(Almost) Hands-Off
Information Integration for the Life Sciences

Ulf Leser, Felix Naumann

Department for Computer Science, Humboldt-Universität zu Berlin
Rudower Chaussee 25, 12485 Berlin, Germany
{leser,naumann}@informatik.hu-berlin.de

Abstract

Data integration in complex domains, such as the life sciences, involves either manual data curation, offering highest information quality at highest price, or follows a schema integration and mapping approach, leading to moderate information quality at a moderate price. We suggest a radically different integration approach, called ALADIN, for the life sciences application domain. The predominant feature of the ALADIN system is an architecture that allows almost automatic integration of new data sources into the system, i.e., it offers data integration at almost no cost.

We suggest a novel combination of data and text mining, schema matching, and duplicate detection to combat the reduction in information quality that seems inevitable when demanding a high degree of automatism. These heuristics can also lead to the detection of previously unknown or unseen relationships between objects, thus directly supporting the discovery-based work of life science researchers. We argue that such a system is a valuable contribution in two areas. First, it offers challenging and new problems for database research. Second, the ALADIN system would be a valuable knowledge resource for life science research.

1 Introduction

Data integration approaches in the life sciences can be classified into two broad types. Of the first type are projects that achieve a high standard of quality in the integrated data through manual curation, i.e., using the experience and expertise of trained professionals. We call this type of projects "data-focused". A prominent example for this class of integrated systems is Swiss-Prot [BBA+93], collecting and integrating data on protein sequences from journal publications, submissions, personal communications, and other databases by means of approximately two dozen human data curators. Data-focused projects are typically managed by domain experts, e.g., biologists. Database technology plays an only minor role. All effort is put into acquiring and curing the actual data that is usually maintained in a text-like manner. If detailed schemata are developed and used, they are not exposed to the user for structured queries.

Projects of the second type use database technology, and are typically initiated and managed by computer scientists. These projects, which we call "schema-focused", are aimed at providing integration middleware rather than building concrete databases. They mostly deal with schema information, using techniques such as schema integration, schema mapping, and mediator-based query rewriting. Examples of such schema-focused projects for the life sciences are TAMBIS [BBB+98] and OPM [KCMS98]. Both require for each integrated data source the creation of some sort of wrapper for query processing, and a detailed semantic mapping between the heterogeneous source schemata and a global, mediated schema. The mappings must be specified in a special language (respectively OPM*QL views and the GRAIL description logic in the mentioned projects), which makes them inaccessible for most domain experts. Their focus is exclusively on schema information and only little attention (and money) is paid to data values, their consistency, and their correctness.

Data-focused projects are very successful in the biological scene, but are also quite costly. Schema-focused systems on the other hand have hardly ever been used for any serious integration project in public life science research and did not achieve the broad attention they might have deserved. We believe that the major reason for this failure lies in their schema-centricity. Creating the necessary semantic mappings is tedious, especially when larger schemata are involved, requires learning proprietary and complicated languages, and often leaves biologists with a

Proceedings of the 2005 CIDR Conference

general feeling of uselessness, because they distrust the necessary abstraction steps encoded in the global schema.

We propose a novel, third type of integration approach for the life sciences, which we call ALADIN (ALmost Automatic Data INtegration). Its essential feature is that the integration of data sources is an almost automatic process without greater loss in terms of information quality. Instead, its automatic mechanisms for finding links between objects often lead to unexpected findings, which are of highest importance in a discovery-oriented field, such as the life sciences.

ALADIN's architecture consists of several components that together allow for the automatic integration of heterogeneous data sources into a global, materialized repository of biological objects and links between them. Its architecture and key mechanisms rely on several characteristics of databases in biological and medical research that are often more a semi-structured, text-centric data collection than a structured, data-centric database:

- Data sources, especially those containing the most valuable information, are typically centered on one primary class of objects, such as genes, proteins, or DNA sequences.
- The objects of this primary class always have a unique and publicly visible key—the "accession number".
- The primary objects are further described by a set of nested fields, called annotations. We call these annotations the "secondary objects". Apart from such annotations, relationships among objects within a data source rarely exist.
- Many of the annotations are text fields, such as description, functional annotation, source of biomaterial, etc. Especially important fields with long strings are those containing DNA, RNA, or protein sequences.
- Databases heavily cross-reference each other, using the global keys of primary objects together with the referenced databases as pointers.
- Databases overlap in the objects they represent, storing sometimes redundant and sometimes conflicting data.

ALADIN uses a relational database as its basis. It can integrate every sort of data source for which a relational representation exists or can be generated, including XML data and flat-file databases with an appropriate parser. As data management in the life sciences has matured considerably in the last decade, almost all important databases are nowadays managed in relational database management systems, and XML is common for data exchange. Furthermore, for almost all flat-file representations there are freely available parsers for importing the data into an RDBMS. Thus, our approach encompasses most data sources in the life science area.

Integration in ALADIN does not depend on predefined integrity constraints structuring a schema, but uses techniques from schema matching [RB01] and data and text mining [BN05; DLD+04] to detect the relation containing the primary objects for each data source, to infer different types of relationships between relations and objects within one source, and to infer relationships between objects in different sources. The system does not rely on any particular schema for its data sources. Even generic parsers may be used, such as generic XML-to-relational mapping tools [NJM03].

ALADIN integrates data sources in a five-step process. First, it imports a data source into a relational format using known parsers. From these relational representations, it then determines the relation (or relations) that represent the primary objects within a data source. In the third step, the fields containing annotation for the primary objects are detected. Here, existing integrity constraints are exploited, if they are available; otherwise, the relationships are guessed from data analysis. In the fourth step, links between the objects of the primary relations of different data sources are searched for. This step includes the discovery of predefined cross-references already contained in the original data. Furthermore, links are generated based on similarity between values of text fields and based on sequence homology between sequence fields. In the fifth step, duplicates are detected across different data sources, thus removing commonplace redundancy.

Once the data is imported into ALADIN, the process is completely automatic. It results in an integrated information system that is best explained in analogy to the Web: The discovered objects correspond to Web pages, and the discovered links correspond to HTML links. Users may traverse this web of biological objects using a generic front-end very much like they travel the web using their browser. Just like in the Web, a specialized search engine can "crawl" the links and index biological objects and their data and textual annotation, thus providing search capability. ALADIN also supports structured queries, detects and flags duplicate objects, and adds a wealth of additional links between objects that are undiscoverable when looking at databases in isolation.

Clearly, the resulting warehouse of data contains errors in those parts where the system needs to guess, i.e., in the definition of primary relations and in the linkage of objects in and across data sources. We expect that the most pertinent errors will be links that are lost between relations during the import step and wrong or missed cross-references between data sources. In consequence, we do not envisage that the resulting database should be directly fed into data mining or data analysis algorithms. However, ALADIN can readily be browsed without any schema knowledge, which is the method of choice for accessing databases for most biologists [SMB+04]. The system develops its strength in scenarios where an explorative approach is necessary, but not in settled environments where the focus is on automated, large-scale data analysis with guaranteed quality bounds.

The rest of this paper is organized as follows: In Section 2 we describe and evaluate properties of biological databases and the different approaches to data integration in the life sciences in more detail. Section 3 outlines the architecture of ALADIN and sketches its main building blocks. Section 4 discusses each of these building blocks in more detail and points out how existing technologies can be used to achieve the required functionality. Section 5 contains a case study for an integrated system of protein structure annotation. Finally, Section 6 discusses related work and summarizes the main challenges of and benefits.

2 Data Integration in the Life Sciences

Data integration in the life sciences has been a topic of intensive research for many years (see [Ste03] for a recent survey). It remains a promising area for database researchers, as there is probably no other area in which such a large number of complex databases is freely available. Further, the importance of research in life sciences is constantly growing due to the direct effect on medical applications and, hence, human health and wellness. This research depends heavily on data integration, as valuable information is scattered over literally hundreds of heterogeneous and autonomous data sources [Aug01].

Databases in life science research have a number of characteristics that need to be taken into account when designing an integration system. Most importantly, life science databases typically store only one primary type of object. Objects of this type are described by a rich and sometimes deeply nested set of annotations. One particularly important type of annotation are cross-references to other databases [BLM+04]. "Hubs" in the life science data world, such as Swiss-Prot, provide links to more than fifty other databases, pointing from the primary object of Swiss-Prot, i.e., proteins, to further information, such as protein structure, publications, taxonomic information, the gene encoding the protein, related diseases, known mutations, etc. These cross-references, internally stored as a pair (*target-database, accession-number*) and presented as hyperlinks on web pages, are used intensively by humans to collect information. Links are determined in a number of ways, ranging from completely automatic methods, for instance based on the computation of similarity between sequences, to fully manual methods requiring the reading and understanding of primary literature.

As observed quite some years ago, links are the natural first choice for integration approaches [Rob95], as they connect biological objects, for instance adding the function to a gene or the cause to a disease. This viewpoint is strictly different from typical database-oriented integration approaches whose primary aim is the removal of semantic redundancy in schemata. For instance, mediator-based approaches start by designing a semantically non-redundant global schema and connect elements of this schema to elements of source schemata to allow for concise querying. However, this task is extremely difficult in the life sciences, where many of the prime object types are largely under-defined. Even ubiquitous concepts, such as a "gene" or a "protein", are defined differently by different databases. Merging two gene classes into one blurs these differences, leading to uncertainty of biological users about what they see.

This uncertainty does not affect the need for duplicate detection. In many cases, even though definitions are vague, objects in different databases are commonly considered as duplicates. For instance, largely the same proteins used to be stored in Swiss-Prot [BBA+93] and PIR [WHA+02]. Detecting duplicate objects is a valuable task; however, contrary to typical duplicate detection scenarios, here duplicates should be only flagged and not merged.

Keeping objects from different sources carefully separated and concentrating on linking of objects from different sources (where "duplicate" is one type of link) avoids confusion while offering most of the expected benefits from data integration: It allows for queries spanning different databases, and it potentially detects previously unknown links, thereby directly contributing to progress in life science research.

	Data-focused	Schema-focused	ALADIN
Focus of attention	Removal of redundancy in data	Removal of redundancy in schemata	Detection of object links
Structure of data	Semi-structured, meant for reading and browsing	Structured data, meant for analysis and queries	Derived structure, allows for both querying and browsing
Cost of integration	High, manual data curation and annotation	Medium; requires wrapper and schema mappings	Minimal cost
Examples[1]	Swiss-Prot, Omim, GeneCards, …	Tambis, DiscoveryLink, BioKleisli	n/a

Table 1: Spectrum of integration approaches

We revisit our classification of integration approaches based on these observations (see Table 1). The data-focused approaches disregard schema information and perform removal of redundancy in data, i.e., merging of information, manually by human experts. They are of highest value for the research community, but costly and grow at a much slower rate than raw experimental data is produced [RML04]. Schema-focused approaches aim at minimization of semantic redundancy in schemata, trying to reduce redundancy in data without having to deal with the data itself. As pointed out, this approach probably misses the main need in life science data integration.

Our viewpoint on the most useful type of data integration is supported by the success of the SRS system for data integration [ZLAE00]. SRS runs in more than 400

[1] We are aware that this comparison can be misleading as the mentioned schema-focused projects aim at developing middleware and methods for data integration and are not actual integrated databases. See also in the text.

installations world-wide, integrating up to 150 databases into one system[2]. The system takes flat-file databases as input. For each database, a parser needs to be written in the proprietary Icarus programming language to define the database structure. Using these parsers, SRS computes indexes that allow for fast search over many databases with one query. Data from different databases is never merged, and queries can "join" different databases by following links taken from especially tagged fields. Thus, it follows the same design principles as our ALADIN project in this respect; however, in SRS all structures and links need to be explicitly specified and no automatic integration takes place (see also Section 6.1).

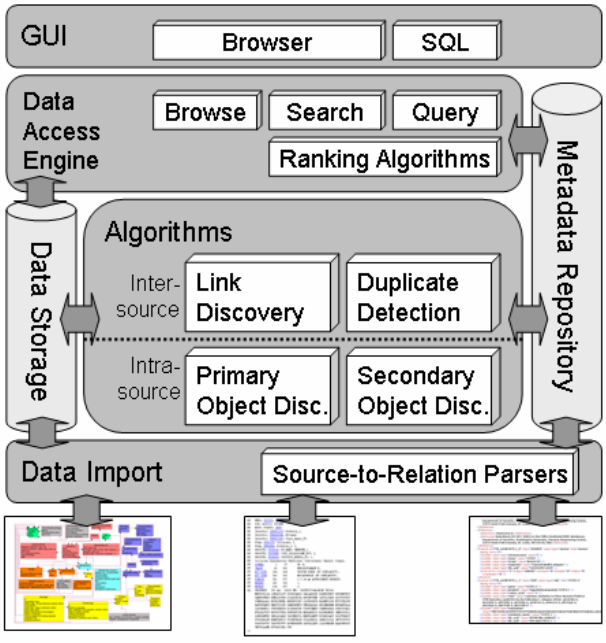

Figure 1. The architecture of ALADIN.

3 System Architecture

ALADIN builds on a local data warehouse that integrates heterogeneous data sources. Its architecture (see Figure 1) is derived from the characteristics of the life sciences domain as described in the previous section: It is intended for browsing, focused on the discovery of unseen relationships between objects, and at the same time allows for searching and structured queries.

ALADIN, unlike previous approaches, is strictly oriented towards an almost maintenance free data integration procedure. From this principle, a number of design requirements are derived: Integration of new data sources should need only minimal human intervention. Specifically, the integration process should not require the definition of semantic relationships between schema elements and it should not require the manual specification of object links and duplicates. Instead, such relationships should be automatically guessed from the data without human intervention. The system should not assume that the data sources come with a clear definition of object types, as such definitions would require custom data importers. Thus, the process must automatically detect several things:

- The primary objects within data sources
- the (secondary) objects and fields containing annotation for the primary objects,
- formerly known and unknown links between objects in different data sources, and
- duplicate objects within and across different data sources.

Clearly, relying so much on automatic methods comes at a price; the detection algorithms are likely to make mistakes: There will be missing links (false negatives) and wrong links (false positives). There will be unrecognized secondary objects, undetected duplicates, and in the worst case, no or only an incorrect primary relation is found for a source. This effect seems unattractive for database researchers, but is a common situation in information retrieval and data mining applications. The standard procedure in such situations is to estimate the amount of errors of the system using performance measures, such as precision and recall. We show in Section 6 how such measures can be estimated using an existing integrated database.

However, "guessing" the various types of links also leads to the detection of unseen relationships between objects. Such discovery is a main feature of ALADIN. We think useful "guesswork" is achievable using a mixture of data integration, text mining, information retrieval, and data mining techniques.

We now list the main components of the architecture in the order they will typically be used when adding the data of a new source into the system. In this section we give the big picture by concentrating on how the components work together (see Figure 2); details for the individual components are given in the next section. There, we also show how far research in related work has advanced in achieving the components.

Data import. Each data source needs to be imported into the relational database system. In cases where no downloadable import method exists, this is the one point where ALADIN does require human work. However, our experience in integration projects [BLM+04; RMT+04] shows that this situation is rare, and when it occurs it is sufficient for ALADIN to use a quick-and-dirty parser. No elaborate schema design or re-design is necessary.

Discovery of primary objects. Once a data source is imported, we need to identify its primary objects stored in the primary relation. Intuitively, primary relations contain data about the main objects of interest in the source, such as "proteins" or "diseases". Usually these relations store a primary key but no foreign keys—they are the "sinks" of

[2] See http://srs.ebi.ac.uk

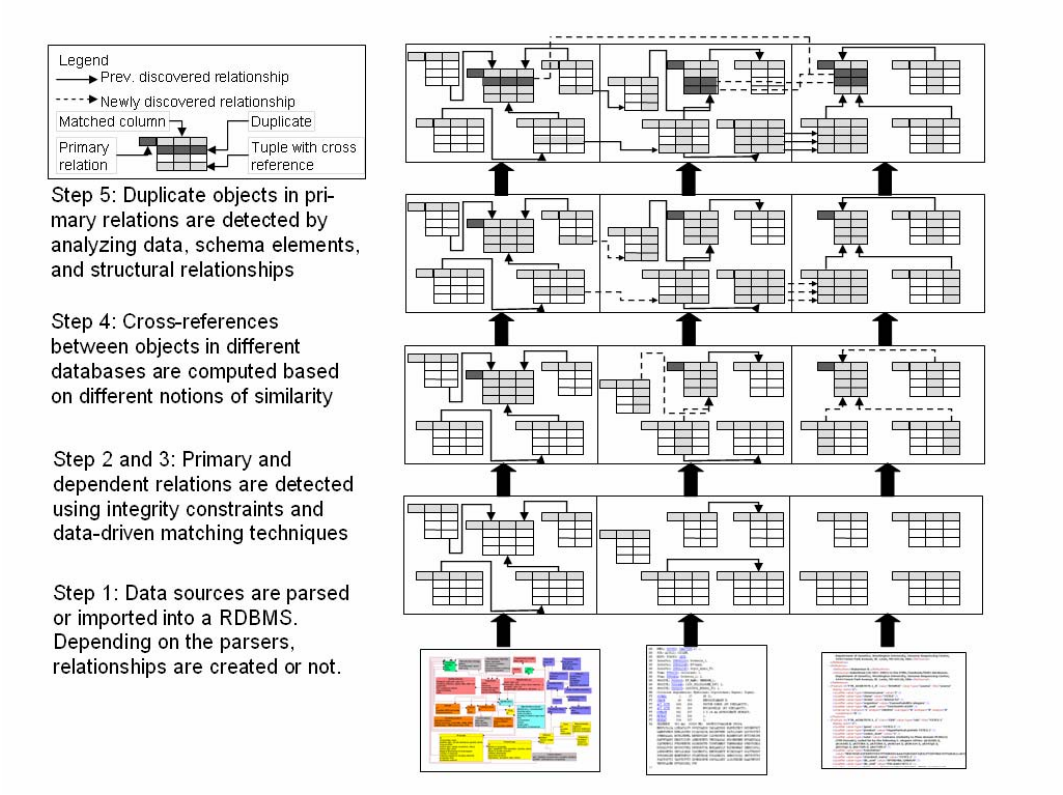

Figure 2. Integration steps in ALADIN.

the graph of foreign-key constraints (for details see Section 4.2). Their discovery cannot rely on predefined integrity constraints, as they often do not exist and generic parsers often cannot generate constraints due to missing semantic knowledge. In consequence, integrity constraints are used if they are known, but missing ICs are automatically detected by analyzing any known schema information and the attribute values. Primary relations are the focal point of the next discovery step.

Discovery of secondary objects. In this step, the previously detected network of integrity constraints is revisited, taking the identified primary relations as additional input. For each primary relation, the set of (possibly nested) dependent objects and attributes is determined. Secondary objects store additional information about primary objects. Additionally, cardinalities of relationships are determined in this step. At the end of this step, the internal structure of a newly imported data source is known (of course, there might be errors). Note that until this point, no data or metadata from other data sources was involved. This ability is important to allow for efficient incremental addition of new data sources.

Link discovery. In the fourth step we search for attributes that are cross-references to objects in other data sources. We assume that cross-references always point to primary objects in other databases as these are the only objects with stable and public IDs. Thus, the results from the second step are necessary input to determine all possible link targets. This assumption is an obvious pruning possibility to cope with the theoretic requirement of comparing all pairs of attributes from all sources. Other pruning strategies that rely on attribute value distributions and statistics are discussed in Section 4.4. These statistics need to be computed only once for each data source and can then be reused for subsequently added data sources.

Duplicate detection. In analogy to the previous link discovery step, in the fifth step we search for a special kind of "links" between primary objects in different data sources, i.e., those indicating that the database objects represent the same real world object. Such duplicate links are established if two objects are sufficiently similar according to some similarity metric (see Section 4.5). Literature defines several domain-independent similarity measures usually based on edit distance. Knowledge of duplicates enhances the users browsing and querying experience by avoiding redundant information and combining complementary data about an object.

Browse, search, and query engine. Once data is integrated into the system, we provide three modes of accessing the data. *Browsing* simply displays objects and different kinds of links (to secondary objects, to related objects, to duplicates) that users can follow. *Search* allows a full-text search on all stored data and a focused search restricted to certain partitions of the data (only certain data sources, only certain fields, etc.). Finally, *querying* allows full SQL queries on the schemata as

imported. All three access modes are supported by appropriate user interfaces.

Metadata repository: The process of discovering new structures and links produces much metadata that is stored in a central repository. In the spirit of the "Corpus" in the Revere project [HED+03], it contains not only known and discovered schemata, but also information about primary and secondary relations, statistical metadata, and sample data to improve discovery efficiency. Finally, a large part of storage space will be consumed by the discovered links on the object level.

In this section we have described the discovery steps as an iterative process. Of course in an implementation of such a system, there is high potential for parallelization and combination of these steps. In particular the discovery of primary and secondary objects can go hand in hand in a single processing step. Figure 2 summarizes this section by representing the overall architecture and pointing out how the components collaborate to achieve (almost) hands-off data integration.

4 System Components

In this section we describe the components of ALADIN in more detail, point out existing technology, and name open challenges.

4.1 Data Import

The task of the data import component is to read a data source into a relational database. It is neither necessary that the relational schema or its elements conform to any standard, nor is it necessary that integrity constraints, such as UNIQUE, PRIMARY KEY, or FOREIGN KEY, are present in the schema. Subsequent steps add this information wherever possible.

Therefore, a variety of known import procedures can be used. We believe that they encompass most of the major life science databases and give some examples that support this claim[3]:

- Some databases, such as Swiss-Prot, the GeneOntology, or EnsEmbl, provide direct relational dump files.
- For text-based exports, in many cases readily downloadable parses are available. For instance, Genbank, Swiss-Prot, or the NCBI Taxonomy Database can be read by the BioPerl and BioSQL packages.
- Some databases provide parsers with their export files, such as the OpenMMS parser for the Protein Structure Database (PDB) or the ArrayExpress parsers for the MGED XML format for microarray data.
- Databases exported as XML files can be parsed using a generic XML shredder. This XML parsing is for instance helpful to import protein-interaction data from the BIND database or pathway data from KEGG.

Until today, we have met only very few sources for which no parser exists. For instance, we are not aware of any parser for the CATH or SCOP databases on protein classification; however, their format is trivial to parse. Again, as ALADIN has almost no expectations to the resulting schema, straight-forward mappings to tables are sufficient.

4.2 Discovery of Primary Relations

As described in the introduction, most life science databases are centered on only one type of (primary) object. The task of the second step is to detect this type in the relational representation without relying on any support from the parsers. We call this type the *primary relation* for the data source; cases with more than one primary relation are seldom and discussed later.

Finding the primary relation relies on a set of heuristic rules using only the schema and the actual data as input. The rules are derived from our experiences with the integration life science databases [BLM+04; LLRC98; RMT+04]. Concrete examples are given in Section 5. The output of the algorithm is the name of the primary relation and a set of properties and guessed relationships between the tables of the data source

As the first step, the algorithm detects "unique" attributes by issuing a SQL query for each attribute in the schema that has no known UNIQUE constraint. Attributes that are unique are marked as such.

Primary objects of a life science database usually are biologically or medically important objects, such as sequences, proteins, diseases, or genes. These objects and their description are long lasting and the target for database cross-referencing. Therefore, these objects carry a stable, publicly visible accession number as ID. We have observed that accession numbers usually contain alphanumeric characters, whereas internal IDs generated by the parsers to connect relations consist only of digits. Therefore, we analyze for each unique attribute whether each of its values contains at least one non-digit character and is at least four characters long[4]. As accession numbers within one database usually all have the same length, we finally require the values of the attribute to differ by at most 20 percent in length.

Attributes fulfilling these conditions are marked as accession number candidates. Each table may have only one accession number candidate; if more than one candidate was found, only the one with the longer average field length is considered.

[3] For space limitations we omit URLs or references for these databases. A search in the web using the database name and the keywords mentioned in the text find the appropriate web site.

[4] These are the shortest accession numbers we are aware of (used in the PDB database).

Next, the algorithm tries to deduce foreign key relationships and their cardinalities. Note that foreign key relationships are directed and represent either a 1:1 or a 1:N relationship[5]. Existing foreign key constraints are found using the data dictionary. Then, all unique attributes are considered as potential targets for such a relationship and all attributes are considered as potential sources. The values of each potential source are compared to the values of each potential target. If the values of a potential source are a true subset of the values of a potential target, we assume a 1:N relationship, i.e., the source attribute is a foreign key of the target attribute. If the values of a potential source are the same set as the values of a potential target, we assume a 1:1 relationship. More sophisticated techniques are described for instance in [KM92; MLP02].

Clearly, this algorithm is prone to make mistakes, e.g., if many dictionary tables are used to define the set of allowed values for an attribute. Those tables typically have as primary key integer numbers from 1 to the number of tuples in the table. As these sets are quite similar, confusion about which is the primary key for a foreign key representing the value-restricted attribute may arise. However, note that for this confusion to happen, all values of the foreign key must also appear in the primary key attribute. This case happens only if the number of values in two dictionary tables are identical – a rather rare event.

The next step chooses the primary relation from the set of tables containing an accession number candidate. Therefore, the network formed by the guessed foreign key relationships is analyzed. We choose as the primary relation the table with highest in-degree of all tables containing an accession number candidate. This heuristic is based on the observation that life science databases contain mostly fields that describe some primary objects. These fields are often multi-valued and structured, e.g., lists of scientific references, lists of database cross-references, or lists of keywords describing the function of a protein in a controlled vocabulary. If such a field is single-valued, for instance to store the sequence of a large stretch of DNA in a separate table, then there is a 1:1 relationship with the primary relation as target. If a field is multi-valued, then the primary relation either has a 1:N relation to the table containing the field or it has a M:N relationship, appearing as 1:M, to a bridge table. Thus, many tables necessarily point to the primary relation.

The detected primary relation and the set of relationships are used as input for the third step.

In some cases data sources have more than one primary relation. For instance, the EnsEmbl database focuses both on sequenced clones and the genes lying on those clones [HBB+02]. Such cases require a more complex metric for finding primary relations, using for instance the difference of the in-degree of a relation to the average in-degree. Schema information, for instance schema elements containing the substring "ID" in their name or elements that match partially to another schema element, could also help. In the following, we further assume that there is only one primary relation for each data source.

[5] M:N relationships from a conceptual model appear as two 1:N relationships in the relational representation.

4.3 Discovery of Secondary Relations

Having discovered the primary relation of a data source, we need to connect the objects in this relation to the other relations in the data source. This step determines the description and annotation that is displayed together with a primary object in the web interface (see next).

We revisit the set of relationships we have detected in the previous step. We compute the path(s) from the primary relation to each of the other relations of the data source using transitivity of relationships, ignoring direction and cardinality. In theory, no such path needs to exist, as the relations of the data source could fall into non-overlapping partitions. However, this would imply that a data source stores two completely unrelated sets of information – a situation we have yet to encounter. Note that this situation is different from a database having more than one primary relation, as those relations will always be connected – establishing this connection is probably one of the main purposes of such a database.

The paths are stored in the metadata repository. As described earlier, those paths are used to join together the information that is presented as belonging to an object. If multiple paths exist, all are stored. The paths may also be used to guide the construction of structured queries.

4.4 Link Discovery

There are two kinds of links between objects that we want to discover in this step: explicit links and implicit links.

First, among life sciences databases there are many existing, explicit cross-references. Usually such a cross-reference is stored as the accession number of the object it points to together with an indication of the database holding this object. Often, both are encoded into one string, such as in "*ENSG00000042753*" or "*Uniprot:P11140*". Thus, already here string matching techniques are needed, for instance for finding common substrings. Because cross-references use public, globally unique, and stable identifiers – in contrast to keys within databases – target candidates are exactly the previously discovered unique fields in primary relations of other databases.

Second, in the biological universe there are many relationships between objects that are not explicitly stored. We discover such implicit relationships by searching for similar data values among the sources. Similarity, and especially sequence similarity, is a ubiquitous indication for object relationships in the life sciences. For instance, similarity between protein sequences or protein structures is the most important way of inferring the function of a new protein [AMS+97]. Of course, not all attributes are promising targets for a comparison, either because similarity doesn't mean anything (similarity of keys), or be-

cause similarity is not detectable by comparing single values (similarity of 3D protein structures). We currently focus on three specific types of comparisons. First, the values of attributes containing DNA, RNA, or protein sequences are compared to each other. Finding sequence fields is simple, as those contain only strings over a fixed alphabet (A, C, T, G for genes). Second, attributes containing longer text strings, such as textual descriptions, can be analyzed by using techniques from information retrieval and text mining. In particular, methods for finding names of biological entities in natural text can be used for extracting names that are matched with unique fields of primary relations potentially holding the name of objects [CSA04; HBP+05]. Third, life science researches are increasingly aware of the importance of using standardized vocabularies across databases. Such vocabularies, called ontologies in this area, are now widely used for annotating gene function (Gene Ontology), microarray data (MGED ontology), and will be used for proteomics data (PSI-MI ontology). The resulting values make excellent links, connecting proteins with similar function or experiments with similar parameters, provided that the ontologies are themselves integrated as data sources.

Conceptually, to discover all such links, we need to look at each pair of attributes among two databases. However, substantial pruning can be applied based on data characteristics. For instance, the attribute representing the target of a cross-reference is always a primary key in the respective table. Further, attributes with few distinct values should be excluded from being a link source, as are attributes with purely numeric values to avoid misinterpretation of surrogate keys.

All links prior to this step result in "normal" joins, whereas here we must store the discovered links on the object level in the metadata repository to avoid repeated discovery and computation at query time.

The link discovery task is closely related to schema matching, especially to those projects using instance-based techniques. Schema matching discovers correspondences between elements of heterogeneous schemata. The resulting mappings between schemata improve the overall performance of query answering in terms of how much data can be accessed and how relevant it is to the query. Examples of schema matching projects are iMAP [DLD+04], similarity flooding [MGR02], and Clio's schema matcher [NHT+02].

4.5 Duplicate Detection

The links mentioned in the previous section connect objects based on individual data values, in particular based on pairs of similar attribute values. The underlying assumption is that the objects are somehow connected if they have similar, non-trivial attribute values. Another interesting relationship between objects is duplicity. Not only do many sources overlap in their intension, but also in their extension, storing data about same real world objects. For instance, the last years have seen an explosion in databases storing data on observed or predicted protein interaction, such as BIND, MIND, WIT, or Reactome. These databases have largely overlapping content, i.e., they partly store the same interactions. However, the data is differently modeled, and there are differences in the set of attributes that are managed in each database.

Duplicate detection techniques are well known in the relational database world. However, duplicate detection in the biological domain cannot rely on well-structured and fine-granular attribute values to identify duplicates. Instead, values of unstructured text fields must be analyzed, and the heterogeneous and only partly overlapping model of the data must be accounted for.

Duplicate detection in semi- and unstructured data is more difficult than in the relational case. It is not a priori clear, which attribute values of one object to compare with which attribute value of the other object. Thus, common similarity measures employed to identify duplicates cannot be applied immediately. Methods considering the nested structure of objects [WN04] and applying more advanced similarity measures taken from then information retrieval are currently developed [BN05].

Knowledge about duplicates is very useful in an environment such as ALADIN. First, duplicates might complement each other: One source knows one thing about an object; the other source might know some additional information. Also, duplicates provide a chance to choose between alternatives. In answering a query, only one representative of each duplicate cluster can be returned. On the other hand, duplicates give rise to data conflicts. Different sources might contradict each other in the data they store about an object. Usually it is up to the experts to decide which of the values (or both) is correct. Especially the life sciences domain produces many such contradicting values, due to the different experimental methods to obtain the values. Exploring such contradictions is of great interest to biologists. A concrete example of the importance of duplicate detection is given in the case study (Section 5).

4.6 Data Access Engine

The data access component connects users with the data. Users have three modes of access, namely browsing, searching and querying. Browsing and searching capabilities are usually already offered by the Web front ends of the individual sources, albeit only on their own data. Explicit cross references between data source are realized as HTML links, allowing users to browse from one source to the other. The ALADIN browser provides uniform access to all stored data. It can follow not only cross-references, but all four types of relationships between objects:

1. **Same relation.** Trivially, users can jump from object to object within a relation.

2. **Dependency.** Users can follow links to secondary objects to obtain more information about a certain object.
3. **Duplicates**. Users can follow links to other database objects that talk about the same real world object. Conflicts are highlighted, and data lineage is shown.
4. **Linked**. Finally, users can follow links between data sources, from one object to another.

Search allows a full-text search on all stored data and a focused search restricted to certain vertical (e.g., a single attribute-type) and horizontal partitions (e.g., only on primary objects) of the data. Ranking algorithms order the search results based on similarity of the result to the query. Searching requires efficient indexes on very large amounts of data. For this goal, we rely on commercial vendor software on top of a database, such as DB2's Search Extender or Oracle 10g's regular expression search capabilities.

Posing queries against such a vast, semi-structured and semi-connected database, is a difficult task. Of particular importance is to rank results according to certainty values derived from the different discovery steps during data import. The situation is similar to that of peer-data-management-systems, where different data sources are connected with mappings. Answers to queries against a data source at one peer are answered by querying data stored locally, but also by following mappings and retrieving information at the other peers.

All three access modes are supported by appropriate user interfaces. In particular the SQL interface must be simple enough to allow even novice users to formulate meaningful queries. The query interface of the COLUMBA prototype [RMT+04] is a good example allowing users iterative query refinement by adding more and more filter conditions.

5 A Case Study on Protein Data

Many of the design decisions of ALADIN are based on experiences drawn from integration projects in the life sciences domain that the authors pursued in the past. Here, we shortly discuss the most recent one, the COLUMBA project [RMT+04] [6].

COLUMBA is an integrated, relational data warehouse that annotates protein structures taken from the Protein Data Bank (PDB) with data on classification of structures, protein sequence and sequence features, functional annotation of proteins, participation in metabolic pathways, and some more. The workflow of integration consists of the steps data import, mapping of source schemata into the COLUMBA target schema, and connecting PDB entries to data from other databases through cross-references.

Developing the data import routines takes relatively little time. All but two of the data sources in COLUMBA are available either directly as dump files or as flat-files together with a proper parser. The two exceptions, i.e., the protein classification databases CATH and SCOP, consist mostly of a hierarchy of classes of protein domains and links to the actual proteins; writing a parser took only a few hours in both cases. This experience shows that relying on a relational database as starting point for the hands-off integration process is a reasonable assumption.

The import generates a relational representation of a data source using a source- and parser-specific schema. However, we decided to store only a fraction of all data in COLUMBA to keep queries simple and fast. In the COLUMBA target schema each data sources has its own subschema. The extraction and transformation from the initial data source schema into the target schema is currently hard-coded, and its development required a large fraction of the overall effort. First, understanding the schemata is very difficult, as they are often poorly documented and frequently use metadata-like structures that cannot be understood by looking only at the schema [Mil98]. Second, transforming the data requires operations that are beyond current schema mapping languages [MHH00]. Transformations in COLUMBA are therefore carried out by a mixture of SQL and Java.

COLUMBA focuses on high-quality data, thus the tedious and manually specified transformation is a necessary step. However, ALADIN aims at automatic data integration accepting some compromises in terms of quality. Our experience from reengineering more than a dozen relational life science schemata makes us convinced that the heuristics described in Section 4.2 and 4.3 are justified. As an example, consider a fraction from the BioSQL schema used for storing imported data from Swiss-Prot and EMBL (see Figure 3)[7].

In the full schema, there are three tables with an in-degree above five: *BioEntry* for storing the primary objects, *OntologyTerm* for storing functional descriptions, and *SeqFeature* storing a meta-representation of sequence features (not visible in the Figure). Of those, only *BioEntry* has an accession number candidate (*accession*), whose values are mixed characters and integers and all have the same length. The other fields in *BioEntry* are either non-unique (e.g. *taxon_id*), have no alphanumeric character (e.g. *bioentry_id*), or have varying length (e.g. *name*). Thus, this table would correctly be identified in ALADIN as the primary relation. Secondary relations could be correctly connected to the primary relation once the primary key – foreign key constraints are detected by analyzing the scope of the different attributes storing surrogate keys. In BioSQL, dictionary tables for various types of keywords are filled only with those terms that are actually referenced, and no two dictionary tables have an equal number of tuples (in the current release). Thus, relationships could be correctly guessed.

In the next step in COLUMBA, protein structures are connected to annotations using either existing cross-

[6] See http://www.columba-de.de

[7] See http://obda.open-bio.org.

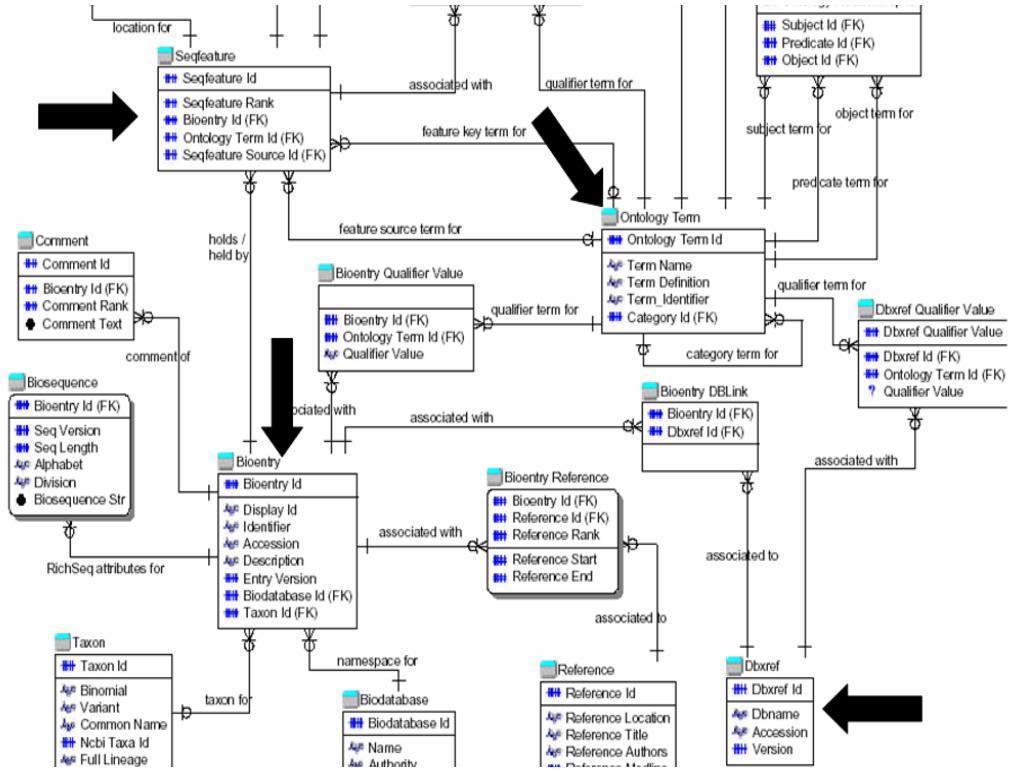

Figure 3. Part of the BioSQL schema. Arrows indicate candidates for primary relations and for cross-references.

references or by sequence similarity. This step seems to be straight-forward, but is not for two reasons. First, many of the secondary databases are maintained by human data curators and there is a considerable backlog in annotating structures. This backlog appears as missing links in COLUMBA. Automatic methods for computing database links are therefore a very important aspect. Second, in many cases there is more than one source linking two databases. For instance, there exist at least five different sets of links from Swiss-Prot to PDB [Mar04]. These sets overlap, but also differ to a considerable degree. Ranking of results based on the strength of evidence is thus a very important feature to support users adequately.

Our heuristic for connecting objects across databases can be demonstrated again using the BioSQL schema with its fields for linking out to other databases. The target of existing cross-references, the *accession* field, would be identified as potential target for links from other sources because of its characteristics (see preceding). The source of external cross-references in BioSQL is the field *DBRef.accession*, where external Ids are indeed stored as described in Section 4.4. Note that we would not be able to use the information in the attribute *DBRef.database* (which specifies the target database), because we analyze only single attributes, not combinations. However, we also do not need this information, as matching the values of *DBRef.accession* against all unique fields of primary relations automatically would find the correct target database.

The BioSQL schema also contains several attributes whose values are excellent candidates for finding implicit links, such as *OntologyTerm.Term_definition*, linking into biological ontologies, *BioEntry.description*, linking to disease or gene-focused databases, and *Biosequence.Biosequence_str*, containing the actual DNA or protein sequence.

Finally, duplicate detection is an important step in COLUMBA, because the protein structures from the PDB are actually available in three different flavors: the original PDB files, a cleansed version available with a parser [BBF+01], and another cleansed version available as dump files [BDF+03]. These three versions, even though derived from the same data set, differ greatly due to different cleansing procedures. Selecting the proper value for each data field is an important problem [MLF04]. Detecting duplicate objects is easy in this case, because the original PDB accession number is available in all three representations. Duplicate detection becomes more difficult for gene and disease data, for which many more and more heterogeneous databases are available than for protein structure [LLRC98].

Apart from the databases integrated into COLUMBA, we have also studied a number of other databases, especially in the area of genomic maps [LLRC98] and gene

expression databases [MSZ+01]. We are convinced that the heuristics described earlier work very well in most of the cases, although we have no proof for this claim yet. The COLUMBA database shall serve as a "learning" test set for estimating the performance of ALADIN's various analysis algorithms. Thus, precision and recall methods for finding primary relations, secondary relations, cross-references, and duplicates can be derived.

6 Conclusions

We have proposed the ALADIN architecture and framework as a novel approach to data integration in the life sciences. It is designed for (almost) hands-off data integration, relying on a mixture of techniques from database, data mining, and information retrieval. We argued that this approach is feasible, because life science databases share certain properties that can be cast into effective heuristics for an automatic execution of most tasks necessary to achieve data integration.

ALADIN offers clear added-value to the biological user when compared to the current data landscape. It enables structured queries crossing several databases for users knowing the schemata (and offers support for those who do not), suggests a wealth of new relationships interlinking all areas of the life sciences, and offers ranked search capabilities across databases for users that prefer Google-style retrieval.

We make one more important point regarding queries across different databases and across different object types. Consider a query for all genes of a certain species on a certain chromosome that are connected to a disease via a protein whose function is known. For each of the object types in the query, several potential data sources exist, as there are several potential sources for the links connecting the objects. However, no current data integration system is capable of dealing with this variability in a transparent fashion; instead, the database administrator usually decides which source to integrate, thus leaving no freedom of decision to the end user. In contrast, ALADIN helps the user in several ways: First, as integration requires almost no manual intervention, integrating all databases is feasible. Second, duplicates are detected automatically, thus reducing the redundancy in the result. Third, data sources concentrating on links, such as Integr8 or DBLink, can be integrated as well with almost no effort. Finally, query results can be ordered based on the number, consistency, and length of different paths between two objects, as suggested in [BLM+04].

We conclude the paper with a survey of related work and an overview of bottlenecks and challenges to our proposal.

6.1 Related work

The problem of data integration in the life sciences has been explored for many years. Various types of integration middleware and methods have been proposed. DiscoveryLink is an extension of IBM's DB2 database system that is capable deploying wrappers to access data outside of the database through SQL from inside DB2 [HSK+01]. Many wrappers exist for life science databases, accessing the data either locally or via the Internet. OPM offers an object-oriented data model and the query language OPM*QL, which can join to remote databases for which an OPM wrapper exists [CKMST98]. TAMBIS maps data source schemata into a global schema formulated in the description logic GRAIL, thus allowing for more semantics in queries [BBB+98]. All these projects have in common that they focus on schema information and do not make use of data in any fashion.

The Sequence Retrieval System (SRS) is a system for integrated access to biological flat-file databases [ZLAE00]. Each database must be wrapped by a specific parser written in the proprietary Icarus programming language. Primary and secondary relations as well as database cross-references need to be explicitly specified in the parser code, which contrasts to our approach where this information is deduced automatically. SRS does not treat duplicate data. Unlike ALADIN, SRS is essentially a schema-driven approach; like ALADIN, SRS focuses on cross-references.

An interesting system is the GenMapper, which integrates data from more than forty data sources into a 4-table schema [DR04]. GenMapper also concentrates on cross-references; however, no attempt is made to find implicit relationships and data import requires the definition of manual mappings into the four-table target schema. A similar approach, although with a more elaborate model of target schemata, defined using the Protégé knowledge base system, is BioMediator [SMB+04]. Recently, two projects proposed a star-schema-like integration model for the life science domain, where information from various sources is connected to a central class using database cross-references. Our COLUMBA database [RMT+04] integrates twelve protein-related databases, whereas the EnsMart project aims at a more generic solution [KKS04]. ALADIN has in common with these projects that data is integrated only via cross-references; however, the research focus of those approaches is completely different.

Apart from individual algorithms and techniques already cited at appropriate places, in the database research area, the project closest to our proposal is probably the Revere project [HED+03]. Based on the PDMS called Piazza, the specific user interface generator Mangrove, and a corpus of metadata and tools thereon are added. The user interface displays unstructured or semi-structured data and entices users to add more structure (and thus semantics) to the data. With more semantics, more interesting queries are answerable with more underlying data.

6.2 Challenges and Bottlenecks

Our proposal for building a new type of data integration system that mostly relies on data characteristics instead of schema information leaves open several questions, the most important one being "Will it work?". Although most of the discovery steps presented here rely on existing algorithms and techniques that have been successfully applied in many different application domains, the ALADIN system is a true challenge in terms of size, number, and complexity of the data sources to be integrated. How well these techniques interoperate in a complex domain such as the life sciences is yet to be found out. In particular, errors in earlier steps propagate and might influence the quality of later results. For instance, incorrectly identifying the primary or secondary relations leads to incorrect targets for the link discovery. Incorrect links in turn influence the precision of duplicate detection.

Nevertheless, we believe that the particular nature of life science data sources and prior experiences offer good arguments for optimism. We also plan to incorporate user feedback into the query interface. Users browsing the data or query results from ALADIN might indicate that a link between two objects or even between two schema elements was inserted incorrectly. Thus, especially false links between relations can be removed quickly.

Correspondence with users of the COLUMBA database ensures that the functionality provided by ALADIN suits typical biologists well and provides added-value at small to zero maintenance cost. For instance, typical microarray experiments produce a set of 50-100 genes. Biologists then manually browse a large number of web sites following hyper links for each gene. Such browsing, enriched with many more links, reduced redundancy due to duplicate detection, and the full capability of SQL queries would be perfectly supported by ALADIN.

One important issue not addressed in the paper is that of performance. While we point out that data sources can be incrementally added, each addition in its own involves an enormous number of calculations. Because we assume no schema, in principle each inclusion step involves calculations on all data of the source: Data import makes at least one pass over the data. Discovery of primary and secondary objects conceptually performs a sort on each attribute and then a join-operation on each pair of attributes from different relations. Link discovery first employs schema matching techniques (with differing complexity) on all attributes of all data sources and then conceptually again a join operation on all matching attributes. In all, the efficiency of adding a new data source will be low, but much better than manually adding a new source and establishing connections between its data and the data of the other already integrated sources. Furthermore, sampling can be used, and there are various heuristics for eliminating attributes from consideration for the different tasks.

Another subtle problem is that of data changes. In principle, all links must be recomputed even if only a small fraction of the data of a data source changes. This re-computation is clearly infeasible. We envisage a threshold on the number of changes to a data source before a new analysis is carried out.

In summary, we believe that the ALADIN system offers both interesting challenges to the database community and high value to the life science community. We plan to take up these challenges in the near future.

References

[AMS+97] Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. J. (1997). "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs." Nucleic Acids Research 25(17): 3389-402.

[Aug01] Augen, J. (2001). "Information technology to the rescue!" Nature Biotechnology 19(6).

[BBB+98] Baker, P. G., Brass, A., Bechhofer, S., Goble, C., Paton, N. and Stevens, R. (1998). "TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources". 6th Int. Conf. on Intelligent Systems for Molecular Biology, Montreal, Canada, AAAI Press, Menlo Park.

[BBF+01] Bhat, T. N., Bourne, P., Feng, Z., Gilliland, G., Jain, S., Ravichandran, V., Schneider, B., Schneider, K., Thanki, N., Weissig, H., et al. (2001). "The PDB data uniformity project." Nucleic Acids Res 29(1): 214-8.

[BN05] Bilke, A. and Naumann, F. (2005). "Schema Matching using Duplicates". International Conference on Data Engineering, Tokyo, Japan (to appear).

[BLM+04] Bleiholder, J., Lacroix, Z., Murthy, H., Naumann, F., Raschid, L. and Vidal, M.-E. (2004). "BioFast: Challenges in Exploring Linked Life Science Sources." SIGMOD Record 33(2).

[BBA+93] Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M. C., Estreicher, A., Gasteiger, E., Martin, M. J., Michoud, K., O'Donovan, C., Phan, I., et al. (2003). "The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003." Nucleic Acids Research 31(1): 365-70.

[BDF+03] Boutselakis, H., Dimitropoulos, D., Fillon, J., Golovin, A., Henrick, K., Hussain, A., Ionides, J., John, M., Keller, P. A., Krissinel, E., et al. (2003). "E-MSD: the European Bioinformatics Institute Macromolecular Structure Database." Nucleic Acids Research 31(1): 458-62.

[CSA04] Chang, J. T., Schutze, H. and Altman, R. B. (2004). "GAPSCORE: finding gene and protein names one word at a time." Bioinformatics 20(2): 216-25.

[CKMST98] Chen, I.-M. A., Kosky, A. S., Markowitz, V. M., Szeto, E. and Topaloglou, T. (1998). "Advanced Query Mechanisms for Biological Databases". 6th Int. Conf. on Intelligent Systems for Molecular Biology, Montreal, Canada, AAAI Press, Menlo Park.

[DLD+04] Dhamankar, R., Lee, Y., Doan, A., Halevy, A. Y. and Domingos, P. (2004). "iMAP: Discovering Complex Mappings between Database Schemas". SIGMOD Conference, Paris, France.

[DR04] Do, H.-H. and Rahm, E. (2004). "Flexible Integration of Molecular-biological Annotation Data: The GenMapper Approach". Int. Conf. on Extending Database Technology, Heraklion, Greece.

[HSK+01] Haas, L. M., Schwarz, P. M., Kodali, P., Kotlar, E., Rice, J. and Swope, W. C. (2001). "DiscoveryLink: A System for Integrated Access to Life Sciences Data Sources." IBM Systems Journal 40(2): 489-511.

[HBP+05] Hakenberg, J., Bickel, S., Plake, C., Brefeld, U., Zahn, H., Faulstich, L., Leser, U. and Scheffer, T. (2004). "Systematic Feature Evaluation for Gene Name Recognition." BMC Bioinformatics, to appear.

[HED+03] Halevy, A. Y., Etzioni, O., Doan, A., Ives, Z., Madhavan, J., McDowell, L. and Tatarinov, I. (2003). "Crossing the Structure Chasm". 1st Conference on Innovative Data Systems Research (CIDR), Asilomar, CA.

[HBB+02] Hubbard, T., Barker, D., Birney, E., Cameron, G., Chen, Y., Clark, L., Cox, T., Cuff, J., Curwen, V., Down, T., et al. (2002). "The Ensembl genome database project." Nucleic Acids Research 30(1): 38-41.

[KKS04] Kasprzyk, A., Keefe, D., Smedley, D., London, D., Spooner, W., Melsopp, C., Hammond, M., Rocca-Serra, P., Cox, T. and Birney, E. (2004). "EnsMart: a generic system for fast and flexible access to biological data." Genome Research 14(1): 160-9.

[KM92] Kivinen, J. and Mannila, H. (1992). "Approximate Dependency Inference from Relations". International Conference on Database Theory.

[KCMS98] Kosky, A. S., Chen, I.-M. A., Markowitz, V. M. and Szeto, E. (1998). "Exploring Heterogeneous Biological Databases: Tools and Applications". 6th Int. Conf. on Extending Database Technology, Valencia, Spain.

[LLRC98] Leser, U., Lehrach, H. and Roest Crollius, H. (1998). "Issues in Developing Integrated Genomic Databases and Application to the Human X Chromosome." Bioinformatics 14(7): 583-590.

[MSZ+01] Mangalam, H., Stewart, J., Zhou, J., Schlauch, K., Waugh, M., Chen, G., Farmer, A. D., Colello, G. and Weller, J. W. (2001). "GeneX: An Open Source Gene Expression Database and Integrated Tool Set." IBM Systems Journal 40(2): 552-569.

[MLP02] Marchi, F. D., Lopes, S. and Petit, J.-M. (2002). "Efficient Algorithms for Mining Inclusion Dependencies". Int. Conf. on Extending Database Technology (EDBT).

[Mar04] Martin, A. C. (2004). "PDBSprotEC: a Web-accessible database linking PDB chains to EC numbers via SwissProt." Bioinformatics 20(6): 986-8.

[MGR02] Melnik, S., Garcia-Molina, H. and Rahm, E. (2002). "Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching". Int. Conf. on Data Engineering (ICDE).

[Mil98] Miller, R. J. (1998). "Using Schematically Heterogeneous Structures". ACM SIGMOD Int. Conference on Management of Data 1998, Seattle, Washington.

[MHH00] Miller, R. J., Haas, L. M. and Hernandez, M. A. (2000). "Schema Mapping as Query Discovery". 26th Conference on Very Large Database Systems, Cairo, Egypt, ACM Press, New York.

[MLF04] Mueller, H., Leser, U. and Freytag, J. C. (2004). "Mining for Patterns in Contradictory Data". Workshop on Information Quality in Information Systems, Paris, France.

[NHT+02] Naumann, F., Ho, C.-T., Tian, X., Haas, L. M. and Megiddo, N. (2002). "Attribute Classification Using Feature Analysis". Int. Conf. on Data Engineering (ICDE).

[NJM03] Naumann, F., Josifivski, V. and Massmann, S. (2003). "Super-Fast XML Wrapper Generation in DB2 (demo)". Int. Conference on Data Engineering (ICDE), Bangalore, India.

[RB01] Rahm, E. and Bernstein, P. A. (2001). "A survey of approaches to automatic schema matching." The VLDB Journal 10(4): 334-350.

[Rob95] Robbins, R. J. (1995). "Information Infrastructure for the Human Genome Project." IEEE Engineering in Medicine and Biology 14(6): 746-759.

[RML04] Rother, K., Michalsky, E. and Leser, U. (2004). "How well are protein structures annotated in annotation databases." Submitted.

[RMT+04] Rother, K., Müller, H., Trissl, S., Koch, I., Steinke, T., Preissner, R., Frömmel, C. and Leser, U. (2004). "COLUMBA: Multidimensional Data Integration of Protein Annotations". Int. Workshop on Data Integration in the Life Sciences (DILS), Leipzig, Germany, Spinger.

[SMB+04] Shaker, R., Mork, P., Brockenbrough, J. S., Donelson, L. and Tarczy-Hornoch, P. (2004). "The BioMediator System as a Tool for Integrating Biologic Databases on the Web". Workshop on Information Integration on the Web (IIWeb04), Toronto, CA.

[Ste03] Stein, L. D. (2003). "Integrating biological databases." Nature Rev Genet 4(5): 337-45.

[WN04] Weis, M. and Naumann, F. (2004). "Detecting Duplicate Objects in XML Documents". Workshop on Information Quality in Information Systems, Paris, France.

[WHA+02] Wu, C. H., Huang, H., Arminski, L., Castro-Alvear, J., Chen, Y., Hu, Z. Z., Ledley, R. S., Lewis, K. C., Mewes, H. W., Orcutt, B. C., et al. (2002). "The Protein Information Resource: an integrated public resource of functional annotation of proteins." Nucleic Acids Research 30(1): 35-7.

[ZLAE00] Zdobnov, E. M., Lopez, R., Apweiler, R. and Etzold, T. (2000). "The EBI SRS Server - Recent Developments." Bioinformatics 18(2): 368-373.