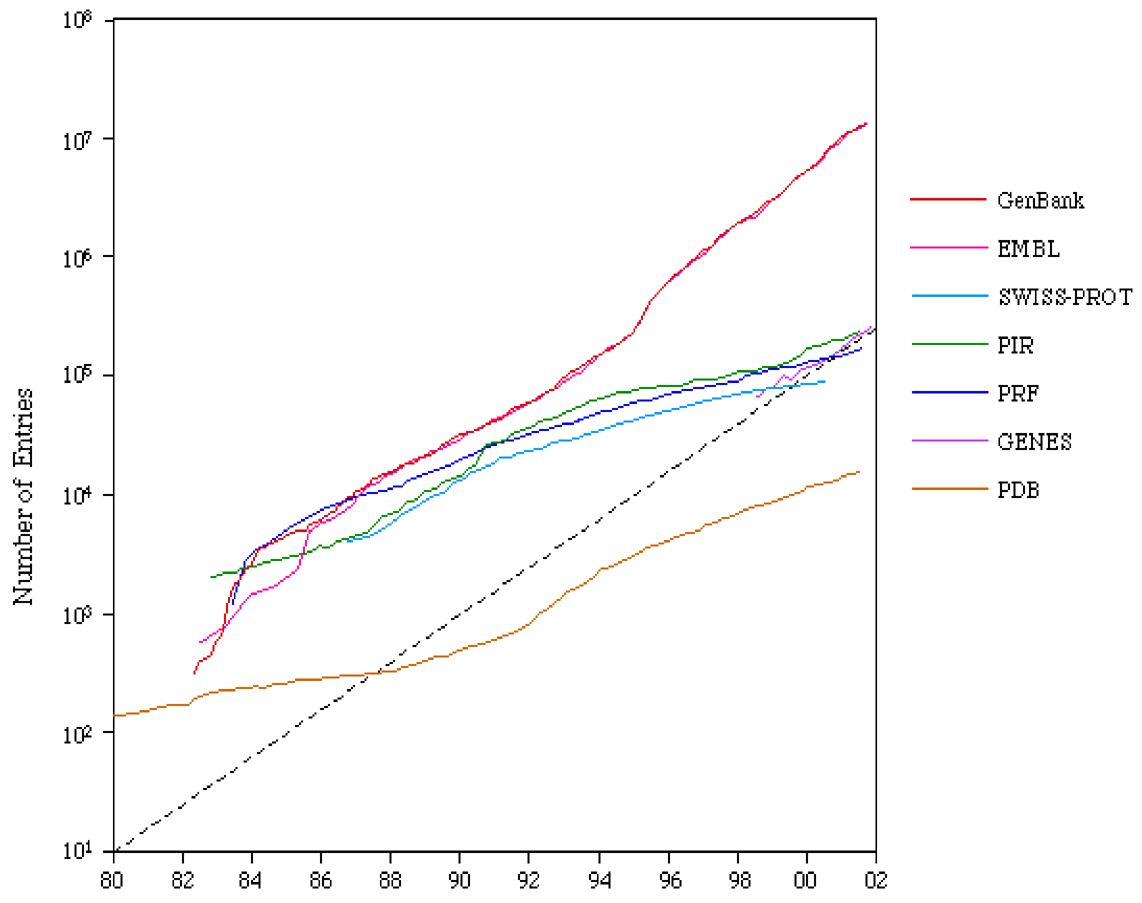


Figure 1: Exponential growth of major molecular biological databases (logarithmic scale). See http://www.genome.ad.jp/dbget/db_growth.html.

In conjunction with the rapid progress of biotechnologies and the human genome project (Aldhous 1990), an increasing amount of data is being generated (Figure 1). The amount of new data is that big that human genetics journals are increasingly reluctant to publish mutation reports (Krawczak et al. 2000). However much data is often published in publicly accessible data sources. At the moment, more than 400 molecular biological databases exist (Baxevanis 2002). The various data sources are maintained by many different institutions and companies and vary widely in their content, formats and access methods. They contain data about metabolic pathways, protein structures, DNA sequences, organisms, diseases, etc. These databases do not only vary with regard to their content, but also in the way they are stored and how they can be accessed.

2.1 Molecular Biological Databases

Why does molecular biology generate such an increasing amount of new data? There are mainly two driving factors: on the one hand, molecular biology is a highly complex field of research. Thousands of enzymes, genes, chemical compounds, diseases, species, cell types, organs etc. exist, interact and are related in many different ways. On the other hand, new molecular biological methods are permanently developed. By automating these methods robots are developed that enable scientists to gain more and more data and insights into biological systems per time. Such robots are used for DNA sequencing, gene expression profiling, drug screening etc. These two factors and the (expected) results of molecular biological research, explain why so much molecular biological data of various types is being generated.

In molecular biology, one can discriminate between primary and secondary databases. Primary databases directly store the experimental results of scientists, whereas secondary databases are derived by one or several of the subsequent procedures:

1) manual or automatical enrichment of data (annotation)
2) removing redundancy of primary databases and validation
3) manually collecting data from literature references (curated databases)
4) compiling data from several databases

Three primary nucleotide databases exist: EMBL, GenBank and DDBJ. They include both sequences submitted directly by scientists and sequences taken from literature and patents. Comparatively little error checking is applied and there is a fair amount of redundancy. The entries in the EMBL, GenBank and DDBJ databases are synchronised on a daily basis.

The fact that the data in the primary nucleotide databases is fairly redundant needs further explanation. In relational databases, redundancy of data can be avoided simply by setting unique key constraints and equivalent mechanisms can comparatively easily be implemented in non relational DBMSs. However, the main reason why redundancy cannot easily be overcome in primary nucleotide databases is that it is non trivial to decide when two nucleotide sequences should be considered to be equal: Are sequences which exactly match, but where one sequence is a bit longer than the other redundant? Or can sequences that differ only in one nucleotide be considered as equal under certain conditions? Those are questions, which cannot be unequivocally answered. In order not to lose data, which might become valuable in the future, the primary nucleotide databases store data redundantly when in doubt and little error checking is applied. They leave it to the secondary databases to clean up the data following their own policy. Therefore, the quality of the data stored in molecular biological databases has recently become a matter of concern (Harger et al. 1998, Andrade et al. 1999). Errors can originate at several levels, starting from the experimental generation of data (PCR errors) and ending at the reporting of data to databases (Aboa et al. 2000, Cotton and Horaitis 2000). Quality of data also applies to the incompleteness and missing links between well-known databases (Macaulay et al. 1998) such as the MGD (Blake et al. 2000) and GDB (Letovsky et al. 1998). Bork showed that data that is gained by automated methods such as sequence analysis, often do not hold experimental validation (Iyer et al. 2001). This leads to the development of databases that emphasize on data quality, such as HGVBase (Fredman et al. 2002) and SWISS-PROT (Bairoch and Apweiler 2000).

An example for a secondary nucleotide database is UniGene (Schuler 1997) which attempts to process the GenBank sequence data into a non-redundant set of gene clusters. Each UniGene cluster contains a set of sequences, which represent a unique gene along with related information for this gene.

Several molecular biological databases of many different types exist: SWISS-PROT (Bairoch and Apweiler 2000) contains general protein data, PDB (Persson 2000, Westbrook et al. 2002) contains protein 3D structures, OMIM (Hamosh et al. 2002) contains genetic disease, BRENDA (Schomburg et al. 2002a, Schomburg et al. 2002b) contains enzyme data and metabolic reactions etc. In total more than 400 molecular biological databases exist (Baxevanis 2002).

The various data sources are maintained by many different institutions and companies, and vary widely in their content, formats and access methods. Whereas only a few years ago proprietary solutions often based on flatfiles were used for data storage, nowadays RDBMS are the de facto standard. Many biological databases were started in the early 80s, i.e. at times when the Internet was not widely used, and DBMSs by themselves required advanced technical skills. Data was made available by proprietary methods, later via static web pages, and even later when flatfiles grew too big, server side scripts like CGI scripts, were used for searching and data retrieval from flatfiles. For data exchange, usually proprietary flatfile formats were used, and several flatfile formats evolved.

2.2 Database Heterogeneity

Molecular biological databases are heterogeneous on several levels: in their storage methods (flatfiles and different DBMS), the structure and naming of database tables and attributes, the data entries and in their access methods.

2.2.1 Storage

Whereas many different DBMS exist, only the DBMS and storage methods, which are actually used in molecular biological databases are introduced in this section.

2.2.1.1 Flatfiles

Only a few years ago, molecular biological data was most commonly stored in ASCII text files. These flatfiles are structured by using letter codes at the beginning of each line or paragraph (see Figure 2).

Nowadays, the number of "databases" which are implemented as flatfiles dicreases and many databases are moved from their old flatfile representations to DBMSs. Flatfiles are no longer considered to be an appropriate alternative to DBMSs, but rather as a data exchange format (see 2.2.3) between molecular biological databases. However, flatfiles are still not obsolete, since many molecular biological applications operate on flatfiles. Many biologists start database searches by searching for specific patterns in sequences or by searching for sequences that are similar to a given sequence. Sequence analysis tools such as for example BLAST (Altschul et al. 1990), FASTA (Lipman and Pearson 1985) and REPuter (Kurtz et al. 2001) generally operate on flatfiles.

Searching, analysing and comparing nucleotide or amino acid sequences is not possible within relational databases, although recently some systems have been developed which facilitate collaboration of sequence analysis and relational databases (Banerjee 2000, Xie et al. 2000, Inman et al. 2001). The implementation of such hybrid systems is generally non trivial and requires that proprietary DBMS specific techniques are used. Therefore it is likely that in spite of the many good reasons against flatfiles, it is likely that flatfiles will remain the de facto standard for data exchange.

```
ID   APHSFRAG   standard; RNA; VRL; 368 BP.
XX
AC   L11360;
XX
SV   L11360.1
XX
DT   12-OCT-1993 (Rel. 37, Created)
DT   04-MAR-2000 (Rel. 63, Last updated, Version 3)
XX
DE   Aphthovirus S fragement RNA.
XX
KW   .
XX
OS   Foot-and-mouth disease virus (strain A12)
OC   Viruses; ssRNA positive-strand viruses, no DNA stage; Picornaviridae;
OC   Aphthovirus; Foot-and-mouth disease virus A.
XX
RN   (1999)
RP   1-368
RX   MEDLINE; 94353645.
RA   Bunch T., Rieder E., Mason P.;
RT   "Sequence of the S fragment of foot-and-mouth disease virus type A12";
RL   Virus Genes 8(2):173-175(1994).
XX
FH   Key             Location/Qualifiers
```

```
FH
FT   source          1..368
FT                   /db_xref="taxon:12114"
FT                   /organism="Foot-and-mouth disease virus (strain A12)"
FT                   /strain="119ab"
FT   5″UTR           1..367
XX
SQ   Sequence 368 BP; 53 A; 125 C; 105 G; 85 T; 0 other;
     ttgaaagggg gcgctagggt ctcacccctaa gcacaccaac gacagtccct gcgttgcact        60
     ccacacttac gttgtgcgta cgcggggccc aatggacctt cgttcaccca cctacagctg       120
     gactcacggc accgcgtggc cattttagct ggactgtgcg gacgaacgcc gcttgcgcaa       180
     ctcgcgtgac cggttagtac tcttaccact ctccgcctac ttggtcgtta gcgctgtctt       240
     gggcactcct gttgggggcc gttcgacgct ccacgggttc ccctgtgcgg caactacggt       300
     gatggggccg tttcgcgcgg gctgaccgcc tggtctgttt cggctgtcac ccgacgtccg       360
     cctttcac                                                                368
//
```

Figure 2: Example for an EMBL flatfile entry. Whole databases can be stored by concatenating such flatfiles. At the start of each line, a two-letter code is used to distinguish between different kinds of information about the sequence. Besides the nucleotide sequence (SQ), EMBL stores identifier (ID), accession number (AC), timestamp (DT), description (DE), keywords (KW), organism name (OS), taxonomic data (OC), and literature references (RX,RN,RP,RA,RT,RL). Features (FT) describes features found in the sequence, and when applicable the positions of those features in the sequence. At the end the sequence (SQ) is provided.

2.2.1.2 ACeDB

The best description of ACeDB can be found at its homepage http://www.acedb.org/:

> "ACeDB is a genome database system developed since 1989 primarily by Jean Thierry-Mieg (CNRS, Montpellier) and Richard Durbin (Sanger Institute). It provides a custom database kernel, with a non-standard data model designed specifically for handling scientific data flexibly, and a graphical user interface with many specific displays and tools for genomic data. AceDB is used both for managing data within genome projects, and for making genomic data available to the wider scientific community.
>
> ACeDB was originally developed for the C.elegans genome project, from which its name was derived (A C. elegans DataBase). However, the tools in it have been generalized to be much more flexible and the same software is now used for many different genomic databases from bacteria to fungi to plants to man. It is also increasingly used for databases with non-biological content."

ACeDB is the incarnation of the German chimera "Eierlegende Wollmilchsau". It runs on Windows, several Unix versions and Macintosh, provides Java, C, CORBA and Perl interfaces, is published under the GNU license and includes various custom methods for visualisation and analysis of molecular biological data.

A small but significant proportion of the molecular biological databases are implemented using ACeDB (Bry and Kröger 2001). It seems to be especially suitable for small to medium sized in-house databases. Further information about ACeDB can be found in (Walsh et al. 1998, Kelley 2000).

2.2.1.3 Object Oriented Database Management Systems

According to (Bry and Kröger 2001), about 7% of all molecular biological databasese are implemented on Object Oriented Database Management Systems (OODBMS), and another 3% are implemented on object relational DBMS. Although OODBMS are relatively new, they are reasonably well standardised and have a sound theoretical background. Many OODBMS can be accessed via JDBC.

In OODBMS any complex data types that can be implemented in the object oriented programming language that is used to generate the objects, can be stored simply by storing objects. In addition, along with the data the methods of the objects can be stored. Thus the commonality between the application programmed in an object oriented language and the databases type systems preserves the datatypes when data is stored in an OODBMS.

An in depth discussion of OODBMS is beyond the scope of this short introduction.

2.2.1.4 Relational Database Management Systems

Most molecular biological databases are implemented on relational database management systems (RDBMS) (Bry and Kröger 2001). Since a considerable amount of molecular biological databases are based on proprietary flatfile solutions, relational DBMS are not as much used for molecular biological databases as in other application domains of databases (for example economics). Recently more and more flatfile databases are migrated to relational DBMS.

In contrast to OODBMS, in relational databases new datatypes cannot be defined. Date (Date 2000) criticises this fact, since this is not an inherent property of relational databases (Date 1982, Louis and Pirotte 1982, Pirotte 1982), but rather a result of how relational DBMS were implemented.

Relational databases were firstly introduced in 1970 (Codd 1970). Since then, a strong theoretical background and many RDBMS have been developed. Nowadays most databases are implemented on relational DBMS. Subsequently it will be assumed that the reader is familiar with the basic concepts of relational databases. For further information the reader is referred to the already existing literature on relational databases, for example (Heuer and Saake 2000, Vossen 2000).

2.2.2 Semantic Heterogeneity

In computer sciences, the semantics of a programming language describes the relationship between the syntax and the model of computation. Whereas the syntax of a programming languages can usually well be formalised, the semantics of programming languages more or less defy formalisation. However, with regard to linguistics, the term semantics applies to the meaning of words. In a linguistic sense, semantic conflicts occur for example when the same symbol is used for different things (mouse as computer device or an animal), or when different symbols exist for the same thing (Mus Musculus and house mouse). Thus dealing with semantics in a linguistically sense, involves for example dealing with synonyms or with the disambiguation of the meaning of homonyms.

Such semantic conflicts that are due to inconsistent naming of database tables, attributes or entries also occur between relational databases. In Figure 3, examples for the various semantic conflicts that occur between relational databases are given. These conflicts occur in an equivalent way between other database types. These semantic conflicts are described in more detail in section 2.2.2.1 - 2.2.2.3, and chapter 3 describes methods that are suited to overcome semantic heterogeneity of databases. Semantic conflicts between relational database schemata are well known since (Kim and Seo 1991), but have not yet been solved sufficiently.

In (Hammer and McLeod 1993) semantic heterogeneity is described as "By this (semantic heterogeneity) we mean variations in the manner in which data is specified and structured in different components. Semantic heterogeneity is a natural consequence of the independent creation and evolution of autonomous databases which are tailored to the requirements of the application system they serve."

A more vivid description is given in (Kim 1995): "A schema contains a semantic description of the information in a given database. It is possible to define equivalent schemas in as many ways as there are data models. Further, the same (or similar) information can be represented in many ways in the same data model. Given such inter- and intra-model variability, it is indeed a formidable task to integrate many schemas into a homogeneous schema."

2.2.2.1 Attributes

At the attribute level, different attributes can have the same name and semantically equivalent attributes may have different names.

In Figure 3, the attribute ID in the table SEQUENCES and the attributes MEDLINE in the table DNA both store Medline IDs, but are named differently. On the other hand, both tables have an attribute ID that contains different data. In the table SEQUENCES, ID stores the Medline ID, whereas in the table DNA the attribute ID stores an internal identifier.

Further conflicts at the attribute level arise when equivalent attributes use different datatypes. For example a Medline ID can either be stored as an integer or a string.

2.2.2.2 Tables

Similar conflicts can occur between database tables. Databases can contain different tables, which have the same name, or tables that store equivalent data but use different names.

In addition, table structure conflicts arise when similar tables have partly different attributes. For example in Figure 3 the table SEQUENCES has an attribute AB that does not exist in the table DNA and on the other hand, DNA has an attribute ID which is used for different kind of data entries the table SEQUENCES.

After (Kim and Seo 1991) "missing but implicit conflicts" occur when database tables miss an attribute which would always have the same entry. Such situations occur in databases which for example store data about one species. Such database tables often do not have an attribute for the species name, since it would always have the same value.

Further syntactic conflicts arise when tables use different primary, unique or foreign key constraints. These conflicts do not matter when databases are integrated for read only access, which is the case for the integration of molecular biological databases (Karp 1995). Thus for the integration of molecular biological databases such constraints can simply be ignored.

SEQUENCES				
AU	SQ	AB	ID	OC
Lenz, A	cctgga...	The prot...	82247835	rat
Coen, A	ctggat...	Analysis...	81245818	mouse

DNA				
ID	AUTHOR	SEQUENCE	MEDLINE	SPECIES
22	Lenz, A	cctgga...	82247835	Rattus Norvegicus
23	Coen, A	ctggat...	81245818	Mus Musculus

Figure 3: Semantic conflicts between relational database schemas. Although the two database tables store basically the same data, to integrate those two tables into one table, conflicts at several levels have to be solved. The different conflicts between the two tables are described in the text.

2.2.2.3 Data entries (Attribute values)

Different people use different words for the same things. An interesting example is the use of species names. Common English names, systematic species names, a species identifier or a mix of common English names and systematic species names are often used. Even within systematic species names differences exist: Sometimes genus names are abbreviated, sometimes the subspecies is included etc. In addition, since species names are dependent from their phylogenetic relationship, species names are unstable, i.e. are subject to change over time. Whereas this unsystematic naming helps biologist to learn phylogenetic relationships, such unsystematic "systematic naming" makes database integration difficult.

Besides the use of different vocabularies, measurement units and the precision at which data is stored may vary. Temperature might be stored in Celsius or Fahrenheit and with a different precision. Therefore, scientists in the field of bioinformatics spend an significant amount of their time for data conversion (Stevens et al. 2001).

2.2.3 Access Methods

Most Databases can be accessed and searched via web pages. Usually appropriate HTML forms support the user at querying the databases. Whereas web pages are appropriate for humans, other interfaces are more suitable for computers. Common interfaces of relational and object orientated databases are JDBC and ODBC.

Most DBMS nowadays have built in support for various data exchange methods, such as JDBC, ODBC, XML etc. Probably mainly due to security concerns, access via those structured query methods is generally not granted. The reasons for such concerns have recently become insubstantial, since JDBC type 3 drivers which provide restricted and secure access to source databases have recently become available for all major RDBMS (see also section 11.2.1).

It can be summarised that in spite of the fact that most DBMS support structured access methods which are well suitable to overcome “technical” heterogeneity, access to these interfaces is not granted. The most common way of data exchange is by using flatfiles. Most molecular biological databases provide flatfiles or database text dumps which can be downloaded via ftp.

2.3 Database Integration

A prerequisite for database integration is the appropriate availability of the databases that are to be integrated. Although this seems to be trivial, it should not be left unmentioned, since this is often the most difficult part of database integration. Molecular biological databases are either the result of many molecular biological experiments or of manual extraction of literature data. Thus it took often several man-years to acquire the data. Therefore it is understandable that in many cases even owners of databases, which have been developed within publicly funded research projects, do not make their data completely available. The fact that most databases can be searched and queried via a web page, does not mean that the data is completely available. Compared to the total size of the databases, often only small amounts of data can be retrieved per query via the HTML frontend. Since outside of Europe copyright protection of databases is non-existent (Maurer et al. 2001), database providers often use the limited accessibility via web pages as a way to protect their intellectual property, i.e. data retrieval is limited by the speed with which a human can interact with a web page.

Different approaches to database integration of molecular biological databases are discussed and reviewed in (Davidson et al. 1995, Karp 1995, Jakobovits 1997, Parent and Spaccapietra 1998, Freier et al. 2002b).

Karp (Karp 1995) discriminates four approaches for database integration. These are:

- Hypertext Navigation, i.e. the HTML frontends of molecular Biological Databases are interlinked.
- Data Warehouse, i.e. physically merging (converting, importing...) of several databases into one big database.
- Multi Database Queries, i.e. querying several databases at the same time.
- Federated Databases. In contrast to “Multi Database Queries”, federated databases integrate database schemata in a federation layer, although like in multi database approaches each database remains autonomous.

Pro and cons of these approaches are discussed by (Karp 1995). However, the aims of these approaches are the same: providing a technique to overcome the several kinds of data heterogeneity to build an unique data retrieval environment for biologists to support their research activities.

2.3.1 Hypertext Navigation Systems

At present, most databases are connected to the Internet and can be accessed via web pages. Many databases provide hypertext links to entries in other databases. In most cases, AC numbers (accession numbers) or other database specific identifiers which are generated when a new entry is added to a database are used for interlinking. Those identifiers are often used for interlinking between databases via their web pages. Due to the fact that in many cases different databases use different identifiers or terms for equivalent entries, interlinking databases is labour intensive. Usually, pair-wise mappings between database entries have to be generated in order to be able to provide links between databases. Therefore, databases usually only provide links to the “most relevant” databases by using accession numbers.

In addition, besides accession numbers, many other database attributes which use common controlled vocabularies such as EC numbers (NC-IUBMB 1992), CAS Registry numbers (Buntrock 2001), GO terms (Ashburner et al. 2000, Gene-Ontology-Consortium 2001), etc. are suitable for linking between databases. Even when databases use the same controlled vocabularies, they are often not used for linking between databases. This is due to the simple fact that in order to be able to generate and maintain pair wise links between the more than 400 molecular biological databases (Discala et al. 2000, Baxevanis 2002) all database providers would have to be aware of the all other relevant databases.

None the less, in spite of the fact that the databases are not as much interlinked as would be useful, (Williams 1997), interlinked web pages are the most often used way of database "integration". According to (Bry and Kröger 2001) 97% of all databases provide at least some links to other databases.

2.3.2 Indexing Systems

Database integration solutions based on indexed flatfiles such as DBGET/LinkDB (Kanehisa 1997a, b, Kanehisa et al. 2002), SRS (Etzold et al. 1996, Zdobnov et al. 2002) and SIR (Ramu 2001) are the de facto standard for the integration of high numbers of heterogeneous databases.

The main principle of indexing systems is simple: The databases to be integrated are provided as flatfiles. The integration system indexes these flatfiles using a script which has to be provided for each database. Thus an indexing system can support various flatfile formats. The indexing script is also responsible for discriminating datatypes and for generating links to other relevant databases. For example within SRS, a proprietary indexing scripting language called Icarus exists which has to be used for writing the indexing scripts. Therefore the aforementioned low degree of interlinking between databases also applies to indexing systems. Based on the indexes, users can search all indexed databases in one step. When indexing systems discriminate between datatypes, the databases can also be searched using common datatype specific comparison operators. In addition, indexing systems can be searched using sequence similarity search tools and the results are visualised in an appropriate way.

Since indexing systems do not require the maintenance of an integrated database schema of all integrated databases, indexing systems enables the addition or removal of any number of flatile databases without affecting other databases in the integration system.

SRS can be accessed either via a comparatively user-friendly HTML frontend or via a command line utility. Figure 4 shows some molecular biological databases accessible via SRS. The degree of interlinking seems to be high at the first glance, but many important databases are only linked to SWISSPROT, and many databases of the same application domain are not directly interlinked. Whereas Figure 4 shows about 30 databases, at present about 500 databases can be accessed via SRS, and typical SRS implementations integrate between 50 and 100 databases.

The hardware requirements of SRS are comparatively moderate: a minimum server can be installed on an Intel Pentium computer with 200 GB of hard disk space and 256 MB of RAM. The SRS server installed at the EBI (Zdobnov et al. 2002), runs on 2 Compaq ES40 servers (4 processors each), 300 GB of hard disk space and 256 MB of RAM in each server. The EBI server supports 2.7 million hits per month (see also http://www.lionbioscience.com/repository/srs-product-sheet.pdf).

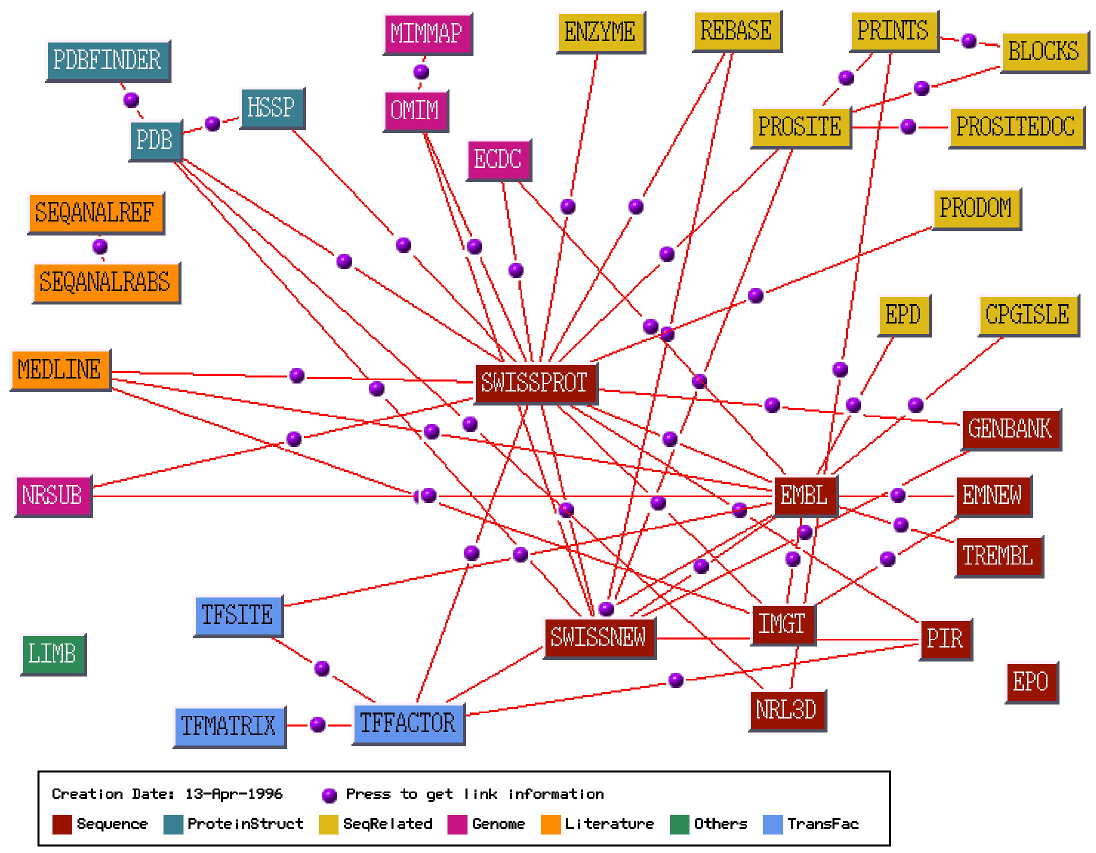


Figure 4: Molecular biological databases accessible via SRS. Each node represents one database (Etzold et al. 1996). The colours encode the application domain of the databases. The edges represent links between databases. The degree of interlinking seems to be high at the first glance, but many important databases are only linked to SWISSPROT, and many databases of the same application domain are not directly interlinked.

2.3.3 Database Mediation and Federation

Database integration systems which use mediation or federation typically consist of three elements: wrappers, an integration layer and a query interface (see Figure 5). The wrappers provide uniform access to the heterogeneous data sources. The integration layer decomposes user queries, sends them to the relevant wrappers and finally integrates the query results before the result is returned to the user via the query interface. In addition, often several other components such as administrative tools and query optimiser exist.

In contrast to mediated databases, in federated databases wrappers mainly map the different interfaces between the data sources. Thus in database federations it is required that the data sources provide the main search and query functionality via different interfaces so that the wrappers mainly have to translate between the different interfaces.

In many cases the data sources do not provide suitable search or query methods. Examples are web pages, flatfiles and other more or less unstructured data sources. In mediated databases, as the term mediation emphasizes, the wrappers play a more active role and implement when necessary missing search or query methods for the data sources.

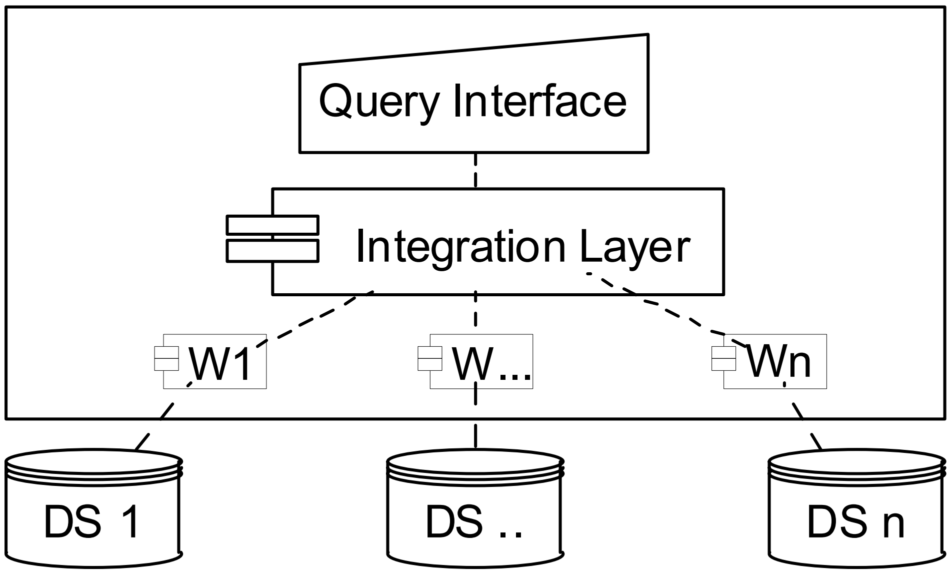


Figure 5: Typical architecture of database mediation and federation systems. The wrappers provide uniform access to the data sources. The integration layer decomposes user queries, sends them to the relevant wrappers and finally integrates the query results before the results are returned to the user.

Several different integration systems exist. They vary in the degree of integration, query interfaces they provide, cost based query optimisation, access methods to the source databases and the integration of data source specific methods.

An elaborate commercial system which has been developed with "life sciences" in mind, is IBMs DiscoveryLink (Haas et al. 2001) which has evolved out of the Garlic project (Haas et al. 2000). In addition, several mediated and federated integration systems exist (Hammer and McLeod 1993, Croft et al. 1995, Jakobovits 1997, Chung and Wong 1999, Freier et al. 1999, Matsuda et al. 1999, Critchlow et al. 2000, Wong 2000, Freier et al. 2002b). These federated databases do not systematically solve the above-mentioned issues related to semantics.

2.3.4 Data Warehouses

Data Warehouses and DataMarts are mostly used in business informatics. Data Warehouses integrate and aggregate data of several different DBMS into one system. In the creation of Data Warehouses, direct access to source databases is usually granted, and techniques like views and materialised views can be used. Usually an elaborate integrated database is developed once. This integrated schema is more or less stable once it has been filled with data.

Systems, which integrate databases in a Data Warehouse approach, are usually restricted to integrating only a few source databases and manage to achieve a higher degree of interoperability of the integrated systems. Warehouse systems are limited by the fact that it is generally not possible to integrate new databases without changing the schema of the Data Warehouse, since this is usually associated with many conflicts between the new database schema and the Data Warehouse schema. Even updating such closely integrated databases from source databases can be difficult and sometimes impossible, especially when database schemas of the source databases change, which is often the case in molecular biological databases (Karp 1995).

The requirements for molecular biological databases are completely different. In (Critchlow et al. 2000) the reasons why the traditional warehouse approach is not applicable to molecular biological databases are summarised:

> "First, schema integration is more difficult for scientific databases than for business sources, because of the complexity of the concepts and the associated relationships. While this difference has not yet been fully explored, it is an important consideration when determining how to integrate autonomous sources. Second, scientific data sources have highly dynamic data representations (schemata). When a data source participating in a warehouse changes its schema, both the mediator transferring data to the warehouse and the warehouse itself need to be updated to reflect these modifications. The cost of repeatedly performing these updates in a traditional warehouse, as is required in a dynamic environment, is prohibitive."

Another situation that Data Warehouses cannot handle, is the integration of varying combinations of databases. For example in SRS it is possible to add or remove new databases to the integration systems on the fly.

Therefore traditional Data Warehouses play practically no role for molecular biological database integration.

2.4 Ontologies

In Artificial Intelligence ontologies are data structures for knowledge representation, which originated from philosophy and are related to conceptual graphs and semantic nets. In the following, ontologies are introduced informally. A formal definition is given in section 3.2. According to Gruber "an ontology is a specification of a conceptualisation" (Gruber 1993b, a). The notion about ontologies, their formal notation and how they should be implemented varies between people and research groups (Noy and Hafner 1997), although most ontologies share a few core items (Stumme and Maedche 2001). Most ontologies consist of a set of concepts which represent real world things such as "enzyme", "amylase" or "amino acid". These concepts are connected by typed relations, which describe how the concepts are related. For example a relation of type "is a" can be used to state that "amylase" "is a" "enzyme" and by using the "is part of" relation type one can state that "amino acid" "is part of" "enzyme".

In order to be language independent, ontologies use concepts and not words. Two things are semantically equal if they address the same concept, i.e. the same real world entity. For example, “zebrafish” and “Danio rerio” address the same fish species, although by a different name. A homonymous example is the term “hybrid”: On the one hand “Hybrid” can be hybridised DNA as “DNA hybrid”, on the other hand “hybrid” can be crop which results from crossing crop varieties or lines that have little or no direct relationship with each other as in “hybrid crop”.

Further informal definitions for ontologies are given in (Schulze-Kremer 1997b, Stevens et al. 2000b, Schulze-Kremer 2002). A more formal definition and examples are given in (Guarino 1998) and in section 3.2.

The vision of the Semantic Web is based on ontologies http://www.semanticweb.org/introduction.html.

> "The Semantic Web is a vision: the idea of having data on the web defined and linked in a way, that it can be used by machines - not just for display purposes, but for using it in various applications.
>
> ...
>
> Indeed, we have the technology available for realizing the Semantic Web, we know how to built terminologies and how to use metadata. The whole vision depends on agreeing on common standards - something that is used and extended everywhere."

Thus Barners-Lee who invented the World Wide Web(Berners-Lee et al. 1992) has the vision that the Semantic Web becomes the "next generation" of the internet, where data on the web is defined and linked in a way, that it can be used by machines. Therefore ontologies have recently developed to a big topic in commercial information technologies.

Also in bioinformatics, ontologies found many applications. They are used for Medical image searching (Greenes et al. 1992, Frankewitsch and Prokosch 2001), metabolic pathways (Karp et al. 1999, Karp et al. 2000, Karp 2001), protein database annotation (Xie et al. 2002) and metasearches to taxonomic/biodiversity data (Edwards et al. 2000). In addition several data formats are being developed based on ontologies such as for storing and exchanging macromolecular structures (Westbrook and Bourne 2000), results of microarray experiments (Brazma et al. 2001) and medical image processing results (Aubry and Todd-Pokropek 2001). Further applications of ontologies in molecular biology are reviewed in (Volot et al. 1998). The use of ontologies for database integration is reviewed in the next section.

2.4.1 Using Ontologies for Database Integration

Ontologies can on the one hand be used to define a common controlled vocabulary and on the other hand to semantically define databases.

Homologous genes are generally differently named in different species. The fact that for the annotation of genes and other entities in molecular biological databases no common controlled vocabulary existed, which made it difficult to search databases for special traits, genes, functions, cell organelles etc. Therefore the Gene Ontology Consortium http://www.geneontology.org was founded by a group of database providers with the goal to "produce a dynamic controlled vocabulary that can be applied to all organisms even as knowledge of gene and protein roles in cells is accumulating and changing". Using ontologies as controlled vocabularies, is similar to using EC numbers, CAS-registry numbers etc. The hierarchical structure of ontologies helps users to find concepts in the ontology. In addition, by using ontologies it is possible to add relations between concepts. The Gene Ontology Consortium consists of about ten institutions, among others the EBI, TIGR and the MGD. Further information on the Gene Ontology can be found in (Ashburner et al. 2000, Gene-Ontology-Consortium 2001).

Once a database uses a controlled vocabulary, ontologies can also be used to translate between different vocabularies. Giudicelli (Giudicelli and Lefranc 1999) describe a system where an ontology is used within a frontend to a relational database, which enables users to use their own terminology.

Thus, ontologies and other controlled vocabularies can be used to overcome heterogeneity of the entries of molecular biological databases. The use of ontologies to define the semantics of databases at the schema level has been suggested by (Karp 1995, Kashyap and Sheth 1996a, Kashyap and Sheth 1996b, Schulze-Kremer 1997a, b, Goksel and McLeod 1999, Hakimpour and Geppert 2001).

TAMBIS (Baker et al. 1998, Baker et al. 1999, Stevens et al. 1999, Stevens et al. 2000a) is an example for such a system. TAMBIS uses the GALEN ontology (Rector and Nowlan 1994) to semantically define databases. Based on a graphical representation of the ontology, the user can construct appropriate database queries which are subsequently processed against the source databases. TAMBIS uses the Kleisli system (Chung and Wong 1999, Wong 2000), a mediated database integration system to process the user queries against the data sources.

Another ontology based database integration system (Ludäscher et al. 2001) was implemented by using F-LOGIC to semantically define and query wrapped XML data sources. This system, TAMBIS and the differences to the approach described in this thesis will be discussed in more detail in the discussion.

2.4.2 Ontologies and Standards

In this section, the efforts to standardise ontologies are discussed. The notion of the data structure of ontologies varies widely. Therefore, a definition of ontologies that is valid for the scope of this thesis is given in section 3.2.

In (McEntire et al. 2000) a list of some ontology exchange data formats/languages is given and evaluated and a standard for conceptional graphs has been proposed http://www.bestweb.net/~sowa/cg/cgstand.htm. In (Hendler and McGuinness 2000) the "semantic web and its languages" are discussed. Interestingly, at present most "real world" applications do not make use of these formats or standards. Almost none of the existing ontologies or knowledge resources such as WordNet (Fellbaum 1998) or the Unified Medical Language System (UMLS) supports any of the exchange formats, but rather provide database dumps, tab-delimited ASCII files or proprietary data formats. This might on the one hand be due to the fact that most systems, which can handle large ontologies, are implemented as databases, i.e. both for importing and exporting, database dumps and tab-delimited files are easier to handle than the elaborate exchange formats. On the other hand, some ontologies use proprietary flatfile formats, which were co-developed with software tools or applications.

Since there are as many different formats for storing ontologies as there are ontologies, exchanging ontologies is tedious. Therefore the obvious solution would be to agree upon one generally accepted exchange language. However, the expressiveness and the data structures of the ontologies widely varies, and so do their proprietary exchange formats. Thus a standard exchange language which can store most ontologies would have to be quite complex, probably so complex and hard to use that most people would not use it. Therefore also (Hendler and McGuinness 2000) conclude "It is unlikely that a single ontology language can fulfill all the needs of the semantic Web's large range of users and applications".

Often in applications that use ontologies, the data that represents the ontologies and the applications are not strictly separated. In some systems relations cannot easily be modified or added since relations between concepts are often more or less hardcoded. However, even such hardcoded ontologies should be capable of exporting the ontology.

Alternatively an ontology exchange language might include only the "most important" features which most ontologies have in common. Such a language could not substitute the many more or less proprietary data formats, but would allow to exchange at least some data. The RDF standard http://www.w3.org/RDF/ that was recently released by the W3 consortium might play this role. The RDF standard is a dataformat based on XML that is well suited as an interchange format of ontologies and controlled vocabularies. Although it is likely that RDF will become the lingua franca for the exchange of controlled vocabularies and ontologies, at present from the ontologies listed in Table 7, only the Gene Ontology (Ashburner et al. 2000, Gene-Ontology-Consortium 2001) can already be downloaded in this format. Recently many tools and applications which are based on RDF have been developed, such as for example the feature rich Redland RDF application framework (Beckett 2001) (http://www.redland.opensource.ac.uk/), the RQL query language (Karvounarakis et al. 2002), SquishQL (http://swordfish.rdfweb.org/rdfquery/) and several others as discussed in (Silvonen and Hyvönen 2001a).

There are also attempts to standardise the content of ontologies. The goal of the SUO working group (IEEE P1600.1, http://suo.ieee.org/) is to develop a "Standard Upper Ontology" which is used as a standardised set of high level ontology concepts. The idea is, that the interoperation between different application specific ontologies can be facilitated, when they agree upon this basic sets of concepts. Due to the fact that several older and more widely used ontologies such as CYC (http://www.cyc.com/cyc-2-1/toc.html) exist, it is not clear whether or not the SUO will de facto become a standard toplevel ontology.

2.4.3 Ontology Editors

Ontology editors are important components of semantic database integration systems, i.e. for database integration systems that are suitable to overcome semantic heterogeneity in databases as introduced in section 2.2.2. Ontology editors allow on the one hand building and maintaining an ontology, which is used to semantically define databases. On the other hand, ontology editors can be used to edit ontologies like the Gene Ontology, which serve as controlled vocabularies.

Whereas several ontology editors, browsers and related tools have been built, the lack of a common ontology exchange format makes it hard to exchange ontologies, although recently several tools started to support the RDF format. The various ontology exchange formats and the lack of a common standard has already been discussed in section 2.4.2.

Ontology editors usually implement the “is a” hierarchy of ontologies as a tree in which the concepts are the nodes which are connected by the relations. Several methods to edit the ontology are provided. Figure 6 displays a screenshot of the OE (Schulze-Kremer 1997b) as an example for an ontology editor. In this ontology editor the functionality is implemented in PROLOG. Therefore, PROLOG can be used to derive inferences on the ontology or to define constraints on the ontology structure.

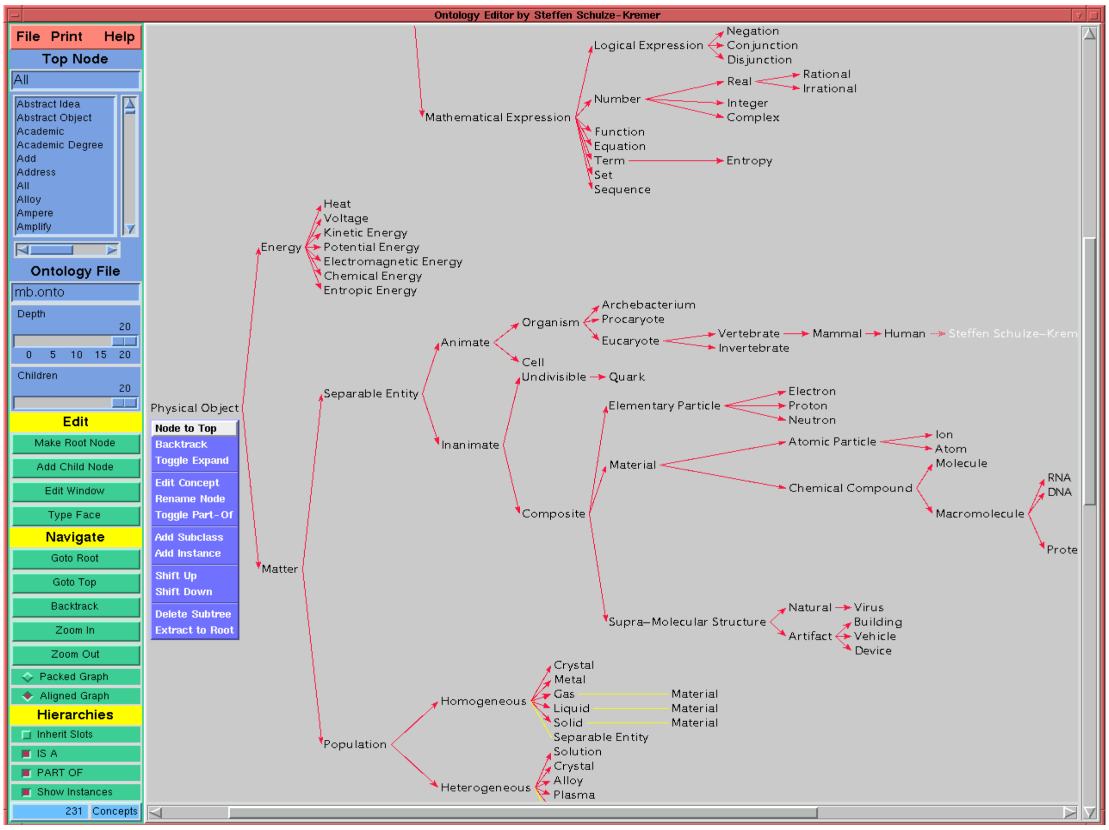
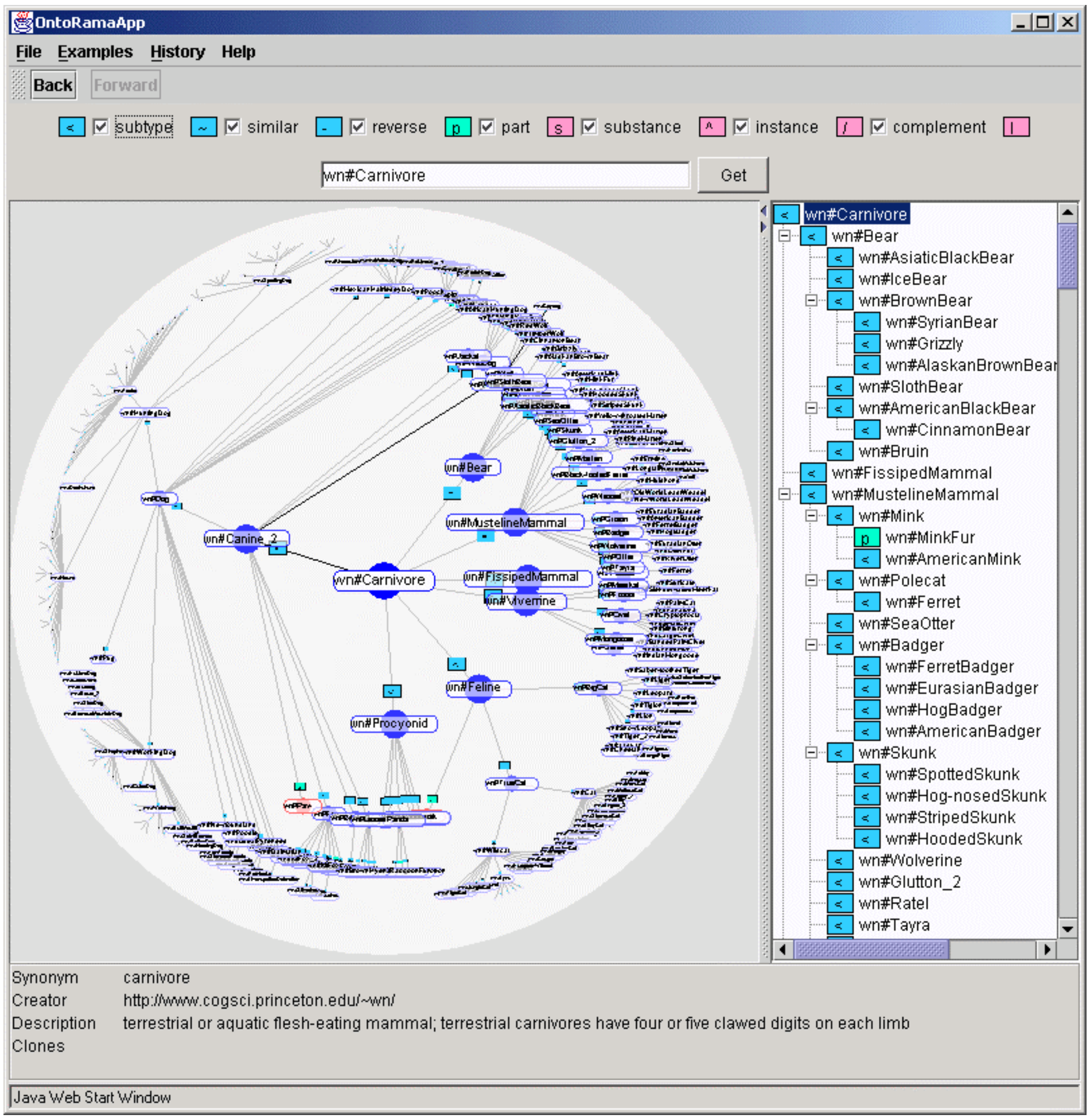


Figure 6: Example for an ontology editor. Screenshot of the OE ontology editor (Schulze-Kremer 1997b). The "is a" hierarchy of the ontology is visualised as a tree, which can be edited by drag and drop and by using the buttons on the left hand side.

The focus of the ontology editor of the Gene Ontology (GO Edit) lies in building and maintaining large ontologies, which are used as controlled vocabularies. In order to be able to handle the increasing size of the Gene Ontology, the Gene Ontology is stored in a relational database. The support for collaborative ontology editing in the ontology editor of the Gene Ontology is weak, since the tool that exist for editing the Gene Ontology does not operate directly on the ontology which is stored in the database, but imports the whole ontology, edits it and subsequently writes it back to the database. When the ontology is written back, and other users have also updated the ontoloy meanwhile, conflicts at different levels have to be resolved. When biologists want to suggest that a new concept is introduced to the Gene Ontology, they can do so by a HTML form or by writing an email to the person in charge of maintaining the ontology. Thus the ontology editor used for the Gene Ontology is more or less a "stand alone" system, although an ontology editor which supports collaborative ontology editing via the internet would be more appropriate for building and maintaining such ontologies that serve as controlled vocabularies for several research groups and database providers.

The size of ontologies is also a challenge for the visualisation of ontologies. Therefore techniques for the visualisation of large graphs (Herman et al. 2000) have been applied to ontologies, for example in Jambalaya (Storey et al. 2001), in Thinkmap® http://thesaurus.plumbdesign.com/index.html or in the OntoRama ontology browser (Figure 7).

Several other ontology editors exist: PROTEGE (Musen et al. 1995, Li et al. 2000a) Ontolingua (Farquhar et al. 1996, 1997) and WebOnto (Domingue 1998) are just a few examples. Comprehensive reviews of the existing ontology editors and related tools can be found in (Duineveld et al. 1999, Silvonen and Hyvönen 2001b).



wn#Carnivore

- wn#Carnivore
  - wn#Bear
    - wn#AsiaticBlackBear
    - wn#IceBear
    - wn#BrownBear
      - wn#SyrianBear
      - wn#Grizzly
      - wn#AlaskanBrownBear
    - wn#SlothBear
    - wn#AmericanBlackBear
      - wn#CinnamonBear
    - wn#Bruin
  - wn#FissipedMammal
  - wn#MustelineMammal
    - wn#Mink
      - wn#MinkFur
      - wn#AmericanMink
    - wn#Polecat
      - wn#Ferret
    - wn#SeaOtter
    - wn#Badger
      - wn#FerretBadger
      - wn#EurasianBadger
      - wn#HogBadger
      - wn#AmericanBadger
    - wn#Skunk
      - wn#SpottedSkunk
      - wn#Hog-nosedSkunk
      - wn#StripedSkunk
      - wn#HoodedSkunk
    - wn#Wolverine
    - wn#Glutton_2
    - wn#Ratel
    - wn#Tayra

Synonym	carnivore
Creator	http://www.cogsci.princeton.edu/~wn/
Description	terrestrial or aquatic flesh-eating mammal; terrestrial carnivores have four or five clawed digits on each limb
Clones	

Figure 7: For displaying large knowledge sources, recent ontology browsers like OntoRama use advanced graph visualisation techniques. http://www.webkb.org/ontorama/demo/index.html

3 Principles of Semantic Database Integration

No common generally accepted definition for ontologies exists. Therefore, this chapter introduces ontologies and gives a definition of controlled vocabularies and ontologies. Subsequently the main principles of semantic database integration are described.

3.1 Database Metadata

Database metadata is data about a database, which describes the logical structure, and other relevant information about a data source. The term metadata will be used in the following for database schema information, data about the DBMS, and for relevant technical data required to access a data source. Database metadata does not include data entries of the source databases.

The schema of relational databases consists of datatypes (domains) and tables (relations). Tables consist of attributes (fields) with an associated datatype as domain, and may contain data within the limits of these domains (Date 1982, Louis and Pirotte 1982, Pirotte 1982).

3.2 Controlled Vocabularies and Ontologies

As no generally accepted definitions for controlled vocabularies and ontologies exist, definitions of these terms are provided in the following as a prerequisite for further discussions.

Controlled vocabularies are named lists of terms that are well defined and may have an identifier. The elements of a controlled vocabulary are called concepts. Concepts can either be defined implicitly or by explicitly listing them. An example for a controlled vocabulary, which is defined implicitly without listing all concepts is temperature in °C. The terms or identifiers of controlled vocabularies are often used as database entries.

Definition:

Controlled Vocabulary CV:= named set of concepts c, with c:= (term, definition, identifier, synonyms)

Example:

An example for a controlled vocabulary is the Enzyme Nomenclature (International-Union-of-Biochemistry 1992). Each concept (enzyme) has a term (recommended name), an identifier (EC number) and synonyms (systematic name, other names). The definition of an enzyme is given by references to literature, which describe the enzyme in more detail.

In practice, controlled vocabularies can be defined using a text definition. Thus it is not necessary that all elements of the controlled vocabulary are explicitly listed in the integration system, i.e. a text definition is sufficient: "A number which uniquely identifies enzymes following the conventions of the Nomenclature Committee, International Union of Biochemistry and Molecular Biology, "(International-Union-of-Biochemistry 1992).

As opposed to a controlled vocabulary, an ontology consists of concepts which are linked by directed edges, thus forming a graph. The edges of an ontology in fact specify in which way, e.g. “is a” or “part of”, concepts are related to each other. This definition of an ontology is similar to (Stumme and Maedche 2001) and differs somewhat from (Guarino 1998) who defines ontologies and their interoperability on a higher abstraction level.

Definition:

Ontology O := G(CV,E), with $E \subseteq CV \times CV$ and a totally defined function $t: E \to T$ which defines the types of the edges. T is the set of possible edge types, i.e. the semantics of an edge in natural language and its algebraic relational properties (transitivity, symmetry and reflexivity). All ontologies have an edge type “is a” $\in$ T. If two concepts c1, c2 $\in$ CV are connected by an edge of this type, the natural language meaning is “c1 is a c2”, or more exactly “c1 is subclass of c2”. O is *formal*, if in its projection to the “is a” hierarchy all concepts are connected so that all concepts are arranged as a tree or directed acyclic graph (Kingston 1998) in which all definitions of super-concepts are valid for all sub-concepts.

Example:

In Figure 8, the concepts vertebrate, animal and organism are connected by transitive "is a" relations, i.e. vertebrate "is a" animal and animal "is a" organism. The transitive "is a" relations can then be used to derive the fact that vertebrate "is a" organism. Examples for informal ontologies are the Gene Ontology (Ashburner et al. 2000), UMLS (Srinivasan 1999). Examples for formal ontologies are the MBO (Schulze-Kremer 1998), EcoCyc/MetaCyc (Karp et al. 2000, Maranas and Burgard 2001) and WordNet (Fellbaum 1998).

The natural language meaning of “is a” can either relate to “subclass of” or “instance of”. In the following, “is a” always relates to “subclass of”. Whereas classes describe categories of things, an instance of a class is an individual element in a class. For example Homo sapiens is a “subclass of” vertebrate and Jacob Köhler is a “instance of“ Homo sapiens. Therefore, in the following the terms sub-concept and super-concept refer to sub-classes and super-classes.

As can be seen from the definition of ontologies and controlled vocabularies, ontologies can be reduced to controlled vocabularies simply by dropping information. This “reducibility” can be used when ontology editors and browsers are developed: a database schema, which can store ontologies, can also store non formal ontologies or controlled vocabularies.

3.3 Semantic Database Definitions

How can databases be semantically defined using ontologies and controlled vocabularies? The main idea to do so is to map tables and attributes of a database to a given ontology. This ontology should be formal with respect to the implementation of a transitive “is a” hierarchy, which connects all concepts. Although other hierarchies can be defined, only the “is a” hierarchy is used for query processing.

In the following, principles of semantic database integration are defined and examples are given to explain how semantic definitions can be used to overcome semantic heterogeneity as introduced in section 2.2.2. The example queries use the syntax “a:t” which has the meaning to search in the database attribute “a” for the term “t”. Such queries are sufficient to illustrate the principles of semantic database queries, although the query interface of SEMEDA is implemented in a more user friendly and powerful way.

3.3.1 Attribute semantics

Databases attributes can be semantically defined by linking them to concepts of an ontology. In consequence, attributes cannot only be addressed by a mapped concept c, but also by the sub- and super-concepts of c.

Definition:

A *semantic attribute definition* is a tuple (A, c) where $c \in O$ is a concept of the Main Formal Ontology and A is a database attribute.

Example Query:

“animal:mouse” - search in the attribute “animal” for “mouse”. In Figure 8 no database attribute is defined as animal. However, some sub- and super-concepts of animal are mapped to database attributes and should therefore be searched for “mouse”. For this query, the fact that “invertebrate” is unlikely to contain mouse data cannot be derived, thus both “invertebrate” and "vertebrate" should be searched. "Plant" will not be searched because it is not a sub- or super-concept of animal, and also because no attribute is defined as "Plant". If the user query is more specific, for example "vertebrate:mouse", the irrelevant database attribute "invertebrate" would not have to be searched.

3.3.2 Table Semantics

The content of a database table can be *refined* by linking the table to a concept of the ontology.

Definition:

A *semantic table definition* is a tuple (T, c) where T is a database table and $c \in O$ is a concept of the ontology O.

Example query:

'animal:mouse' (Figure 9). An imaginary 'mouse enzyme database' only contains information about mice; therefore no attribute for species exists. The database table 'enzyme_tab' is refined with a *semantic table definition* to contain mouse data. Refining a table this way is similar to adding an attribute to a table, which has the same value for each row.

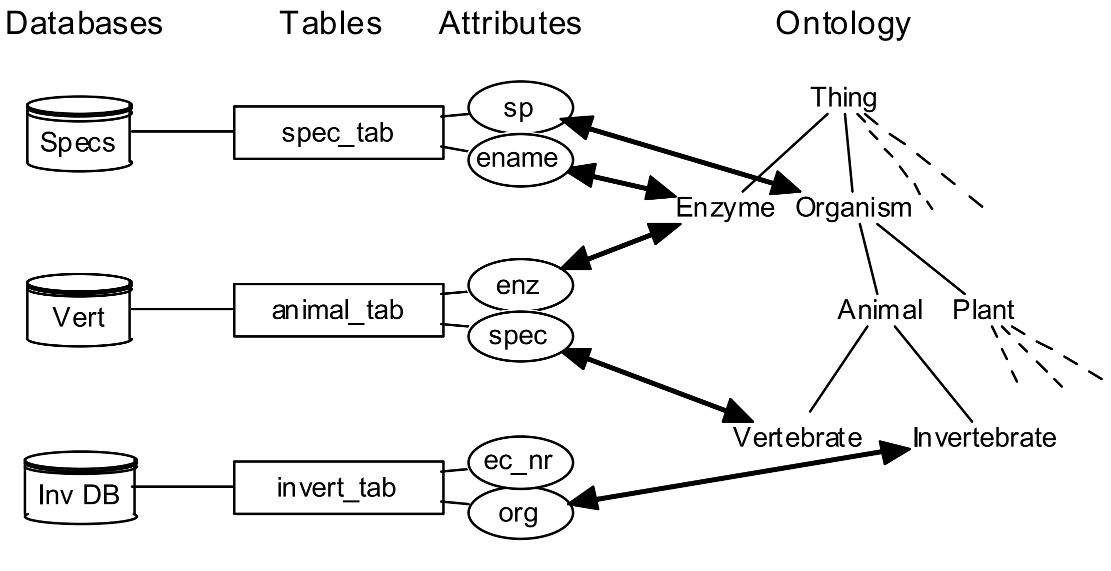


Figure 8: Databases attributes can be defined by linking them to concepts of an ontology (thick arrows). 'ename' and 'enz' are defined in this example as the same concept 'Enzyme', i.e. they both contain enzyme names. 'org' contains only invertebrates, whereas 'sp' contains both animal and plant.

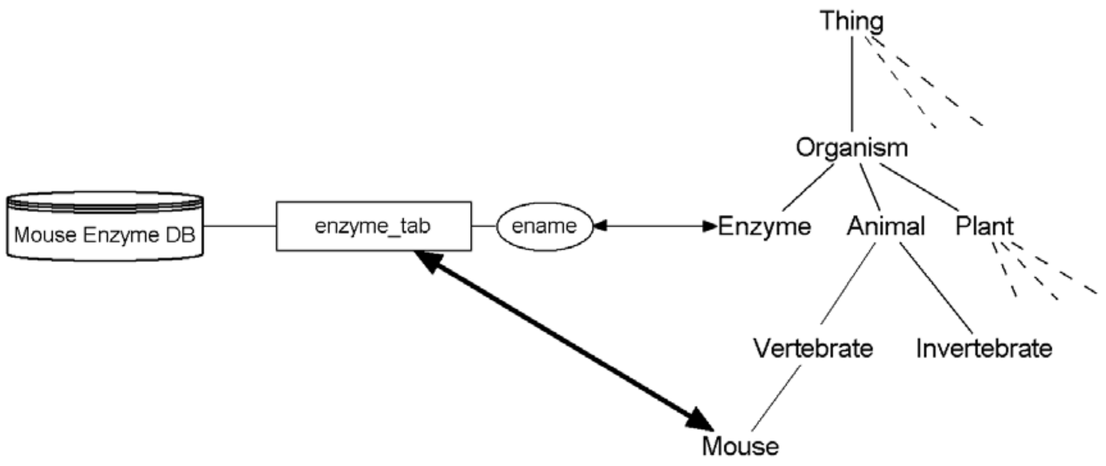


Figure 9: The content of a database table can be refined by linking the table to a concept of the ontology. The database table 'enzyme tab' only contains mouse data in this example, but does not have an attribute 'species'. For database integration purposes the table has therefore to be refined in order to indicate that it contains mouse data.

3.3.3 Attribute Value Semantics

To define the semantics of attribute values (all entries of an attribute of a database), the simplest (but not easiest) approach would be to map all entries of a database attribute to an ontology O. This would require that all database entries of an attribute have a synonymous concept c ∈ O. Since the construction of large formal ontologies is a time consuming work that takes several man-years, this approach will only work for database attributes with few distinct entries. An easier approach to semantically define attribute values is to use controlled vocabularies as datatypes for attributes in the first place, which is actually done in many existing databases.

Definition:

> If the datatype D(A) of an attribute A is a controlled vocabulary CV, i.e. D(A) = CV, the controlled vocabulary semantically defines the values of A *(attribute values definition).*

Example:

> Searching a database attribute 'organism' with the controlled vocabulary 'systematic species name' assigned as datatype for 'Mus Musculus' makes sense, since 'Mus Musculus' is a systematic species name. Searching this attribute for 'mouse' makes no sense, because 'mouse' is not an element of 'systematic species name'.

By listing synonymous concepts between controlled vocabularies in a translation list it is possible to relate entries between databases that use different terms for the same things. Concepts, which do not have a synonym counterpart, cannot be translated.

Definition:

Translation list := {(c1, c2) | c1 $\in$ CV1, c2 $\in$ CV2 and c1 is a synonym of c2}

CV1 and CV2 are controlled vocabularies.

Example:

Without a *translation list* a search in both databases in Figure 10 would only find ’mouse’ in the attribute ‘english_spec_name’ of DB1. By using the translation list it is possible to translate ’mouse’ to ’Mus musculus’ and in consequence also search the attribute ’systematic_spec_name’.

With this approach, controlled vocabularies do not need a synonym counterpart in the underlying ontology. Still, the development of translation lists between controlled vocabularies can be a time consuming work. However, pairwise translations between several controlled vocabularies exist, e.g. CAS Registry numbers (Buntrock 2001) versus EC numbers, the Gene Ontology (Ashburner et al. 2000) versus SWISS-PROT (Bairoch and Apweiler 2000), EC numbers, InterPro (Apweiler et al. 2000), etc. If transitivity and symmetry relations are used, this method can be extended for translations between more than 2 controlled vocabularies.

Some controlled vocabularies can be translated by using context specific functions.

Definition:

A *translation function tf* is a function tf:CV1 $\rightarrow$ CV2 with tf(t1) = t2, t1 = term of c1, t2 = term of c2, c1 $\in$ CV1, c2 $\in$ CV2 and c1 and c2 are synonyms. CV1 and CV2 are controlled vocabularies.

Examples:

Temperature in F can be converted by a translation function to temperature C°, upper case notation can be converted to lower case notation, etc.

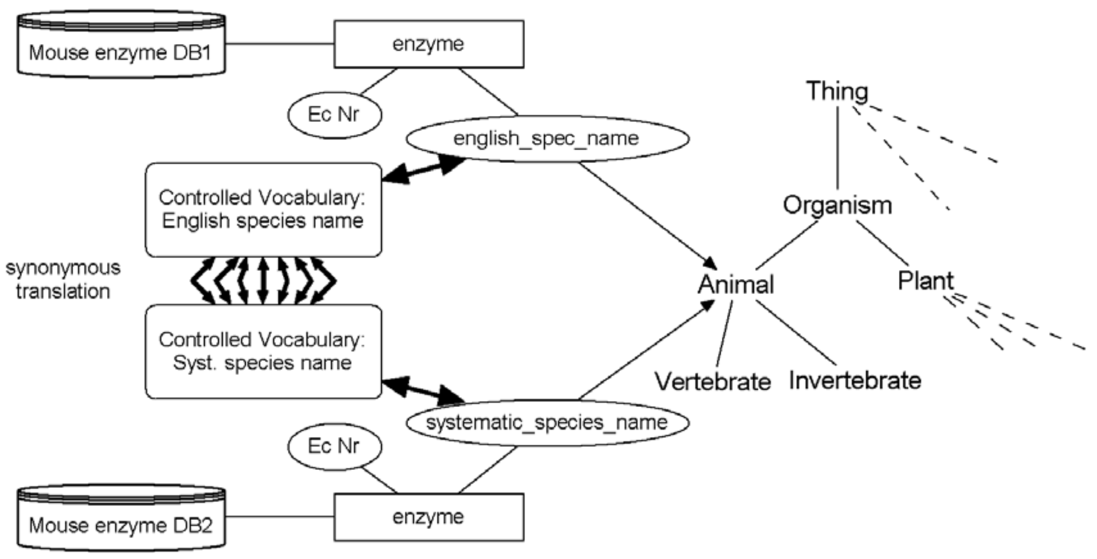


Figure 10: By mapping synonymous concepts of controlled vocabularies, it is possible to relate database entries that use different terms for the same things.

3.3.4 Database Links and Cross-references

The previous definitions can be used to derive *cross-references* for a database attribute A, i.e. the set of all attributes which share the same *semantic attribute definition* and use the same controlled vocabulary (*attribute values definition*). It is important to note that these cross-references explicitly include attributes which are mapped to sub- and super-concepts of A. Cross-references can be used to automatically generate database links as well as to automatically derive which database tables can be joined.

Example:

> In Figure 11, the query 'animal:mouse' would find the EC number of mouse enzymes. By using semantic cross-references, a system can automatically generate links to other database tables that contain further information about EC numbers. In the example additional information can therefore be found in the 'enzymes' database.

Semantic cross references go further than "ordinary" referential integrity constraints since they can be applied between databases, and because all semantic cross-references can automatically be generated once a database has been semantically defined.

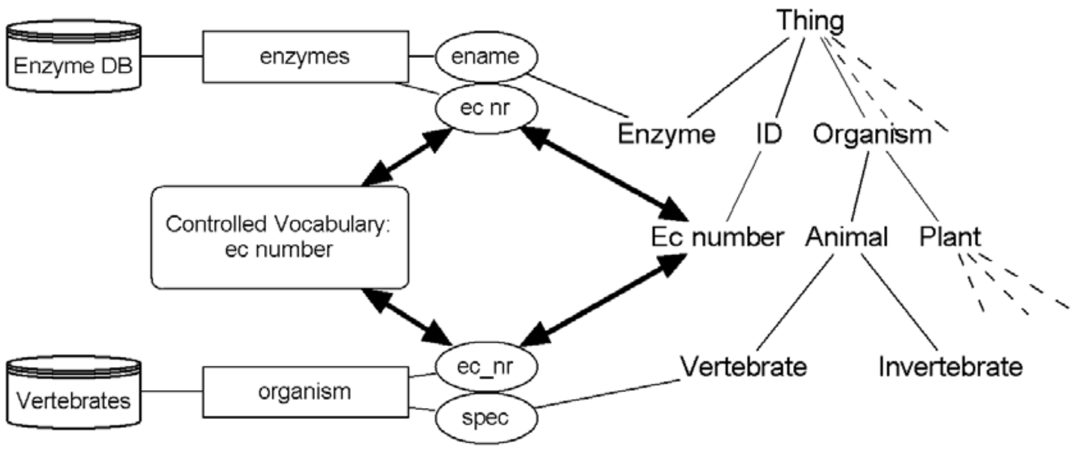


Figure 11: Database attributes which are defined as the same concept and share the same controlled vocabulary as their domain can be used for cross-referencing between database attributes.

4 Requirements Analysis

In the previous chapter, principles and methods for semantic database integration were introduced independent of their implementation. These principles are well suited to be used within existing database integration systems such as SRS (Etzold et al. 1996, Etzold and Verde 1997, Zdobnov et al. 2002) or IBMs DiscoveryLink (Haas et al. 2001).

In this chapter, the requirements for the implementation of a semantic database integration system SEMEDA (Semantic Meta Database) are introduced. General requirements for the integration of molecular biological databases are discussed with regard to the feasibility of their implementation in general and in SEMEDA. Further requirements specific to SEMEDA are introduced. It is also analysed how an existing system, the BioDataServer, can be used for homogeneous SQL access to heterogeneous data sources which cannot be directly accessed via JDBC, like flatfiles and web pages. Thus SEMEDA only has to deal with semantic heterogeneity.

The sequence in which the chapters “Requirements Analysis” and ”Design and Implementation” are presented might imply that this is the sequence in which SEMEDA was developed, but this was not the case. The requirement analysis and the design and implementation evolved more or less in several iterative steps. Thus the software engineering process followed, could be described best as the spiral process (Boehm 1988), although it was not followed in a formal way.

For example, being able to port SEMEDA to any relational DBMS that uses JDBC was originally a requirement. This requirement was sacrificed in order to be able to use an efficient method for "tree processing" within SQL (see below). Another example is the frontend: originally it was a requirement to be able to use "drag and drop" for ontology editing, i.e. the implementation of the frontend as a Java applet, but this conflicted with the requirement to be able to access SEMEDA via the Internet from within as many browsers (and browser versions) as possible, since Java versions differ between browsers.

In addition, at the beginning it was intended to access all data sources via the BioDataServer. After first tests, it turned out that also databases that are accessed by the BioDataServer via JDBC respond slowly. Thus, the system was extended to access relational DBMS directly via JDBC, whereas heterogeneous data sources still were accessed via the BioDataServer.

In addition, many features were added/modified after the frontend was implemented and after the first users used it to semantically define databases and to query databases.

4.1 Prototype

Requirement 1: SEMEDA should be implemented as a prototype that fulfils the purpose of semantic database integration as described in the previous chapter.

The main goal of the development of SEMEDA is to demonstrate the potential of the principles of semantic database integration.

Requirement 2: It should be possible to extend the system so that it can be used as a real world application, i.e. the system should be scaleable without being completely rewritten.

Well-established features, which already exist in other systems, like views, integration of bioinformatic analysis tools/applications, are beyond the focus of this work. Systems which already provide such functionality are for example OPM/MQS (Topaloglou et al. 1999), IBMs DiscoveryLink (Haas et al. 2000, Haas et al. 2001) and SRS. Database integration Systems like SRS were developed in many man-years by many developers, i.e. for practical reasons this prototype does not have to implement all features which are generally useful in database integration systems. Nonetheless, all features that are required to demonstrate the main principles of Semantic Database Integration should be implemented and the architecture should allow that general features useful for database integration systems can be added.

Requirement 3: The architecture should be flexible, i.e. it should be possible to incorporate new ideas and principles as easily as possible.

The principles for Semantic Database Integration as introduced in the previous chapter are new and some were actually developed during the implementation process. Therefore the System had to be implemented in a way that it could integrate new methods as seamless as possible.

Requirement 4: The system should be modular in a way that functionality and components of the System can be reused for the development of related applications.

Examples for applications which might be implemented based on one or more components of SEMEDA are discussed in section 11.2.

4.2 General Requirements

Requirements for the Integration of Molecular Biological Databases in general exist. In (Karp 1995) several requirements and assumptions about molecular biological databases and integration systems are listed (see Table 1). Since some of the requirements contradict each other, these cannot be met at the same time.

In order to process multidatabase queries (Requirement 5 and Requirement 7) and querying source databases “timely” (Requirement 6), all relevant databases would have to be able to answer complex declarative queries via Internet (Assumption 3).

However, as (Karp 1995) also mentions, most source databases do not support “complex queries” and in many databases even attributes exist which cannot be searched.

In addition, the methods by which databases can be searched via the internet vary widely: some databases can be searched by regular expressions, some can only be searched case sensitive, whereas others can only be searched case insensitive. In addition, some databases discriminate between data types and allow that datatype specific operators are used (for example the < operator on integer values), whereas others do not discriminate between data types.

Therefore the BioDataServer supports the “smallest common denominator” for database queries, i.e. in a timely manner only “one step accessible” attributes can be searched and only the = operator is supported, i.e. for example it is not possible to search all integer entries in a database which are smaller than a certain value. Due to these facts and due to the performance issues intrinsic to database mediation as discussed in section 2.3.3 “timely” multidatabase queries which query several data sources in one step in real time cannot be implemented. Therefore Requirement 5, Requirement 6 and Requirement 7 cannot be met at the same time. Therefore in section 11.2.1 an architecture for “database mirrors” which enables multidatabase queries at the price of older data is described.

However, Requirement 8 - Requirement 10 are feasible.

Table 1: Requirements and Assumptions for Molecular biological database Integration (Karp 1995).

Requirement 5	Users must be able to issue complex declarative multidatabase queries.
Assumption 1	Write access to member DBs is not required by most users, and will be provided by special "backdoor" mechanisms.
Requirement 6	Updates to member DBs occur frequently (roughly every day), and users place high priority on timely access to the newest data.
Assumption 2	The schemas of member DBs change quickly --- on the average of two or three times per year.
Requirement 7	Users should not be forced to circumscribe their queries in advance to a relatively small number of DBs (on the order of one or two dozen DBs).
Requirement 8	Special high-level tools must be provided for handling the rapid pace of schema change (multiple changes across all member DBs).
Assumption 3	All relevant DBs can answer complex declarative queries via Internet

A*ssumption 4*	Sophisticated user interfaces will emerge to help biologists compose complex queries
Requirement 9	Database heterogeneity is here to stay, at a variety of levels. We need powerful tools for managing heterogeneity.
Requirement 10	We should not require users (who formulate queries) to know of the existence, or the physical location, or the access mechanisms, for every DB that is relevant to their query.
Requirement 11 (Williams 1997)	Mechanisms for automatic generation of links between databases are needed.

Another requirement is that mechanisms for automatic link generation between databases are needed (Requirement 11). Due to the lack of a systematic linking mechanism, even up to date database integration systems such as SRS (Etzold et al. 1996) (Zdobnov et al. 2002), KEGG (Kanehisa 1997a, b, Kanehisa et al. 2002) and PEDANT (Frishman et al. 2001, Mewes et al. 2002), only link or merge the "most important" database attributes. Therefore, as already mentioned the degree of interlinking in these database integration systems is sub-optimal.

Whereas the requirements listed in this section are requirements, which apply to most database integration systems, in the subsequent sections specific requirements for SEMEDA are given.

4.3 Using the BioDataServer to Access Data sources

Although at present most molecular biological databases are implemented using DBMS, which enable structured access methods like JDBC or ODBC, most database integration solutions use flatfiles or other proprietary methods for data exchange. For example in SRS the source database has to export the data and the integration system has to import the data. Both for exporting and importing, source database specific scripts have to be used.

The situation that the structured access methods are rarely used seems to be paradox and has mostly historical reasons (see chapter 2), but still at present time most database owners would not grant JDBC access to their databases. The main reasons for this are security concerns although nowadays all major DBMS have mechanisms to restrict user rights. Usually it is possible to restrict read only access to a few tables, and to hide table attributes and even tablerows by using views. By using user specific grants, it is possible to grant different privileges on the user level. In addition, JDBC type 3 drivers have recently become available for all major DBMS http://industry.java.sun.com/products/jdbc/drivers. JDBC type 3 drivers usually support encryption and provide additional new methods (besides user grants) to deny write access.

To be able to access data sources that cannot be accessed via JDBC, the BioDataServer of the MARGBench system can be used. The BioDataServer is a mediated integration architecture that provides SQL access to heterogeneous data sources. The BioDataServer was developed within the MARGBench (Modeling and Animation of Regulatory Gene Networks) project.

Requirement 12: Data sources, which can be accessed directly via JDBC, should be accessed directly by JDBC. Other data sources should be accessed via the BioDataServer of the MARGBench system.

Subsequently a brief overview of the MARGBench is given, and performance and special features of the BioDataServer are discussed as far as they are relevant for SEMEDA. The information in this section is based on (Freier et al. 1999, Freier et al. 2002b) and on personal communication (Matthias Lange, IPK Gatersleben, Germany, January 2002).

The BioDataServer is a system, which provides wrapper based SQL access to heterogeneous distributed data sources. As already discussed, Molecular Biological Data sources use many different interfaces: whereas some databases provide SQL access via JDBC or ODBC, at present most molecular biological databases can still only be accessed via form based dynamic web pages, structured (XML) or proprietary flatfile formats. The BioDataServer supports all those data sources, although the amount of work required for writing different wrapper types varies.

The main factors influencing performance of queries to distributed databases are the client side costs for data extraction, and server side costs for data provision. Table 1 compares those factors and estimates the work required for implementing adapters for different data sources. Those characteristics are intrinsic properties that are independent of the way adapters are implemented (Davidson et al. 1995).

Table 2: Comparison of costs for querying different types of heterogeneous data sources, and the work required for the implementation of adapters.

Data source	Data extraction costs at the BioDataServer	Data source side performance costs	Work for Adapter implementation
Dynamic web pages	high	high	very high
Dynamic XML files	low	high	medium
Static XML files	low	low - medium	low
Proprietary flatfiles	medium	low - medium	high
DBMS	very low	low - medium	low

Some dynamic web pages cause further download and extraction costs: some attributes cannot be searched directly within the source database, i.e. when such an attribute is queried, an adapter has to implement the access method to the attribute. This access method may require downloading and parsing all entries of an attribute, i.e. depending on the source database, querying such attributes may take several minutes up to several hours. Such attributes are said to be not *one step accessible*. The concept of o*ne step accessible* attributes is important and will be used often in SEMEDA, since only *one step accessible* attributes can be queried at a speed which is acceptable for interactive database queries.

But also remote DBMS, which can be directly accessed by SQL via JDBC, can cause high costs when tables between different DBMS have to be joined. DBMSs use indexes to improve the performance of the join operation. This is not possible in heterogeneously distributed databases: indexes on source databases cannot be used, and global indexes cannot be generated since the data is not known before it is queried. In addition, it may be necessary that all values of the attributes that are used for joining are compared, what may cause high download and extraction costs. In some situations, probe based optimisations (Shahabi et al. 2000) or cost based query optimisations based on metadata are possible (Haas et al. 2000, Haas et al. 2001), although they cannot overcome all performance issues of database mediation.

The BioDataServer was developed as a component of the data integration layer of MARGBench, a workbench for bioinformatic tools, i.e. it was important that imported data from source databases always reflect the current state of the source database. For this scenario, response times between a few seconds and several minutes (Freier et al. 2002b) are acceptable and the price for “fresh” data.

What has to be implemented to be able to use the BioDataServer?

Requirement 13: Import methods for reading database metadata from the BioDataServer have to be implemented.

Some databases consist of hundreds of tables and attributes. Although it should also be possible to add and enrich database metadata in SEMEDA manually, metadata should be imported whenever it is possible. The BioDataServer uses a proprietary protocol for retrieving database metadata for the databases that are accessible via the BioDataServer.

Requirement 14: Methods to generate and submit a global integrated database schema to the MARGBench are required.

The BioDataServer can be used by several users and applications, which may require different database schemas. Therefore, for each user a global integrated database schema has to exist which is used to model user specific views of the integrated databases. Database queries are based on these global integrated databases schemas.

4.4 Multi User Support

Requirement 15: Several users should be able to simultaneously query the data sources.

This is a prerequisite to make SEMEDA available as a web service.

Requirement 16: Several users should be able to edit the ontology and the database metadata collaboratively.

The provider of a database usually knows the content and the data formats of his database best. Therefore database providers should be able to semantically define their own databases. Therefore three groups of users with different permissions have to be differentiated:

Everybody: All users should have the permission to query databases and to browse all metadata. Confidential database information (host, port, login/password) may only be browsed by the database owner and by administrative accounts.

DB Provider: Objects, which are generated by a DB provider, should be treated as "suggested objects” which can only be released by administrative accounts. It should not be possible for other users to add objects, which are dependent on the suggested objects. For example, it should not be possible to add sub-concepts to an ontology concept, which was suggested by another user before the concept is released by an administrative account

Admins: Full permission on everything. Only few people should use administrative accounts, and only administrative accounts should have the permission to release suggested objects. After an administrative account has released objects, which were suggested by a database provider, the database provider should no longer be able to edit these objects. SEMEDAs administrative accounts are only used for issues related to semantically defining databases and making database available via SEMEDA. Therefore, in addition to SEMEDAs user groups, accounts for managing the resources that are used to implement and deploy SEMEDA are needed. Such accounts (UNIX root accounts, administrative database accounts etc.) are not integral part of SEMEDA and may differ between different deployments of SEMEDA.

4.5 User Interface

SEMEDA should have mainly two user interfaces, one for querying databases and one for semantically defining databases.

4.5.1 Querying Databases: SEMEDA-query

The query interface should give an example how the different semantic database definitions introduced in chapter 3 can be used to semantically integrate source databases at query time.

Requirement 17: Both the query interface and the interface for semantic database definitions should be accessible via the Internet.

This is especially important for the query interface since SEMEDA should be available as a web service. Potential users should be able to access SEMEDA from within a web browser as it is the case in successful database integration systems like SRS (Etzold et al. 1996, Etzold and Verde 1997, Zdobnov et al. 2002) and PEDANT (Frishman et al. 2001).

By making the interface for semantic database definitions available via the Internet, database providers can semantically define their own databases.

Requirement 18: The query interface should not rely on any other relation type than the "is a" hierarchy in the ontology.

The common smallest denominator of existing ontologies is the existence of an "is a" hierarchy (see Table 7). The query interface might use other relation types than the "is a" hierarchy when an ontology provides them. However it should not require the existence of other hierarchies for database queries, since this would make it a priori impossible to use many of the existing ontologies from within SEMEDA. The mechanisms of semantic database integration as introduced in the previous chapter do not make use of any other hierarchies than the "is a" hierarchy.

Requirement 19: The query interface should hide the ontology from the user.

One lesson that had to be learned in the GALEN project (Rector et al. 1998), was that the data structure of ontologies could not be understood by most medical doctors.

4.5.2 Semantically Defining Databases: SEMEDA-edit

The Use Case diagram in Figure 12 gives an overview of the operations, which are required to add databases to SEMEDA, and summarises some of the requirements given in the previous sections.

The Use Cases are simplified, and most of the Use Cases displayed in the diagram actually consist of several Use Cases: For example the Use Case "Edit ontology" consists of adding, deleting and modifying concepts, relations and relation types.

Most methods, which are required for each Use Case are more or less self-explanatory. Therefore it is sufficient to state here that the Use Cases had to be implemented in a way, that the semantic database definitions as defined in Chapter 3 can be applied. In addition it has to be possible to query data sources once the database metadata has been imported and databases have been semantically defined.

Requirement 20: Adequate methods for browsing and displaying the hierarchical structure of ontologies are required.

Ontologies can become big and thus hard to survey. Ontologies have typically a few hundred up to 1 million concepts (see Table 7), i.e. more concepts than can be displayed at the same time at a computer screen. Therefore adequate methods for browsing the ontology are required.

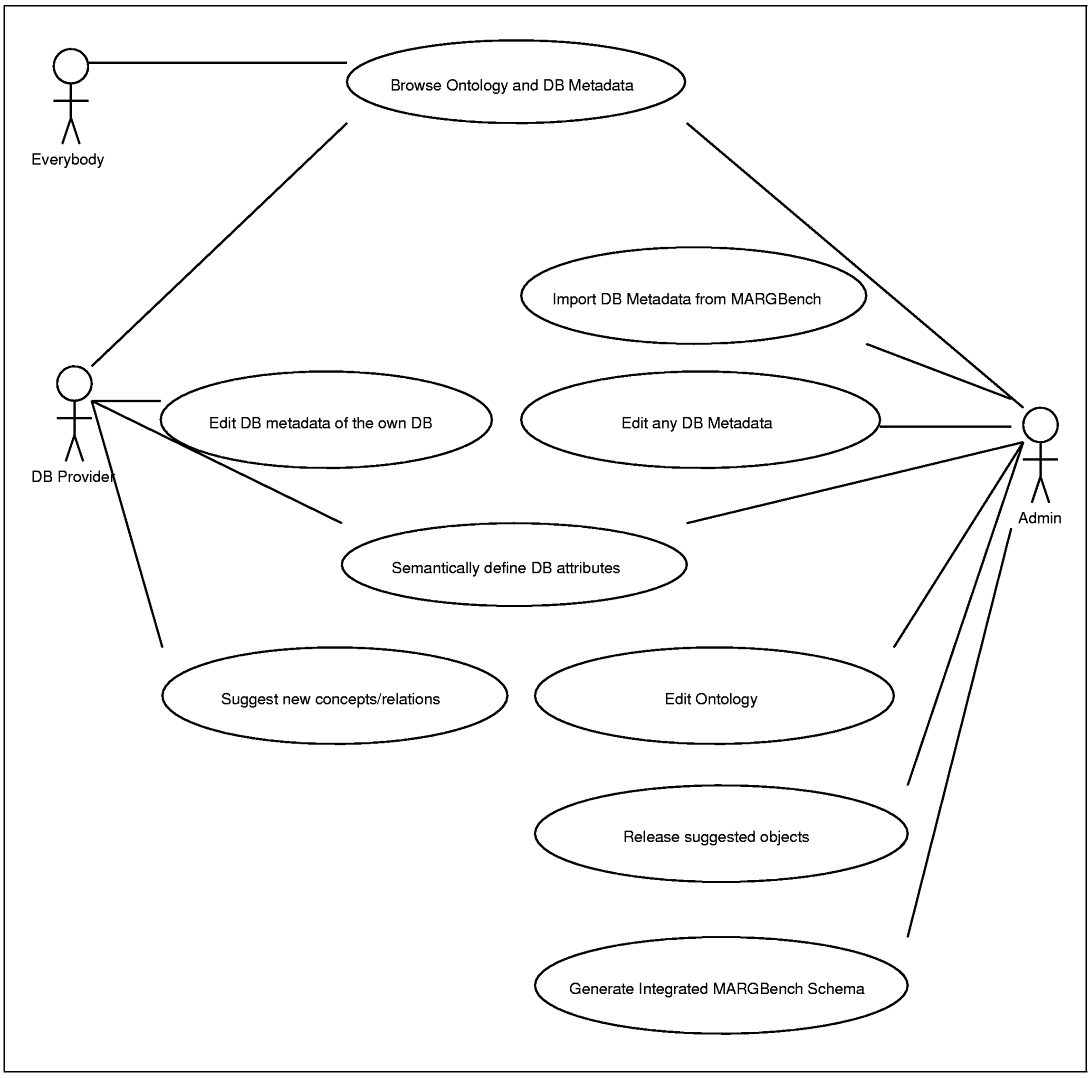

Everybody
Browse Ontology and DB Metadata
Import DB Metadata from MARGBench
DB Provider
Edit DB metadata of the own DB
Edit any DB Metadata
Admin
Semantically define DB attributes
Suggest new concepts/relations
Edit Ontology
Release suggested objects
Generate Integrated MARGBench Schema

Figure 12: Use Cases of the operations, which are needed to add databases to SEMEDA. The Use Cases are displayed from top to bottom in the sequence in which they have to be used to add a database.

4.6 Tool Interface

The database metadata in SEMEDA and the semantic database definitions may also be useful for other applications. For example the database metadata might be used to support writing global integrated MARGBench Schemata (see previous section). For example, a user who models an integrated MARGBench schema might need to find all data sources, which contain Enzyme numbers, i.e all database attributes which contain information about a certain concept.

Requirement 21: Tool interface which supports: a) searching for a concept in SEMEDA b) retrieving database metadata for a database c) retrieving database attributes which are defined as a given concept.

In addition, several other methods may be useful for other applications. Therefore it should be possible to add further methods when they are required.

5 Methods

Whereas chapter 3 describes principles for semantic database integration, in chapter 4, the requirements for the database integration system are derived. In this chapter, the methods that are suitable to implement the introduced principles are described as a prerequisite for their realisation in SEMEDA.

To do so, an appropriate data structure for storing ontologies, database metadata and the semantic definitions as described in Chapter 3 is developed. Subsequently, it is discussed how this data structure can be edited and queried.

5.1 Data Structure

An Entity-Relationship (ER) diagram (Chen 1976) was chosen for the representation of the data structure, since it allows to represent the data structure in an implementation independent manner, although it is relatively easy to derive an appropriate table structure for a relational database, based on an ER diagram.

Figure 13 shows the data structure that is appropriate for storing ontologies, database metadata and the semantic definitions as described in chapter 3. The left part of the ER diagram contains the parts, which are relevant for representing the ontology. Concepts (NODE) are connected by relations (EDGE), and relations are typed (EDGE_TYPE). Concepts, relations and relation types belong to an ONTOLOGY (Element of), which allows storing several ontologies. For ontologies that are imported, the URL where they were imported from can also be stored. Concepts (NODE) have a name and a definition. In addition, a subclassifying criterion (Schulze-Kremer 2002) can also be stored. The type of a relation (EDGE_TYPE) is defined by its name and its algebraic relational properties such as transitivity, reflexivity, symmetry (as defined in section 3.2).

The right part of the schema shows the structure to handle database metadata: Each database (DB) can contain several tables (DB_TABLE), and each table can contain several attributes (DB_ATTRIBUTE). The datatypes that attributes can have (String, Integer, Boolean etc.) are stored in the table INTERNAL_DATATYPE. In this table, also different labels used for equivalent datatypes are stored, i.e. JDBC datatypes are mapped to BioDataServer datatypes and Oracle datatypes. In the database table (DB) relevant information about the database is stored: the name of the database, the type of the DBMS, technical data needed to access the database (host, port, login password etc) and whether the database should either be directly accessed or via the BioDataServer. The database table (DB_TABLE) stores the name of a table and also data, which is relevant for generating BioDataServer user schemata (see section 4.3). Metadata of database attributes are the name, the fact if an attribute is a database key and if it is *one step accessible* (see section 4.3). To know if an attribute is *one step accessible* is important, since it determines how fast a database can be queried, i.e. if an attribute responds fast enough for interactive database querying. In addition functional dependencies between attributes are stored. A database attribute is functionally dependent from another, if its entries are uniquely determined by the other. The information about *functional dependencies* is important for generating an integrated user schemata for the BioDataServer (which in turn is a prerequisite for using the BioDataServer to query data sources).

Semantic database definitions are represented by the relations DEFINE_ATTRIBUTE, REFINE_TABLE and ATTRIBUTE_VALUE_DEFINITION. DEFINE_ATTRIBUTE links database attributes to ontology concepts and thus semantically defines them (see section 3.3.1). In an equivalent way, REFINE_TABLE refines the content of database tables (see section 3.3.2). The Table DATATYPE defines which vocabulary is used for which database attribute (see section 3.3.3). Whereas the table DATATYPE_REPRESENTATION stores generic datatypes such as String, Integer or Boolean, the table DATATYPE stores which vocabulary is used, i.e. it defines the values of a database attribute more precisely (Enzyme Number, Systematic Species Name, CAS registry number etc.).

ONTO_USER is important for multi user support (see section 4.4), and contains a list of all users. Therefore it stores all user related information such as login, password, session timeout, activation state of user accounts etc. For reasons of conciseness, not all attributes of the entities in Figure 13 are displayed. All tables that require support for multi user editing (i.e. all tables but DATATYPE_REPRESENTATION, ONTO_USER and CONFIGURATION) have the attributes OWNER and EDITED_BY. OWNER_FK is a foreign key to the ONTO_USER who suggested an entry. In the ER diagram (Figure 13) this OWNER_FK is represented by the BELONGS_TO relations. Until the OWNER_FK of an entry is not NULL, the entry is treated as "suggested" (see also section 4.4). In every insert or update statement, the attribute EDITED_BY is set to the user name of the user who tries to execute the insert or update statement. Thus, database check triggers can evaluate if the user has the permission to execute the respective data manipulation and if necessary raise an error and prevent the modification (see section 5.2). In addition, the tables ONTOLOGY, DB and DATATYPE have an attribute LOCKED_BY as a foreign key to ONTO_USER. Setting this attribute to an ONTO_USER locks the respective ontology, database or vocabulary for other users, i.e. only the user who set the lock can edit elements of the respective ontology, database or vocabulary, until the lock is released by setting LOCKED_BY to NULL.

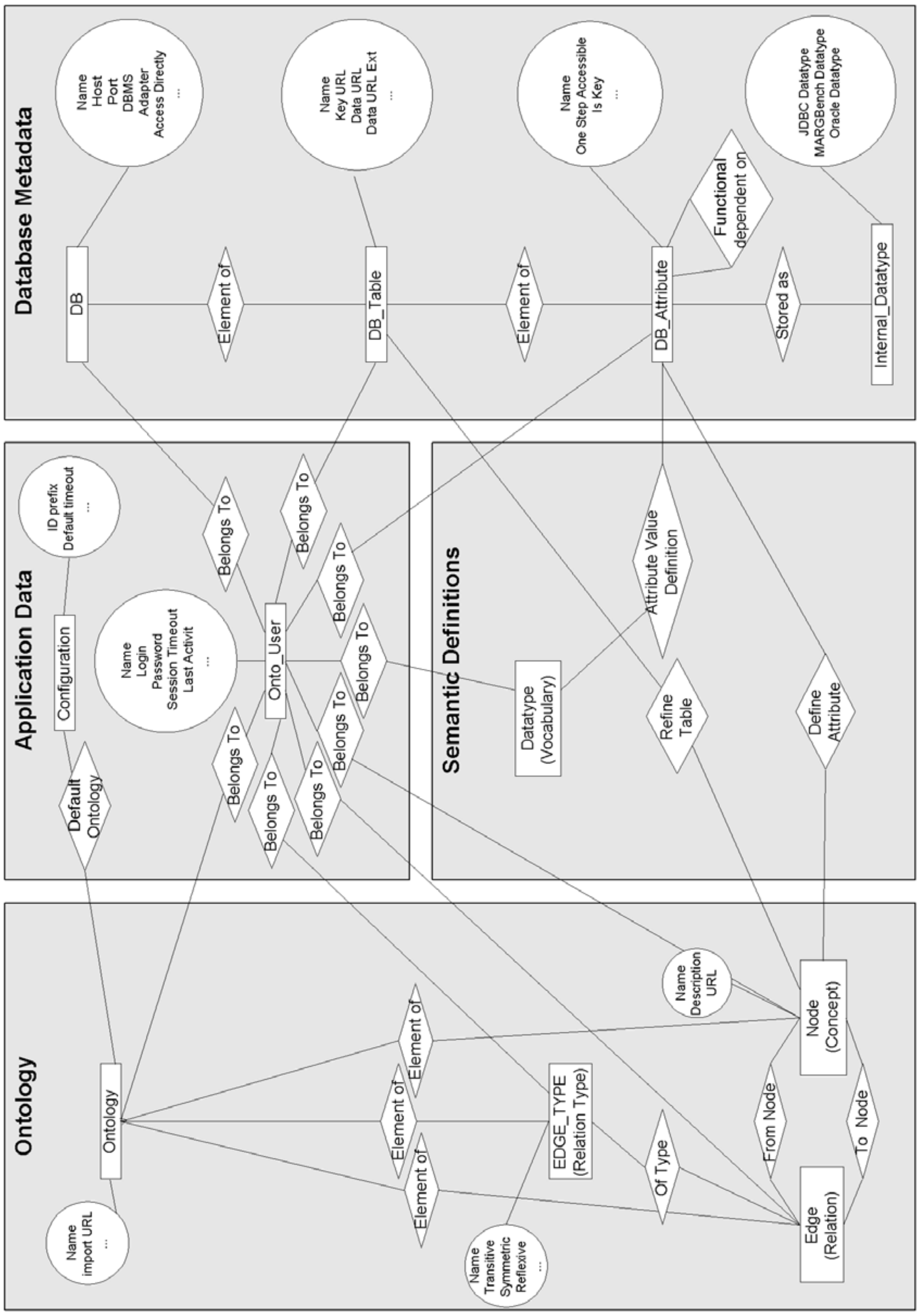


Figure 13: ER-schema of SEMEDAs backend. The diagram is further explained in the text.

5.2 Editing Data and Multi User Support

Based on the data structure introduced in the previous section, the algorithms for editing the data are mainly self-explanatory. With regard to the implementation of the data structure on a relational database, modifying data relates to SQL insert, update and delete statements. For example, to add a concept to the ontology, an entry is added to the entity NODE. In order to relate two concepts of the ontology, an ontology relation is added by an insert statement to the entity EDGE.

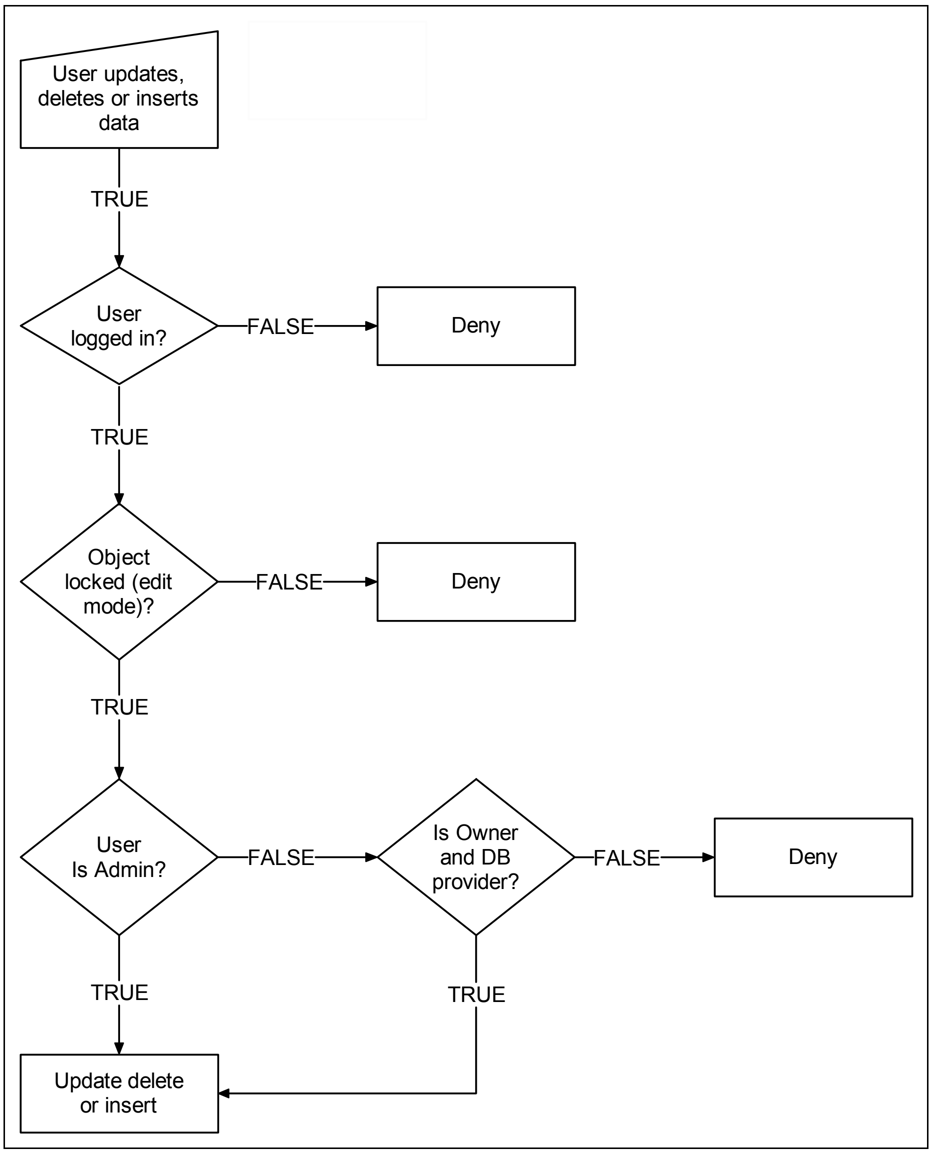


Figure 14: Permission checks required for multi user support.

For multi user support, it has to be assured that whenever a user tries to modify data, he has the right grants to do the data manipulation. Based on the requirements for multi user support (see section 4.4) several checks have to be performed before a user is allowed to modify data (Figure 14). First of all, the user has to be logged in, and the user session should not have timed out. When this is true, it has to be checked if the account is still activated. As described in the previous section, locks had to be implemented. This is necessary to prevent conflicts that arise when two users edit the same data elements at the same time. When all those checks have succeeded, administrative accounts may modify the data, whereas database providers may only update or delete elements that they suggested themselves.

5.3 Querying the Data Structure

Based on the data structure of Figure 13, many different possibilities to query SEMEDA exist. Examples for common queries are:

1) Get the semantics of a given database attribute (see section Attribute semantics, 3.3.1).
2) Get all database attributes of a given database table.
3) List all database attributes that have the same semantic attribute definition and use the same vocabulary (to generate Database Links and Cross-references, see section 3.3.4).
4) Get all sub-concepts of a given ontology concept (this is for example useful to display the hierarchical structure of the ontology).

With regard to the data structure displayed in Figure 13, the queries 1-3 require selecting data from the respective entities, and connecting those elements by traversing the relevant relations. For example in query 1, the respective entry from the entity DB_ATTRIBUTE is selected, and by following the relation DEFINE_ATTRIBUTE, the semantics of the database attribute is found. With regard to the implementation of the ER schema as a relational database, those queries can be formulated using SQL 92 queries.

Query 4 cannot be formulated using standard SQL queries, since it requires traversing the entities NODE and EDGE recursively via the relations FROM_NODE and TO_NODE. However, by using recursive SQL queries (Wang and Zaniolo 1998), which were defined in SQL 99, such queries can also be handled within relational databases, if the standard is fully implemented.

6 Design and Implementation

6.1 System Architecture

6.1.1 Overview

Figure 15 shows how SEMEDA and the BioDataServer of the MARGBench interoperate. The BioDataServer is mainly used for SQL access via JDBC to heterogeneous data sources. Thus SEMEDA could focus on semantic heterogeneity, whereas the BioDataServer provides homogeneous SQL access to heterogeneous data sources (flatfiles, web pages, XML files etc). SEMEDA provides mainly two user interfaces: SEMEDA-edit for semantically defining databases following the principles described in chapter 3 and SEMEDA-query for querying databases. SEMEDAs query interfaces guides the user to the relevant databases for his query, offers appropriate HTML forms for querying the data sources and converts the form based queries to appropriate SQL queries, which are subsequently processed by the BioDataServer or directly by the source databases if direct SQL access is granted by the source databases.

SEMEDA itself was implemented as a 3-tiered system consisting of a relational database (Oracle 8i backend) and JSP 1.1 (Java Server Pages) as the middle tier, which dynamically generates the HTML frontend (Figure 16). The backend (relational database) stores the ontologies, database metadata and the semantic definitions of the databases, whereas the middle tier connects the HTML frontend to the database by mapping and validating the HTML form based user actions to appropriate SQL statements. Much functionality of SEMEDA was implemented using Oracles PL/SQL functions, procedures and triggers. PL/SQL is a proprietary procedural programming language, which is an integral part of Oracle.

By using this architecture the requirements introduced in the previous sections can be met: The system is modular, extensible and flexible (see Requirement 1, Requirement 2 and Requirement 4), i.e. the relational backend could be (re)used independently of the JSP middle tier component. In order to keep the functionality independent of the JSPs, much functionality was implemented as PL/SQL procedures, and the JSPs were mainly used for user session tracking and for dynamic generation of the HTML frontend. In addition, keeping the functionality in the relational DBMS improves performance (see section 6.1.3), which is crucial for efficient “tree generation”. A disadvantage of using Oracles PL/SQL programming language is that the system cannot so easily be ported to other DBMS, although it should be possible in a straightforward manner to port the relational backend and the data to other DBMS.

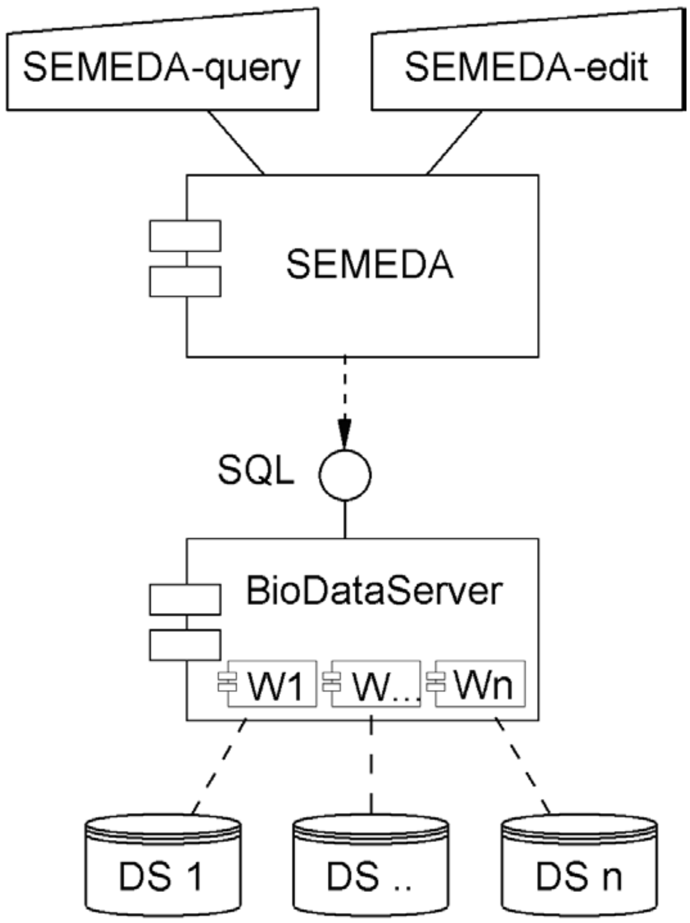


Figure 15:Interoperation of SEMEDA and the BioDataServer. The interface SEMEDA-edit is used to semantically define databases, whereas the interface SEMEDA-query is used to query databases. The BioDataServer provides SQL access to the mediated data sources. W1, ..., Wn are the wrappers of the data sources DS 1, ..., DS n.

Further requirements closely related to the system architecture are accessibility via the Internet (Requirement 17) and the provision of a tool interface to SEMEDA (Requirement 21). Accessibility via the Internet is achieved via the dynamically generated HTML frontend, whereas the tool interface is implemented by granting JDBC read access to the relevant parts of the databases and to relevant PL/SQL procedures (see section 6.4).

The BioDataServer was used for data access (Requirement 12) using its JDBC interface from within the JSP middle tier component.

Further advantages of this architecture are that data (ontologies and database metadata) can be consistently stored independently from the application and that data can also be retrieved or imported by using the various built in interfaces and tools of the DBMS. In addition this design keeps the application scalable, since both the relational backend (Oracle) and the middle tier (JSP) can be deployed on different systems and thus are both scalable.

The other requirements introduced in the previous section are more or less independent of the system architecture. In the subsequent sections of this chapter it will be explained how the different components of SEMEDA were implemented to meet the requirements introduced in the previous chapter.

6.1.2 Scenario

In this section it will be described how the main components of SEMEDA and the BioDataServer interoperate to provide the required functionality. In order to query databases via SEMEDA, database metadata of the relevant databases has to be incorporated to SEMEDA and semantically defined. Subsequently databases can be queried.

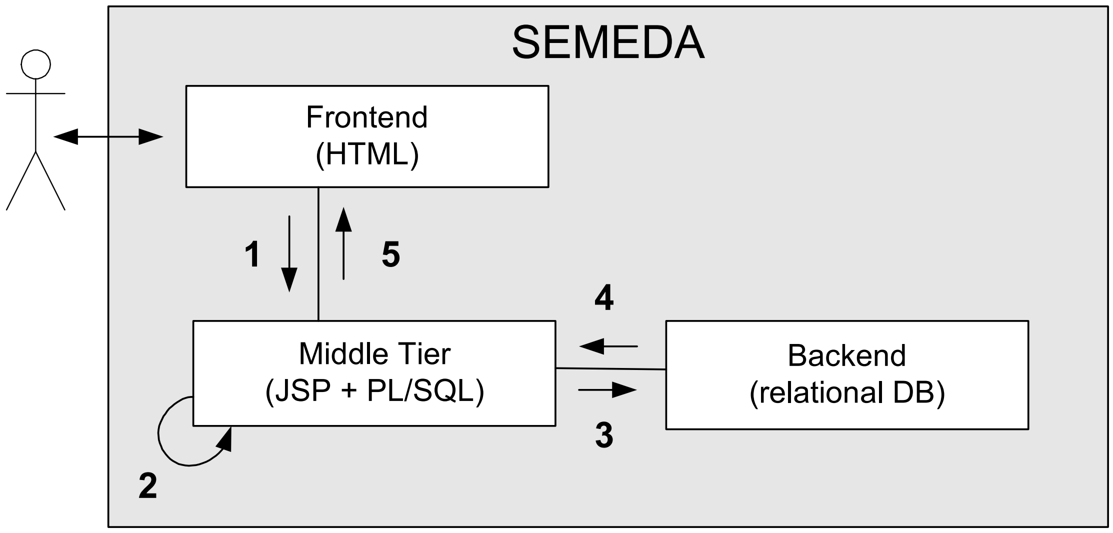


Figure 16: Editing ontologies and database metadata: Collaboration of SEMEDAs components.

Figure 16 shows the scenario for editing data in SEMEDA. Data are ontologies, database metadata and vocabularies.

1.) User inserts, updates or deletes data in SEMEDA via the HTML frontend using HTML forms. For example, the user wants to add a new ontology concept or to semantically define a database attribute. The HTML form data is sent to the middle tier.

2.) Using JSP and PL/SQL procedures, it is validated if the user has the required permissions to perform the required operation. In addition, it is checked that the data does not create inconsistencies in SEMEDA.

3.) SEMEDAs backend (relational DB) is updated.

4 + 5.) The success of the operation is reported to the user and the data is refreshed in the frontend.

Once databases are semantically defined in SEMEDA, an integrated BioDataServer schema for all databases, which are semantically defined in SEMEDA, have to be submitted to the BioDataServer. After this step, databases can be queried via SEMEDAs query frontend (SEMEDA-query). Figure 17 shows how the different components of SEMEDA and the BioDataServer interoperate to query databases:

1,2): SEMEDA guides the user based on semantic attribute definitions (section 3.3.1) and semantic table definitions (section 3.3.2) to appropriate database tables. For example if the user wants to search databases by EC Numbers, he can choose a database table from a list of all database tables which contain attributes which are semantically defined as EC Numbers.

3.) User submits the database query by using a source database specific query form. In this query form the table attributes are labelled using concept names besides the attribute names of the source database. In addition, the vocabulary of the data-sources is displayed by using the attribute value semantics (see section 3.3.3).

4.) The JSP processes the user query and translates it to an appropriate SQL query, which is sent to the BioDataServer via the JDBC, interface of the BioDataServer.

4a.) If the source database can be directly accessed via JDBC, the query is directly sent to the source database.

5.) Based on the integrated BioDataServer database Schema, the query processor determines which adapters have to be used to process the query.

6.) The appropriate subqueries will be generated by the subquery builder and sent to the adapters. The Integration layer sends the SQL query to the respective adapter.

7.) The adapters translate and send the subqueries to the appropriate databases.

8.) The databases send the replies to the adapters.

9.) The adapters send the replies to the "result set integration", where the different formats of the replies will be unified. From databases which provide HTML files as replies, the relevant information will be extracted. Finally the result sets will be merged to a global result set.

10.) From the “Integration Layer” the BioDataServer returns these replies to the query processor.

11, 11a) The result set is returned to the middle tier component of SEMEDA.

12.) The JSP returns the results to the user. Along with the results, links to other relevant databases are displayed which are generated by using semantic cross-references (see section 3.3.4).

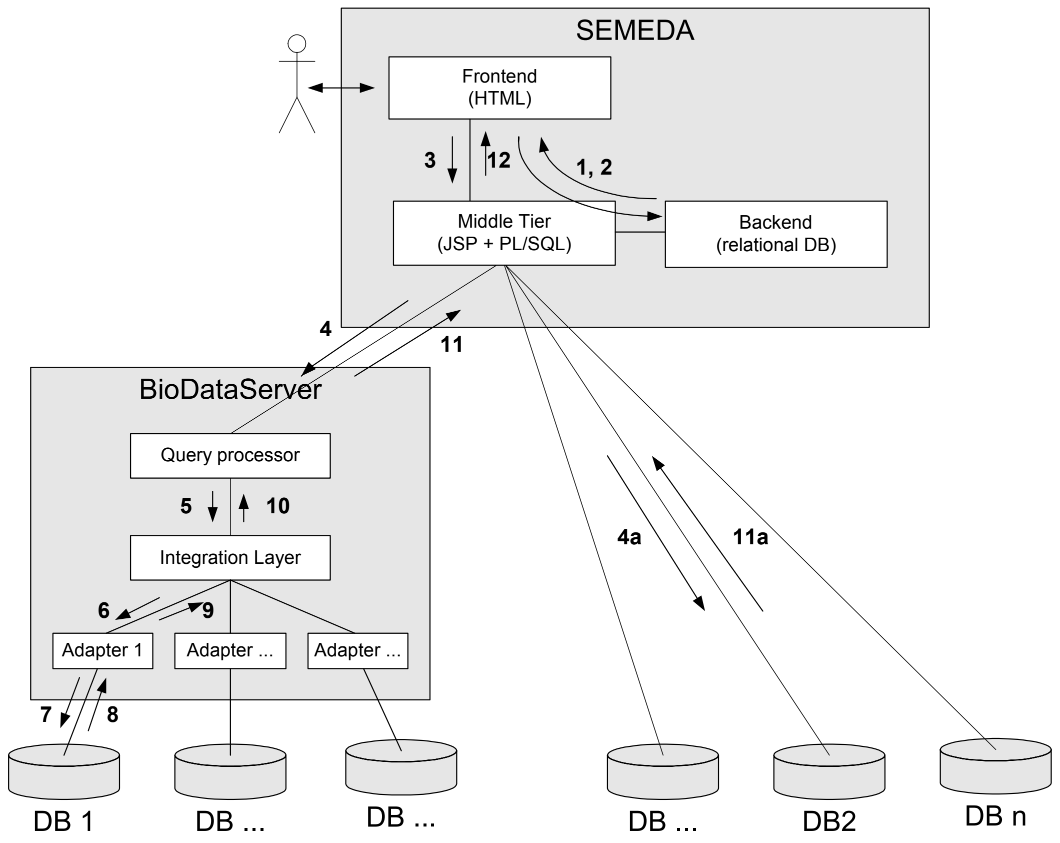


Figure 17: Querying databases: collaboration of the components of SEMEDA and the BioDataServer.

6.1.3 Backend

The data structure introduced in chapter 5 was implemented using Oracle 8i as a relational DBMS system, although any other database system could also have been used for storing the data. Whereas often the "application logic" is located in the middle tier, in SEMEDA it was partly implemented in JSP and partly by using DBMS features such as constraints and PL/SQL (trigger, procedures, functions). This results in better performance (fewer JDBC calls, query optimisation by the DBMS) and has the advantage that much of the implemented functionality can be used by other applications without having to use the middle tier. On the other hand, the use of DBMS features makes it more difficult to port SEMEDA to other DBMS. However, this does not affect raw data exchange of ontologies and database metadata between SEMEDA and other systems.

Database identifiers are often considered to be an unimportant issue, but in order to facilitate data- and ontology exchange, finding a good solution was crucial and several requirements had to be met. IDs should be referable from external applications. Therefore they should not change and be unique, also between SEMEDA implementations. Thus IDs do not have to be adjusted when an ontology is transferred between two SEMEDA implementations or with other ontology applications. The IDs should further be able to cope with IDs of other ontologies to facilitate import of ontologies (a number can be stored as a string, a string cannot easily and unequivocally be stored as a number).

In order to account for these requirements, it was decided to use a prefix, which is unique for each SEMEDA deployment. Such a prefix could be similar to a stock symbol. For example, concepts generated at a SEMEDA version deployed at the Resource Center Primary Databases might use the prefix RZPD, followed by an integer. Such concept IDs should not be modified when ontologies or metadata are traversed between SEMEDA implementations. Also, when ontologies are imported from other systems, the original IDs should be used and only when necessary be extended by a unique, source ontology specific prefix. Thus SEMEDAs identifiers are similar to Gene Ontology accession numbers, but whereas all GO concepts use the same prefix “GO:”, independent of the tools that are used to edit ontologies, SEMEDAs prefixes are specific for each installation of SEMEDA. To make sure that these prefixes remain unique is rather a “political” than a technical issue: recently several different ontologies are being developed, and a mechanism for assigning such prefixes should be negotiated between the research groups involved. Technically this could be solved using a web page that automatically generates a prefix after successful registration of a user.

To improve the consistency of the data stored in SEMEDA, database integrity constraints were used. Some inconsistencies cannot be caught by using constraints. When this was the case, PL/SQL "check triggers" (Oracle-Corp. 1997) were implemented. These and the functionality of the middle tier are described in the subsequent section.

6.1.4 Middle Tier

The middle tier indirectly connects the frontend with the backend, i.e. maps HTTP GET and POST requests to the appropriate SQL/DML statements and PL/SQL procedures via JDBC. In addition, JSP is used for session tracking.

6.1.4.1 PL/SQL Procedures, Functions and Triggers

PL/SQL stands for "Procedural Language extensions to SQL." PL/SQL is available primarily as an "enabling technology" within Oracle, i.e. it does not exist as a standalone language. It can be used in the Oracle relational database, in the Oracle Server, and in client-side application development tools, such as Oracle Forms. PL/SQL is closely integrated into Oracles SQL implementation and adds programming constructs that are not native to SQL. PL/SQL allows combining SQL statements with "standard" procedural constructs such as iteration, loops and if-then-else statements.

PL/SQL can be used to write procedures, functions and triggers. Procedures, functions and triggers are program units, which can execute one or several blocks of code. Procedures and functions can have parameters and only differ in the fact that functions can return a value. Triggers are blocks of code, which are executed on predefined events: such events can for example be SQL insert, update or delete statement on predefined tables or table columns.

Therefore, triggers can be used to check if an insert, update or delete statement may be executed, and if necessary a trigger can raise an error which prevents the execution of the code ("check triggers").

PL/SQL procedures, functions and triggers can be called via the various interfaces Oracle provides. Whereas SEMEDAS JSPs access the procedures and functions via JDBC, other tools could be implemented which access SEMEDAs PL/SQL functions and procedures via other interfaces. Therefore, whenever it was possible, the application logic was implemented in PL/SQL. Thus SEMEDA remained modular in a way which would allow to reuse SEMEDAs application logic without requiring the use of JSP or other components of the frontend. The disadvantage of this architecture is that this makes SEMEDA more difficult to port to another DBMS than Oracle.

In total about 150 PL/SQL functions, procedures and triggers have been implemented. Table 3 lists the most important procedures, functions and triggers of SEMEDA and describes their functionality. These functions could also be used by external tools (see section 6.4).

The details of the implementations of the procedures, functions and triggers will not be discussed any further. Only the mechanism of "tree walking" which is used in several procedures and functions (generation of semantic cross-references, browsing the ontology, keeping the "is a" hierarchy cycle free etc.) will be described in more detail. As can be seen in the ER schema (Figure 13) ontologies are basically stored as a set of nodes, which are connected by edges. However, a tree or a net representation of the ontology is needed when a user browses an ontology, or when all "child nodes" of a given hierarchy have to be selected. In simple words, large hierarchical data structures like ontologies require efficient methods for tree-processing *within* the DBMS. It is important that these computations are performed within the DBMS since this enables that controlled vocabularies and ontologies, which consist of millions of concepts, can be handled by the system. SEMEDA makes intensive use of Oracles proprietary "connect by prior" extension (Oracle-Corp. 1997) for tree processing, which performs "excellent" compared to other methods for "tree-walking" (NHS Information Authority 2000).

Subsequently a simplified minimalistic example query of the "connect by prior" statement is given:

```
SELECT id, level
  FROM edge
  WHERE LEVEL <= 5 and etype = "is_a" and onto= "GO"
  CONNECT BY PRIOR from_node = to_node
START WITH edge.id = "root";
```

This example statement generates an "is a" tree from the imported Gene Ontology of depth <= 5 starting with the "root" node. The actual implementations of the procedures, which use the “connect by prior” statement, were quite complicated since several SQL constructs do not work in conjunction with the "connect by prior" statement. For example, Oracle 8i does not allow to use the "connect by prior" statement in conjunction with the join operator, a problem which can be worked around by using inline views.

ANSI/ISO SQL99 introduced an equivalent feature called "recursive queries". This feature would facilitate porting SEMEDA to other DBMS, although only few DBMS have implemented "recursive queries". In addition it is likely that the first implementations do not perform very well.

Besides using PL/SQL procedures, functions or triggers, much functionality was implemented by using Oracles cascading delete. This proprietary oracle extension allows that tablerows that have a foreign key to another tablerow can be deleted automatically when the foreign key is deleted. Thus when for example an ontology is deleted, all relations, relation types and concepts of this ontology can be deleted in one step.

Table 3: List of the most important PL/SQL functions, procedures and triggers of SEMEDA

Name of the function, procedure or trigger	**Description**
Function MARGBSCHEME_ALL (): MargBench Schema Function MARGBSCHEME (database): BioDataServer Schema	Returns an integrated MARGBench Schema either for all databases in SEMEDA or for a specific database.
Function CROSS_REFERENCES (attribute, ontology): db attributes	Lists the semantic cross-references of an database attribute (see section 3.3.4).
Function LOGIN (login, pw):user ID	Logs a user in if the login/pw is correct and the account is activated.
Procedure LOCK_O (user, ontology) Procedure LOCK_DB (user, database) Procedure LOCK_DT (user, datatype)	Locks the ontology, database or datatype (vocabulary) for other users, i.e. makes sure that only one user at a time can edit an ontology, database or datatype. Denies lock if the object is already locked by another user.
Procedure UNLOCK_O (user, ontology) Procedure UNLOCK_DB (user, database) Procedure UNLOCK_DT (user, datatype)	Unlocks the ontology, database or datatype. Denies unlock if the object is locked by a different user than the user who tries to unlock an object.
Procedure RELEASE_IDLE_LOCKS ()	Is called by a oracle job at an configurable interval. Releases locks of users who have been idle for a certain amount of time. The trigger LAST_ACCESS_TABLENAME is a prerequisite for this method.
Trigger LAST_ACCESS_TABLE	For each table this trigger is fired whenever an attribute is inserted, updated or deleted. Sets in the table ONTO_USER the attribute LAST_ACCCESS to the current date.
Trigger CK_I_PERMISSION_TABLE (ID) Trigger CK_U_PERMISSION_TABLE (ID) Procedure CK_D_PERMISSION_TABLE (ID)	Checks if a user has the permission to execute an insert, update or delete statement (See Figure 19) on a table. For each table in SEMEDAs database schema, all three triggers/procedures were implemented.

Name of the function, procedure or trigger	**Description**
Trigger CK_I_E (all attributes of the edge to be inserted) Trigger CK_U_E (all attributes of the edge to be updated)	Inserts/updates a new edge and checks for consistency of the ontology. The consistency cannot be checked by triggers because triggers cannot query the table which fired the trigger.
Trigger CK_I_U_DEPEND_E	Checks on insert or update of edges if 1.) the nodes which the edge connects is released or suggested by the user who inserts the edge 2.) the edgetype of the edge is released or belongs to the user who inserts the edge 3.) if the user who inserts the edge is a SYS Account, he is allowed to omit these checks.
Trigger CK_I_U_DEPEND_D	Checks on insert or update of a semantic attribute definition if the database attribute which is defined and the node as which it is defined is released or suggested by the user who inserts/updates the definition
Trigger CK_I_U_DEPEND_R	Checks on insert or update of a semantic table definition if the database table which is refined and the node as which it is refined is released or suggested by the user who refines the database table.
Procedure REFRESH_EDGES ()	This procedure has to be executed every time an edge, node or edge type has been inserted, deleted or updated. This procedure is a prerequisite for the execution of the "connect by prior" statement. The details of this procedure are of technical nature and will not be discussed here any further.
Function FIND (ontology, reg. Expression): List of ontology concepts	Searches concept names and identifiers of an ontology by a regular pattern using SQL92 syntax and returns the matching concepts.

Name of the function, procedure or trigger	**Description**
Function BROWSE (node, ontology, edge type, inverse, depth):ontology subtree	Generates a depth and relation type filtered tree-representation for browsing the ontology
Procedure CK_CYCLES (node, ontology, edge type, inverse, depth)	This check makes sure that the ontologies in SEMEDA are directed acyclic graphs (DAG), i.e. it prevents that situations like A "is a" B, B "is a" C and C "is a" A occur. The check is executed each time a relation (edge) is inserted or updated

6.1.4.2 Java Server Pages

Java Server Pages (JSP) is a technology for building dynamic web pages, similar to cgi, Microsofts Active Server Pages (ASP) or PHP. It is based on Java, i.e. developers who use JSP have the full Java API at their disposal. JSP is a technology developed by Sun Microsystems which is based on the Java Servlet technology (http://java.sun.com/products/jsp/). In principle, there is no difference between a JSP and Servlets, since JSPs are translated to a Servlet. When a request is mapped to a JSP page, it is handled by a special servlet that first checks whether the JSP page”s servlet is older than the JSP page. If it is, it translates the JSP page into a servlet class, compiles the class and executes the servlet.

The main advantage of JSP compared to Servlets lies in the development mode. Whereas in Servlets the developer has to use many "println" commands to generate the HTML source code at request time, in JSPs the developer can first develop a static HTML page, and subsequently substitute the parts of the HTML source code, which has to be generated at request time. This enables to develop a static HTML prototype of the frontend, which can already be presented to potential users and readily be adjusted to meet the user requirements. Once such a static HTML version of the frontend exists, all the HTML pages can one by one be transferred to JSPs by substituting the parts of the source code, which has to be generated at request time.

As already mentioned, the main application logic of SEMEDA was implemented by using PL/SQL procedures, functions and triggers. In order to access SEMEDAs relational backend, JDBC was used. SQL cannot only be used to access databases by using SQL, but also to call the PL/SQL procedures and functions. Thus the main functionality of the JSPs implemented was to mediate between the HTML frontend and the relational backend. Therefore JSP was used to dynamically generate the HTML frontend and for session tracking. In addition, for SEMEDAs database query interface, the source databases were queried from within JSP via JDBC, either directly or via the BioDataServer.

Since different parts of SEMEDA required different HTML representations (a page for querying databases has a different structure than a page for browsing an ontology), several different JSPs had to be developed.

However, adding, editing, deleting or showing ontology concepts, attributes of databases, database tables, database attributes, relation types etc. is technically similar, since it always applies to a single row SQL insert/update/select or delete statement. Therefore these functions were implemented by a few generic JSPs (Table 4) which could be configured by request parameters, i.e. each page is called with two HTTP request parameters: table = name of the database table + id of the respective tablerow. For example:

www-bm.ipk-gatersleben.de/semeda/browse.jsp?table=ontology&id=GO

displays the tablerow of the table ONTOLOGY with the identifier "GO". Respectively

www-bm.ipk-gatersleben.de/semeda/browse.jsp?table=datatype&id=MGDB48015

displays the properties of the vocabulary with the internal identifier "MGDB48015" (enzyme number). To test these examples you have to login to SEMEDA first.

In addition, some functionality which was the same on most pages, such as connecting to the relational database backend or printing a HTML"head" was implemented by using the JSP include directive, which allows to include external code at compile time of a JSP. Thus the database connection of all JSPs can be changed without having to edit every JSP separately.

Table 4: Generic JSPs which are used to display, insert, update and delete data in SEMEDA.

JSP Name	**Description**
browse.jsp	show the attributes of an object (i.e. attributes of a tablerow).
insert.jsp update.jsp	generate appropriate html forms dynamically. Once a user filled in the html form data and pressed the "submit" button, the attribute/value pairs are posted to the process_insert.jsp and process_update.jsp files.
process_insert.jsp process_update.jsp process_delete.jsp	process the forms, i.e. execute the insert/update or delete operation. These JSPs can either be called from the generic insert.jsp/update.jsp forms, or directly by providing the appropriate parameters.

6.1.5 Frontend (User Interface)

For the frontend, dynamically generated HTML was used. By using Cascading Style Sheets, the layout (fonts types, font sizes, font colour, background colour etc.) of the frontend can easily be adjusted.

SEMEDA used the same page layout for all pages. By using tabbed panes for the top-level menus, and by arranging the submenus below the selected top-level menu, SEMEDAs frontend stayed in a way modular, which allows to add or delete menus without requiring that the frontend is redesigned (see Figure 18). The rest of the screen was used for the application, which was selected via the menus. Using HTML frames further enhanced the modularity of SEMEDA, i.e. the lower part of the window was divided into three equal sized HTML frames. Thus single frames could be edited without affecting other parts of SEMEDAs frontend. Since SEMEDA is a frontend for a database integration, it is rather an application than a website. Therefore it is not required that individual pages of SEMEDA can be bookmarked or searched by search engines. Therefore, although HTML frames are generally not recommended for building websites, in SEMEDA the use of frames had more advantages than disadvantages.

Basic Software ergonomy principles such as warning a user before he updates or deletes data and visually grouping related form elements have been followed. For visually grouping of HTML form elements, which belong together, such form elements which are submitted together are grouped by underlying them using a background colour which differed from the main background colour.

The use of JavaScript could not completely be avoided. JavaScript had to be used to display warning messages, checking HTML form data, display ToolTips and to submit HTML form data across HTML framesets. All JavaScript functions were implemented Netscape 4.5+ and Internet Explorer 5.5+ compliant.

SEMEDAs main functionality also runs on Opera 6.01+. For the query interface, the overlib (http://www.bosrup.com/web/overlib/) was used to display additional information about databases, database attributes, concepts etc. by using mouse over effects. The overlib supports Netscape/IE 4.x and newer browser versions, but due to Operas limited JavaScript support, Opera does not work with the overlib.

The main top-level menus of SEMEDA from the left to the right are the database query interface (“Query DBs”), the interface for editing database metadata and ontologies (“Meta DB”), an user interface for various administrative tasks (“Admin tools”) and in “about” information about SEMEDA is provided.

The details of SEMEDAs frontend will be described in chapter 7.

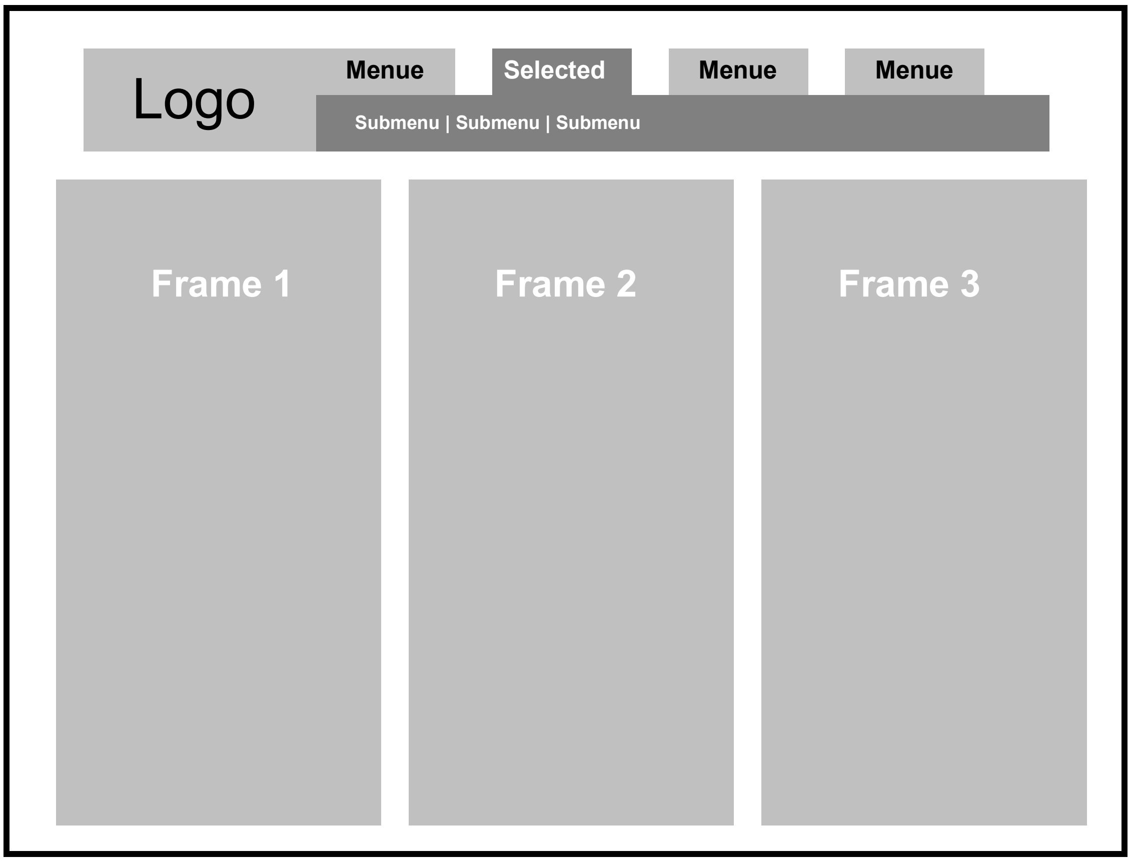


Figure 18: Page layout of SEMEDAs frontend. The top-level menus were arranged as tabbed panes. When submenus were needed, these were provided as links below the selected main menu. The rest of the screen was used for the application, which was selected via the menus. This area was divided in three areas, which corresponded to HTML frames. HTML form elements, which are submitted together, are grouped by using a different background colour than the main background.

6.2 BioDataServer access

The BioDataServer was used to query heterogeneous data sources and to import database metadata. In addition as a prerequisite for database queries, an integrated BioDataServer schema had to be submitted to the BioDataServer.

Whereas for the database queries the JDBC interface of the BioDataServer could be used, the metadata import and the transfer of the integrated schema to the BioDataServer had to be executed via a proprietary telnet protocol of the BioDataServer (see section 4.3).

6.3 Multi User Support

The implementation of multi-user support required a high proportion of the total time used for the implementation of SEMEDA. According to Requirement 16 for multi-user support, the three user groups (admins, DB providers and everybody) were distinguished. Table 5 lists the groups and permissions implemented in SEMEDA. In addition, a fourth user group ("system accounts") has been implemented which completely omits all consistency checks and is exclusively used by internal procedures.

All SEMEDA users are allowed to query databases and to browse ontologies, database metadata and vocabularies. Only confidential database metadata such as port, login, password etc. for accessing a database are exclusively accessible by the owner of a database and by administrative accounts.

As already discussed (Requirement 16), users can only suggest objects. Before other users can also use the suggested objects, they have to be released by an administrative account. The idea was first to make objects "invisible" until they have been released by an administrative account, but it was decided later on that this is not a good solution since it might be confusing: for example a database provider might want to add a new concept, but SEMEDA would have to refuse to generate this concept because this concept already exists, but is invisible. Therefore it was decided that it is better to show all nodes, but to implement it in the above-mentioned way, i.e. to indicate when a concept has been suggested and hence may only be used by the owner until it has been released.

Table 5: User groups and permissions in SEMEDA.

	Query databases	Browse ontologies and database Metadata	Suggest changes to ontologies, databases and vocabularies	Release suggested changes
Administrators	✓	✓	✓	✓
DB providers	✓	✓	✓	
Everybody	✓	✓		

SEMEDA can handle several ontologies, databases and vocabularies. Each ontology, database and datatype can be edited by only one user at the same time. Before a user can edit one of these objects he has to "lock" it, i.e. these objects are usually accessed in "browse mode", and when a user wants to edit one of the objects, he has to switch to the "edit mode". When a user locks one of these objects (enters one of the objects in edit mode), only he and no other user can edit it. Ontologies, databases or vocabularies are locked, by setting the attribute LOCKED_BY to the user id of the user who locks the object.

In order to make sure that a user cannot permanently lock an object, the lock is released when a user was inactive for more than 30 minutes. Different lock-timeouts can be defined at the user level. The "unlocking" of objects was implemented by using a PL/SQL job, which sets all LOCKED_BY, attributes for all idle users to NULL. In order to be able to determine how long a user was idle, each successful insert update or delete operation of a user sets the attribute LAST_ACTIVITY in the table ONTO_USER to the current time.

In Figure 19 the checks, which are executed each time a user tries to update, delete or insert data in SEMEDA are displayed. First it is checked if the user is logged in, and if he is logged in it is checked if his session has not timed out. If the session timed out, he is redirected to the login page. These two checks were implemented by using JSPs session tracking mechanisms (see http://java.sun.com/products/servlet/2.3/javadoc/javax/servlet/http/HttpSession.html).

All other checks were implemented using Oracles PL/SQL either within the procedures, which execute the modifications, or within PL/SQL triggers, which are fired on, insert or update of data. The fact if an user account is activated is stored in the table ONTO_USER. Thus it is possible to deny editing data by simply setting in table ONTO_USER the attribute ACCOUNT_ACTIVATED to false. If the account of the user is activated, it is checked if the user is in "edit mode", i.e. whether he has locked the respective ontology, database or datatype. Thereafter it is checked if the user who tries to update, or delete an object has an administrative account or is an database provider who tries to update or delete data which "belongs" to himself, i.e. data which he himself had suggested.

It is also checked if dependent objects are only suggested or already released, i.e. if for example a concept that shall be used to semantically define a database attribute is only suggested, thus may only be used by the owner, unless it has been released by an administrative account. This prevents potentially faulty concepts being used. Thus it is possible to undo changes of one user without having to deal with multiple dependencies, which may occur when other users use faulty concepts. This step is not displayed in Figure 19 since it is different for different objects. For example when a new concept is added and connected by an edge, it has to be checked if the relation type ("is a", "is part of" etc.) by which it is connected to another concept is released, and also if the concept to which it is connected is released. When a database attribute is semantically defined (see section 3.3.1), it has to be checked if the concept as which it is defined has already been released, and when the vocabulary of a database attribute is defined (see section 3.3.3) it has to be checked if the vocabulary has been released etc.

Once an insert, update or delete operation has passed all those checks it is executed and the respective object is suggested. Thereafter administrators can list all concepts, relations, relations types, vocabularies etc. which have not been released via a special interface. From the listed concepts the administrator can select and release the objects, which are correct.

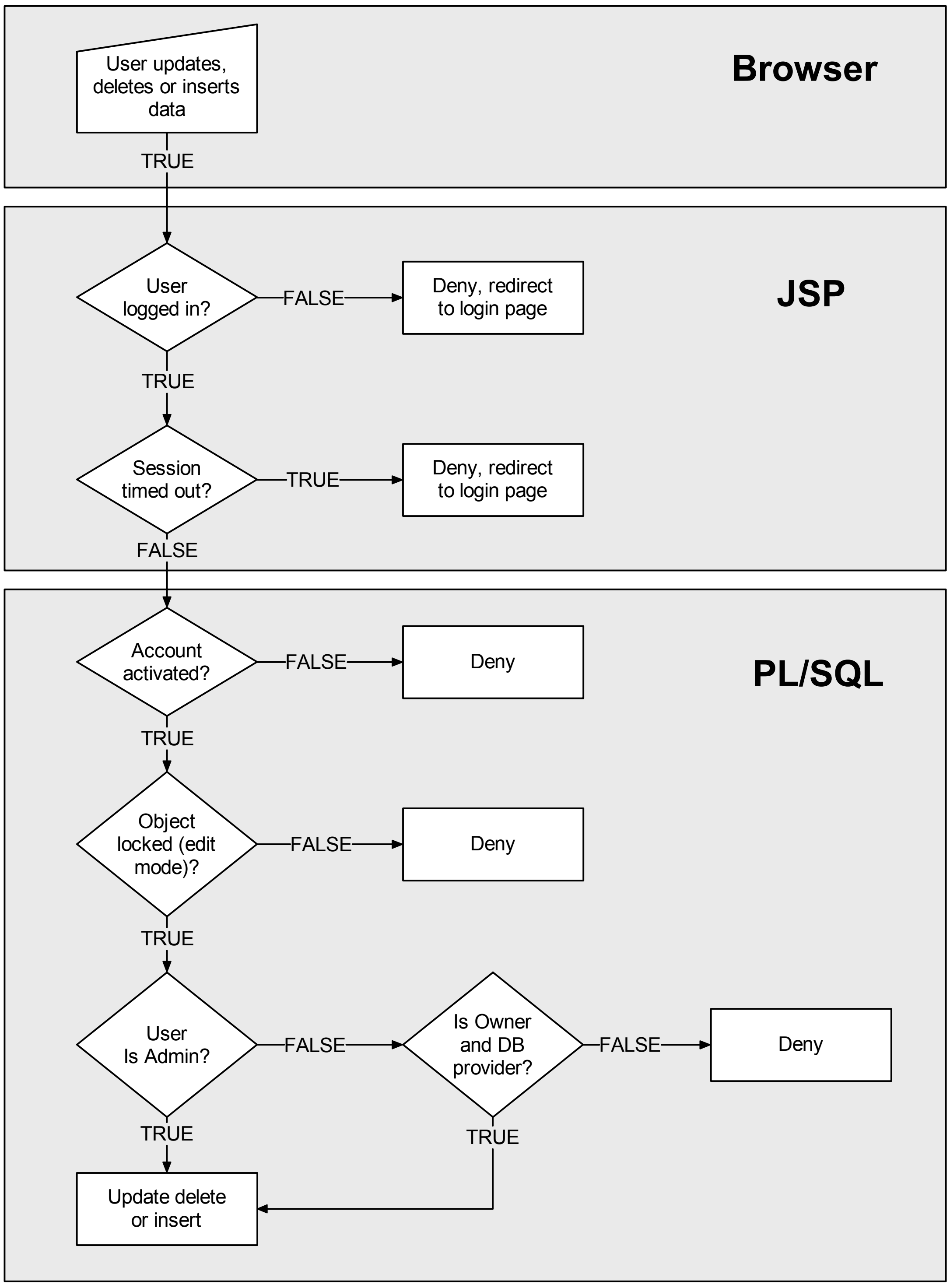


Figure 19: User authentication on update, insert or delete. Besides DB Provider and Admin accounts, System accounts exist, which have full access on everything, unless the account is deactivated.

In addition, further checks can relatively easily be implemented by adding appropriate PL/SQL triggers or procedures, which are executed on insert, update or delete SQL statements. This is useful for implementing consistency checks of the ontology: for example a check, which prevents relations to connect concepts as loops, has been implemented (see Table 3) procedure CK_CYCLES). This check makes sure that the ontologies in SEMEDA are directed acyclic graphs (DAG), i.e. prevents that situations like A "is a" B, B "is a" C and C "is a" A occur. The check is executed each time a relation (edge) is inserted or updated.

6.4 Tool Interface

Other applications can access SEMEDAs relational backend directly via JDBC. This does not only enable other applications to read data directly from SEMEDA backend via SQL, but also to use the many PL/SQL procedures and functions implemented in SEMEDA. As already mentioned, much of the application logic of SEMEDA was implemented using PL/SQL procedures and functions.

6.4.1 Read-only Access

Read only access to SEMEDA is provided by granting read access to an Oracle account on all database tables and attributes of SEMEDAs backend. Tables such as the ONTO_USER table (user data of SEMEDA which contains login, passwords, user rights etc.) or confidential database metadata such as host, port, login etc. of source databases can be hidden. In addition, access to PL/SQL procedures/functions, which do not modify data can be granted. Such PL/SQL procedures are for example procedures for searching an ontology concept (see Table 3, function FIND), for generating subtrees of ontologies in SEMEDA (Table 3,, function BROWSE), for creating integrated BioDataServer schemata (Table 3, function MARGBSCHEME_ALL) etc.

6.4.2 Write Access

In principle tools could be granted write access in an equivalent way as read access is granted, i.e. by granting read, insert and update permissions to an oracle account. Write access for external applications is neither required for the functionality of SEMEDA, nor would it be useful for the MARGBench to get write access in SEMEDA.

However, "good willing" tools that follow certain conventions might be granted write access. Those conventions are:

1.) Before a tool starts editing data in SEMEDA it has to use the LOGIN procedure and use the user ID returned by this procedure subsequently.

2.) The tool has to LOCK (see Table 3) the database, ontology or datatype (vocabulary) before it edits data of these objects by SQL statements or PL/SQL procedures.

3.) In each SQL insert or update statements, the attribute EDITED_BY has to be set to the user ID. Thus, respective database triggers can check user permissions and throw an exception if a user does not have the permission to insert or update an entry.

4.) SQL delete statements may not be used directly. Instead of using the delete statement, for each database table in SEMEDA a procedure CK_D_TABLENAME(user_id, row_id) exists which does the permission checking and consistency checking before an entry is deleted.

When an external application follows these 4 conventions, the tool permissions are checked in the same way as user permissions are checked when a user edits data in SEMEDA via the frontend.

6.5 Resources and Programming Languages

SEMEDA was implemented using Oracle 8i, but also runs on newer Oracle versions. SEMEDA was recently deployed at the IPK Gatersleben on Oracle 9i without requiring any modifications of the database or the PL/SQL code. To deploy SEMEDA on older Oracle Versions than Oracle 8i would require substantial modifications because features that are only available since Oracle 8i were used.

Due to the use of PL/SQL, SEMEDA cannot easily be ported to other DBMS systems. In addition, the fact that most other relational DBMS do not provide adequate methods for "tree processing" such as oracles "connect by prior" statement would make it especially difficult to port SEMEDA to other relational DBMS. However, with SQL 99 "recursive SQL" queries have been introduced, which will presumably be suitable for "tree generation".

Resin was used to run the JSPs although in principle any other JSP engine, which supports the JSP 1.1 standard, should be capable of running SEMEDA. Resin was configured to run with the Apache webserver although Resin can also be deployed standalone. A very useful feature of resin was its database connection pooling support, which was much used. A prerequisite for running Resin is the installation of the Java Programming language. SEMEDA has been tested using Java 1.3 and Java 1.4.

In addition, taglibs were used. In simple words, taglibs are custom defined JSP tags similar to HTML tags. Every tag is mapped to a particular Java class file, which is executed whenever the tag is encountered in a JSP file.

For SEMEDAs query interface (SEMEDA-query) the overlib was used for mouse over tool tips. The overlib is a highly configurable JavaScript Library, which works both in Netscape and Internet Explorer.

Table 6 summarises the resources and programming languages that were used for the implementation of SEMEDA.

Table 6: Resources and programming languages used for the implementation of SEMEDA.

Resource	**URL**	**Description**
Oracle 8i	http://www.oracle.com/database/oracle8i/	Relational DBMS, includes PL/SQL
Apache	http://www.apache.org/	Webserver
Resin	http://www.caucho.com/	JSP Engine
Java 2	http://java.sun.com/	Programming Language Java
Jakarta Taglibs	http://jakarta.apache.org/taglibs/	Taglibs for database connection, accessing session variables, etc were used
JavaScript	http://www.netscape.com/eng/mozilla/3.0/handbook/javascript/	Some functionality of the frontend was implemented using JavaScript.
Overlib	http://www.bosrup.com/web/overlib/	JavaScript Library for mouse over tool tips

7 Using SEMEDA

In this chapter the usage of SEMEDA is described. The different usages of SEMEDA will be described in the sequence of a user who wants to add a new database to the system: First database metadata has to be imported from the BioDataServer (7.2). Then the database metadata can be complemented and semantically defined using SEMEDA-edit (7.3). To semantically define a database, it may also be necessary to suggest new vocabularies or modifications to SEMEDAs main ontology. These suggested modifications can subsequently be released by an administrative account (7.4). Finally an integrated BioDataServer Schema for all databases in SEMEDA has to be generated and submitted to the BioDataServer (7.5) before the databases can be queried (**Error! Reference source not found.**).

SEMEDA can be found at http://www-bm.ipk-gatersleben.de/semeda/. A demo account (login:semeda, password:pw) can be used to access SEMEDA in read only mode, i.e. with this account it is possible to query databases, browse database metadata and ontologies, but not to edit database metadata, ontologies etc.

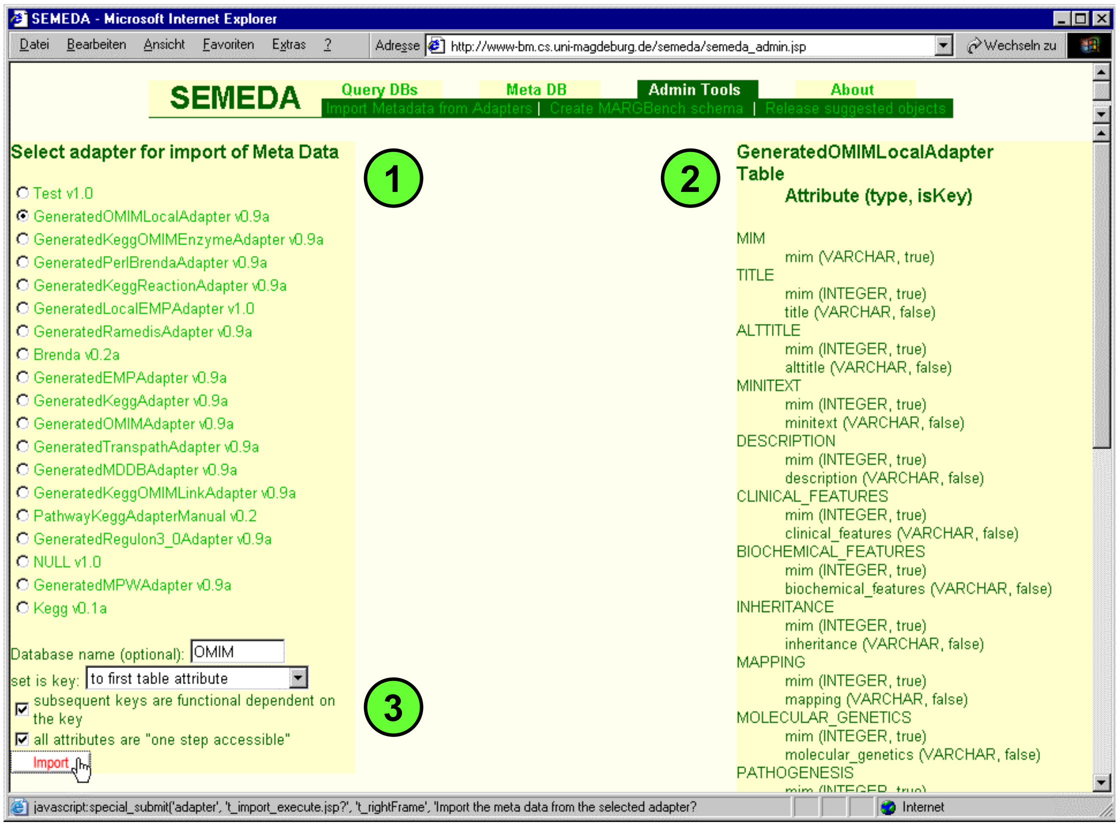
7.1 Client requirements

SEMEDA was tested with Netscape 4.7, 6.2, IE 5.5+ and Opera 6.01. Netscape was tested on SUN, Windows and Linux computers, but as already mentioned, the JavaScript mouse over tool tips based on the overlib do not work for Opera.

7.2 Import Metadata from the BioDataServer.

Database Metadata can be imported using "Import Metadata from Adapters" from the "Admin Tools" menu (Figure 20). Alternatively it is possible to enter all database metadata manually using SEMEDA-edit (see below). The database schema information of a database adapter (the names of the tables, the names of the attributes, the datatypes of the attributes etc.) is imported from the BioDataServer. Only administrative accounts can import database metadata to SEMEDA.

When a user selects this menu, SEMEDA connects to the BioDataServer, reads a list of all adapters that are available and displays them to the user. By clicking on a specific adapter, a user can preview the database metadata of the adapter without importing it. Before a user executes the import by clicking the "import" button, he can adjust several import options: Set the name of the database (the BioDataServer lists adapter names, but adapter names sometimes differ from database names), determine which database attributes are key attributes and which attributes are functional dependent on the keys, and determine if attributes are *one step accessible* (see section 4.3). If "all attributes are one step accessible" is not selected, only the key attributes are set to *one step accessible*. As a rule of the thump, only in relational databases that can be accessed directly via JDBC, this checkbox should be set.


SEMEDA - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Adresse http://www-bm.cs.uni-magdeburg.de/semeda/semeda_admin.jsp
Wechseln zu
SEMEDA
Query DBs
Meta DB
Admin Tools
About
Import Metadata from Adapters | Create MARGBench schema | Release suggested objects
1
2
3
Import
javascript:special_submit('adapter', 't_import_execute.jsp?', 't_rightFrame', 'Import the meta data from the selected adapter?
Internet

Select adapter for import of Meta Data

- ○ Test v1.0
- ● GeneratedOMIMLocalAdapter v0.9a
- ○ GeneratedKeggOMIMEnzymeAdapter v0.9a
- ○ GeneratedPerlBrendaAdapter v0.9a
- ○ GeneratedKeggReactionAdapter v0.9a
- ○ GeneratedLocalEMPAdapter v1.0
- ○ GeneratedRamedisAdapter v0.9a
- ○ Brenda v0.2a
- ○ GeneratedEMPAdapter v0.9a
- ○ GeneratedKeggAdapter v0.9a
- ○ GeneratedOMIMAdapter v0.9a
- ○ GeneratedTranspathAdapter v0.9a
- ○ GeneratedMDDBAdapter v0.9a
- ○ GeneratedKeggOMIMLinkAdapter v0.9a
- ○ PathwayKeggAdapterManual v0.2
- ○ GeneratedRegulon3_0Adapter v0.9a
- ○ NULL v1.0
- ○ GeneratedMPWAdapter v0.9a
- ○ Kegg v0.1a

Database name (optional): OMIM
set is key: to first table attribute
☑ subsequent keys are functional dependent on the key
☑ all attributes are "one step accessible"

GeneratedOMIMLocalAdapter
Table
Attribute (type, isKey)

```
MIM
        mim (VARCHAR, true)
TITLE
        mim (INTEGER, true)
        title (VARCHAR, false)
ALTTITLE
        mim (INTEGER, true)
        alttitle (VARCHAR, false)
MINITEXT
        mim (INTEGER, true)
        minitext (VARCHAR, false)
DESCRIPTION
        mim (INTEGER, true)
        description (VARCHAR, false)
CLINICAL_FEATURES
        mim (INTEGER, true)
        clinical_features (VARCHAR, false)
BIOCHEMICAL_FEATURES
        mim (INTEGER, true)
        biochemical_features (VARCHAR, false)
INHERITANCE
        mim (INTEGER, true)
        inheritance (VARCHAR, false)
MAPPING
        mim (INTEGER, true)
        mapping (VARCHAR, false)
MOLECULAR_GENETICS
        mim (INTEGER, true)
        molecular_genetics (VARCHAR, false)
PATHOGENESIS
        mim (INTEGER, true)
```

Figure 20: Import of Database Metadata from BioDataServer adapters: 1) list of available BioDataServer adapters. 2) A preview of the database metadata of an adapter can be displayed by clicking on one of the adapters. 3) Import options.

7.3 SEMEDA-edit

SEMEDA-edit can be found in the "Meta DB" menu. The frontend consists of three frames: the left frame is used for database metadata editing and editing of vocabularies, the middle frame is used for ontology editing and browsing and the right frame is used context dependent on both frames. Figure 21 shows the start menu of SEMEDA-edit where users can choose a specific database, ontology or vocabulary to be opened, deleted or added. Since several users can co-operatively edit databases, ontologies or vocabularies, a refresh button exists which can be used to update the respective dropdown menus and thus display changes other users may have incorporated meanwhile. In addition, some properties of databases, ontologies or vocabularies can be displayed ("Properties") or edited ("Edit Prop"). By using this option for databases, it can be specified if the database should be directly queried via JDBC, or if a database should be accessed via the BioDataServer.

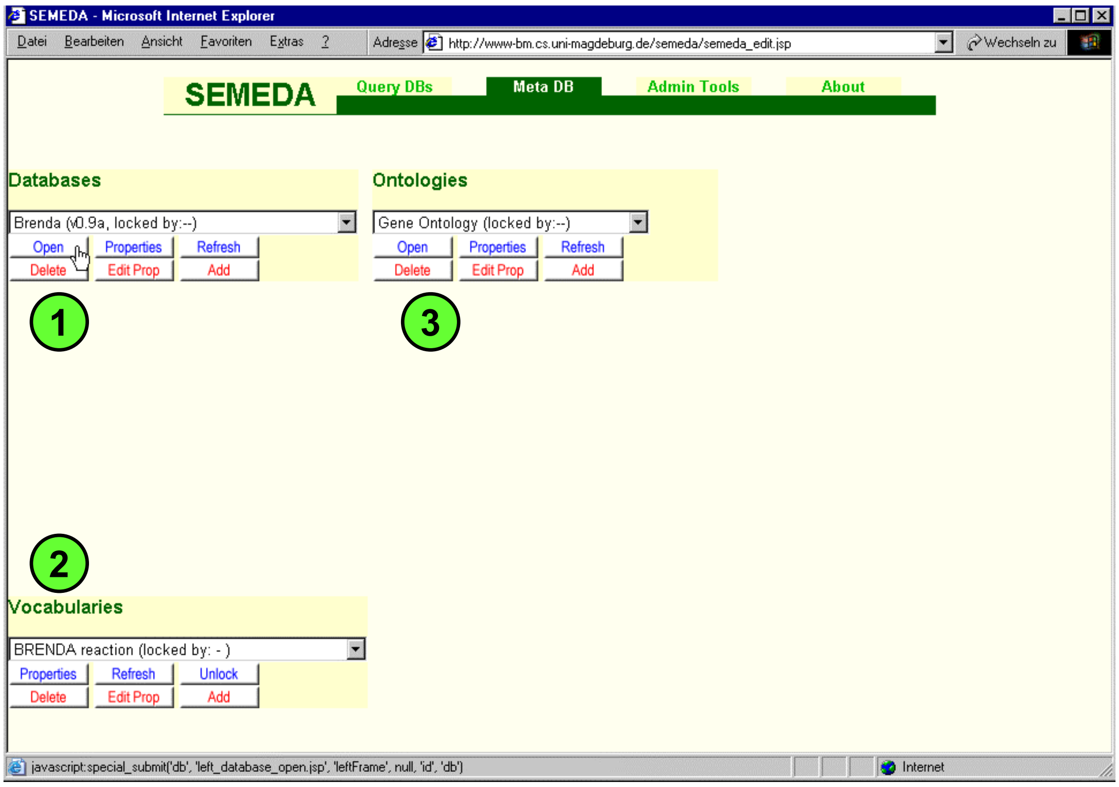

SEMEDA - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Adresse http://www-bm.cs.uni-magdeburg.de/semeda/semeda_edit.jsp
Wechseln zu
SEMEDA
Query DBs
Meta DB
Admin Tools
About
Databases
Brenda (v0.9a, locked by:--)
Open
Properties
Refresh
Delete
Edit Prop
Add
Ontologies
Gene Ontology (locked by:--)
Open
Properties
Refresh
Delete
Edit Prop
Add
1
3
2
Vocabularies
BRENDA reaction (locked by: -)
Properties
Refresh
Unlock
Delete
Edit Prop
Add
javascript:special_submit('db', 'left_database_open.jsp', 'leftFrame', null, 'id', 'db')
Internet

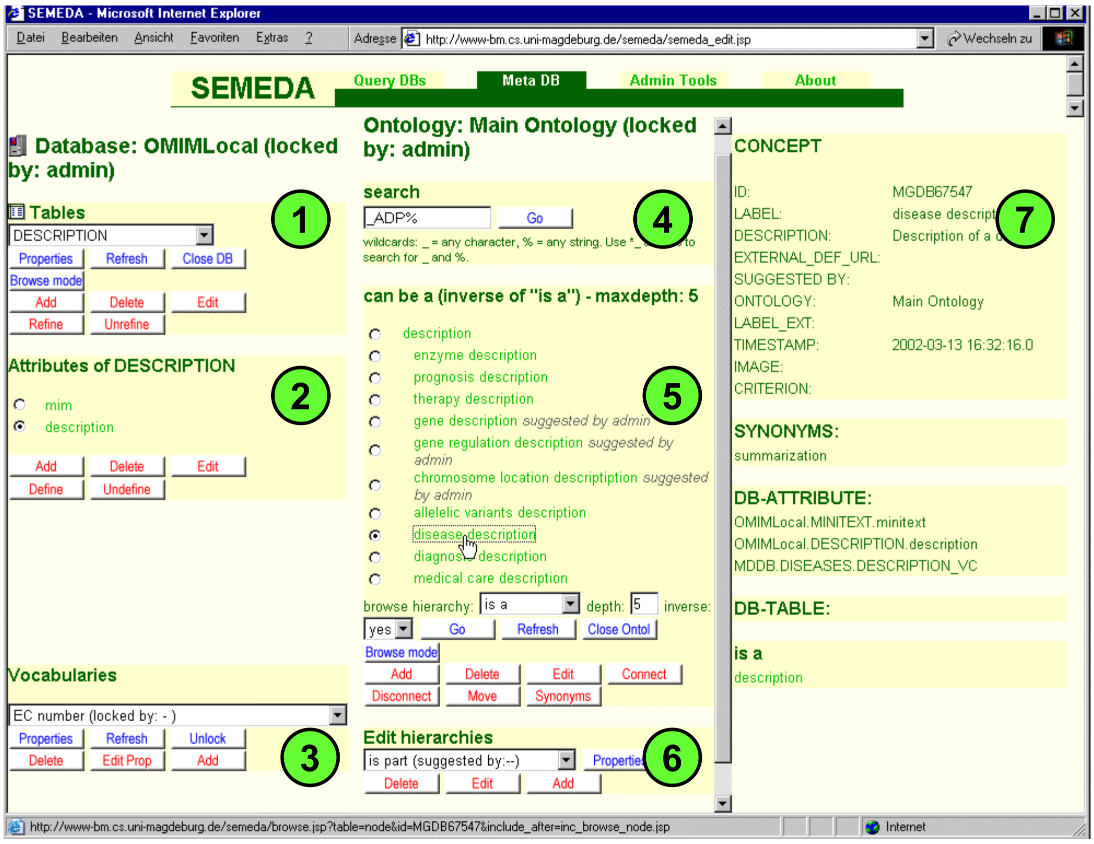
Figure 21: Start screen of SEMEDA-edit. When a user enters SEMEDA-edit, he can choose if he wants do open, add or delete a specific database (1), ontology (2) or vocabulary (3).

After a user has selected a specific database or ontology and opened it, he can browse the database metadata or ontology. After opening the database or ontology, he can see all database metadata/ontology concepts but not edit it. In order to edit the database metadata or ontology, the user has to click the button "Edit Mode". This button switches to edit mode, i.e. it displays buttons required for editing the ontology or database metadata and locks the ontology/database metadata for other users. When a database, ontology or datatype is locked for a user, other users cannot edit it at the same time. Thus locking avoids conflicts which otherwise might arise from several users editing the same objects. Figure 22 shows SEMEDA in edit mode.

Both in the ontology and the database metadata editor frames, concept, database, table and attribute information etc. can be viewed in the right frame by clicking the appropriate objects. These objects can be edited by selecting the appropriate radio-button or dropdown menu option before clicking on the appropriate "*add*", "*edit*" or "*delete*" button.

In the left frame that displays the database metadata, a dropdown list of all database tables of the selected database is displayed. Below this dropdown list, a list of all attributes of the database table, which is selected in the dropdown menu, is displayed.

The middle frame is used for ontology editing. The ontology editor, can be used to add/delete/edit concepts, connect concepts, add synonyms to concepts, and to move concepts. Connecting concepts inserts a relation between two concepts whereas moving a concept disconnects a concept from its parent concept and connects it to the concept to which it is moved.


SEMEDA - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Adresse http://www-bm.cs.uni-magdeburg.de/semeda/semeda_edit.jsp
Wechseln zu
SEMEDA
Query DBs
Meta DB
Admin Tools
About
1
2
3
4
5
6
7

Database: OMIMLocal (locked by: admin)

Tables

DESCRIPTION

Properties Refresh Close DB
Browse mode
Add Delete Edit
Refine Unrefine

Attributes of DESCRIPTION

- mim
- [✓ description]

Add Delete Edit
Define Undefine

Vocabularies

EC number (locked by: -)

Properties Refresh Unlock
Delete Edit Prop Add

Ontology: Main Ontology (locked by: admin)

search

_ADP% Go

wildcards: _ = any character, % = any string. Use *_ *% to search for _ and %.

can be a (inverse of "is a") - maxdepth: 5

- description
  - enzyme description
  - prognosis description
  - therapy description
  - gene description *suggested by admin*
  - gene regulation description *suggested by admin*
  - chromosome location descriptiption *suggested by admin*
  - allelelic variants description
  - [✓ disease description]
  - diagnosis description
  - medical care description

browse hierarchy: is a depth: 5 inverse: yes Go Refresh Close Ontol
Browse mode
Add Delete Edit Connect
Disconnect Move Synonyms

Edit hierarchies

is part (suggested by:--) Properties
Delete Edit Add

CONCEPT

ID:	MGDB67547
LABEL:	disease descript
DESCRIPTION:	Description of a d
EXTERNAL_DEF_URL:	
SUGGESTED BY:	
ONTOLOGY:	Main Ontology
LABEL_EXT:	
TIMESTAMP:	2002-03-13 16:32:16.0
IMAGE:	
CRITERION:	

SYNONYMS:

summarization

DB-ATTRIBUTE:

OMIMLocal.MINITEXT.minitext
OMIMLocal.DESCRIPTION.description
MDDB.DISEASES.DESCRIPTION_VC

DB-TABLE:

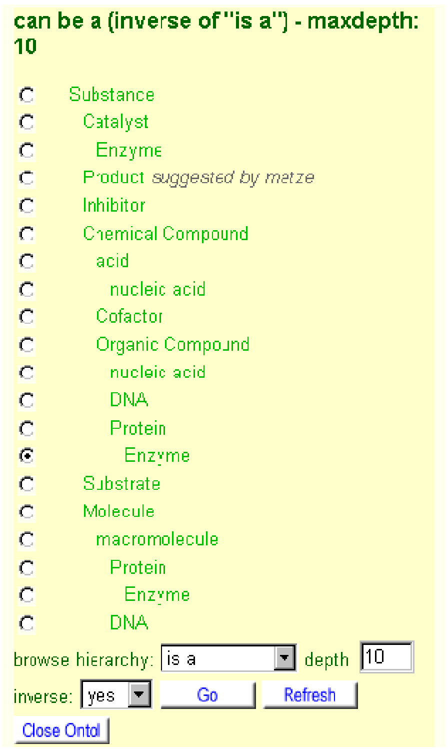
is a

description

http://www-bm.cs.uni-magdeburg.de/semeda/browse.jsp?table=node&id=MGDB67547&include_after=inc_browse_node.jsp Internet

Figure 22: Screenshot of SEMEDA in edit mode. Left: Database metadata editor, middle: Ontology Editor. Right: context dependent frame (7). 1) Browse, edit and semantically define database tables. 2) Browse, edit and semantically define database attributes of the selected database table. 3) Browse and edit vocabularies. 4) Search for an ontology concept. 5+6) Browse and edit ontologies and ontologies.

Ontologies are often large (see Table 7). Thus for a visual representation of the ontology, only a subset of all concepts can be displayed at a time. For visual representation of the ontology a by depth and by relation type filtered tree is used, which is equivalent to a depth limited closure of a node over a given hierarchy. To browse the ontology, a user has to select the radio button of an ontology concept, choose the hierarchy (relation type) he wants to browse and choose how deep the ontology should be expanded before he clicks the "browse" button. The hierarchical structure of the ontology is displayed by indenting sub-concepts relative to their parent concepts. The option to browse "inverse" enables the user to select whether he wants to browse all super-concepts or all sub-concepts of a selected concept. The inverse relation of "is a" relations is for example "can be a". All proteins are organic compounds (protein "is a" organic compound), but organic compounds are not necessarily proteins (organic compound "can be a" protein).



can be a (inverse of "is a") - maxdepth: 10

- Substance
  - Catalyst
    - Enzyme
  - Product *suggested by metze*
  - Inhibitor
  - Chemical Compound
    - acid
      - nucleic acid
    - Cofactor
    - Organic Compound
      - nucleic acid
      - DNA
      - Protein
        - [✓ Enzyme]
  - Substrate
  - Molecule
    - macromolecule
      - Protein
        - Enzyme
      - DNA

browse hierarchy: is a depth 10

inverse: yes

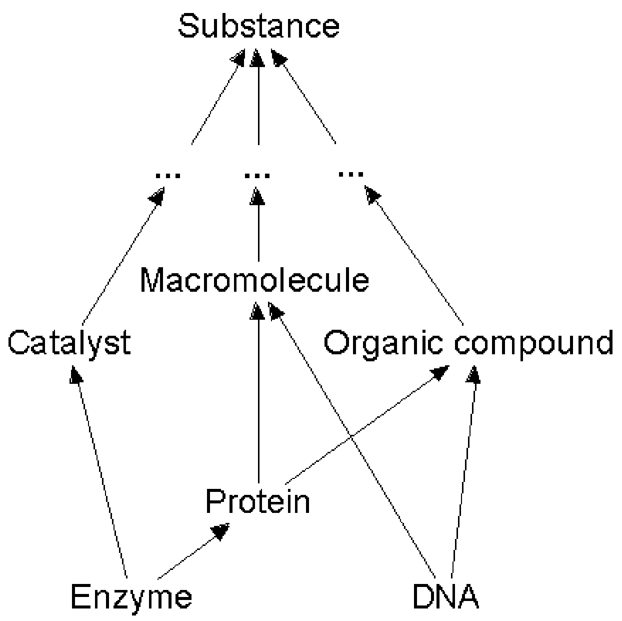


Figure 23:" is a" hierarchy of the sub-concepts of "Substance" in SEMEDAs "Main Ontology". Left: screenshot. The hierarchical structure of the ontology is displayed by indenting sub-concepts relative to their parent concepts. In addition, the relation typeand the inverse relation type is displayed (explanation see text). Right: equivalent graph visualisation.

Although ontologies in SEMEDA are directed acyclic graphs (DAG), ontologies are represented as trees in SEMEDA, i.e. concepts that have two parent concepts are displayed under each parent concept. Figure 23 shows some concepts of SEMEDAs "Main Ontology" and the equivalent graph representation. Since the "is a" relation is transitive, SEMEDA can derive: (enzyme "is a" protein) and (protein "is a" organic compound) → (enzyme "is a" organic compound). On the other hand, if a user wants to find all organic compounds, all sub-concepts of "organic compound" can be derived (in this example: protein, enzyme, DNA). This is for example useful, when a user wants to know which databases contain attributes which can be searched for organic compounds, since all relevant database attributes are defined as "organic compound" or as sub-concepts of "organic compound”. A clear distinction between browse and edit mode exists and the user always sees in which mode he is working. "Suggested objects" are visually differentiated and the username of the owner, i.e. the user who suggested an object is always displayed, thus a user can see if he has the permission to modify an object. Objects, which do not "belong" to somebody specific, can only be edited by administrative accounts. Such nodes should be 100 % correct and not be modified. Thus a DB provider can rely on the semantics of the concepts in SEMEDA, and he can be sure that the semantics will not change after he defined database attributes as concepts.

7.3.1 Attribute Semantics

Database attributes can be defined using the "*define*" button after selecting both the attribute and the appropriate concept in the ontology. By "defining" a database attribute as a concept, the database provider states: "entries of this database attribute are ‘is a’ children or instances of the selected concept". Whether the attribute contains instances, sub-concepts or both, can be defined indirectly by defining the vocabulary used by a database attribute (see below, section 7.3.3). A database attribute can be defined more than once, since database attribute contents sometimes are heterogeneous: for example the attribute "source" in the Protein Data Bank (Berman et al. 2000) may contain tissue, species, cell line etc. information. Another example is the table REFERENCES in the MDDB (Hofestädt et al. 1998, Hofestädt et al. 2000), which is used to store literature reference data. Whereas separate attributes for author, title and year of a publication exist, only one attribute ANNOTATION exists which is used to store journal name, volume, issue and the page numbers.

7.3.2 Table Semantics

The "*refine*" button refines the content of a database table, i.e. makes statements about all entries of a table. Database tables are refined in a similar way as attributes are defined, i.e. to define a database table a user has to select the database table he wants to define from the dropdown menu and the concept to which he wants to refine the attribute and finally press the “refine” button. Refining a table for example to mouse means that all data in this table is mouse data. Tables can also be refined to more than one concept, since for example a table may contain protein data from mice.

7.3.3 Attribute Value Semantics

The vocabulary, which is used in a database attribute, is defined by selecting the database attribute in the left frame, clicking the “edit” button. This displays a form where several properties of the database attribute can be manually edited, among other things the vocabulary (datatype) used can be specified by selecting the appropriate vocabulary from a dropdown menu. In addition to the vocabulary, the JDBC datatype of the source database can be defined. In simple words, the JDBC datatype stores generic datatypes such as String, Integer or Boolean, whereas the vocabulary is much more specific and stores which vocabulary is used, i.e. it defines the values of a database attribute more precisely (Enzyme Number, Systematic Species Name, English species name, CAS registry number etc.).

If a vocabulary does not already exist in SEMEDA, it is possible to add new vocabularies at the bottom of the left frame. After a user suggested a new vocabulary, he can use it to define attribute values of database attributes.

7.4 Release Suggested Objects

Modifications to the ontology and vocabularies are only suggested, i.e. the modifications have to be released by an administrative account before other users can edit these objects. Suggested objects can be released by selecting "Release suggested objects" from the "Admin Tools" menu (see Figure 24). This displays a dropdown list from which the user can select the ontology for which he wants to release objects. Subsequently the user can choose if he wants to release Concepts, Edges (relations), E-Types (relation types), Attribute Definitions, Table Definitions or Vocabularies. By clicking the respective option, all suggested objects of one of these categories are displayed. By using checkboxes, the user can select which objects he wants to *release* and finally release the objects by clicking the “release” button.

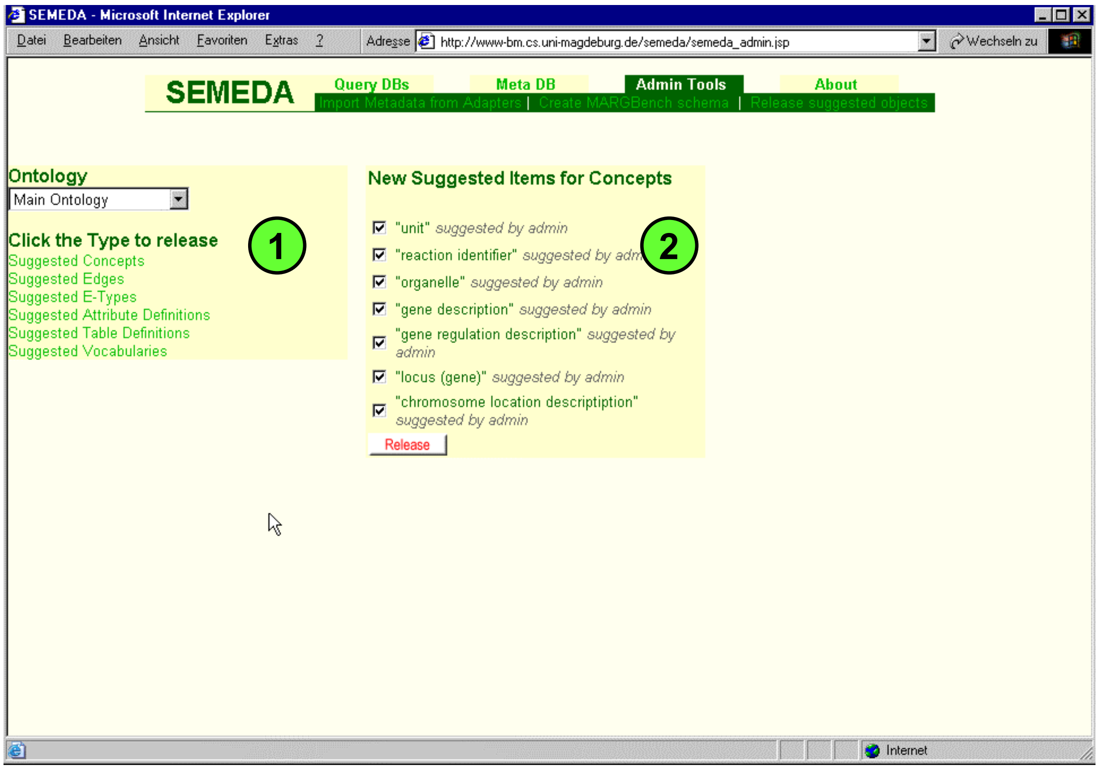
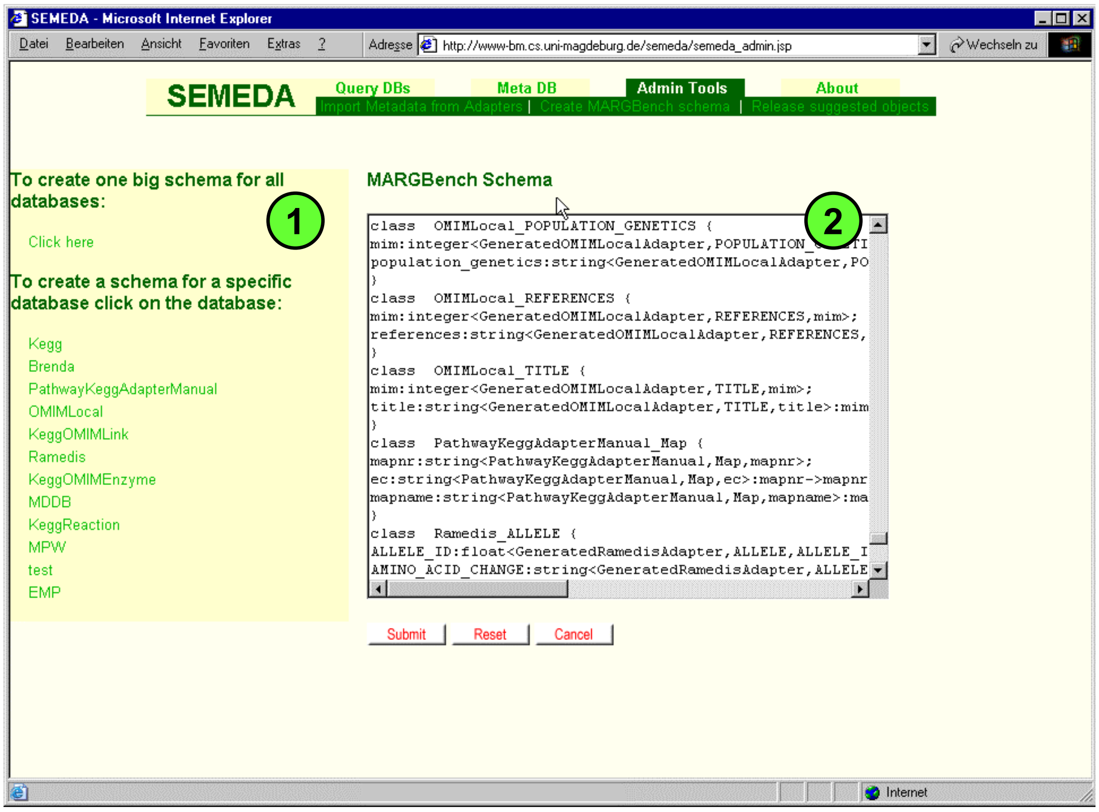

SEMEDA - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Adresse http://www-bm.cs.uni-magdeburg.de/semeda/semeda_admin.jsp
Wechseln zu
SEMEDA
Query DBs
Meta DB
Admin Tools
About
Import Metadata from Adapters | Create MARGBench schema | Release suggested objects
Ontology
Main Ontology
Click the Type to release
Suggested Concepts
Suggested Edges
Suggested E-Types
Suggested Attribute Definitions
Suggested Table Definitions
Suggested Vocabularies
1
New Suggested Items for Concepts
2
"unit" suggested by admin
"reaction identifier" suggested by admin
"organelle" suggested by admin
"gene description" suggested by admin
"gene regulation description" suggested by admin
"locus (gene)" suggested by admin
"chromosome location descriptiption" suggested by admin
Release
Internet

Figure 24: User interface where administrative accounts can release suggested modifications. 1) Select ontology of which suggested concepts, edges (relations), attribute definitions etc. are to be listed. 2) Select the objects to be released.

7.5 Submit BioDataServer Schemata

Before SEMEDA can use the BioDataServer to query heterogeneous data sources by SQL, an integrated database schema has to be created and submitted to the BioDataServer. This can be achieved by selecting "Create MARGBench schemata" from the "Admin Tools" menu (see Figure 25). Users can choose to either create one big integrated database schema for all attributes, or to create a BioDataServer schema for a specific database. When the user clicks the "submit" button, SEMEDA connects to the BioDataServer and transfers the integrated schema to the BioDataServer. Only administrative accounts can submit integrated BioDataServer schemata.


SEMEDA - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Adresse http://www-bm.cs.uni-magdeburg.de/semeda/semeda_admin.jsp
Wechseln zu
SEMEDA
Query DBs
Meta DB
Admin Tools
About
Import Metadata from Adapters | Create MARGBench schema | Release suggested objects
1
2
Submit
Reset
Cancel
Internet

To create one big schema for all databases:

Click here

To create a schema for a specific database click on the database:

Kegg
Brenda
PathwayKeggAdapterManual
OMIMLocal
KeggOMIMLink
Ramedis
KeggOMIMEnzyme
MDDB
KeggReaction
MPW
test
EMP

MARGBench Schema

```
class  OMIMLocal_POPULATION_GENETICS {
mim:integer<GeneratedOMIMLocalAdapter,POPULATION_GENETI
population_genetics:string<GeneratedOMIMLocalAdapter,PO
}
class  OMIMLocal_REFERENCES {
mim:integer<GeneratedOMIMLocalAdapter,REFERENCES,mim>;
references:string<GeneratedOMIMLocalAdapter,REFERENCES,
}
class  OMIMLocal_TITLE {
mim:integer<GeneratedOMIMLocalAdapter,TITLE,mim>;
title:string<GeneratedOMIMLocalAdapter,TITLE,title>:mim
}
class  PathwayKeggAdapterManual_Map {
mapnr:string<PathwayKeggAdapterManual,Map,mapnr>;
ec:string<PathwayKeggAdapterManual,Map,ec>:mapnr->mapnr
mapname:string<PathwayKeggAdapterManual,Map,mapname>:ma
}
class  Ramedis_ALLELE {
ALLELE_ID:float<GeneratedRamedisAdapter,ALLELE,ALLELE_I
AMINO_ACID_CHANGE:string<GeneratedRamedisAdapter,ALLELE
```

Figure 25: Interface for creating and submitting integrated BioDataServer schemata. 1) Select whether to create one big integrated schema for all databases or only for a specific database. 2) The generated integrated schema is displayed in an HTML formfield where it can be manually edited before it is submitted to the BioDataServer.

Creating a database schema for a specific database is not useful for SEMEDA. It was implemented since it might be useful for applications of the MARGBench project. In the MARGBench project a schema editor for building and editing integrated schematas was developed. In this schema editor a user has to enter the whole schema manually. Therefore this option of SEMEDA might be useful to generate a template schema for a specific database, which can be copied and pasted to the MARGBench schema editor, where it can subsequently be adjusted.

7.6 SEMEDA-query

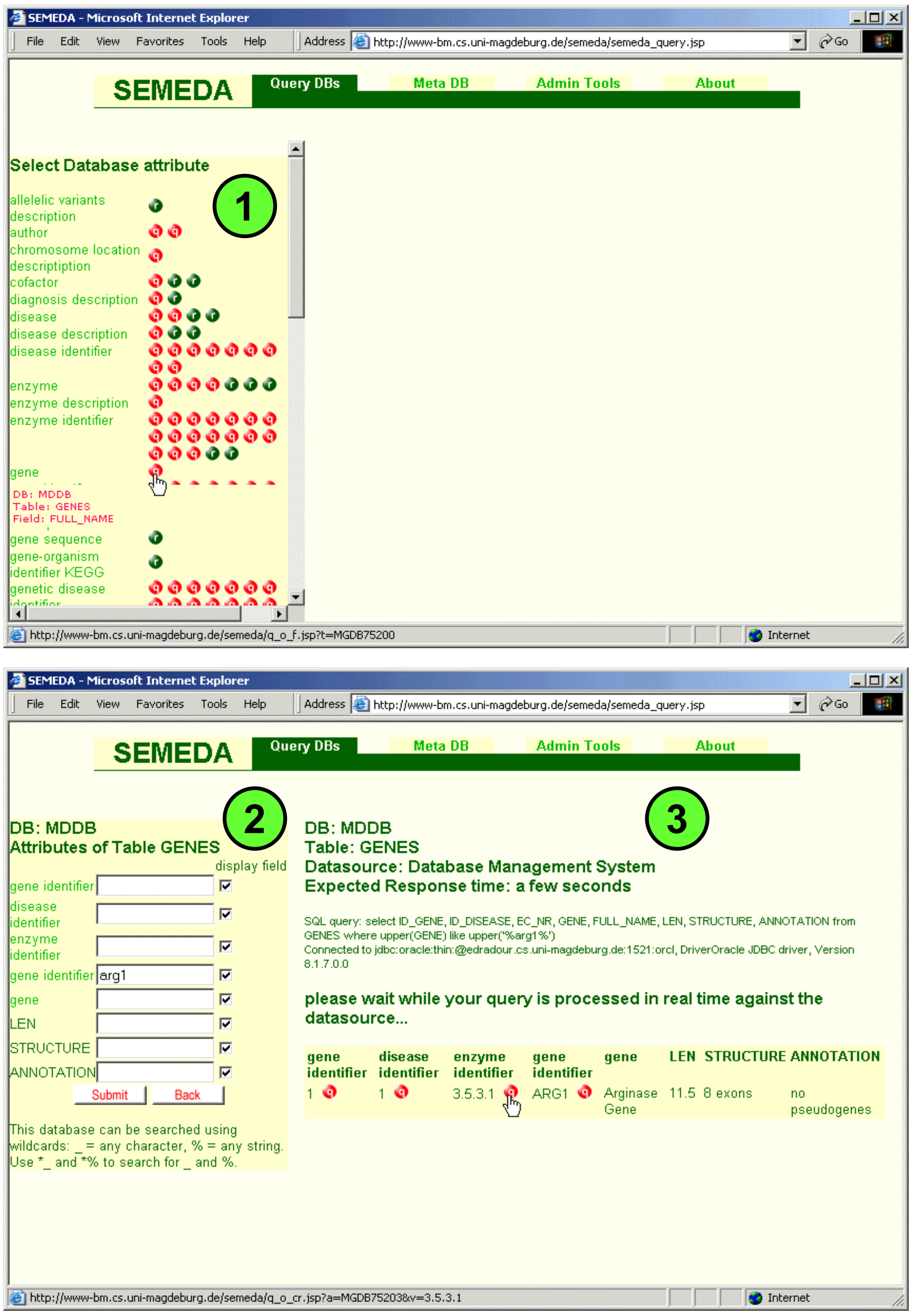
SEMEDAs database query interface makes use of the semantic database definitions of the previous sections.

The most useful features of semantic database integration as introduced in chapter 3 are the semantic attribute definitions and the semantic cross-references (which in turn are based on semantic attribute and attribute value definitions). In addition, by storing database metadata (database names, table names attributes and their datatypes, etc.), a source database specific form based query interface supports user queries. In addition, BioDataServer specific information such as whether or not a database attribute is *one step accessible*, enables to discriminate between attributes which can be searched within a response time which is acceptable for human interaction. It is also differentiated between databases that can be directly accessed via JDBC, and data sources that are accessed via the BioDataServer. Databases that can be directly accessed via the JDBC, can be searched using SQL 92 pattern matching.

The query interface (see Figure 26 and Figure 27) was implemented to illustrate the potential of semantic database integration based on these main principles.

SEMEDA-query can be found in the "Query DBs" menu. When a user selects this menu option, a list of concepts is displayed. Behind each concept, a set of colored icons is displayed. Each icon represents one database table, which has an attribute for the concept. These icons are based on semantic attribute definitions (see 3.3.1). Database tables in which this attribute can only retrieved in the result set, but which cannot be searched are represented by a green icon. Otherwise the table is represented by a red icon. This colour code is based on the fact whether or not an attribute is *one step accessible*. A mouse over ToolTip displays the names of the database, database table and attribute. After clicking one of the icons, an appropriate query form for the respective database table can be used to query the database.

The query form displays all database attributes of the database table and the user can select the database attributes to be included in the result. The database attributes which are semantically defined (see 3.3.1) are labelled using concept names rather than the often misleading attribute names of the source databases. Mouse over ToolTips of the attribute labels display further information about a database attribute, such as its datatype (=attribute value definition, see 3.3.3), a description of the datatype, an example entry of the attribute and also the attribute name used in the source database. A text informs the user when a database can be searched using SQL 92 patterns matching, which is the case for databases, which can directly be accessed via JDBC. In addition, when semantic table definitions (see 3.3.2) exist for a database table, these are displayed. Semantic table definitions indicate if the table contains only data of a certain group, e.g. for example only data for a certain species or a certain compound class.


SEMEDA - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www-bm.cs.uni-magdeburg.de/semeda/semeda_query.jsp
Go
SEMEDA
Query DBs
Meta DB
Admin Tools
About
1
2
3
Submit
Back
Internet

Select Database attribute

allelelic variants description
author
chromosome location descriptiption
cofactor
diagnosis description
disease
disease description
disease identifier
enzyme
enzyme description
enzyme identifier
gene

DB: MDDB
Table: GENES
Field: FULL_NAME

gene sequence
gene-organism identifier KEGG
genetic disease identifier

http://www-bm.cs.uni-magdeburg.de/semeda/q_o_f.jsp?t=MGDB75200

DB: MDDB
Attributes of Table GENES

		display field
gene identifier		☑
disease identifier		☑
enzyme identifier		☑
gene identifier	arg1	☑
gene		☑
LEN		☑
STRUCTURE		☑
ANNOTATION		☑

This database can be searched using wildcards: _ = any character, % = any string. Use *_ and *% to search for _ and %.

DB: MDDB
Table: GENES
Datasource: Database Management System
Expected Response time: a few seconds

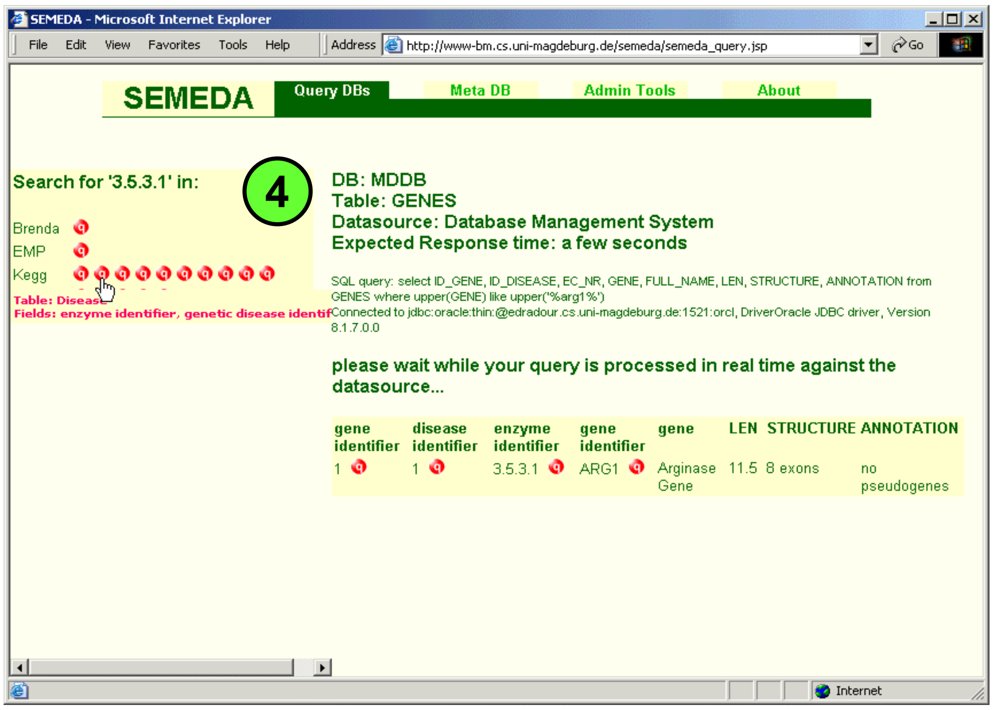
SQL query: select ID_GENE, ID_DISEASE, EC_NR, GENE, FULL_NAME, LEN, STRUCTURE, ANNOTATION from GENES where upper(GENE) like upper('%arg1%')
Connected to jdbc:oracle:thin:@edradour.cs.uni-magdeburg.de:1521:orcl, DriverOracle JDBC driver, Version 8.1.7.0.0

please wait while your query is processed in real time against the datasource...

gene identifier	disease identifier	enzyme identifier	gene identifier	gene	LEN	STRUCTURE	ANNOTATION
1	1	3.5.3.1	ARG1	Arginase Gene	11.5	8 exons	no pseudogenes

http://www-bm.cs.uni-magdeburg.de/semeda/q_o_cr.jsp?a=MGDB75203&v=3.5.3.1

Figure 26: SEMEDAs database query interface. 1) All concepts for which database attributes exist are listed. Each of the round icons represents a database table, which contains attributes for the concept. 2) After clicking one of the icons, an appropriate query form for the respective database can be used to query the database. 3) Result set. Round icons can be used to cross-reference to other relevant databases (see next Figure).


SEMEDA - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www-bm.cs.uni-magdeburg.de/semeda/semeda_query.jsp
Go
SEMEDA
Query DBs
Meta DB
Admin Tools
About
4
Internet

Search for '3.5.3.1' in:

Brenda
EMP
Kegg

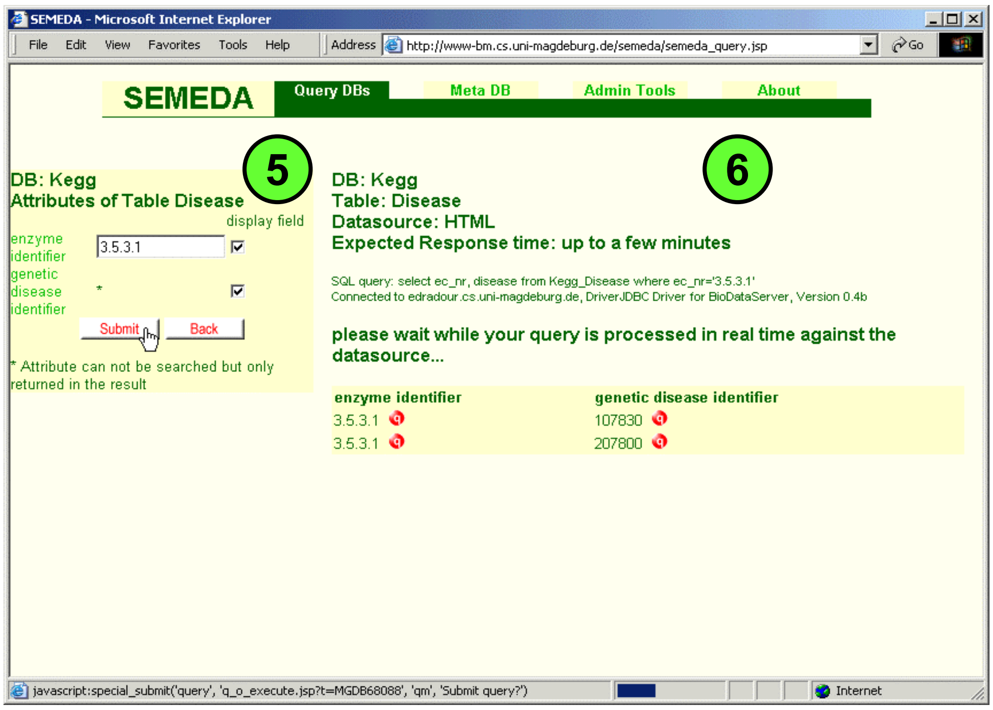
Table: Disease
Fields: enzyme identifier, genetic disease identif

DB: MDDB
Table: GENES
Datasource: Database Management System
Expected Response time: a few seconds

```
SQL query: select ID_GENE, ID_DISEASE, EC_NR, GENE, FULL_NAME, LEN, STRUCTURE, ANNOTATION from
GENES where upper(GENE) like upper('%arg1%')
Connected to jdbc:oracle:thin:@edradour.cs.uni-magdeburg.de:1521:orcl, DriverOracle JDBC driver, Version
8.1.7.0.0
```

please wait while your query is processed in real time against the datasource...

gene identifier	disease identifier	enzyme identifier	gene identifier	gene	LEN	STRUCTURE	ANNOTATION
1	1	3.5.3.1	ARG1	Arginase Gene	11.5	8 exons	no pseudogenes


SEMEDA - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://www-bm.cs.uni-magdeburg.de/semeda/semeda_query.jsp
Go
SEMEDA
Query DBs
Meta DB
Admin Tools
About
5
6
Submit
Back
Internet

DB: Kegg
Attributes of Table Disease

		display field
enzyme identifier	3.5.3.1	✓
genetic disease identifier	*	✓

* Attribute can not be searched but only returned in the result

DB: Kegg
Table: Disease
Datasource: HTML
Expected Response time: up to a few minutes

```
SQL query: select ec_nr, disease from Kegg_Disease where ec_nr='3.5.3.1'
Connected to edradour.cs.uni-magdeburg.de, DriverJDBC Driver for BioDataServer, Version 0.4b
```

please wait while your query is processed in real time against the datasource...

enzyme identifier	genetic disease identifier
3.5.3.1	107830
3.5.3.1	207800

```
javascript:special_submit('query', 'q_o_execute.jsp?t=MGDB68088', 'qm', 'Submit query?')
```

Figure 27: SEMEDAs database query interface (continued, see previous figure). 4) After clicking one of the cross-reference icons (see previous figure), the user can link to other relevant database tables. 5) These can again be searched using an appropriate form, and from the result set (6) the user can link again to other appropriate databases.

After the user entered the search terms and submitted the query form, the source database is queried and the result set is displayed. Along with the result, round icons provide cross-references (see 3.3.4) to other relevant databases. If for example, a result set contains an Enzyme Number, the user can click the icon to get a list of other database tables that can also be searched using this Enzyme Number. Thus the user can search for further information in other databases just by linking to them. By clicking one of these icons, a query form for the selected database table is displayed with the Enzyme Number field filled out which was used for cross-referencing. Now the user can further refine the query before he submits the form and gets the results for this second database table. From these results, the user can again link to other databases etc.

Thus the user can browse across several source databases without having to know the databases. However, the system is transparent in a way that it always displays from which database the data was retrieved. This is important, since most users would not trust a system, which retrieves data without reporting where the data was retrieved from.

The main limitation of SEMEDA-query is the low speed of the BioDataServer. Therefore, in a scenario where more than a few users use SEMEDA, the performance of the BioDataServer would have to be improved. One way how this could be achieved is discussed in the "database mirror" section of the discussion (see 11.2.1).

8 Evaluation of Existing Ontologies

In the previous chapter, SEMEDAs user interface was described. In order to use SEMEDA in the described way, an appropriate ontology is needed. Therefore, several ontologies were evaluated for their usability in SEMEDA. The intention was to find out if a suitable ontology can be found and imported or whether it is more appropriate to build a custom ontology for SEMEDA.

Since generating large ontologies is a labour intensive time-consuming task (Schulze-Kremer 1997b), it would be preferable to use an existing ontology. In order to keep the ontologies up to date, it should be possible to re-import ontologies when they have been updated in the source ontology. It is important to realise that although SEMEDA can handle several ontologies, all databases should be semantically defined by using the same ontology.

In this chapter, the specific requirements to ontologies for SEMEDAs database integration approach are derived. Subsequently ontologies which might be appropriate for semantic database integration are evaluated, then the ontologies are matched versus SEMEDAs requirements and it is finally discussed how and if ontologies can be merged and supplemented to meet SEMEDAs requirements, or whether it is more appropriate to build a custom ontology for SEMEDA.

8.1 Criteria

Subsequently, criteria to ontologies and "knowledge sources" for SEMEDA are listed. General requirements for molecular biological ontologies are given in (Schulze-Kremer 1997b, Rector et al. 1998, Schulze-Kremer 1998), although requirements for ontologies to be used in SEMEDA are more specific:

1. IDs: unique id of concepts or unique label of ontology concepts.
2. Stability of concepts: Whereas the text of concept definitions and the name (label) of concepts may change the semantic of a concept should remain the same. This, and the "unique id" criteria are important for "re-importing" ontologies and referencing ontology concepts.
3. Valid “is a” hierarchy: the “is a” hierarchy is important, for "intelligent" database queries; hence the transitive closure of the “is a” hierarchy has to be mathematically sound.
4. Size: The ontologies may be small as long as they cover the database attributes to be integrated. Small ontologies are easier to survey, but a high number of synonyms would be very useful, since it would enable the users to use their own terminology.

5. Availability: The ontology has to be available, free of charge is a plus but not a prerequisite.
6. Wide use: wide acceptance and use is not essential, but makes sure that the potential users are familiar with the terminology/structure of the imported ontology.
7. Maintenance: the ontology should still be maintained and updated and it should be possible to suggest new concepts. Although it is possible to add new concepts within SEMEDA, it would be helpful if the newly needed concepts could also become part of the public domain/source ontology.
8. Precise definition of concepts.

In addition, the ontologies to be imported have to cover the field of research of molecular biological databases. Since SEMEDA maps database attributes, not database entries, the ontologies generally do not have to be very deep. In order to get an overview of the terminology needed in SEMEDA, the most common attributes from some molecular biological databases were compiled. The SRS implementation of the European Bioinformatics Institute gives a good overview of molecular biological databases (http://srs.ebi.ac.uk/srs6bin/cgi-bin/wgetz?-page+databanks+-newId). From each database category (group) the attributes of the databases, which have the highest number of entries, were checked. In addition, databases which are already connected to the MARGBench were investigated: Brenda, RegulonDB (Salgado et al. 2001), TRANSFAC/TRANSPATH (Wingender et al. 2000, Schacherer et al. 2001) and MDDB (Hofestädt et al. 1998, Hofestädt et al. 2000).

The aim was to get a broad overview of the knowledge domain needed for database integration. Therefore the database attributes were not exactly matched nor are all attributes of the selected databases listed.

Common database attributes are: Molecule, Enzyme, Organelle, Gene, Transcription Factor, Transcription Binding Sites, Chromosome, Author, Title, Journal, Abstract, MedlineID, Reference, Comment, Specie, Organism, Tissue, Organ, Reaction, Compound, Age, Date, Sequence, Sequence length, Protein Chain, Evidence, EC-Number, Link, Sequence Type, Cell Lines, NMR Experimental Data, Atomic coordinates, Crystallographic Protein Coordinate Information, Protein Residues, Codon Change, Codon Change Position, Substrate, Product, Cytogenetic Location, Phenotype, Mutation, Mutation Type, Amino Acid Changes, Pathology, Therapy, Clotting Activity, Validated, Allele, Reaction Equation, Reaction Direction, Specific Activity, Purification Steps, Storage Conditions, Temperature, Coenzyme, Optimal PH, Michaelis Constant, Molecular Mass, Restriction Enzyme, Restriction Position, Vendor, Accession Number, Reference.

8.2 Evaluation

In Table 7 some ontologies, controlled vocabularies and other structured knowledge sources are listed and evaluated against the above-mentioned criteria and terms. The criterion "good definition of concepts" is more or less subjective and therefor not discussed in the table.

Some other ontologies which are listed and evaluated in (Schulze-Kremer 1998) or (Noy and Hafner 1997) were not further investigated because they obviously did not cover the knowledge domain of molecular biology.

Table 7: Evaluation of ontologies and knowledge sourcae for semantic database integration.

Ontology	**Evaluation**
MBO	**URL:** http://igd.molgen.mpg.de/~www/oe/mbo.html **Info:** Uses IDs, fairly stable, transitive “is a”, size: 1272 concepts available for free, rarely used, is maintained. **Comment:** Covers fundamental molecular biological concepts and some general terms and concepts.
"upper" CYC	**URL:** http://www.cyc.com/cyc-2-1/toc.html **Info:** Uses IDs, fairly stable, transitive “is a”, size: 3000 concepts, available for free, sometimes used, but not as much as WordNet, is maintained. **Comment:** The "upper CYC" ontology is a general "top-level ontology", commercial extensions to the upper CYC ontology are available.
WordNet 1.7.1 (Fellbaum 1998)	**URL:** http://www.cogsci.princeton.edu/~wn/ **Info:** Uses IDs, fairly stable, fairly transitive “is a”, size: 66025 concepts (noun synsets), available for free, widely used, is maintained. Covers most common English words. **Comment:** Fundamental molecular biological concepts are also covered. Specific molecular biological concepts are not covered. Word, id, and lex_file_num together identify concepts. "Semantic Concordance Package" can be used to trace updates between WordNet versions, but its complicated proprietary structure and flatfile format make imports and updates tedious.

Ontology	**Evaluation**
EcoCyc/ MetaCyc(Karp et al. 2000, Maranas and Burgard 2001)	**URL:** http://ecocyc.pangeasystems.com **Info:** IDs: uses IDs from the sources where it was compiled from (for example EC Numbers), fairly stable, fairly transitive "is a", available free for academic use, mainly used within the CYC system, is maintained. **Comment:** small but "deep". All entries are sub-concepts of Metabolic Pathways, Signaling Pathways, Reactions, Enzymes, Genes, tRNAs, Compounds or Citations. Data is *Escherichia coli* centric.
GO (Ashburner et al. 2000)	**URL:** http://www.geneontology.org **Info:** Uses IDs, fairly stable, non-transitive "is a", size: about 11000 concepts, 14000 relations (June 2002) available for free, widely used in genome databases, is maintained. Covers exclusively genetic and closely related terminology. **Comment:** The Gene Ontology actually consists of three ontologies where some concepts are defined in more than one of the ontologies, but have different IDs in each ontology. The "is a" hierarchy is not a formal "is a" hierarchy.
UMLS	**URL:** http://www.nlm.nih.gov/research/umls/umlsmain.html **Info:** Uses IDs, fairly stable, non-transitive "is a", size: about: 800 000 concepts, available for free, widely used, is maintained. **Comment:** The Unified Medical Language System maps the terminology of 60 different biomedical source vocabularies.
MeSH	**URL:**http://www.nlm.nih.gov/mesh **Comment:** The Medical Subjects Headings are a subset of UMLS.
MmCIF, (Westbrook and Bourne 2000)	**URL:** http://ndbserver.rutgers.edu/mmcif/ **Info:** Uses IDs, fairly stable, non-transitive "is a", available for free, widely used in protein-databases and protein structure related sciences, is maintained. **Comment:** The Macromolecular Crystallographic Information File is a very specific large detailed ontology of crystallographic information, closely integrated to the Protein Data Bank (Berman et al. 2000)

Ontology	Evaluation
Controlled Anatomical Vocabulary	**URL:** http://www.cbil.upenn.edu/anatomy.php3 **Info:** Uses IDs, fairly stable, mixed “is a”/”is part” hierarchy, size: about 500 concepts, available for free, concepts compiled from several sources, is maintained. **Comment:** Concept Definitions have to be looked up in the appropriate sources from which the controlled vocabulary has been compiled.
LinkBase (Ceusters 2001)	**URL:** www.landc.be **Info:** Uses IDs, fairly stable, transitive “is a”, size: 950.000 concepts, commercially available, used in L&C products and custom tailored systems, is maintained. **Comment:** Biomedical terminology as well as basic general concepts are covered. In addition to the "is a" hierarchy, LinkBase implements several other relation types in a strictly formal way and 2.100.000 instantiated relations exist. In addition, 300.000 cross references to foreign systems exist.
GALEN Common Reference Model (CRM) (Rector and Nowlan 1994)	**URL:** www.opengalen.org **Info:** Uses IDs, fairly stable, transitive “is a”, size: about 1200 concepts, available for free (GALEN Open Source License), widely used, is maintained. **Comment:** Extensible Core Model of Biomedical Terminology. Many instantiated relations exist. The core model is a tree and not a directed acyclic graph, i.e. multiple inheritance is not used.
SNOMED	**URL:** www.snomed.org **Info:** Uses IDs, fairly stable, information on transitive “is a” was not available, size: about 200 000 concepts, commercially available, widely used, is maintained. **Comment:** Biomedical Terminology.
Taxonomy (Benson et al. 2000)	**URL:**http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html **Info:** Uses IDs, fairly stable, phylogenetic trees can be seen as excellent transitive “is a” trees, size: huge, most relevant species to molecular biology are covered, available for free is maintained **Comment:** Changes are documented from update to update. Is part of NCBI and thus closely integrated and referenced to and from other applications.

Ontology	Evaluation
Tree of Life	**URL:**http://phylogeny.arizona.edu/tree/phylogeny.html **Info:** IDs: Phylogenetic groups seem to be mapped as a file structure, thus no ids exist, fairly stable, phylogenetic trees can be seen as excellent transitive “is a” trees, size: huge, unclear license restrictions **Comment:** Taxonomic multiauthored phylogenetic tree. Uses also a picture for phylogenetic groups and implements a review process and much useful information. The information seems to be stored as plain html files, which would make importing the phylogeny labour intensive. Larger than necessary for the purpose of semantic database integration.

8.3 Conclusion

Of the aforementioned criteria, formal transitive “is a” hierarchies and coverage of database knowledge domains are the most important criteria for "intelligent" ontology based database queries. Therefore, non-commercial ontologies, which might be suited for our purposes, are the Molecular Biological Ontology (MBO), WordNet and GALENs Common Reference Model. Commercial systems are LinkBase and SNOMED. The size and the fact that LinkBase covers many other relation types in a formal way, makes it especially interesting. However, for SNOMED information concerning the transitivity of the “is a” hierarchy was not available, neither from the documentation nor on request. None of the ontologies, which are listed in Table 7, covers all relevant database concepts and thus would have to be supplemented either manually or by merging ontologies (Uschold et al. 1998), although merging ontologies is a nontrivial task (Russ et al. 1999). In ontologies, which consist mainly of an “is a” hierarchy, merging specialised ontologies by substituting whole “is a” branches is possible. For example NCBIs Taxonomy could be merged into Wordnet although this would have to be done with some care: for example substituting WordNets "organism" hierarchy with NCBIs Taxonomy would also erase Author, since "Author ‘is a’ Human and Human ‘is a’ ... ‘is a’ organism". Since parts of non-formal ontologies such as GO are often correct “is a” hierarchies, merging parts of such hierarchies into a top-level ontology would also be possible. A labour intensive approach for using ontologies with valuable concepts (UMLS, GO, MmCIF) but non-formal “is a” hierarchies would be to introduce a formal “is a” hierarchy.

Even though it is possible to transfer any of the mentioned ontologies to formal ontologies, the time needed would be a matter of many days (MmCIF) or several "man-years" (UMLS). In ontologies, which were generated by merging ontologies, keeping track of updates and re-importing the updated source ontologies would be cumbersome.

Therefore two realistic options existed: either to build a small custom ontology or to import one of the ontologies, which has a formal “is a” hierarchy. For importing, non-commercial candidates are GALENs CRM, MBO and WordNet. The commercial LinkBase might be best suited due to its formal modelling of the “is a” hierarchy and since due to its size it covers the knowledge domain best.

However, it was decided to try to build a small custom ontology for SEMEDA first. If it would have turned out that this approach does not lead to good results, or is too time-consuming, one of the aforementioned ontologies would have had to be used.

In addition the Gene Ontology was imported anyway, to demonstrate SEMEDAs potential to edit big ontologies collaboratively. The Gene Ontology was chosen, because it is widely used and accepted for molecular biological applications.

9 Ontology Design

SEMEDAs semantic database definitions are based on one central ontology. At the end of the previous chapter it was decided to try to build a custom ontology rather than to import an existing ontology. In this chapter, it is described how general ontology design principles were used to build SEMEDAs main ontology. Subsequently the structure of SEMEDAs main ontology is described, and how it was derived based on the implicit semantics of relational database tables.

9.1 General Ontology Design Principles

Before one builds an ontology, one has to decide what kind of ontology is required, and how the ontology should be build.

Domain ontologies describe a particular small model of the world, which is relevant for a specific application. In contrast, top-level ontologies contain generally applicable concepts and relations, which are likely to be used in different applications. Sometimes, top-level ontologies are used as a base ontology, which is extended by adding application specific concepts. It was decided to build a small domain ontology for SEMEDA. This is not only less work, but also such an application specific ontology is easier to survey. In addition, since an ontology, which contains only the concepts required for SEMEDA, is smaller such an ontology can be searched faster, i.e. common queries such as finding all sub-concepts of a concept perform faster.

It further had to be decided whether an ontology should be manually build or automatically from text sources. Several publications deal with automatic ontology building and information extraction from free text (Craven et al. 1998, Ceusters et al. 1999, Sanderson and Croft 1999, Khan and McLeod 2000, Maedche and Staab 2000, Nobécourt 2000, Hahn et al. 2002). The basic idea is, to parse in a first step ontological concepts from texts. In a second step relations between concepts are generated by parsing texts for sentences in which any two concepts occur. By analysing such sentences, the type of the relation between the two concepts is determined ("is a", "is part", "synonym" etc.). This is a very simplistic description of the process of automatic ontology construction. The actual techniques involved are nontrivial and based on Natural Language Processing, although in some more or less formalised texts, for example the Enzyme Nomenclature, patents and laws, hierarchies which can be adopted as “is a” hierarchies already exist. Interestingly, whereas several "hand made" ontologies were found (Table 7), not a single automatically build ontology could be found. Although it does not seem completely unfeasible to automatically construct ontologies from texts, it is unlikely that by using nowadays techniques an ontology could be automatically build which would meet SEMEDAs requirements.

Also one has to decide whether an ontology should be built top-down or bottom-up. Top-down ontologies are started with the concepts which stand at the top of the "is a" hierarchy. Manually constructed ontologies are mostly built top down. In automatically generated ontologies, concepts or relations are added whenever a relevant concept or relation is found. Thus bottom up ontology building is more appropriate for an automated approach. Nothing the less, it was decided to build SEMEDAs ontology more or less bottom up, i.e. to add concepts whenever a new database attribute has to be semantically defined. Manual bottom up building of ontologies is not completely uncommon (van der Vet and Mars 1995, 1998).

Several techniques and conventions for ontology design exist (Gruber 1993a, Schulze-Kremer 1997a, Blázquez et al. 1998, Ceusters 2001, Noy and McGuinness 2001, Hovy 2002, Schulze-Kremer 2002). Subsequently, the most important ontology design principles and how they were used in SEMEDA are given.

Multiple Inheritance

Multiple inheritance should be used whenever it is applicable (see Figure 28). By using multiple inheritance, the ontology becomes less arbitrary and needs fewer concepts to represent the same facts. Using multiple inheritance, often also eliminates the need to invent long unnatural concept names. The downside of it is that the ontology cannot directly be used as a decision tree as suggested by (Schulze-Kremer 2002), although a decision tree could automatically be generated from an ontology, which uses multiple inheritance.

Coherence (valid transitive "is a" hierarchy)

After (Gruber 1993a): “An ontology should be coherent: that is, it should sanction inferences that are consistent with the definitions. At the least, the defining axioms should be logically consistent. Coherence should also apply to the concepts that are defined informally, such as those described in natural language documentation and examples. If a sentence that can be inferred from the axioms contradicts a definition or example given informally, then the ontology is incoherent.”

As already mentioned, a valid "is a" hierarchy is important for the functionality of SEMEDA so that SEMEDA can make transitive inferences such as:

If B "is a" A and C "is a" B, then C "is a" A

This seems to be trivial, but mistakes can easily be made:

EC_NR "is a" Enzyme, Enzyme "is a" Protein -> EC_NR "is a" Protein

Which is an example for how a wrong “is a” relation results in faulty inference, i.e. correctly EC_NR should be a sub-concept of identifier and not of enzyme.

Avoiding cycles in the "is a" hierarchy

A hierarchy has a cycle when a concept A has a sub-concept B and at the same time B is a super-concept of A. Such cycles are automatically caught in SEMEDA, i.e. an error is thrown when a user tries to generate a cycle.

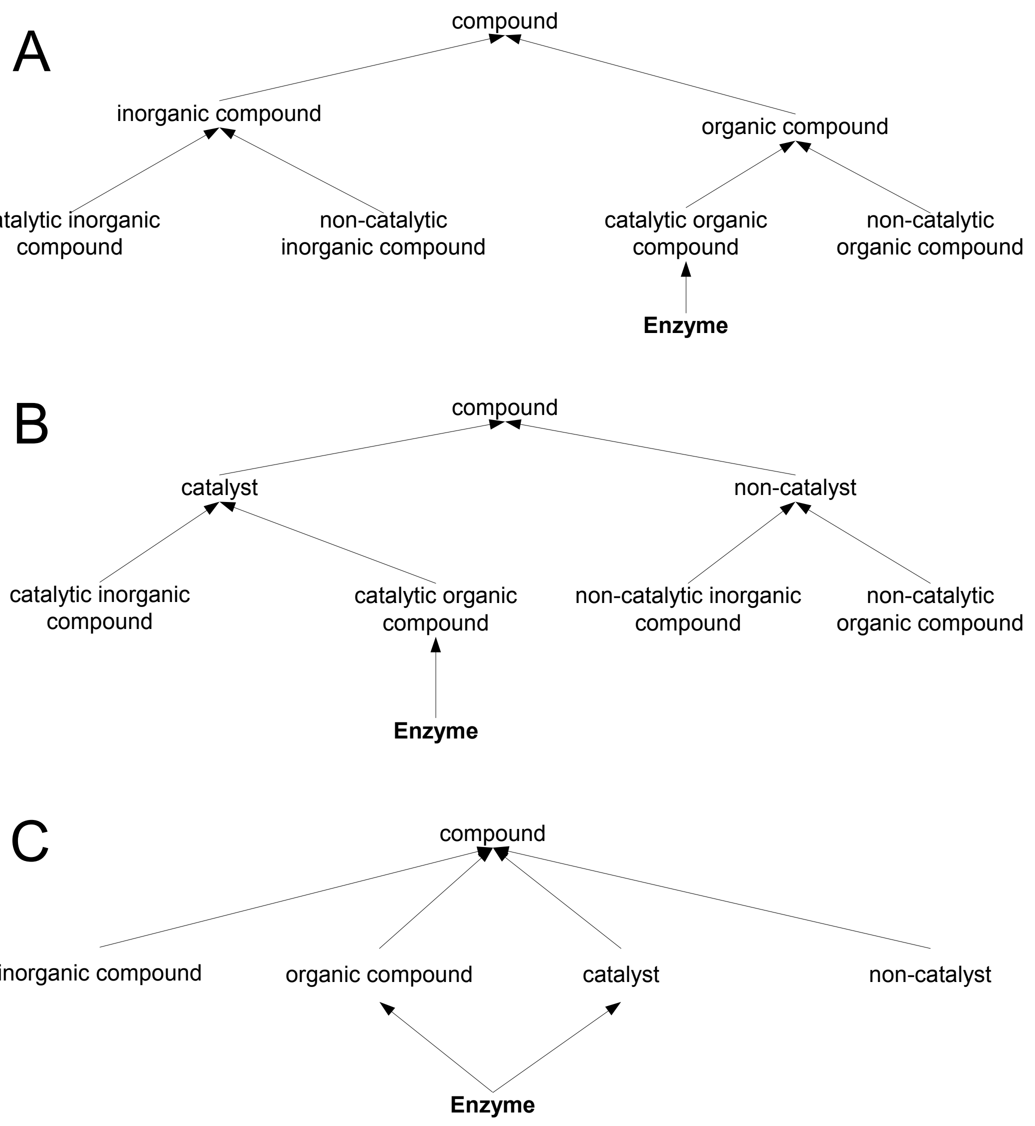

A
compound
inorganic compound
organic compound
catalytic inorganic compound
non-catalytic inorganic compound
catalytic organic compound
non-catalytic organic compound
Enzyme
B
compound
catalyst
non-catalyst
catalytic inorganic compound
catalytic organic compound
non-catalytic inorganic compound
non-catalytic organic compound
Enzyme
C
compound
inorganic compound
organic compound
catalyst
non-catalyst
Enzyme

Figure 28: Multiple inheritance versus single inheritance. By not using multiple inheritance, the structure of the ontology becomes arbitrary, and depends on which concepts are placed higher in the "is a" hierarchy (A versus B). When multiple inheritance is used (C), this problem does not occur. In addition, by using multiple inheritance, the ontology still represents the same facts but needs fewer concepts.

Complete Connectivity

All concepts should be part of the "is a" hierarchy. Although it is possible to introduce any new relation types to SEMEDA, SEMEDAs database query interface makes exclusively use of the "is a" hierarchy. By requiring that all concepts be interconnected, it is avoided that SEMEDAs ontology falls apart.

Branching level

To improve human readability of an ontology, each concept should not have too many sub-concepts in the next level, but also not too few. (Noy and McGuinness 2001) states: "If a class has only one direct subclass there may be a modeling problem or the ontology is not complete. If there are more than a dozen subclasses for a given class then additional intermediate categories may be necessary."

Concept names

Concept names should be self-explanatory, so that a user can identify the concept by its name without having to know its sub-concepts or super-concepts. For example, “organic compound” which is a sub-concept of “compound” should not just be named “organic” (see Figure 28). An exception are names of things: whereas a concept “enzyme identifier” might exist, a concept should not be labelled “enzyme name” but just “enzyme”.

Renaming Concepts

Ontology concepts may be modified even after database attributes have been defined as the concept. This is possible since SEMEDA internally identifies concepts by an identifier and not by their name. This is useful when a better name for a concept is found, to correct spelling mistakes or to improve concept definitions. However, when a concept is edited, the semantics of a concept should not change, i.e. "EC number" may be changed to "enzyme number" since it is just a different word for the same thing, but it may not be changed to "enzyme" which is an entirely different concept.

Concept Definitions

All concepts should have a natural language definition. In addition, when possible a linkout to an online dictionary or another external definition should be provided.

Often "implicit knowledge" is associated with the term of a concept. For example the definition of the term "Gene" is defined in the glossary of (Lewin 2000) as:

"Gene: (cistron) is the segment of DNA involved in producing a polypeptide chain; it includes regions preceding and following the coding region (leader and trailer) as well as intervening sequences (introns) between individual coding segments (exons)"

Although it is not explicitly stated in this definition, the fact that genes usually carry heredity information is generally implicitly associated with the concept. When ontologies are built, implicit knowledge plays a major role, since it is generally not feasible to provide all information about each concept in the definitions.

Capitalization

It is sometimes suggested that concept names be capitalised to improve readability (Noy and McGuinness 2001). Since this differs from written English and from most thesauri, SEMEDAs concept names were not generally capitalised, but only when the word would also have been capitalised in written English. This is practically the same approach, which is recommended by (Schulze-Kremer 2002).

Delimiters

Many concept names consist of more than one word, for example "organic compound" or "nucleic acid". Therefore ontology editors that cannot handle space characters use underscores or run the words together while capitalising each word (OrganicCompound). Using space characters is the most natural way of representing concept names. Since SEMEDA supports space characters, there was no reason not to use them.

Singular or Plural

For concept names always the singular is used.

British English

British English spelling was used. Although this is merely syntactic, by always using the same spelling, the user does not have to search for the same concept twice by using different spellings.

9.2 Implicit Database Table Semantics

Whereas in some cases the name of database attributes is not self-explanatory, the way in which attributes within one table relate to each other is usually obvious, once the semantics of the attributes is understood (see Table 8). This is due to the fact, that database tables usually have been designed carefully by humans. Such implicit table semantics work in most cases well, but is not very useful in highly decomposed database schemata such as of the EMP database (Selkov et al. 1996). EMP basically uses for each attribute a separate database table, although this database design is unique and would for any other purpose than that of the EMP project be considered to be bad database design.

Table 8: Attributes of the table GENE from RegulonDB (Salgado et al. 2001). Whereas for some attributes it is obvious from the table names what kind of data they contain (GENE_ID, GENE_NAME, GENE_SEQUENCE), for other attributes a concept name or a description is helpful (B_NUMBER, GENE_STRAND). However, it is usually clear how the attributes relate to each other, i.e. it is obvious that GENE_POSLEFT is a property of GENE which has the two identifiers GENE_ID and B_NUMBER.

Attribute Name	**Adequate Concept Name**	**Description**
GENE_ID	gene identifier	The identifier of the gene in RegulonDB (ID).
B_NUMBER	gene identifier	ID assigned to each gene.
GENE_NAME	gene	The name of the gene.
GENE_POSLEFT	gene position left	The left absolute position in the genome of the gene.
GENE_POSRIGHT	gene position right	The right absolute position in the genome of the gene.
GENE_STRAND	transcription direction	The direction of the transcription in relation to the orientation of the chromosome, forward means coding strand and reverse lagging strand.
GENE_SEQUENCE	DNA sequence	The gene sequence.
GENE_NOTES	gene description	Any comments about the gene

In general, database attributes store names, identifiers, properties or free text descriptions of real world objects. Names and identifiers serve to identify the real world objects, whereas the properties and descriptions store facts about those objects. Therefore “name”, “identifier”, “description” and “property” are used as top-level concepts, which were direct sub-concepts of the root concept from SEMEDAs main ontology (see below).

These characteristics could have been formalised by also using relation types "is identifier for", "is property of" and "is description of" which would connect sub-concepts of "identifier", "description" and "property" with sub-concepts of "name". Since SEMEDAs query interface does not use any other relation types but "is a", introducing such relation types creates only unnecessary overhead. In addition, during the work on SEMEDA, the experience was made that most people understand the data structure of an ontology, which uses only an "is a" hierarchy reasonably easy, but the existence of other relation types was often a source for confusion. Also in the GALEN project (Rector et al. 1998) it was learned that the data structure of ontologies could not be understood by most medical doctors.

Finally, by not relying on any other hierarchy than the "is a" hierarchy (class hierarchy, see section 3.2), compatibility with other ontologies and ontology editors is facilitated.

9.3 SEMEDAs "Main Ontology"

The top-level concepts of SEMEDAs main ontology are “name”, “identifier”, “description” and “property” and “literature reference data” (see Table 9).

Not all concepts, which are potentially useful for database integration, were created in SEMEDA, i.e. concepts were only added when they were needed to semantically define a specific database. Thus, at present SEMEDAs main ontology is small and easy to survey.

In addition to the “name”, “identifier”, “description” and “property” top-level concepts, the concept “literature reference data” was also used as a sub-concept of the root concept (see Table 9). At first the concepts, which are now below “literature reference data”, were sub-concepts of the other four top-level concepts. However, when a database table that contained literature data had to be defined, some concepts were found below name (journal), others below properties (page number) etc. Putting all “literature reference data” below one single top-level concept was possible, since for those data the semantic cross-references (see 3.3.4) could be generated without using the “is a” hierarchy, i.e. neither the sub-concepts nor the super-concepts of literature reference data could be used to generate semantic cross-references.

In SEMEDAs main ontology, the concepts "compound" and "substance" are considered to be synonyms, since most people do not properly differentiate those terms. The definition and usage of "substance" and "compound" varies even between chemistry textbooks. Since databases usually also do not discrimination these concepts, it was decided to use them as synonyms. Thus, this subtle impreciseness avoids inconsistent definitions of database attributes.

Whereas most databases use specific identifiers (such as for organisms, enzymes, diseases etc.), in few cases unspecific identifiers are used. For example the EMP database (Selkov et al. 1996) uses for each table an attribute COL_ID. COL_ID is used in a way that all entries that have the same COL_ID belong to the same entry. Since EMP entries can be completely different biological objects, such as diseases, enzyme numbers, metabolic pathways etc., a concept "unspecific identifier" is used as a sub-concept of "identifier".

Table 9: SEMEDAs five top-level concepts that are directly linked below the root concept, and some of their sub-concepts.

- **description**
  - enzyme description
  - prognosis description
  - therapy description
  - …
- **name**
  - body part
  - gene
  - organism
  - …
- **literature reference data**
  - issue (publication)
  - author
  - title
  - …
- **property**
  - unit
  - locus (gene)
  - sequence
  - …
- **identifier**
  - metabolic pathway identifier
  - disease identifier
  - unspecific identifier
  - …

9.4 Semantic Definition of Databases

By using SEMEDAs main ontology to semantically define database attributes, two problems are frequently met. Those problems, and how they can be overcome is described subsequently.

Semantically defining all database attributes of all tables of a database can be time-consuming, since often new concepts have to be added to SEMEDA. Therefore, SEMEDA was implemented in a way, that all database attributes of a database table for which at least one attribute is semantically defined can be searched. Thus, to make a database table accessible via SEMEDA, only one database attribute has to be semantically defined. Such, database attributes which are not semantically defined can only be addressed by their original name, and cannot be used to link to other databases by using semantic cross-references. Thus it is possible to make a databases accessible via SEMEDA by just a few mouse clicks, i.e. by semantically defining only their most common attributes such as EC numbers (NC-IUBMB 1992), CAS numbers (Buntrock 2001), OMIMs mim numbers (Hamosh et al. 2002) etc.

Another common situation is that some database attributes contain data of more than one concept. For example in the Metabolic Disease Database (Hofestädt et al. 1998, Freier et al. 2000.), in the database table REFERENCES the attribute ANNOTATION stores the journal, issue, volume and page number of publications. Such database attributes should be semantically defined as all relevant ontology concepts.

9.5 Custom Ontology versus Import

Thus it turned out, that building and using a custom ontology for SEMEDA worked well. By building a custom ontology, the structure and concepts of the ontology could be build to suit the requirements of SEMEDA much better than any of the existing ontologies. Although the SEMEDAs main ontology was kept as small as possible, and the structure of SEMEDA was kept as simple as possible by exclusively using the "is a" hierarchy, SEMEDAs main ontology differs from existing ontologies. Thus, the experiences made are similar like (Russ et al. 1999), who concludes that "it is difficult to simultaneously obtain high usability (i.e., adequacy for a specific use) and reusability (i.e., adequacy for several uses)".

10 Practical Applications of SEMEDA

In this chapter, the practical use of SEMEDA is described by two examples. The first section of this chapter shows how SEMEDA supports the building of user schemata for the BioDataServer. The second section describes how the clone database of the RZPD Berlin (Deutsches Ressourcenzentrum für Genomforschung GmbH) is connected to SEMEDA and thus linked to the other databases.

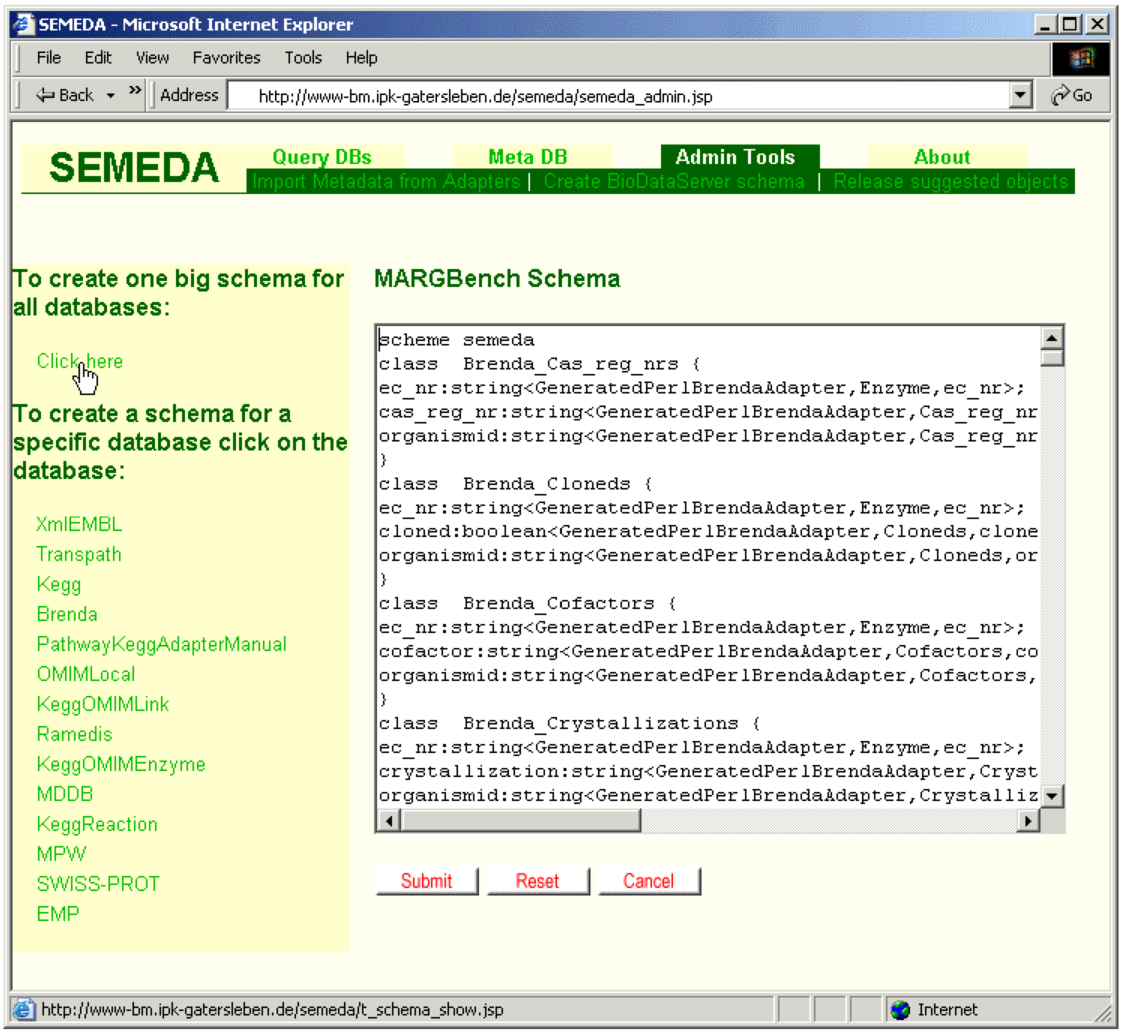
10.1 Modelling Integrated BioDataServer Schemata

Within the MARGBench project, the BioDataServer was developed as a mediator based solution for the integrated retrieval of molecular biological data (see also section 4.3). Since the BioDataServer is designed as a mediator, no single global integrated data schema exists, but a collection of user specific global views called user schemata. For building user schemata, the 'BDSSchemeEditor' has been developed within the MARGBench project. Besides editing user schemata, the BDSSchemeEditor supports loading and sending user schemata to/from the BioDataServer. In addition it checks the syntactical correctness of user schemata and supports schema editing by syntax highlighting and tree-view navigation elements.

User schemata are modelled according to user needs. Therefore, modelling user schemata requires the identification of an appropriate subset of all database tables and attributes from the mediated databases. At present (November 2002) more than 1000 database tables with a total of more than 3000 attributes are accessible via the BioDataServer. Therefore, identifying database attributes appropriate for a specific user schema is often difficult.

This task is supported by SEMEDA. To identify relevant database attributes, SEMEDAs semantic database definitions are used and displayed (see Figure 22). To do so, the appropriate ontology concept is selected, in order to display a list of all databases that contain relevant data. If for example database tables are required that contain enzyme data, the ontology concepts 'enzyme' and 'enzyme identifier' can be selected in order to display a list of tables that can be searched using enzyme names or EC Numbers.

Besides identifying appropriate database tables and attributes, SEMEDA is used to automatically generate user schemata (see also section 7.5). The syntax for user schemata is quite complex. This is due to the intricacy of the mechanisms for user schema based multi-database views (Freier et al. 2002b). Therefore SEMEDAs automatically generated schemata can be used to serve as templates, which are copied to the BDSSchemeEditor where they can be adjusted according to user needs. Figure 29 shows such an automatically generated user schema that was copied to the BDSSchemeEditor (Figure 30).


SEMEDA - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address
http://www-bm.ipk-gatersleben.de/semeda/semeda_admin.jsp
Go
SEMEDA
Query DBs
Meta DB
Admin Tools
About
Import Metadata from Adapters | Create BioDataServer schema | Release suggested objects
Submit
Reset
Cancel
http://www-bm.ipk-gatersleben.de/semeda/t_schema_show.jsp
Internet

To create one big schema for all databases:

Click here

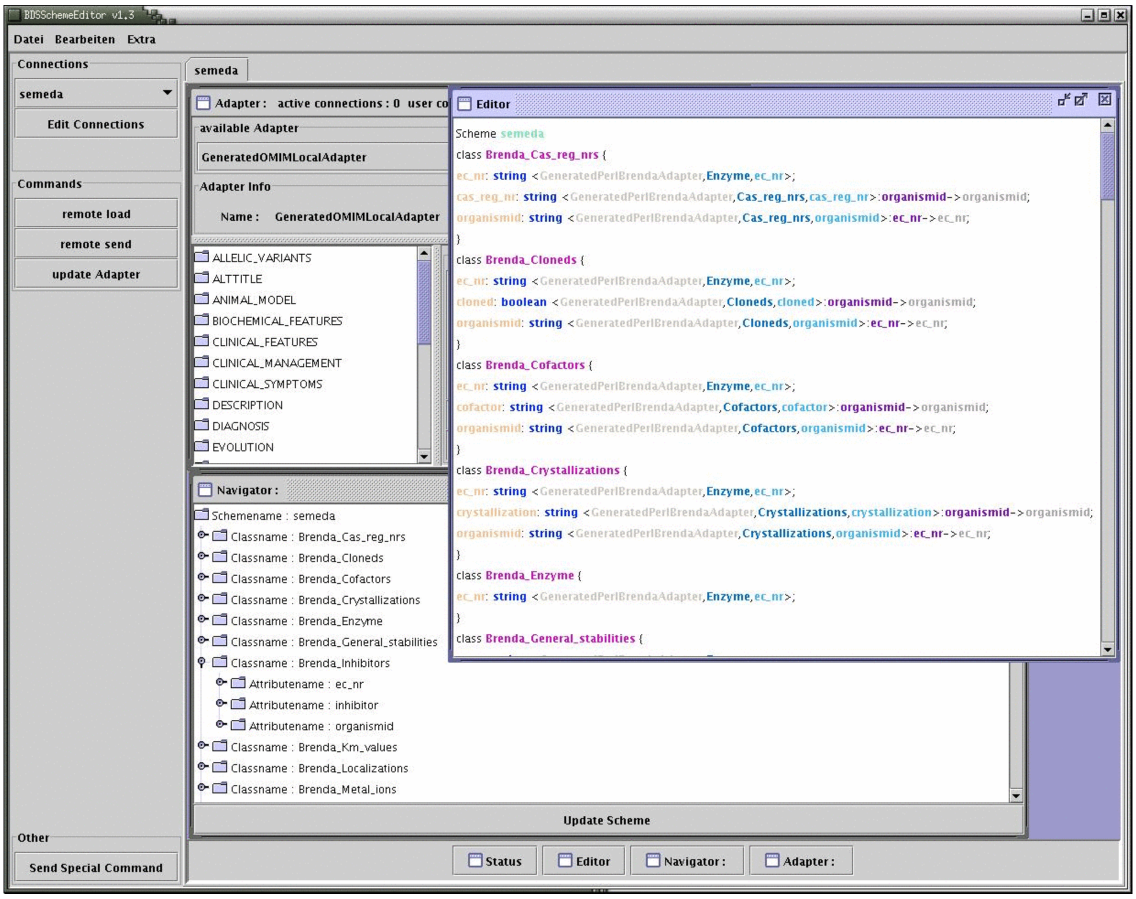
To create a schema for a specific database click on the database:

- XmlEMBL
- Transpath
- Kegg
- Brenda
- PathwayKeggAdapterManual
- OMIMLocal
- KeggOMIMLink
- Ramedis
- KeggOMIMEnzyme
- MDDB
- KeggReaction
- MPW
- SWISS-PROT
- EMP

MARGBench Schema

```
scheme semeda
class  Brenda_Cas_reg_nrs {
ec_nr:string<GeneratedPerlBrendaAdapter,Enzyme,ec_nr>;
cas_reg_nr:string<GeneratedPerlBrendaAdapter,Cas_reg_nr
organismid:string<GeneratedPerlBrendaAdapter,Cas_reg_nr
}
class  Brenda_Cloneds {
ec_nr:string<GeneratedPerlBrendaAdapter,Enzyme,ec_nr>;
cloned:boolean<GeneratedPerlBrendaAdapter,Cloneds,clone
organismid:string<GeneratedPerlBrendaAdapter,Cloneds,or
}
class  Brenda_Cofactors {
ec_nr:string<GeneratedPerlBrendaAdapter,Enzyme,ec_nr>;
cofactor:string<GeneratedPerlBrendaAdapter,Cofactors,co
organismid:string<GeneratedPerlBrendaAdapter,Cofactors,
}
class  Brenda_Crystallizations {
ec_nr:string<GeneratedPerlBrendaAdapter,Enzyme,ec_nr>;
crystallization:string<GeneratedPerlBrendaAdapter,Cryst
organismid:string<GeneratedPerlBrendaAdapter,Crystalliz
```

Figure 29: SEMEDA can automatically generate BioDataServer schemata either for all databases that are semantically defined in SEMEDA or for individual databases.


BDSSchemeEditor v1.3
Datei Bearbeiten Extra
Connections
semeda
Edit Connections
Commands
remote load
remote send
update Adapter
Other
Send Special Command
semeda
Adapter : active connections : 0 user co
available Adapter
GeneratedOMIMLocalAdapter
Adapter Info
Name : GeneratedOMIMLocalAdapter
ALLELIC_VARIANTS
ALTTITLE
ANIMAL_MODEL
BIOCHEMICAL_FEATURES
CLINICAL_FEATURES
CLINICAL_MANAGEMENT
CLINICAL_SYMPTOMS
DESCRIPTION
DIAGNOSIS
EVOLUTION
Navigator :
Schemename : semeda
Classname : Brenda_Cas_reg_nrs
Classname : Brenda_Cloneds
Classname : Brenda_Cofactors
Classname : Brenda_Crystallizations
Classname : Brenda_Enzyme
Classname : Brenda_General_stabilities
Classname : Brenda_Inhibitors
Attributename : ec_nr
Attributename : inhibitor
Attributename : organismid
Classname : Brenda_Km_values
Classname : Brenda_Localizations
Classname : Brenda_Metal_ions
Update Scheme
Status
Editor
Navigator :
Adapter :
Editor

```
Scheme semeda
class Brenda_Cas_reg_nrs {
ec_nr: string <GeneratedPerlBrendaAdapter,Enzyme,ec_nr>;
cas_reg_nr: string <GeneratedPerlBrendaAdapter,Cas_reg_nrs,cas_reg_nr>:organismid->organismid;
organismid: string <GeneratedPerlBrendaAdapter,Cas_reg_nrs,organismid>:ec_nr->ec_nr;
}
class Brenda_Cloneds {
ec_nr: string <GeneratedPerlBrendaAdapter,Enzyme,ec_nr>;
cloned: boolean <GeneratedPerlBrendaAdapter,Cloneds,cloned>:organismid->organismid;
organismid: string <GeneratedPerlBrendaAdapter,Cloneds,organismid>:ec_nr->ec_nr;
}
class Brenda_Cofactors {
ec_nr: string <GeneratedPerlBrendaAdapter,Enzyme,ec_nr>;
cofactor: string <GeneratedPerlBrendaAdapter,Cofactors,cofactor>:organismid->organismid;
organismid: string <GeneratedPerlBrendaAdapter,Cofactors,organismid>:ec_nr->ec_nr;
}
class Brenda_Crystallizations {
ec_nr: string <GeneratedPerlBrendaAdapter,Enzyme,ec_nr>;
crystallization: string <GeneratedPerlBrendaAdapter,Crystallizations,crystallization>:organismid->organismid;
organismid: string <GeneratedPerlBrendaAdapter,Crystallizations,organismid>:ec_nr->ec_nr;
}
class Brenda_Enzyme {
ec_nr: string <GeneratedPerlBrendaAdapter,Enzyme,ec_nr>;
}
class Brenda_General_stabilities {
```

Figure 30: Screenshot of the BDSSchemeEditor. The user schema displayed in the editor was automatically generated using SEMEDA (see Figure 29) and serves as a template that is adjusted according to user requirements.

10.2 Integration of the RZPD Clone Database

Connecting the RZPD database directly via JDBC to SEMEDA serves to demonstrate that real world databases that contain business-critical and confidential data can safely use SEMEDA to provide semantically integrated real time access to their data.

By connecting the RZPD clone database to SEMEDA, the clone information can be directly linked to other databases within the system using HUGO gene names and Gene Accession numbers. At present (November 2002), this allows to link RZPD clones, SWISSPROT protein data and MDDBs metabolic disease data using HUGO gene names. As soon as other databases that use HUGO gene names or GenBank accession numbers are connected to SEMEDA, these databases are also automatically linked to RZPD clone data.

In addition, connecting the RZPD database to SEMEDA allows the RZPD database and all other databases available in the system to be queried using one and the same query interface.

10.2.1 The RZPD Clone Database

The RZPD is an important provider of clones. According to information of the RZPD, the RZPD (http://www.rzpd.de/about/)

> "… harbors one of the most comprehensive clone collections world-wide. 225 cDNA libraries and 126 genomic libraries from 32 organisms contain about 30 million clones, which are arrayed in 384-well microtiter plates. Each clone stored in our freezers can be instantly addressed via the Primary Database regarding storage location, condition for distribution, additional data available etc. Since RZPD is one of only 5 authorized distributors of I.M.A.G.E. ESTs, this collection includes with more than 4.6 million clones, the largest public EST-collection currently available."

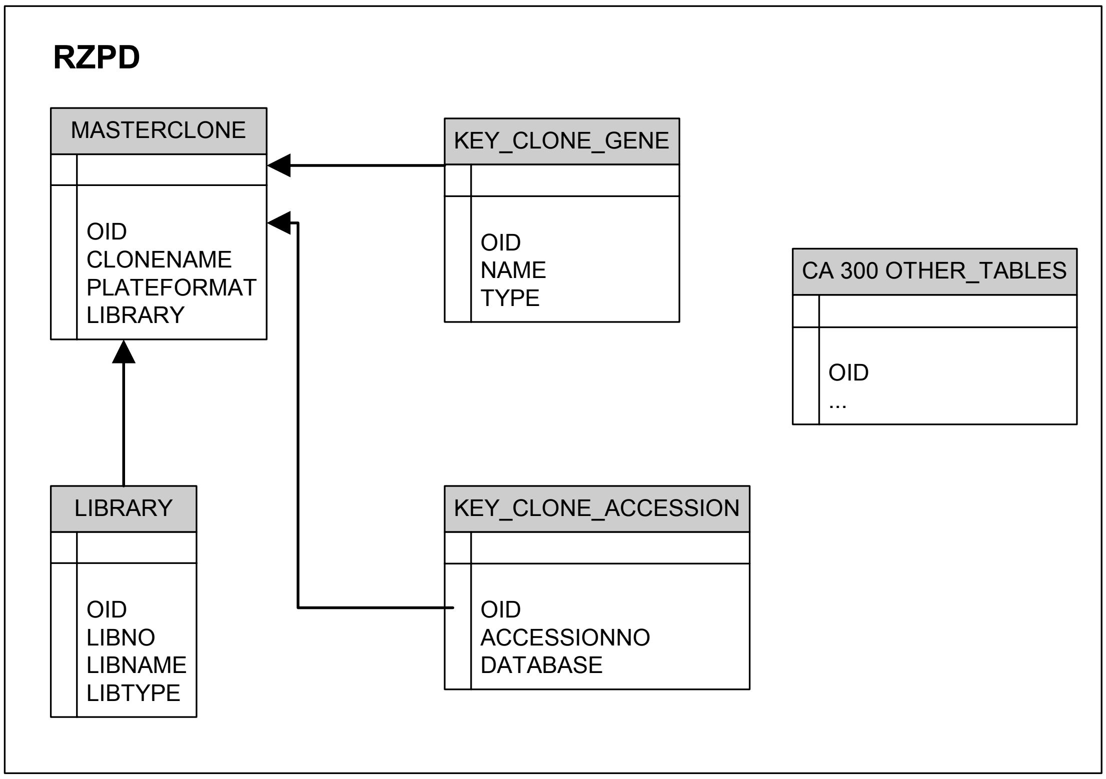


Figure 31: Database schema of the RZPD tables that are connected to SEMEDA (IDEF1X notation (Bruce 1992)).

The RZPD database is deployed on Oracle 8i and consists of more than 300 database tables. Many of those tables contain data such as timestamps, customer data and order status of clones. Such tables are either not of general interest or strictly confidential. Therefore, only a subset of the RZPD database tables was connected to SEMEDA. In addition, data and system security had to be guaranteed.

The core of the RZPD clone database consists of four database tables (see Figure 31). The table MASTERCLONE lists all clones that are available via the RZPD. The tables KEY_CLONE_ACCESSION links clones to GenBank/EMBL/DDBJ accession numbers, the table KEY_CLONE_GENE links clones to HUGO gene symbols (Cotton and Horaitis 2002) and the table LIBRARY stores information about the clone libraries of the respective genes. For the connection to SEMEDA, the HUGO gene symbols and the GenBank/EMBL/DDBJ accession numbers are especially important, since they are well suited to link the RZPD clone data to other databases of SEMEDA.

10.2.2 Connecting the RZPD to SEMEDA

The following steps are required to connect the RZPD database to SEMEDA (see also Figure 12 and chapter 7):

RZPD database administrator

1) Open the firewall to SEMEDA.

2) Create a RZPD database account for SEMEDA. This account is only granted read access to selected database tables. In addition it can also create views.

3) An administrative RZPD account creates views for those tables that are appropriate to be accessed via SEMEDA and that filter out confidential data. Read only access to those views is granted to SEMEDAs RZPD account.

4) Create views that filter out irrelevant data using SEMEDAs RZPD account.

SEMEDA administrator

5) Generate an adapter for the RZPD database.

6) Import database Meta Data from the adapter to SEMEDA.

7) Semantically define the RZPD database tables in SEMEDA.

8) Test whether the RZPD database is correctly connected to SEMEDA using SEMEDAs database query interface.

The practical work for connecting the RZPD clone database tables to SEMEDA required an estimated four hours of work from the RZPD database administrators (steps 1-4), and about the same amount of time from the SEMEDA administrators (steps 5-8). Thus, connecting the RZPD database required comparatively many steps. When other databases were added, many of those steps were not required. Step 1 can be omitted when the database port of the firewall is already open. In many databases a "guest account" already exists that provides read only access to selected database tables and entries. In such databases, steps 2-4 are not required.

However, although the amount of time required to connect databases to SEMEDA is comparatively low, it should be mentioned that usually additional time is required for consultation and discussing security issues with administrators and officials in charge of the database to be connected to SEMEDA.

In principle, it would have been possible to merge the four RZPD database tables using one or two views. Although this would have improved the understandability of the database tables, this was not possible due to serious performance deficiencies that occur in Oracle when views are used in conjunction with the outer join operator (see also http://home.clara.net/dwotton/dba/ojoin2.htm).

11 Discussion

11.1 Comparison with other Systems

11.1.1 Ontology Editors/Browser

At the beginning of this section, SEMEDAs ontology editor is compared to GOs tools for ontology editing and browsing, since GO is the best known and most widely used ontology in the field of molecular biology. GO and its ontology editing tools are being developed since 1999. At present (November 2002), 18 GO tool developers are registered at http://sourceforge.net/projects/geneontology.

Figure 32 compares collaborative ontology editing between SEMEDA and the Gene Ontology tools. Both SEMEDA and the Gene Ontology can use a relational database to store the ontology. In SEMEDA, all users who edit the ontology work on the same instance of the ontology, whereas in the GO tools the ontology is imported to the ontology editor, where it is edited and finally written back to the database. When several users edited ontologies using the GO tools, conflicts between different ontology versions have to be resolved. In addition, in SEMEDA users see changes that other users apply to the ontology immediately, whereas in the GO tools modifications are only seen when the ontology is written back to the database. Collaborative ontology editing is also facilitated in SEMEDA by the implemented “review process” most users may only “suggest” concepts, and only administrative accounts can release newly suggested concepts.

Another advantage of SEMEDAs architecture is, that SEMEDA has the potential to handle large ontologies, since it does not require that the whole ontology be loaded to the frontend. Whereas this might be not important for the current size of GO, editing large ontologies with millions of concepts using DAG-edit is most probably impossible.

Whereas GOs DAG-edit is mainly used for editing ontologies, several html based GO ontology browsers exist. SEMEDAs performance could not directly be compared to the GO browsers. The fact that SEMEDA is fast at tree-processing is due to the fact, that SEMEDA uses an efficient proprietary feature of the Oracle database for tree processing (see section 6.1.4.1), whereas the GO browsers either generate trees from the database backend by several iterative SQL queries, or they load the whole ontology to the middle tier application, which dynamically generates the html frontend. In simple words, SEMEDA can generate a hierarchy within the database using one transaction, whereas the GO browsers either need approximately one database transactions per ontology concept, or the middle tier has to load the whole ontology to memory.

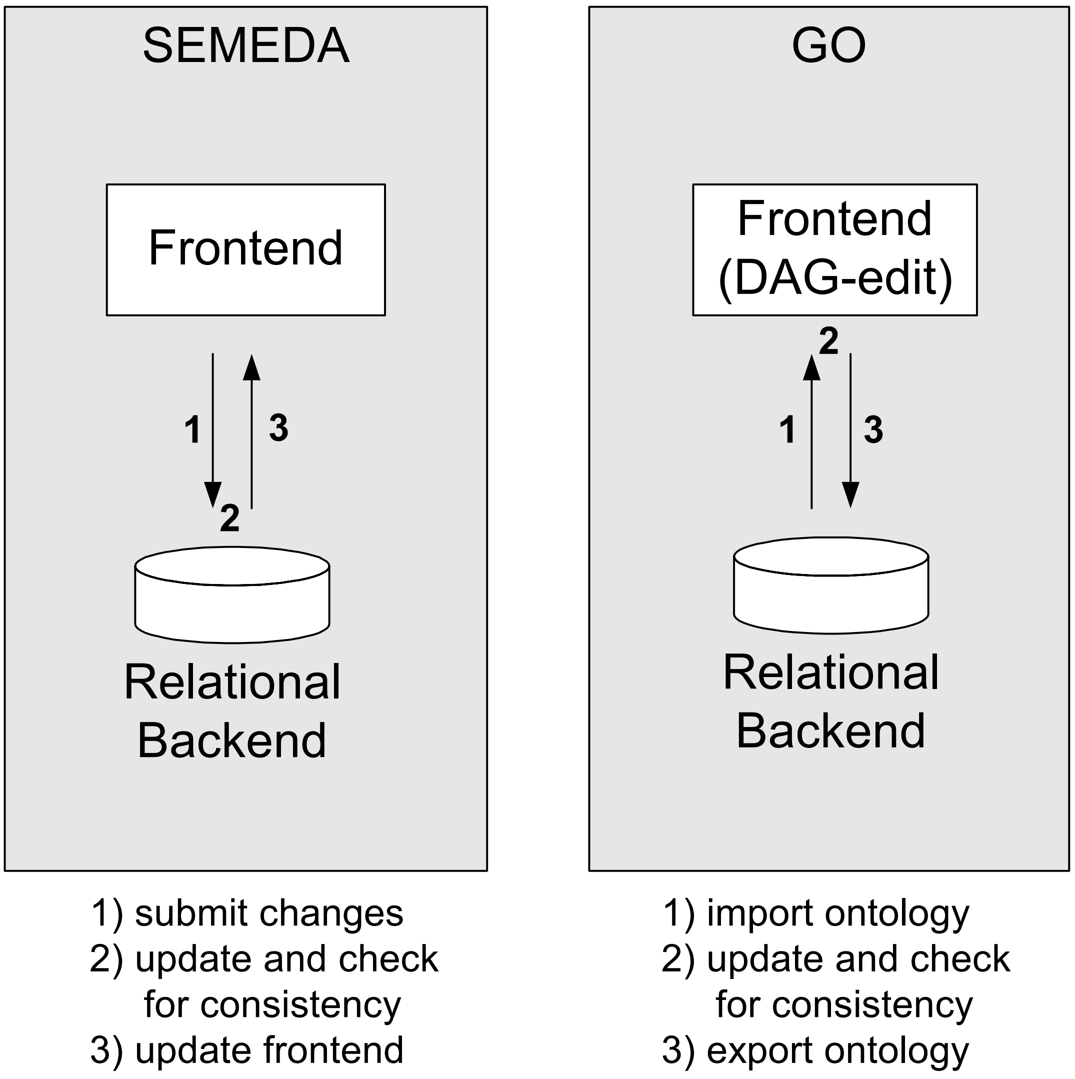


Figure 32: Comparison of collaborative ontology editing between SEMEDA and the Gene ontology tools. In SEMEDA, all users who edit the ontology work on the same instance of the ontology, whereas in the GO tools the ontology is imported to the ontology editor, where it is edited and finally written back to the database. When several users edit ontologies using the GO tools, conflicts between different ontology versions have to be resolved when the ontology is written back to the database.

Whereas SEMEDAs strength lies in collaborative editing of large ontologies, some other ontology editors are richer in features. SEMEDAs notion of an ontology is similar to RDF ontologies, see section 3.2. RDF is sometimes described as a "lightweight ontology format", i.e. whereas several research groups have developed different data structures with different features, RDF is not the language which unites all those features, but rather serves as the "smallest common denominator" of most ontologies. Therefore, most ontology editors use their own format for storing data (see section 2.4.2). Although most ontologies can be imported to SEMEDA by dropping information, SEMEDA does not support all advanced features of different existing ontology editors. For example in the PROLOG based OE ontology editor (Schulze-Kremer 1997b), powerful constraints can be declared using PROLOG.

11.1.2 Database Integration Systems

SRS is the most widely used molecular biological database integration system. SRS provides many useful features that are not implemented in SEMEDA, such as for example views, the integration of bioinformatic analysis tools/applications, and the download of results in different data formats. On the other hand, SRS does not support semantic database integration as introduced in SEMEDA.

In a limited way, in SRS attribute semantics can be defined by assigning equivalent database attributes equal names, although this does not systematically cover the situation where one database attribute is more general than another. An example would be an attribute organism that stores species names of plants, animal and fungae versus an attribute animal that stores only names of animal. In SRS such situations cannot be treated in an appropriate way, whereas in SEMEDA animal could be defined as a sub-concept of organism. SRS also does not provide a standardised set of attribute names and it is up to the database provider to define the name for a database attribute in the Icarus script he has to provide for indexing the database. In SEMEDA, all database attributes are semantically defined versus the same main ontology. When a database provider does not find an appropriate concept for a database attribute in SEMEDA, he can add a new concept which can subsequently also be used by other database providers.

SRS does not differentiate between different vocabularies used in equivalent attributes of different databases. One database might for example use English species names for an attribute “organism”, whereas another database might use systematic species names. In SEMEDA the vocabulary used can be defined and mechanisms for translating between different controlled vocabularies are suggested (see section 3.3.3).

In SRS indexing scripts, attributes can be directly linked to attributes of other databases. Thus, each database would have to provide links to all other relevant databases. Considering the fact, that at present more than 500 databases can be integrated using SRS, a database provider would have to know all other databases in order to be able to decide to which other databases links can be provided. In SEMEDA, database attributes only have to be defined once to the main ontology. Based on these definitions and the mechanism for generating semantic cross-references, links to all other relevant databases can be generated automatically. If n is the number of database attributes which can be used for linking between databases, the number of integration steps (database links to be defined) in SRS increases exponentially with the number of attributes to be interlinked (integration steps = (n*(n-1))/2), whereas in SEMEDA it is sufficient to define each database attribute only once versus the main ontology (integration steps = n). This applies also to other database integration systems such as for example PEDANT (Frishman et al. 2001). The only system that has similar link generation capabilities like SEMEDA is KEGGs DBGET/LinkDB (Fujibuchi et al. 1998). DBGET/LinkDB automatically generates links between databases using EC Numbers and EMBL/Genebank Accession Numbers, but unlike SEMEDA it cannot use any other database attributes for link generation.

Subsequently existing systems that have the aim to semantically integrate databases will be discussed and compared to SEMEDA. These systems are mainly TAMBIS (Stevens et al. 2000a) and a system where the semantics of databases are defined and mapped using F-LOGIC (Ludäscher et al. 2001).

Several differences between SEMEDA and these two systems exist. SEMEDA makes only use of the “is a” hierarchy for defining database attributes, whereas both other systems also use several other relation types. By using several other relation types, the semantic definitions are more precise, whereas in SEMEDA by not using relation types that are specific to a particular application domain, SEMEDAs database integration principles are more generic. Another reason why for SEMEDA only the “is a” hierarchy was used, is that this made it possible to keep the work required for semantic database definitions on the one hand minimal, and on the other hand simple enough that it is possible to define the semantics of database attributes via a relatively easy to use a GUI frontend.

In contrast to the two other systems, SEMEDA does not require that all attributes of a table have to be semantically defined, in order to make them accessible via the integration system. SEMEDA avoids this, by using the table structures and by relying on “implicite table semantics” (see section 9.4) of the underlying databases. Thus only one database attribute per table has to be semantically defined to make a table accessible via SEMEDA. This reduces the work of adding or removing databases significantly. In addition to this, databases can be added to SEMEDA at runtime without the need for source code modifications.

To query the system described in (Ludäscher et al. 2001), F-LOGIC is used as a query language. F-LOGIC is very expressive, but requires programming language skills. Therefore, form based HTML frontends are used on top of predefined F-LOGIC queries. However, these HTML frontends are highly specific to queries and data sources.

The TAMBIS system uses a graphical representation of the underlying ontology to interactively build database queries. Although this frontend is powerful, building queries is not trivial and may be too difficult for medical doctors, i.e. in the GALEN project (Rector et al. 1998), it was learned that medical doctors had difficulties understanding the data structure of ontologies.

As a consequence of the difficulties in building queries and query frontends with existing ontology based systems for semantic database integration, the methods introduced in the course of this thesis, as well as the development of SEMEDA, was driven by ergonomic needs of non computer scientists. In consequence, query frontends are automatically generated once a database has been semantically defined. Furthermore, the user of SEMEDA's query interface is not confronted with the data structure of the underlying ontology.

Recently, (Hakimpour and Geppert 2001) suggested an approach where the semantics of each database in a separate ontology. This creates much unnecessary overhead, since it requires ontology merging and does not facilitate that the databases use and maintain the same ontology. Ontology merging by itself is a nontrivial task which can only partly be automated and requires much human intervention (Noy and Musen 1999, McGuinness et al. 2000, Stumme and Maedche 2001). Therefore the suggested system (Hakimpour and Geppert 2001) probably causes more problems than it solves.

Although the work to semantically define databases was minimised as much as possible in SEMEDA, it would be useful if this step could be automated. Methods for automated schema matching are reviewed in (Raham and Bernstein 2001), and (Li et al. 2000b) implemented a system where neural networks are used to automatically map database schemata. However, although such automated schema matching performs in many situations well, it does not always align database schemata correctly and its performance depends much on the database schemata to be matched. Therefore, at least at present manual schema matching as applied in SEMEDA, can in the best case only be supplemented, but not be substituted by automatic schema matching algorithms.

11.2 Outlook

11.2.1 Improving Performance using a Database Mirror

For the use of SEMEDA, two application scenarios are possible: the implemented scenario is useful for a workgroup of scientists who need to query source databases in real-time and where slow response times are acceptable. The reasons why the BioDataServer responds slowly to some database queries are discussed in section 4.3. To use SEMEDA as a system, which is intensively used by many people via the internet, i.e. as a alternative to indexing database integration systems, speedier data access than that which is intrinsic to database integration by database mediation (Davidson et al. 1995) would be required. This could be achieved by using a database mirror (Figure 33).

Such a system mirrors the database tables, maps the datatypes of the source databases to the datatypes of the mirroring DBMS but drops any other information such as procedures, functions and other features, which are generally DBMS specific. Metadata (database schema information) can be imported via JDBC or ODBC to SEMEDA where this metadata can be supplemented, semantically defined and adjusted to the requirements of the mirroring DBMS. Subsequently this schema information can be used to create a mirror schema on the mirroring DBMS, which can then be filled with data via JDBC. Heterogeneous data sources, which are connected via the BioDataServer, can in some cases not be directly mirrored, due to the lack of an underlying relational schema, i.e. the BioDataServer may have to implement an appropriate relational schema.

It avoids schema conflicts, since the generation of one big integrated database schema is not required. Therefore, in principle, any number of databases can be integrated, although this approach is limited by disk space and the available computer hardware. Database federation systems like IBMs DiscoveryLink (Haas et al. 2000, Haas et al. 2001), adjust query plans to take into account specific costs for querying different types of heterogeneous data sources, but still cannot completely rule out performance issues which are intrinsic to database integration by database federation. The proposed database mirror could automatically make use of built in query optimisation mechanisms of the mirroring DBMS. In addition, SEMEDAs semantic cross-references can be used to create indexes on relevant columns.

In addition, by implementing a database mirror the user interface could be implemented much more powerful, since it would not be restricted by the limited SQL subset the BioDataServer has implemented. Thus, as a side effect, SQL 92 like pattern matching and datatype operators (<, >, <=, >= etc.) could be used for the implementation of the database query forms.

This architecture would be especially useful, when source databases can be accessed by JDBC or other systematic access to database data and metadata is available, although by using the BioDataServer, heterogeneous data sources can also be accessed. At present time, due to security concerns, most database owners do not grant JDBC access to their databases. However, nowadays all major DBMS have mechanisms to restrict user rights. Usually it is possible to restrict read only access to a few tables and to hide table attributes and tablerows by using views. In addition, JDBC type 3 drivers have recently become available for all major DBMSs http://industry.java.sun.com/products/jdbc/drivers. JDBC type 3 drivers usually support encryption and can also be used to deny write access.

Several database providers do not grant direct access to their database as a measure to protect their intellectual property. They often spent several man-years for the generation of the database, and copyright protection by law is weak in most countries (Maurer et al. 2001). Those databases can also not be systematically accessed by other integration systems. However, a high percentage of the databases listed in (Baxevanis 2002) are freely distributed, usually as tab-delimited flatfiles, SQL database dumps, XML etc. Obviously, the owners of those databases do not mind to share their data.

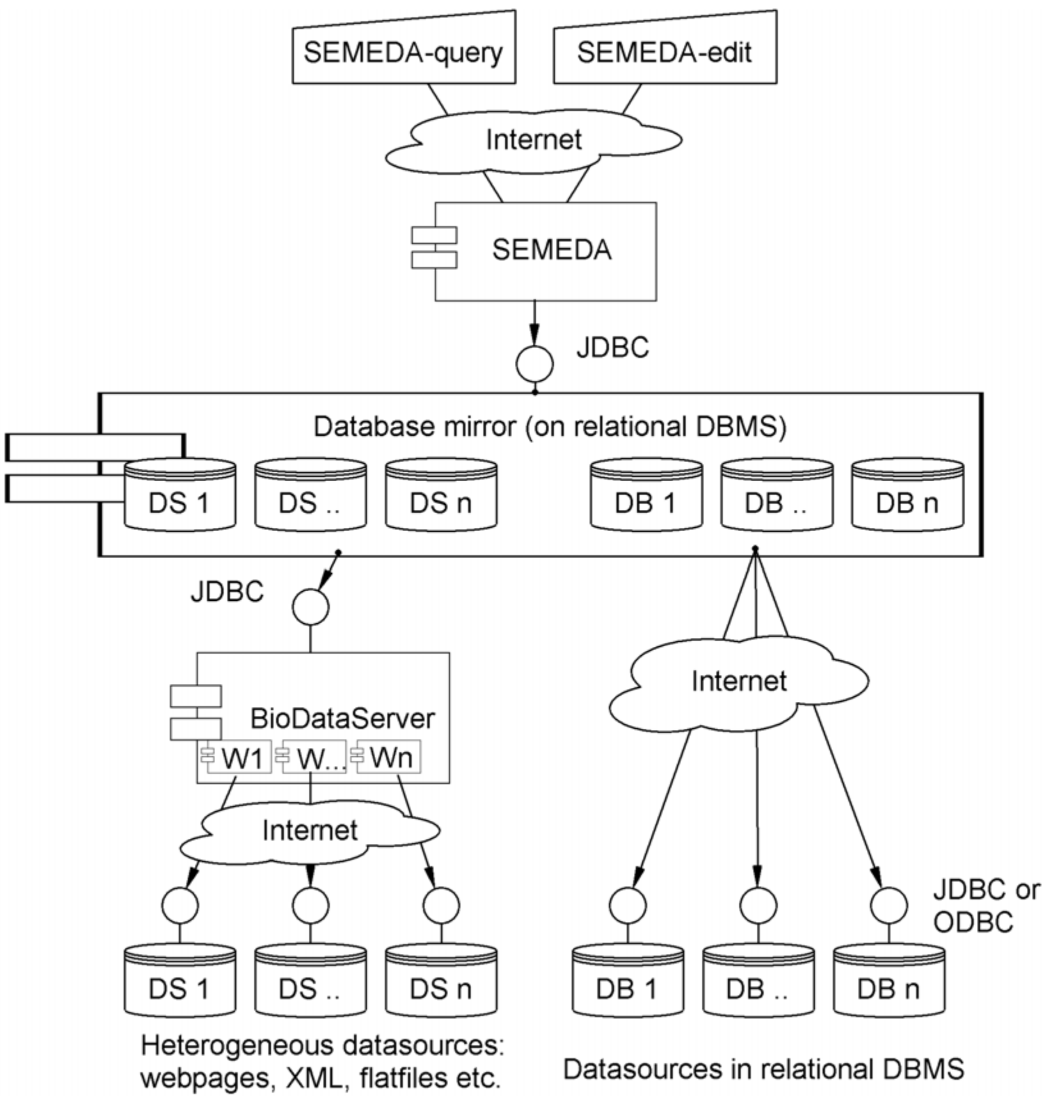


Figure 33: Suggested architecture for database mirrors. Whereas in the implemented architecture (Figure 17) queries are processed against the mediated source databases, in this system queries would be processed against a local copy of the source databases. The mirror is updated at intervals.

When downloading much data from the same source, many database providers blocked the BioDataServer. This may be due to two reasons: 1.) the database provider does not want that all his data is downloaded systematically. 2.) the download of the data is causing too much load on the server. In the latter case, this might be avoided by updating the different data sources not one after the next, but rather by updating them at the same time. For example, search engines do not crawl a source in one step, but rather retrieve one page per five minutes. In a system where data sources are accessed directly, this would be too slow for user interaction, but by using a database mirror this could be implemented.

Whereas indexing systems such as SRS (Etzold et al. 1996) (Zdobnov et al. 2002) require that proprietary flatfiles are built at regular intervals, and that scripts for the integration systems are provided, the work required to install and configure JDBC type 3 drivers is low. Therefore it is likely that in the near future JDBC and other direct data exchange methods will be used between molecular biological databases, not only within, but also between institutions. In addition, the fact that the discussed mirror is based on a relational DBMS makes it possible to use all built in methods and extensions of the DBMS on the mirrored databases. Thus, such a relational database mirror would be much more flexible than indexing systems.

One might argue that the suggested architecture might not be feasible since hard disk size might limit the number of databases to be mirrored. The simple fact that the SRS implementation of the EBI stores about 100 flatfile databases on 300 GB of hard disk, indicates that the amount of data is not too high to be mirrored in relational DBMS.

Recently, a prototypic database mirror based on the principles discussed in this section was implemented within a Diploma thesis (Reinke 2002). Due to some shortcomings of this prototype it is not yet really useful. For example, at present it can only mirror database tables with 6 or less database attributes, database tables which are being updated cannot be accessed during the update process which may take several hours, and since it does not use a systematic naming convention for database tables, conflicts would occur if two database tables of different databases use the same name. In spite of these shortcomings, the result of the database mirror are promising: whereas retrieving all EC numbers and enzyme names from BRENDA via the BioDataServer takes more than 10 hours, the same query took less than a minute via the database mirror. This example query was executed via a proprietary query interface of the database mirror. Even shorter response times can be achieved when the mirrored database tables are accessed directly via the mirroring DBMS, which was in this case Oracle.

11.2.2 Extensions of SEMEDA

In this section, several features are listed which would be useful in SEMEDA. Many of those features already exist in other database integration systems and are therefore not implemented in SEMEDA, i.e. they are not innovative and the aim was to develop a prototype, which demonstrates the potential of semantic database integration.

Useful features for ontology editing are finer granularity of user rights, history tracking of ontology editing and merging of ontology concepts. At present each ontology can only be edited by one user at the same time, and merging ontology concepts like in GOs DAG-edit is not implemented. In order to use SEMEDA to collaboratively edit large ontologies, user permissions, it is necessary to restrict user permissions further. For example, at present everyone who has the permission to edit ontologies, is allowed to add new concepts or relations to all ontologies in SEMEDA. In addition, SEMEDA does not track changes, i.e. does not keep a history of the changes of the ontology, although Oracle can be configured to protocol all changes of data in SEMEDA.

Subsequently, several features for the database query interface are discussed.

Multi Database Views

Multi Database Views require integration of data across several database tables of the same database or of different databases. Thus Views often use the SQL join operator or aggregate functions. The join operator in the BioDataServer is slow, since it requires downloading data from several data sources and since indexes are not available in mediated systems (see section 4.3). Therefore a prerequisite for the implementation of Multi Database Views would be the use of a database mirror as discussed in the previous section.

Translation Lists and Translation Functions

Another feature that requires a database mirror to be implemented efficiently are the concepts of Translation Lists and Translation Functions, which were introduced in section 3.3.3. The best way to implement translation lists would be by mirroring database tables that map different controlled vocabularies. These mirrored tables could then be used to translate between different vocabularies.

Translation Functions could best be implemented using DBMS specific scripting languages, such as PL/SQL in Oracle or PGL/SQL in Postgres. Such functions can then be used within SQL statements. By using a scripting language of the DBMS, such functions could benefit from internal compiler optimisations of the DBMS.

Multi Database Searches

Whereas at present the ontology in SEMEDA is among other things used to guide the user to appropriate tables, it is not possible to query all relevant database tables/attributes in SEMEDA in one step. Even querying “one step accessible” attributes via the BioDataServer usually takes several seconds, i.e. querying several database tables in one step would be too slow for human interaction. Again, by using a database mirror this problem could be overcome.

Data formats

At present, in SEMEDA data is displayed using HTML tables. Although this is an appropriate format for humans, it might be useful to provide data in various formats such as XML or comma separated flatfiles.

Pattern search and comparison operators

The SQL implementation of the BioDataServer only supports the = operator in the SQL where clause. Thus for example, free text comments cannot be searched for the occurrence of a specific keyword. In addition, since no other comparison operators than the = operator is implemented in the BioDataServer, it is for example not possible to search an attribute "enzyme temperature optimum" for all entries where the optimum temperature is above a certain value.

Database Pooling

At present, when SEMEDA connects to a relational database directly and not via the BioDataServer, each time a user submits a database query, a connection to the respective database is opened and closed again, after the result is retrieved. When connection pooling is applied, a predefined number of database connections are permanently kept open. The time required for opening and closing JDBC depends on the database, the internet connection and the JDBC driver. Normally establishing JDBC connections takes less than 2 seconds.

Linkouts

At present, SEMEDA only provides links to other databases within SEMEDA. Links to external databases are not provided, although it would be possible to automatically generate HTML links to external databases using EC numbers or Gene Accession numbers.

11.2.3 Database Integration in Other Knowledge Domains

Whereas SEMEDA was implemented with molecular biological databases in mind, databases from other knowledge domains could also be integrated (Ecology, Chemistry, Agriculture, GIS, Stock-Market related databases, socio-economic etc.). The top-level structure of SEMEDAs main ontology (see Table 9) is independent of the type of databases to be integrated. SEMEDAs four SEMEDAs approach is especially appropriate to semantically integrate databases that come from different knowledge domains. Thus it would be possible to connect data, which usually is not connected: a medical patient database might be connected to drug data, which is connected to databases about drug producing companies, i.e. data of different knowledge domains can be integrated.

11.3 Conclusion

In this thesis, principles for semantic database integration were developed. These principles were implemented in SEMEDA in a way, which enables users to query databases with no knowledge of the database schema of the underlying databases, nor does the user have to know the location of the databases or any other technical details of the databases.

Databases can be added or removed from the system with a minimum amount of work, and all semantic database definitions can be applied without editing source code. Several ontologies were evaluated and an extensible rudimentary custom tailored top-level ontology for the semantic definition of database attributes has been developed. By using SEMEDAs HTML interface, databases can semantically defined using a web browser.

In addition, SEMEDA can be used to collaboratively edit large ontologies such as the Gene Ontology via the Internet, thus enabling database providers to collaboratively edit controlled vocabularies and ontologies. However, at present SEMEDAs ontology editor requires that users who edit ontologies are “good willing”, i.e. user permissions are not yet implemented as stringent as they would have to be implemented if SEMEDA would be used as a generally available WebService.

12 Literature

Aboa, E. E., A. Bairoch, C. W. Barker, S. Beck, D. A. Benson, H. Berman, G. Cameron, C. Cantor, S. Doubet, T. J. P. Hubbard, T. A. Jones, G. J. Kleywegt, A. S. Kolaskar, A. Van Kuik, A. M. Lesk, H.-W. Mewes, D. Neuhaus, F. Pfeiffer, L. F. TenEyck, R. J. Simpson, G. Stosser, J. L. Sussman, Y. Tatento, A. Tsugita, E. L. Ulrich, and J. F. G. Vliegenthart. 2000. Quality control in databanks for molecular biology. Bioessays **22**:1024-1034.

Aldhous, P. 1990. Human genome project. Database goes on-line [news]. Nature **347**:9.

Altschul, S. F., W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. 1990. Basic local alignment search tool. J Mol Biol **215**:403-410.

Andrade, M. A., N. P. Brown, C. Leroy, S. Hoersch, A. de Daruvar, C. Reich, A. Franchini, J. Tamames, A. Valencia, C. Ouzounis, and C. Sander. 1999. Automated genome sequence analysis and annotation. Bioinformatics **15**:391-412.

Apweiler, R., T. K. Attwood, A. Bairoch, A. Bateman, E. Birney, M. Biswas, P. Bucher, L. Cerutti, F. Corpet, M. D. Croning, R. Durbin, L. Falquet, W. Fleischmann, J. Gouzy, H. Hermjakob, N. Hulo, I. Jonassen, D. Kahn, A. Kanapin, Y. Karavidopoulou, R. Lopez, B. Marx, N. J. Mulder, T. M. Oinn, M. Pagni, F. Servant, C. J. Sigrist, and E. M. Zdobnov. 2000. InterPro--an integrated documentation resource for protein families, domains and functional sites. Bioinformatics **16**:1145-1150.

Ashburner, M., C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. 2000. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat Genet **25**:25-29.

Aubry, F., and A. Todd-Pokropek. 2001. Mimos: a description framework for exchanging medical image processing results. Medinfo **10**:891-895.

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

Baker, P. G., A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. 1998. TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. An Overview. *in* sixth International Conference on Intelligent Systems for Molecular Biology, Montreal.

Baker, P. G., C. A. Goble, S. Bechhofer, N. W. Paton, R. Stevens, and A. Brass. 1999. An ontology for bioinformatics applications. Bioinformatics **15**:510-520.

Banerjee, S. 2000. A Database Platform for Bioinformatics. Pages 705-710 *in* 26 Intl Conf. on VLDB, Cairo, Egypt.

Baxevanis, A. D. 2002. The Molecular Biology Database Collection: 2002 update. Nucleic Acids Res **28**:1-7.

Beckett, D. 2001. The Design and Implementation of the Redland RDF Application Framework. *in* Tenth International World Wide Web Conference (WWW10), Hong Kong.

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

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

Berners-Lee, T., R. Cailliau, J. Groff, and B. Pollerman. 1992. World-Wide Web: The Information Universe. Electronic Networking: Research, Applications and Policy **1**:74-82.

Blake, J. A., J. T. Eppig, J. E. Richardson, and M. T. Davisson. 2000. The Mouse Genome Database (MGD): expanding genetic and genomic resources for the laboratory mouse. The Mouse Genome Database Group. Nucleic Acids Res **28**:108-111.

Blázquez, M., M. Fernández, J. M. García-Pinar, and A. Gomez-Perez. 1998. Building Ontologies at the Knowledge Level using the Ontology Design Environment. *in* KAW'98, Banff, Canada.

Boehm, B. W. 1988. A spiral model of software development and enhancement. IEEE COMPUTER **21**:61-72.

Brazma, A., P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C. A. Ball, H. C. Causton, T. Gaasterland, P. Glenisson, F. C. Holstege, I. F. Kim, V. Markowitz, J. C. Matese, H. Parkinson, A. Robinson, U. Sarkans, S. Schulze-Kremer, J. Stewart, R. Taylor, J. Vilo, and M. Vingron. 2001. Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. Nat Genet **29**:365-371.

Bruce, A. T. 1992. Designing quality databases with IDEF1X information models. Dorset House Pub, New York, NY.

Bry, F., and P. Kröger. 2001. A Molecular Biology Database Digest. PMS-FB-2001-3, Institute for Computer Science, University of Munich, Munich.

Buntrock, R. E. 2001. Chemical registries--in the fourth decade of service. J Chem Inf Comput Sci **41**:259-263.

Ceusters, W. 2001. Formal Terminology Management for Language Based Knowledge Systems: Resistance is Futile. Pages 135-153 *in* R. Temmerman and M. Lutjeharms, editors. Trends in Special Language Technology.

Ceusters, W., J. Rogers, F. Consorti, and A. Rossi-Mori. 1999. Syntactic-semantic tagging as a mediator between linguistic representations and formal models: an exercise in linking SNOMED to GALEN. Artif Intell Med **15**:5-23.

Chen, P. P. 1976. The entity relationship model towards a unified view of data. ACM Transactions on Database Systems **1**:9-36.

Chung, S. Y., and L. Wong. 1999. Kleisli: a new tool for data integration in biology. Trends Biotechnol **17**:351-355.

Codd, E. F. 1970. A Relational Model of Data for Large Shared Data Banks. Comm. of the ACM **13**:377-387.

Cotton, R. G., and O. Horaitis. 2000. Quality control in the discovery, reporting, and recording of genomic variation. Hum Mutat **15**:16-21.

Cotton, R. G., and O. Horaitis. 2002. The HUGO Mutation Database Initiative. Human Genome Organization. Pharmacogenomics J **2**:16-19.

Craven, M., D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to Extract Symbolic Knowledge from the World Wide Web. Pages 509-516 *in* 15th National Conference on Artificial Intelligence (AAAI-98), Madison.

Critchlow, T., K. Fidelis, M. Ganesh, R. Musick, and T. Slezak. 2000. DataFoundry: information management for scientific data. IEEE Trans Inf Technol Biomed **4**:52-57.

Croft, W. B., J. P. Callan, and D. B. Aronow. 1995. Effective access to distributed heterogeneous medical text databases. Medinfo **8**:1719.

Date, C. J. 1982. A Formal Definition of the relational Model. SIGMOD Record **13**:18-29.

Date, C. J. 2000. An introduction to database systems, 7th edition. Addison-Wesley, Reading, Mass.

Davidson, S. B., C. Overton, and P. Buneman. 1995. Challenges in integrating biological data sources. J Comput Biol **2**:557-572.

Discala, C., X. Benigni, E. Barillot, and G. Vaysseix. 2000. DBcat: a catalog of 500 biological databases. Nucleic Acids Res **28**:8-9.

Domingue, J. 1998. Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the

Web. *in* KAW 98, Banff, Canada.

Duineveld, A. J., R. Stoter, M. R. Weiden, B. Kenepa, and V. R. Benjamins. 1999. Wondertools? A Comparative Study of Ontological Engineering Tools. *in* Twelfth Workshop on Knowledge Acquisition, Modeling and Management, Voyager Inn, Banff, Alberta, Canada.

Edwards, J. L., M. A. Lane, and E. S. Nielsen. 2000. Interoperability of Biodiversity Databases: Biodiversity Information on Every Desktop. Science **289**:2312-2314.

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

Etzold, T., and G. Verde. 1997. Using views for retrieving data from extremely heterogeneous databanks. Pac Symp Biocomput:134-141.

Farquhar, A., R. Fikes, and J. Rice. 1996. The Ontolingua Server: a Tool for Collaborative Ontology Construction. *in* KAW 96, Nottingham, UK.

Farquhar, A., R. Fikes, and J. Rice. 1997. The Ontolingua Server: a Tool for Collaborative Ontology Construction. Intern. Journal of Hum. Comp. Studies (IJHCS) **46(6), pp. 707-728.**:707-728.

Fellbaum, C. 1998. WordNet : an electronic lexical database. MIT Press, Cambridge, Mass.

FlyBase-Consortium. 1999. The FlyBase database of the Drosophila Genome Projects and community literature. The FlyBase Consortium. Nucleic Acids Res **27**:85-88.

Frankewitsch, T., and U. Prokosch. 2001. Navigation in medical Internet image databases. Med Inform Internet Med **26**:1-15.

Fredman, D., M. Siegfried, Y. P. Yuan, P. Bork, H. Lehvaslaiho, and A. J. Brookes. 2002. HGVbase: a human sequence variation database emphasizing data quality and a broad spectrum of data sources. Nucleic Acids Res **30**:387-391.

Freier, A., R. Hofestädt, M. Lange, and U. Scholz. 2002a. Information Fusion and Metabolic Network Control. Pages 49-84 *in* J. Collado-Vides and R. Hofestädt, editors. Gene Regulation and Metabolism - Post-Genomic Computational Approaches. MIT Press, Cambridge.

Freier, A., R. Hofestädt, M. Lange, U. Scholz, and A. Stephanik. 2002b. BioDataServer: An SQL-based service for the online integration of life science data. In Silico Biol. **2**.

Freier, A., R. Hofestädt, M. Lange, U. Scholz, and T. Töpel. 2000. MD-CAVE - The Metabolic Diseases Database a System for Storing information About Human Inborn Errors. Pages 66ff *in* Second International Conference on Bioinformatics of Genome Regulation and Structure (BRGS2000), ICG Novisibirsk.

Freier, A., R. Hofestdt, M. Lange, and U. Scholz. 1999. MARGBench - An Approach for Integration, Modeling and Animation of Metabolic Networks. Proceedings of the German Conference on Bioinformatics, Hannover.

Frishman, D., K. Albermann, J. Hani, K. Heumann, A. Metanomski, A. Zollner, and H. W. Mewes. 2001. Functional and structural genomics using PEDANT. Bioinformatics **17**:44-57.

Fujibuchi, W., S. Goto, H. Migimatsu, I. Uchiyama, A. Ogiwara, Y. Akiyama, and M. Kanehisa. 1998. DBGET/LinkDB: an integrated database retrieval system. Pac Symp Biocomput:683-694.

Gene-Ontology-Consortium. 2001. Creating the gene ontology resource: design and implementation. Genome Res **11**:1425-1433.

Giudicelli, V., and M. P. Lefranc. 1999. Ontology for immunogenetics: the IMGT-ONTOLOGY. Bioinformatics **15**:1047-1054.

Goksel, A., and D. McLeod. 1999. Semantic heterogeneity resolution in federated databases by metadata implantation and stepwise evolution. VLDB Journal **8**:120-132.

Greenes, R. A., R. C. McClure, E. Pattison-Gordon, and L. Sato. 1992. The findings--diagnosis continuum: implications for image descriptions and clinical databases. Proc Annu Symp Comput Appl Med Care:383-387.

Gruber, T. R. 1993a. Toward principles for the design of ontologies used for knowledge sharing. *in* N. G. a. R. Poli, editor. International Workshop on Formal Ontology. Kluwer Academic.

Gruber, T. R. 1993b. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition **5**:199-220.

Guarino, N. 1998. Formal Ontology and Information Systems. *in* N. Guarino, editor. Formal Ontology in Information Systems. IOS Press (amended version), Trento, Italy.

Haas, L. M., P. Kodali, J. E. Rice, P. M. Schwarz, and W. C. Swope. 2000. Integrating Life Sciences Data - With a Little Garlic. Pages 5-12 *in* Bioinformatics and Biomedical Engineering, Arlington, Virginia, USA.

Haas, L. M., P. M. Schwarz, P. Kodali, E. Kotlar, J. E. Rice, and W. C. Swope. 2001. DiscoveryLink: A system for integrated access to life sciences data sources. IBM System Journal **40**:489-511.

Hahn, U., M. Romacker, and S. Schulz. 2002. Creating knowledge repositories from biomedical reports: the MEDSYNDIKATE text mining system. Pac Symp Biocomput:338-349.

Hakimpour, F., and A. Geppert. 2001. Resolving Semantic Heterogeneity in Schema Integration: an Ontology Based Approach. *in* FOIS - International Conference On Formal Ontology In Information Systems. ACM, Ogunquit, Maine.

Hammer, J., and D. McLeod. 1993. An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. International Journal of Intelligent & Cooperative Information Systems **2**:51-83.

Hamosh, A., A. F. Scott, J. Amberger, C. Bocchini, D. Valle, and V. A. McKusick. 2002. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. Nucleic Acids Res **30**:52-55.

Harger, C., M. Skupski, J. Bingham, A. Farmer, S. Hoisie, P. Hraber, D. Kiphart, L. Krakowski, M. McLeod, J. Schwertfeger, G. Seluja, A. Siepel, G. Singh, D. Stamper, P. Steadman, N. Thayer, R. Thompson, P. Wargo, M. Waugh, J. J. Zhuang, and P. A. Schad. 1998. The Genome Sequence DataBase (GSDB): improving data quality and data access. Nucleic Acids Res **26**:21-26.

Hendler, J., and D. L. McGuinness. 2000. The DARPA Agent Markup Language. IEEE Intelligent Systems **15**:67-73.

Herman, I., G. Melançon, and M. S. Marshall. 2000. Graph Visualisation and Navigation in Information Visualisation. IEEE Transactions on Visualization and Computer Graphics **6**:24-43.

Heuer, A., and G. Saake. 2000. Datenbanken kompakt, 2 edition. International Thomson Publishing, Bonn.

Hofestädt, R., U. Mischke, and U. Scholz. 2000. Knowledge acquisition, management and representation for the diagnostic support in human inborn errors of metabolism. Pages 857-862 *in* A. Hasman, B. Blobel, J. Dudeck, R. Engelbrecht, G. Gell, and H.-U. Prokosch, editors. Medical Infobahn for Europe. IOS Press, Hannover.

Hofestädt, R., M. Prüß, U. Scholz, and H. Urban. 1998. Molekulare Bioinformatik-Molekulare Informationssysteme zur Erkennung von angeborenen Stoffwechselerkrankungen. Magdeburger Wissenschaftsjournal **3**:29-40.

Hovy, E. H. 2002. Comparing Sets of Semantic Relations in Ontologies. Pages cm. *in* R. Green, C. A. Bean, and S. H. Myaeng, editors. The semantics of relationships : an interdisciplinary perspective. Kluwer Academic Publishers, Boston.

Inman, J. T., H. R. Flores, G. D. May, J. W. Weller, and C. J. Bell. 2001. A High-Throughput Distributed DNA Sequence Analysis and Database System. IBM System Journal **40**:464-486.

International-Union-of-Biochemistry. 1992. Enzyme nomenclature 1992 : recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes. Published for the International Union of Biochemistry and Molecular Biology by Academic Press, San Diego.

Iyer, L. M., L. Aravind, P. Bork, K. Hofmann, A. R. Mushegian, I. B. Zhulin, and E. V. Koonin. 2001. Quoderat demonstrandum? The mystery of experimental validation of apparently erroneous computational analyses of protein sequences. Genome Biol **2**:RESEARCH0051.

Jakobovits, R. 1997. Integrating Heterogeneous Autonomous Information Sources. UW-CSE-971205, Univ. of Washington.

Kanehisa, M. 1997a. A database for post-genome analysis. Trends Genet **13**:375-376.

Kanehisa, M. 1997b. Linking databases and organisms: GenomeNet resources in Japan. Trends Biochem Sci **22**:442-444.

Kanehisa, M., S. Goto, S. Kawashima, and A. Nakaya. 2002. The KEGG databases at GenomeNet. Nucleic Acids Res **30**:42-46.

Karp, P. D. 1995. A Strategy for Database Interoperation. J Comput Biol **2**:573-586.

Karp, P. D. 2001. Pathway databases: a case study in computational symbolic theories. Science **293**:2040-2044.

Karp, P. D., M. Riley, S. M. Paley, A. Pellegrini-Toole, and M. Krummenacker. 1999. Eco Cyc: encyclopedia of Escherichia coli genes and metabolism. Nucleic Acids Res **27**:55-58.

Karp, P. D., M. Riley, M. Saier, I. T. Paulsen, S. M. Paley, and A. Pellegrini-Toole. 2000. The EcoCyc and MetaCyc databases. Nucleic Acids Res **28**:56-59.

Karvounarakis, G., S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. 2002. RQL: A Declarative Query Language for RDF. *in* WWW2002, Honolulu, Hawaii, USA.

Kashyap, V., and A. Sheth. 1996a. Schematic and Semantic Similarities between Database Objects: A Context-based Approach. VLDB Journal **5**.

Kashyap, V., and A. Sheth. 1996b. Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies. *in* Cooperative Information Systems: Current Trends and Directions.

Kelley, S. 2000. Getting started with Acedb. Brief Bioinform **1**:131-137.

Khan, L., and D. McLeod. 2000. Disambiguation of Annotated Text of Audio using Ontologies. *in* ACM SIGKDD Workshop on Text Mining, Boston.

Kim, W. 1995. Modern database systems : the object model, interoperability, and beyond. ACM Press ; Addison-Wesley Pub. Co., New York, N.Y., Reading, Mass.

Kim, W., and J. Seo. 1991. Classifying Schematic and Data Heterogeneity in Multidatabase Systems. IEEE COMPUTER **24**:12-18.

Kingston, J. H. 1998. Algorithms and Data Structures Design, Correctness, Analysis, 2nd edition. Addison Wesley.

Köhler, J., M. Lange, R. Hofestädt, and S. Schulze-Kremer. 2000. Logical and Semantic Database Integration. Pages 77-80 *in* D. C. Young, editor. Bioinformatics and Biomedical Engineering, Arlington, Virginia, USA.

Köhler, J., and S. Schulze-Kremer. 2001. The Semantic Metadatabase (SEMEDA): Ontology Based Integration of Federated Molecular Biological Data Sources. *in* GCB, German Conference on Bioinformatics.

Krawczak, M., E. V. Ball, I. Fenton, P. D. Stenson, S. Abeysinghe, N. Thomas, and D. N. Cooper. 2000. Human gene mutation database-a biomedical information and research resource. Hum Mutat **15**:45-51.

Kurtz, S., J. V. Choudhuri, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich. 2001. REPuter: the manifold applications of repeat analysis on a genomic scale. Nucleic Acids Res **29**:4633-4642.

Letovsky, S. I., R. W. Cottingham, C. J. Porter, and P. W. D. Li. 1998. GDB: the Human Genome Database. Nucleic Acids Res **26**:94-99.

Lewin, B. 2000. Genes VII. Oxford University Press, New York ; Oxford.

Li, Q., P. Shilane, N. F. Noy, and M. A. Musen. 2000a. Ontology acquisition from on-line knowledge sources. Proc AMIA Symp:497-501.

Li, W.-S., C. Clifton, and S.-Y. Liu. 2000b. Database Integration Using Neural Networks: Implementation and Experiences. Knowledge and Information Systems **2**:73-96.

Lipman, D. J., and W. R. Pearson. 1985. Rapid and sensitive protein similarity searches. Science **227**:1435-1441.

Louis, G., and A. Pirotte. 1982. A Denotational Definition of the Semantics of DRC, A Domain Relational Calculus. Pages 348-356 *in* Eigth International Conference on Very Large Data Bases. Morgan Kaufmann, Mexico City, Mexico.

Ludäscher, B., A. Gupta, and M. E. Martone. 2001. Model-Based Mediation with Domain Maps. *in* 17th Intl. Conference on Data Engineering (ICDE),. IEEE Computer Society, Heidelberg, Germany.

Macaulay, J., H. Wang, and N. Goodman. 1998. A model system for studying the integration of molecular biology databases. Bioinformatics **14**:575-582.

Maedche, A., and S. Staab. 2000. Discovering Conceptual Relations from Text. *in* 14th European Conference on Artificial Intelligence (ECAI), Berlin, Germany.

Maranas, C. D., and A. P. Burgard. 2001. Review of EcoCyc and MetaCyc Databases. Metab Eng **3**:98-99.

Matsuda, H., I. Imai, M. Nakanishi, and A. Hashimoto. 1999. Querying Molecular Biology Databases by Integration Using Multiagents. IEICE TRANS. INF & SYST. **E82-D**:199-207.

Maurer, S. M., P. B. Hugenholtz, and H. J. Onsrud. 2001. Intellectual property. Europe's database experiment. Science **294**:789-790.

McEntire, R., P. Karp, N. Abernethy, D. Benton, G. Helt, M. DeJongh, R. Kent, A. Kosky, S. Lewis, D. Hodnett, E. Neumann, F. Olken, D. Pathak, P. Tarczy-Hornoch, L. Toldo, and T. Topaloglou. 2000. An evaluation of ontology exchange languages for bioinformatics. Proc Int Conf Intell Syst Mol Biol **8**:239-250.

McGuinness, D. L., R. Fikes, J. Rice, and S. Wilder. 2000. An Environment for Merging and Testing Large Ontologies. *in* Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000), Breckenridge, Colorado.

Mewes, H. W., D. Frishman, U. Guldener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Munsterkotter, S. Rudd, and B. Weil. 2002. MIPS: a database for genomes and protein sequences. Nucleic Acids Res **30**:31-34.

Musen, M. A., J. H. Gennari, H. Eriksson, S. W. Tu, and A. R. Puerta. 1995. PROTEGE-II: computer support for development of intelligent systems from libraries of components. Medinfo **8**:766-770.

NC-IUBMB. 1992. Enzyme nomenclature 1992 : recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes. Published for the International Union of Biochemistry and Molecular Biology by Academic Press, San Diego.

NHS Information Authority. 2000. The Clinical Terms Version 3 (The Read Codes) - Reference Manual, Woodgate.

Nobécourt, J. 2000. A method to build formal ontologies from texts. *in* 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW), Juan-les-Pins, French Riviera, France.

Noy, N. F., and C. Hafner. 1997. The State of the Art in Ontology Design: A Survey and Comparative Review. AI Magazine **18**:53-74.

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

Noy, N. F., and M. A. Musen. 1999. SMART: Automated Support for Ontology Merging and Alignment. *in* KAW '99, Banff, Alberta, Canada.

Oracle-Corp. 1997. Oracle8i SQL Reference. *in*.

Parent, C., and S. Spaccapietra. 1998. Database Integration: an Overview of Issues and Approaches. Communications of the ACM **41**:166-178.

Persson, B. 2000. Bioinformatics in protein analysis. Exs **88**:215-231.

Pirotte, A. 1982. A Precise Definition of Basic Relational Notions and of the Relational Algebra. SIGMOD Record **13**:30-45.

Raham, E., and P. A. Bernstein. 2001. A survey of approaches to automatic schema matching. VLDB Journal **10**:334-350.

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

Rector, A. L., and W. A. Nowlan. 1994. The GALEN project. Comput Methods Programs Biomed **45**:75-78.

Rector, A. L., P. E. Zanstra, W. D. Solomon, J. E. Rogers, R. Baud, W. Ceusters, W. Claassen, J. Kirby, J. M. Rodrigues, A. R. Mori, E. J. van der Haring, and J. Wagner. 1998. Reconciling users' needs and formal requirements: issues in developing a reusable ontology for medicine. IEEE Trans Inf Technol Biomed **2**:229-242.

Reinke, M. 2002. Entwurf und Implementierung eines Caches zum effizienten Zugriff auf integrierte molekularbiologische Daten. Diplom. Magdeburg, Germany, Magdeburg, Germany.

Russ, T., A. Valente, R. MacGregor, and W. Swartout. 1999. Practical Experiences in Trading Off Ontology Usability and Reusability. *in* Twelfth Workshop on Knowledge Acquisition, Modeling and Management, Banff, Alberta, Canada.

Salgado, H., A. Santos-Zavaleta, S. Gama-Castro, D. Millan-Zarate, E. Diaz-Peredo, F. Sanchez-Solano, E. Perez-Rueda, C. Bonavides-Martinez, and J. Collado-Vides. 2001. RegulonDB (version 3.2): transcriptional regulation and operon organization in Escherichia coli K-12. Nucleic Acids Res **29**:72-74.

Sanderson, M., and W. B. Croft. 1999. Deriving concept hierarchies from text. Pages 206-213 *in* Conference on Research and Development in Information Retrieval (SIGIR '99). ACM, Berkeley, CA, USA.

Schacherer, F., C. Choi, U. Gotze, M. Krull, S. Pistor, and E. Wingender. 2001. The TRANSPATH signal transduction database: a knowledge base on signal transduction networks. Bioinformatics **17**:1053-1057.

Schomburg, I., A. Chang, O. Hofmann, C. Ebeling, F. Ehrentreich, and D. Schomburg. 2002a. BRENDA: a resource for enzyme data and metabolic information. Trends Biochem Sci **27**:54-56.

Schomburg, I., A. Chang, and D. Schomburg. 2002b. BRENDA, enzyme data and metabolic information. Nucleic Acids Res **30**:47-49.

Schuler, G. D. 1997. Pieces of the puzzle: expressed sequence tags and the catalog of human genes. J Mol Med **75**:694-698.

Schulze-Kremer, S. 1997a. Adding semantics to genome databases: towards an ontology for molecular biology. Ismb **5**:272-275.

Schulze-Kremer, S. 1997b. Integrating and Exploiting Large-Scale, Heterogeneous and Autonomous Databases with an Ontology for Molecular Biology. Pages 43-56 *in* R. Hofestädt and H. Lim, editors. Molecular Bioinformatics, Sequence Analysis - The Human Genome Project (R. Hofestädt and H. Lim eds). Shaker Verlag, Aachen, pp. 43-56. Shaker Verlag, Aachen.

Schulze-Kremer, S. 1998. Ontologies for molecular biology. Pac Symp Biocomput:695-706.

Schulze-Kremer, S. 2002. Ontologies for molecular biology and bioinformatics. In Silico Biology **2, 0017**.

Selkov, E., S. Basmanova, T. Gaasterland, I. Goryanin, Y. Gretchkin, N. Maltsev, V. Nenashev, R. Overbeek, E. Panyushkina, L. Pronevitch, E. Selkov, Jr., and I. Yunus. 1996. The metabolic pathway collection from EMP: the enzymes and metabolic pathways database. Nucleic Acids Res **24**:26-28.

Shahabi, C., L. Khan, and D. McLeod. 2000. A Probe-Based Technique to Optimize Join Queries in Distributed Internet Databases. International Journal of Knowledge and Information Systems (KAIS) **2**:373-385.

Silvonen, P., and E. Hyvönen. 2001a. Semantic Web Kick-Off - Vision, Technologies, Research, and Applications. Pages 304 *in* E. Hyvönen, editor. HIIT Publications, Helsinki Institute for Information Technology (HIIT), Helsinki, Finland.

Silvonen, P., and E. Hyvönen. 2001b. Semantic Web Tools. Pages 137-152 *in* E. Hyvönen, editor. Semantic Web Kick-Off - Vision, Technologies, Research, and Applications. HIIT Publications, Helsinki Institute for Information Technology (HIIT), Helsinki, Finland.

Srinivasan, P. 1999. Exploring the UMLS: a rough sets based theoretical framework. Proc AMIA Symp:156-160.

Stevens, R., P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N. W. Paton, C. A. Goble, and A. Brass. 2000a. TAMBIS: transparent access to multiple bioinformatics information sources. Bioinformatics **16**:184-185.

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

Stevens, R., C. A. Goble, and S. Bechhofer. 2000b. Ontology-based knowledge representation for bioinformatics. Brief Bioinform **1**:398-414.

Stevens, R., C. A. Goble, N. W. Paton, S. Bechhofer, P. Baker, and A. Brass. 1999. Complex Query Formulation Over Diverse Information Sources Using an Ontology. Pages 83-88 *in* Workshop on Computation of Biochemical Pathways and Genetic Networks. E. Bornberg-Bauer, A. De Beuckelaer, U. Kummer, U. Rost, European Media Lab (EML).

Storey, M.-A. D., M. Musen, J. Silva, C. Best, N. Ernst, R. Fergerson, and N. F. Noy. 2001. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protege. *in* Workshop on Interactive Tools for Knowledge Capture (K-CAP), Victoria, B.C. Canada.

Stumme, G., and A. Maedche. 2001. FCA-Merge: A Bottom-Up Approach for Merging Ontologies. Pages 225-234 *in* International Joint Conference on Artificial Intelligence, Seattle, Washington, USA.

Topaloglou, T., A. Kosky, and V. Markowitz. 1999. Seamless integration of biological applications within a database framework. Proc Int Conf Intell Syst Mol Biol:272-281.

Uschold, M., P. Clark, M. Healy, K. Williamson, and S. Woods. 1998. Ontology Reuse and Application. Pages 179-192 *in* N. Guarino, editor. International Conference on Formal Ontology and Information Systems - FOIS. IOS Press, Amsterdam.

van der Vet, P. E., and N. J. I. Mars. 1995. Bottom-up construction of ontologies: the case of an ontology of pure substances. UT-KBS, University of Twente, Enschede, Holland.

van der Vet, P. E., and N. J. I. Mars. 1998. Bottom-up construction of ontologies. IEEE Transactions on Knowledge and Data Engineering **10**:513-526.

Volot, F., M. Joubert, and M. Fieschi. 1998. Review of biomedical knowledge and data representation with conceptual graphs. Methods Inf Med **37**:86-96.

Vossen, G. 2000. Datenbankmodelle, Datenbanksprachen und Datenbankmanagementsysteme. Oldenbourg-Verlag, München.

Walsh, S., M. Anderson, and S. W. Cartinhour. 1998. ACEDB: a database for genome information. Methods Biochem Anal **39**:299-318.

Wang, H., and C. Zaniolo. 1998. Aggregates in Recursive Datalog and SQL3 Queries. 980043, University of California at Los Angeles, Los Angeles, USA.

Westbrook, J., Z. Feng, S. Jain, T. N. Bhat, N. Thanki, V. Ravichandran, G. L. Gilliland, W. Bluhm, H. Weissig, D. S. Greer, P. E. Bourne, and H. M. Berman. 2002. The Protein Data Bank: unifying the archive. Nucleic Acids Res **30**:245-248.

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

Williams, N. 1997. Bioinformatics: How to Get Databases Talking the Same Language. Science **275**:301-302.

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

Wong, L. 2000. Kleisli, a Functional Query System Journal of functional Programming. Journal of Functional Programming **10**:19-56.

Xie, G., R. DeMarco, R. Blevins, and Y. Wang. 2000. Storing biological sequence databases in relational form. Bioinformatics **16**:288-289.

Xie, H., A. Wasserman, Z. Levine, A. Novik, V. Grebinskiy, A. Shoshan, and L. Mintz. 2002. Large-Scale Protein Annotation through Gene Ontology. Genome Res **12**:785-794.

Zdobnov, E. M., R. Lopez, R. Apweiler, and T. Etzold. 2002. The EBI SRS server-recent developments. Bioinformatics **18**:368-373.