Cluster based Integration of Heterogeneous Biological Databases using the AutoMed toolkit

Michael Maibaum[1], Lucas Zamboulis[2], Galia Rimon[2], Christine Orengo[1], Nigel Martin[2], and Alexandra Poulovassilis[2]

[1] Department of Biochemistry and Molecular Biology, University College London, Gower Street, London WC1E 6BT

[2] School of Computer Science and Information Systems, Birkbeck College, University of London, London WC1E 7HX

Abstract. This paper presents an extensible architecture that can be used to support the integration of heterogeneous biological data sets. In our architecture, a clustering approach has been developed to support distributed biological data sources with inconsistent identification of biological objects. The architecture uses the AutoMed data integration toolkit to store the schemas of the data sources and the semi-automatically generated transformations from the source data into the data of an integrated warehouse. AutoMed supports bi-directional, extensible transformations which can be used to update the warehouse data as entities change, are added, or are deleted in the data sources. The transformations can also be used to support the addition or removal of entire data sources, or evolutions in the schemas of the data sources or of the warehouse itself. The results of using the architecture for the integration of existing genomic data sets are discussed.

1 Introduction

This paper presents work on an architecture for integrating biological data sources, and reports our experience in applying it to an existing application aimed at providing an integrated sequence/structure/function resource that supports analysis, mining and visualisation of functional genomics data (transcriptomic and proteomic).

Biological data sources are characterised by a very high degree of heterogeneity in terms of the type of data model used, the schema design within a given data model, as well as incompatible formats and nomenclature of values. Further, such data sources frequently make use of large numbers of unstable, inconsistent identifiers for biological entities. Our architecture addresses these two issues by combining two data integration techniques supporting both data heterogeneity and inconsistent identifiers.

The database community has done much work on integration of data from heterogeneous data sources. Examples of significant applications to biological data sources include DiscoveryLink [8], K2/Kleisli [12] and Tambis [5]. In practice, the most widely used system is Sequence Retrieval System (SRS) [30]. A recent survey is provided by [17].

SRS represents one approach to integration: it acts as a portal to data sources exploiting indexes built by the system. It therefore has a more restricted aim than DiscoveryLink, K2/Kleisli and Tambis which are all aimed at supporting higher level query facilities across data sources. DiscoveryLink and Tambis aim to achieve this without users needing to be aware of source data schemas: in our own work we also aim to insulate users in this way.

The two traditional approaches to providing such transparent access are to materialise the integrated data in a warehouse, or alternatively to provide virtual integration with mediator software supporting access to data in the original data sources. Materializing integrated data in a warehouse is usually done on performance grounds: not only is distributed access to remote data sources avoided, but also centralised database query optimisation techniques can be applied to enable complex queries to be supported more efficiently. Maintaining a materialised warehouse to correctly reflect updates in data sources can be complex, however. While access to a virtual warehouse is likely to be less efficient than with a materialised warehouse, it may be the only option if it is not possible to extract data from the underlying data sources, or if the storage overheads of materialisation would be too high.

In our own work we have chosen to exploit the AutoMed data integration toolkit[3] to support the integration of heterogeneous biological data sources. The particular strength of AutoMed for this application area is that it supports bi-directional, extensible transformations from data source schemas to an integrated schema enabling integration both through explicit materialisation in a data warehouse as well as virtual integration of data remaining in the original data resources. The extensibility of AutoMed transformations is also the basis for update of schemas within both the data sources and any materialised warehouse.

AutoMed does not in itself provide a solution for transformations between unstable, inconsistent identifiers. There are a number of significant initiatives within the Life Sciences community to address the problem of inconsistent identifiers. For example, the Life Sciences Identifiers (LSID) initiative [25] is aimed at a standardised scheme for assigning and recognising identifiers for biological entities, while the International Protein Index (IPI) [10] is developing stable identifiers for human, mouse and rat proteomes. Meeting the needs of applications that process and analyze transcriptomics and proteomics data is a particular motivation for such work. Extensive work has also been done on standardisation in more specialised areas, for example the work of the Microarray Gene Expression Data (MGED) Society on MAGE-ML for standardised recording of data related to microarray gene expression experiments [11]. However, the legacy of very large numbers of inconsistent non-standardised identifiers will remain.

Hence, in our work we have combined AutoMed with a clustering approach to associate biological entities independently of their identifiers. In our application of this approach so far, we have used gene sequence clustering to establish associations, but the approach is not limited to sequence-based clustering.

---

[3] See `http://www.doc.ic.ac.uk/automed/`

The remainder of the paper is organised as follows. Section 2 introduces those features of AutoMed which have been exploited in our work, together with the basis for combining AutoMed with a clustering approach. Section 3 presents our data integration framework. Section 4 reports on our experience applying this framework to the integration of biological data sources in a warehouse being constructed to support the mining and visualisation of functional genomics data. Conclusions and a discussion of ongoing work are given in Section 5.

2 Background

2.1 The AutoMed Toolkit

AutoMed is a heterogeneous data transformation and integration system which offers the capability to handle virtual, materialised and indeed hybrid data integration across multiple data models. AutoMed supports a low-level hypergraph-based data model (HDM), and provides facilities for specifying higher-level modelling languages in terms of this HDM. These specifications are stored within AutoMed's Metadata Repository [1]. In the specific application described in this paper, the problem addressed has been the integration of relational data sources into a relational data warehouse.

AutoMed provides a set of primitive schema transformations that can be applied to schema constructs. In particular, for every construct of a modelling language $\mathcal{M}$ there is an add and a delete primitive transformation which add to/delete from a schema an instance of that construct. For those constructs of $\mathcal{M}$ which have textual names, there is also a rename primitive transformation. For example, in a simple relational model there may be four kinds of modelling construct, Rel, Att, primaryKey and foreignKey.

Instances of modelling constructs within a particular schema are uniquely identified by their *scheme*, enclosed within double chevrons $\langle\langle ... \rangle\rangle$. AutoMed schemas can be incrementally transformed by applying to them a sequence of primitive transformations, each adding, deleting or renaming just one schema construct (thus, in general, AutoMed schemas may contain constructs of more than one modelling language). Each add or delete transformation is accompanied by a query specifying the extent of the new or deleted construct in terms of the rest of the constructs in the schema. This query is expressed in a functional query language, IQL[4]. AutoMed also provides contract and extend primitive transformations which behave in the same way as add and delete except that they indicate that their accompanying query may only partially specify the extent of the new/removed schema construct. Their query may just be the constant Void, indicating that the extent of the new/removed construct cannot be specified even partially, in which case the query can be omitted.

[4] IQL is a comprehensions-based functional query language, and we refer the reader to [18] for details of its syntax, semantics and implementation. Such languages subsume query languages such as SQL and OQL in expressiveness [2].

A sequence of primitive transformations from one schema S_1 to another schema S_2 is termed a transformation *pathway* from S_1 to S_2, denoted by $S_1 \rightarrow S_2$. All source, intermediate, and global schemas, and the pathways between them, are stored in AutoMed's Metadata Repository.

AutoMed has its theoretical foundations in the schema transformation and integration framework described in [22] where it was shown that this approach generalises all the previous notions of 'schema equivalence'. Intuitively, this is because: (a) sequences of the primitive transformations are able to express syntactically any transformation from one schema to another, with first a 'growing' phase which adds missing schema constructs and then a 'shrinking' phase which removes redundant schema constructs; (b) IQL queries are able to express the semantic relationships between a new schema construct and the existing constructs, or between a removed schema construct and the remaining constructs.

The IQL queries present within transformations that add or delete schema constructs mean that each primitive transformation has an automatically derivable *reverse transformation*. In particular, each add/extend transformation is reversed by a delete/contract transformation with the same arguments, while each rename transformation is reversed by swapping its two arguments. [19] discusses how the queries present within these reversible schema transformation pathways can be used to generate view definitions for global schema constructs in terms of source schema constructs. Essentially, this is by means of query unfolding using the queries within delete, contract and rename transformations along the set of reverse pathways from a global schema to a set of source schemas.

AutoMed pathways can be used to express the data cleansing, transformation and integration processes involved in heterogeneous data integration. The queries within transformations also allow the pathways to be used for materialising and incrementally maintaining a materialised global database, and any materialised databases derived from it, in the face of insertions/ deletions/ updates to the data sources. The queries within transformations also allow the pathways to be used for tracing the *lineage* of data in a materialised global database, or any materialised databases derived from it, to the data sources. We refer the reader to [13, 14] for details of these uses of AutoMed pathways.

In any heterogeneous data integration environment, it is possible for either a data source schema or the global database schema to evolve. This schema evolution may be a change in the schema, or a change in the data model in which the schema is expressed, or both. An AutoMed pathway can be used to express the schema evolution in all of these cases. Once the current transformation network has been extended in this way, the actions taken to evolve the rest of the transformation network and schemas, and any materialised derived data, are localised to just those schema constructs that are affected by the evolution. We refer the reader to [23, 24, 15] for details of how this can be achieved in both virtual [23, 24] and materialised [15] integration scenarios. The algorithms used are mainly automatic, except for input of domain or expert human knowledge regarding the semantics of new schema constructs added to a local or global schema which are not semantically equivalent to any existing constructs in the schema.

For our particular application here, the task has been to support the transformation of biological data source schemas into a global warehouse schema. The data source and warehouse schemas were relational, while we have used an XML-based unifying data model for the intermediate schemas. We made this choice in order to allow the use of AutoMed's facilities for automatically transforming and integrating XML data, which are discussed in detail in [28, 29].

The standard schema definition languages for XML are DTD and XML Schema. However, both of these provide grammars to which conforming documents adhere to, and do not summarise the tree structure of the data sources. In our schema transformation setting, schemas of this type are preferable as this facilitates schema traversal, structural comparison between a source and a target schema, and restructuring the source schema(s) that are to be transformed and/or integrated. Moreover, such a schema type means that the queries supplied with AutoMed primitive transformations are essentially path queries, which are easily generated.

The AutoMed toolkit therefore supports a modelling language *XML Data-Source Schema* (XMLDSS) which summarises the tree structure of XML documents, much like DataGuides [16]. XMLDSS schemas consist of four kinds of constructs (see [28] for details of their specification in terms of the HDM): Element, Attribute, PCData and NestList. The last of these are parent-child relationships between two elements e_p and e_c and are identified by a scheme of the form $\langle\langle \mathsf{i}, \mathsf{e_p}, \mathsf{e_c} \rangle\rangle$, where i is the position of e_c within the list of children of e_p in the XMLDSS schema. In an XML document there may be elements with the same name occurring at different positions in the tree. To avoid ambiguity, in XMLDSS schemas we use an identifier of the form elementName$count for each element, where count is a counter incremented every time the same elementName is encountered in a depth-first traversal of the schema. An XMLDSS schema can be automatically derived from an XML document, as discussed in [28], and it is also possible to automatically derive an XMLDSS schema from a DTD or an XML Schema specification, if available.

2.2 Clustering for Supporting Multiple IDs

While AutoMed is well-suited to the task of supporting transformations of data source schemas into a global warehouse schema, it provides no mechanisms for supporting the equivalence of inconsistent identifiers. Integrating data sources usually results in incomplete matching of related entities in the different data sets, either due to identifier redundancy or due to the use of different reference identifiers. In the case of some biological databases, the percentage of entities that can be matched using a single identifier can be very low. When trying to match proteins from KEGG Gene to the Gene Ontology Gene Products less than 40% match, despite the sources nominally describing the same entities.

Data-based entity clustering provides a general approach to integrating any set of logically related entities and hence supporting multiple identifiers. Under this approach, an appropriate relatedness measure is developed (for example sequence or structure similarity), allowing each entity in the data being integrated to be compared to each of the other entities and a similarity index derived. Once the similarity measure values have been obtained they can be used to organise the entities hierarchically into nested sets. Each level of nesting represents an increasing degree of similarity between the entities contained in the set, allowing each application built on the integrated data source to determine what is an appropriate degree of clustering for that application. In the context of biological data, for example, protein structure is conserved at low levels of sequence similarity compared to function and therefore clusters with lower levels of similarity can be used when structural annotation is desired rather than functional.

Having generated such sets of related entities, information applicable to each set may be extracted and associated with that set. Moreover, an attribute which is only defined for a subset of members may be inferred for remaining members of a set if it is known that the attribute will be shared amongst similar entities.

Use of an appropriate similarity measure and clustering algorithm provides sets of entities that represent the same 'real world' entity that may never have been associated based purely on an identifier mapping. Sets of entities with a lower level of similarity represent entities that are less closely related. While this approach does not allow identification of identical entities, in biological contexts it is often at least, if not more useful to identify similar entities, given the incomplete knowledge about any individual entity.

This type of approach is applicable to many types of data. There is no inherent limitation on the type of clustering or the type or types of similarity measures used to compare entities. For example given a measure of similarity of scientific publications was available, the related articles could be organised into clusters providing links between articles on similar topics. In the simplest case this might be based on keyword matching, but other far more sophisticated approaches are available.

3 Our Data Integration Framework

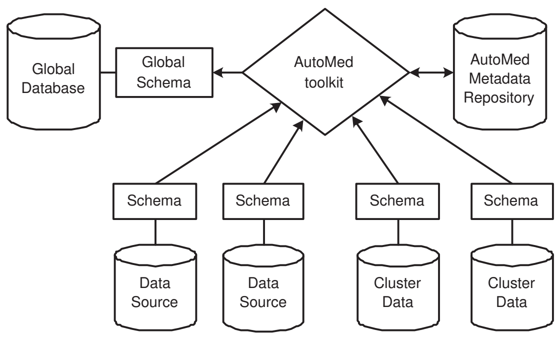

Fig. 1. Architectural Overview of the Data Integration Framework

The architecture of our biological data integration framework is illustrated in Figure 1. There are two principal sources of information for the Global Schema — data sources and cluster data — which are processed in the same way but contain different types of information. Each Data Source is an externally maintained resource that is to be integrated as part of the global database. A data source could be a conventional relational or other structured database, or a semi-structured data source, such as an XML file. Conceptually, a data source describes facts about biological entities. Each Cluster Data resource is constructed from one or more data sources and provides the basis for a generally applicable approach to the integration of data lacking a common reference identifier as discussed in Section 2.2 above. Conceptually, a cluster data resource provides a data-dependent classification of the entities within data sources into related sets.

Each data source is either a structured data source such as a relational database (in which case its associated Schema is a relational schema) or a semi-structured file (in which case it may or may not have an associated schema). In the latter case a schema appropriate for the data source can be generated by the appropriate AutoMed wrapper (see [28, 1] for details of extracting schemas from semi-structured data). Cluster data resources are maintained as relational data with an associated relational schema. The schemas of data source and cluster data resources are processed in the same way, and an arbitrary number of data sources and methods of clustering can be integrated.

Some data sources do not contain a primary key identifier that is persistent between versions of the resource. The lack of a persistent primary key identifier makes the identification of changes between each version difficult. For such data sources a non-volatile, primary key identifier is generated for each entity and added to the data source. Persistent primary key identifiers provide a simple, generic primary key for the higher level tools to use and enables synchronisation of the warehouse with the changing content of the underlying data source.

Wrappers provided by the AutoMed Toolkit automatically generate the AutoMed internal representations of the Schemas and the Global Schema, and store these in the AutoMed Metadata Repository. The AutoMed toolkit is then used to generate the transformation pathways from the Schemas to the Global Schema. These are described in detail with illustration from the example application in Section 4.3 below.

Virtual Integration. After the integration process has been completed, and the transformation pathways from a set of data source schemas to a global schema have been set up, queries formulated with respect to the global schema can be evaluated. Such a query is submitted to AutoMed's Global Query Processor (see [18]) which first reformulates it into a query that can be evaluated over the data sources. This is accomplished by following the reverse transformation pathways from the global schema to the data source schemas in order to generate view definitions of global schema constructs in terms of data source constructs. These view definitions are substituted into the original query, which is then optimised. The query evaluator then interacts with the data source wrappers in submitting to them IQL subqueries which they translate into the local query language for evaluation, returning sub-query results back to the evaluator for any further necessary post-processing and merging.

Materialised Integration. The current version of the BioMap warehouse (see Section 4 below) was materialised using conventional SQL queries on relational sources, before the AutoMed components of our architecture were in place. This approach is labour intensive as the queries must all be manually designed. Upcoming iterations of the warehouse will however be able to benefit from AutoMed's facilities for incrementally maintaining the warehouse. In general, the data sources may be updated by the insertion, deletion or modification of data. Deltas on data sources may result in deltas on cluster data resources also. Both kinds of deltas can be propagated through the AutoMed transformation pathways up to the materialised global database (and to any other materialised databases derived from it). In particular, the queries within add and extend transformation steps can be used to compute a new set of deltas from the current set of deltas, all the way up to the target database (see [14]).

4 Application of the Framework to Gene Family Based Integration

The above architecture has been applied to biological data sources integrated within the BioMap data warehouse. In this section we describe how the architecture has been applied and the results of the work to date.

4.1 The BioMap Warehouse

BioMap is a collaborative project to develop a warehouse integrating protein family, structure, function and pathway/process data with gene expression and other experimental data. The aim is to provide an integrated sequence/structure/function resource that supports analysis, mining and visualisation of functional genomics data (transcriptomic and proteomic). The warehouse is implemented within Oracle, extending techniques developed for the CATH-PFDB database [26] and is designed to serve as a source for data marts which will themselves be constructed using the AutoMed techniques presented in this paper.

Current data sources include the CATH protein structure family database [6], KEGG pathway database [20], Gene3D annotated protein sequence database [21], Gene Ontology [9], EBI Macromolecular structure database (MSD) [4] and ten other resources. Thus far, we have taken CATH, Gene3D, KEGG_Gene, KEGG_Genome, KEGG_Orthology, and also a CLUSTER data source discussed below, representing a significant subset of BioMap data sources describing structural, functional, sequence and ontological information. These contain a diverse set of data structures, formatting conventions and sizes to use for evaluation of our data integration framework.

4.2 The Clustering Approach

There are a variety of methods for classifying biological entities into sets and these methods can be used on the facts within the data warehouse. The facts concerning individual entities within a set will not all derive from precisely the same biological entity, but by choosing an appropriate algorithm to create the sets, the set will contain valuable information about biological entities that are similar (in some way) to each other. One such categorisation method is UniGene [7]. Our categorisation method is based on the PFScape protocol [21] which is in turn based on the TRIBE-MCL algorithm [3]. The PFScape protocol was developed for Gene3D and has been adapted and improved for BioMap. In brief, to construct Gene3D the peptide sequences of more than 120 completed genomes were obtained from the NCBI and from ENSEMBL. An 'all vs all' BLAST was performed using the blastpgp program from the NCBI. The BLAST was performed using a cluster of 50 dual processor machines running GNU/Linux using Sun Grid Engine. An e-value cut off of 0.001 was used. The results were used to create a similarity matrix which was used by TRIBE-MCL to create protein families.

Since then, many more completed genomes have become available, in particular the genome of the Rat. Other genomes have been revised. For the BioMap project an extension of the PFScape protocol has been developed to update the Gene3D families.

The complete genomes of more than 203 Archea, Prokaryotes and Eukaryotes were downloaded from the EBI. For each sequence in Gene3D and the downloaded proteomes an md5 was calculated and an 'all vs all' BLAST performed. The BLAST results were filtered using an 80 percent overlap cutoff to select only the BLAST hits that represented whole chain matches. Each novel sequence was assigned into the best hit family for each of the new sequences, or if no family was identified then a new family was created. Within the protein families multi-linkage clustering was performed based on sequence identity using cluster thresholds of 0.3, 0.35, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1. The clustering was performed using TCluster, a locally developed program.

To integrate the other data sources, a representative sequence was obtained for each entity in the data source and a md5 calculated. The set of md5s that were not present in the genomic sequences was then obtained. The sequences corresponding to those md5s were then compared to the genomic sequences using BLAST as described above. The entities were then classified in terms of the genomic clusters based on their best hits.

4.3 The Integration Process

The integration process consists of the following steps, of which steps 3 to 6 are explained in more detail below. Steps 1 and 2 are carried out automatically by AutoMed's relational wrapper, as mentioned in Section 3.

1. Automatic generation of the AutoMed relational schemas, $LS_1, \ldots, LS_n$, corresponding to the Data Source and Cluster Data Schemas.

2. Similarly, automatic generation of the AutoMed relational schema, GS, corresponding to the Global schema.
3. Automatic translation of schemas $LS_1, \ldots, LS_n$ and GS into the corresponding XMLDSS schemas $X_1, \ldots, X_n$ and GX.
4. Partial conformance of each schema X_i to GX by means of appropriate rename transformations, to ensure that only semantically equivalent schema constructs share the same name, and that all equivalent schema constructs do share the same name. This results in a set of new schemas $X'_1, \ldots, X'_n$.
5. Completing the conformance of each schema X'_i to GX by applying an automatic XMLDSS schema transformation algorithm to each pair of schemas X'_i, GX, creating a set of new schemas $X''_1, \ldots, X''_n$.
6. Application of any necessary data cleansing transformations on each X''_i, creating a set of schemas $GX_1, \ldots, GX_n$. As the integration of the schemas up to this point does not involve any reference to the actual data, the data cleansing does not have to be performed prior to this step.

In Steps 4 - 6, the pathways $LS_1 \to X_1, \ldots, LS_n \to X_n$ generated by Step 3 are extended with further primitive transformations, leading finally to the schemas $GX_1, \ldots, GX_n$ in Step 6.

Each GX_i is identical to the global XMLDSS schema GX from Step 3. The reverse of the pathway $GS \to GX$ generated in Step 3 can finally be appended to each GX_i to transform it into the relational global schema GS.

Step 3: Translating AutoMed relational to XMLDSS schemas. To translate a relational schema into an XMLDSS schema we first generate a graph, G, from the relational schema. There is a node in G corresponding to each table in the relational schema. There is an edge from R_1 to R_2 in G if there is a foreign key in R_2 referencing the primary key of R_1. In the given relation schemas there are no cycles in G — in a general setting, we would have to break any cycles at this point. We create a set of trees, T, obtained by traversing G from each node that has no incoming edges, and we convert T into a single tree by adding a generic root. We finally use T to generate the pathway from the relational schema to its corresponding XMLDSS schema. This last phase consists of traversing T and, for each node t encountered, doing the following:

(i) If t is the root, insert a PCData construct into the current schema, and then insert the root itself as an Element construct.
(ii) else:
  (a) insert t as an Element
  (b) insert a NestList construct from the parent of t to t
  (c) find the columns c_i belonging to the table that corresponds to t, and for each c_i: insert c_i as an Element construct; insert a NestList construct from t to c_i; and insert NestList constructs from c_i to PCData.
(iii) For each child of t, t'_i, treat t'_i as t in step (i).
(iv) Remove the now redundant relational constructs from the schema.

To illustrate the translation, the top of Figure 2 illustrates a part of the schema of the CLUSTER data source (where ASSIGNMENT_TYPE_ID in ASSIGNMENT_TYPES is referenced by CLUSTER_TYPE in CLUSTER_DATA, and the rest of the foreign keys have the same name as the primary keys they reference). At the bottom, the XMLDSS schema that corresponds to this relational schema is illustrated. Similarly, Figure 3 illustrates a part of the relational global schema and the corresponding AutoMed XMLDSS schema.

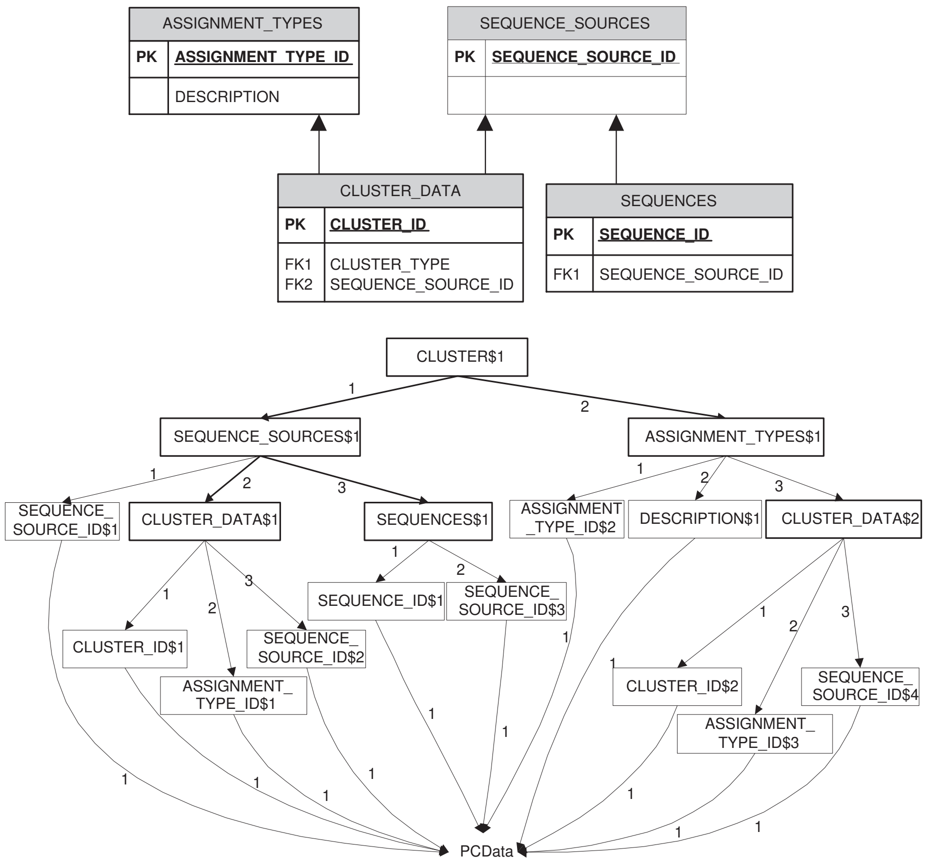

Fig. 2. Top: part of the CLUSTER relational schema. Bottom: corresponding part of the CLUSTER XMLDSS schema.

Step 4: Schema Matching. The XMLDSS schema transformation algorithm used in Step 5 of the integration process assumes that if two schema constructs in a local schema and in the global schema, respectively, have the same name, then they refer to the same real-world concept, and if they do not have the same name, they do not. We do not currently support automatic schema matching in our integration process. Thus, after the XMLDSS schemas are produced, and before the application of the schema transformation algorithm in Step 5,

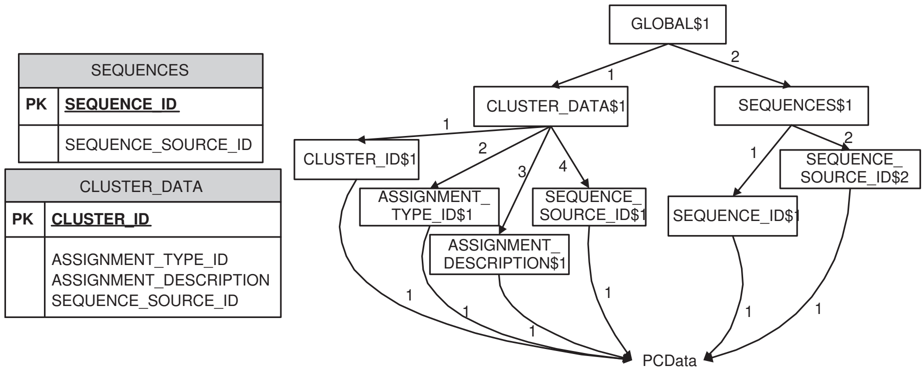

Fig. 3. Left: part of the global relational schema. Right: corresponding part of the XMLDSS schema.

the necessary rename transformations must be manually issued on each source XMLDSS schema. These rename transformations effectively simulate a schema matching phase and in our case they have been produced by a domain expert. However, the AutoMed toolkit also offers a tool for performing semi-automatic schema matching and generating the corresponding AutoMed transformation pathways [1]. We also note that this schema matching step does not have to be performed on the XMLDSS schemas, but could instead be performed on the source relational schemas. The only necessity is for this step to be performed before the application of the schema transformation algorithm.

In our running example, the domain expert produced the following rename transformations on the XMLDSS schema in Figure 2:

```
rename(<<CLUSTER$1>>,<<GLOBAL$1>>);
rename(<<DESCRIPTION$1>>,<<ASSIGNMENT_DESCRIPTION$1>>);
rename(<<SEQUENCE_SOURCE_ID$1>>,<<PSEQID>>);
rename(<<SEQUENCE_SOURCE_ID$2>>,<<SEQUENCE_SOURCE_ID$1>>);
rename(<<SEQUENCE_SOURCE_ID$3>>,<<SSEQID>>);
rename(<<SEQUENCE_SOURCE_ID$4>>,<<SEQUENCE_SOURCE_ID$2>>);
rename(<<ASSIGNMENT_TYPE_ID$2>>,<<PASSID>>);
rename(<<ASSIGNMENT_TYPE_ID$3>>,<<ASSIGNMENT_TYPE_ID$2>>)
```

and the following rename transformation on the XMLDSS schema in Figure 3:

```
rename(<<SEQUENCE_SOURCE_ID$2>>,<<SSEQID>>)
```

Step 5: Automatic XMLDSS-based integration. The algorithm for automatically transforming a source XMLDSS schema S into a target XMLDSS schema T has three phases:

Growing phase: Traverse T in a depth-first fashion and for every schema construct encountered that is not present in S, issue an add or extend transformation, resulting in an intermediate schema S_1.

Shrinking phase: Traverse S_1 in a depth-first fashion and for every schema construct encountered that is not present in T, issue a delete or contract transformation, resulting in an intermediate schema S_2.

Renaming phase: Traverse S_2 in a depth-first fashion and issue the necessary rename transformations needed to rename the ordering labels of the NestList constructs in order to create the correct ordering of these constructs, resulting in a final schema S_T syntactically identical to the target XMLDSS schema T.

For reasons of space, we refer the reader to [29] for a detailed description of this algorithm. To illustrate the algorithm, we list below a part of the pathway generated to transform the XMLDSS schema in Figure 2 to the XMLDSS schema in Figure 3, after the earlier rename transformations of Step 4 have first been applied. Here `makelist` is a built-in IQL function that takes a value v and a number n and produces a list consisting of n copies of v:

```
add(<<0,GLOBAL$1,CLUSTER_DATA$1>>,
       [{v0,v2}|{v0,v1}<-<<GLOBAL$1,SEQUENCE_SOURCES$1>>;
        {v1,v2}<-<<SEQUENCE_SOURCES$1,CLUSTER_DATA$1>>]);
add(<<0,GLOBAL$1,SEQUENCES$1>>,
       [{v0,v2}|{v0,v1}<-<<GLOBAL$1,SEQUENCE_SOURCES$1>>;
        {v1,v2}<-<<SEQUENCE_SOURCES$1,SEQUENCES$1>>]);
extend(<<0,CLUSTER_DATA$1,ASSIGNMENT_DESCRIPTION$1>>,
       [{v1,v2}|{v0,v1}<-<<ASSIGNMENT_TYPES$1,CLUSTER_DATA$2>>;
        {v0,v2}<-<<ASSIGNMENT_TYPES$1,ASSIGNMENT_DESCRIPTION$1>>);
delete(<<1,GLOBAL$1,SEQUENCE_SOURCES$1>>,
       makelist {'GLOBAL$1','SEQUENCE_SOURCES$1'}
                 (count <<SEQUENCE_SOURCES$1>>));
delete(<<1,SEQUENCE_SOURCES$1,PSEQID>>,
       makelist {'SEQUENCE_SOURCES$1','PSEQID'}
                (count <<PSEQID>>));
contract (<<1,PSEQID,PCData>>);
contract (<<PSEQID>>);
delete(<<2,SEQUENCE_SOURCES$1,CLUSTER_DATA$1>>,
       makelist {'SEQUENCE_SOURCES$1','CLUSTER_DATA$1'}
                (count <<CLUSTER_DATA$1>>));
delete(<<3,SEQUENCE_SOURCES$1,SEQUENCES$1>>,
       makelist {'SEQUENCE_SOURCES$1','SEQUENCES$1'}
                (count <<SEQUENCES$1>>));
contract(<<SEQUENCE_SOURCES$1>>);
```

The unwanted edges on the RHS of the XMLDSS schema of Figure 2 are deleted/contracted similarly. A series of rename transformations then follows to create a contiguous ordering of edges beneath a parent element.

Step 6: Data cleansing. After the local XMLDSS schemas have been conformed with the global XMLDSS schema, the domain expert can manually issue any further necessary transformations to remove any representational heterogeneities at the data level. AutoMed transformations can express the transformation of data from one format to another in the same way as they can express the transformation of schema structures. For example, consider in our running example attribute `DESCRIPTION` in relation `ASSIGNMENT_TYPES` (see Figure 2). The extent of this attribute in the data source consists of mixed case strings.

In the CLUSTER XMLDSS schema this attribute is called `DESCRIPTION$1`. After the partial conformance step (Step 4 in Section 4.3), the attribute has been renamed to `ASSIGNMENT_DESCRIPTION$1`. To turn the extent of this attribute to uppercase strings before merging with the other data sources in the global schema, the following transformations can be appended to the transformation pathway resulting from the conformance step (Step 5 in Section 4.3):

```
add(<<0,ASSIGNMENT_DESCRIPTION$1,PCData>>,
    [{v0,stringUpper v1} |
     {v0,v1}<-<<1,ASSIGNMENT_DESCRIPTION$1,PCData>>]);
contract(<<1,ASSIGNMENT_DESCRIPTION$1,PCData>>);
rename(<<0,ASSIGNMENT_DESCRIPTION$1,PCData>>,
       <<1,ASSIGNMENT_DESCRIPTION$1,PCData>>)
```

Here `stringUpper` is a built-in IQL function that converts all the alphabetic characters in a string to upper-case. Several other string-handling functions are supported by IQL e.g. `stringLower`, `stringConcat` and `stringSplit`. The IQL query processor is implemented in such a way that extending it with new built-in functions is straightforward.

In general with AutoMed, these kinds of data cleansing transformations can take place at any stage of the integration process. It is also possible to incorporate materialised correspondences between data values in source and target schemas into data cleansing transformations — this extensional information is treated as another data source.

Implementation and Results. The above integration process was carried out on a Pentium 4 2.8Ghz, with 1Gb RAM and Linux as the operating system. The Gene3D, KEGG_Gene, KEGG_Genome, KEGG_Orthology, CATH and CLUSTER data sources, and the global database are all Oracle databases. The AutoMed repository is stored in a PostgreSQL database, and the AutoMed toolkit itself is written in Java. The integration of each data source took under 15 minutes, resulting in a total running time of about 85 minutes. Many of the algorithms are not yet fully optimised and therefore we expect a major performance improvement as more optimisations are built into the AutoMed toolkit.

5 Conclusions and Future Work

This paper has presented a data integration framework for biological data sources that combines techniques to support the diversity of data models, schemas and formats which are characteristic of biological data together with a clustering approach developed to support distributed biological data sources with inconsistent identification of biological objects.

The work we have described is this paper is currently being extended in a number of areas. First, the approach is being applied to the other data sources noted in Section 4 with further detailed evaluation of the results obtained. The clustering approach is also being extended: while the use of sequence families is described here, other methods of classification could be used including structural and many other approaches. We are currently working on a method that integrates feature and domain recognition (hidden Markov model) approaches to identify attributes of sequences. These attributes (i.e. a structural domain, or a protein active site) can be used to form clades, within which the existing clustering information can be organised. This combination of two clustering approaches will provide the best features of the extremely sensitive, but time consuming scanning approaches with the less sensitive, but much faster simple sequence comparisons.

In the BioMap warehouse we have so far successfully applied the AutoMed-based techniques for the data cleansing, transformation and integration processes as presented in Section 4. We are currently implementing AutoMed-based materialisation and maintenance of the global database, which have been manual processes to date. Use of AutoMed will enable delta changes to be automatically propagated to the global database as well as allowing schema changes to accommodated.

The techniques presented on this paper have not so far been applied to integrating textual data sources such as PubMed abstracts within BioMap. However, work has already been done on extending AutoMed with facilities for integrating unstructured text with structured data [27], and these techniques will be applied to textual biological data sources.

A further collaborative project, ISPIDER, aims to develop Grid-based data integration of biological data resources. The strengths of AutoMed for supporting bi-directional and incrementally constructed transformation pathways are of particular value in a Grid environment, and work is being pursued on developing these techniques and integrating them with existing Web Service and Grid middleware components for service discovery and metadata management.

References

1. M. Boyd, S. Kittivoravitkul, C. Lazanitis, P.J. McBrien, and N. Rizopoulos. AutoMed: A BAV data integration system for heterogeneous data sources. In *Proc. CAiSE'04*, 2004.
2. P. Buneman, L. Libkin, D. Suciu, V. Tannen, and L. Wong. Comprehension syntax. *SIGMOD Record*, 23(1):87–96, 1994.
3. A J Enright, S Van Dongen, and C A Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res*, 30(7):1575–84, 2002.
4. A. Golovin *et al.* E-MSD: an integrated data resource for bioinformatics. *Nucleic Acids Res*, 32 Database issue(1362-4962):D211–6, 2004.
5. C.A. Goble *et al.* Transparent access to multiple bioinformatics information sources. *IBM Systems Journal*, 40(20):532–552, 2001.
6. C.A. Orengo *et al.* CATH–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–108, 1997.
7. D.L. Wheeler *et al.* Database resources of the National Center for Biotechnology: update. *Nucl. Acids Res.*, 32 Database Issue:D35–D40, 2004.
8. L.M. Haas *et al.* DiscoveryLink: A system for integrated access to life sciences data sources. *IBM Systems Journal*, 40(20):489–511, 2001.
9. M. Ashburner *et al.* Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genet.*, 25(1):25–29, 2000.

10. P.J. Kersey *et al.* The International Protein Index: An integrated database for proteomics experiments. *Proteomics*, 4(7):1985–1988, 2004.
11. P.T. Spellman *et al.* Design and implementation of microarray gene expression markup language (MAGE-ML). *Genome Biology*, 3(9):research0046.1–0046.9–1988, 2002.
12. S. Davidson *et al.* K2/Kleisli and GUS: Experiments in integrated access to genomic data sources. *IBM Systems Journal*, 40(20):512–531, 2001.
13. H. Fan and A. Poulovassilis. Tracing data lineage using schema transformation pathways. In *Knowledge Transformation for the Semantic Web*, volume 95 of *Frontiers in Artificial Intelligence and Applications*, pages 64–79. IOS Press, 2003.
14. H. Fan and A. Poulovassilis. Using AutoMed metadata in data warehousing environments. In *Proc. DOLAP 2003*, pages 86–93, 2003.
15. H. Fan and A. Poulovassilis. Schema evolution in data warehousing environments — a schema transformation-based approach. In *Proc. ER'04*, pages 639–653, 2004.
16. R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proc. VLDB'97*, pages 436–445, 1997.
17. T. Hernandez and S. Kambhampati. Integration of biological sources: Current systems and challenges ahead. *Sigmod Record*, 33(3):51–60, 2004.
18. E. Jasper, A. Poulovassilis, and L. Zamboulis. Processing IQL queries and migrating data in the AutoMed toolkit. Technical Report 20, Automed Project, 2003.
19. E. Jasper, N. Tong, P. McBrien, and A. Poulovassilis. View generation and optimisation in the AutoMed data integration framework. In *Proc. CAiSE Forum at CAiSE'03*, pages 29–32. Univ. of Maribor Press, June 2003.
20. M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The KEGG resources for deciphering the genome. *Nucl. Acids Res.*, 32 Database Issue:D277–D280, 2004.
21. D. Lee, A. Grant, R. Marsden, and C. Orengo. Identification and distribution of protein families in 120 completed genomes using Gene3D. *Proteins (In Press)*, 2004.
22. P. McBrien and A. Poulovassilis. A formalisation of semantic schema integration. *Inf. Syst.*, 23(5):307–334, 1998.
23. P. McBrien and A. Poulovassilis. Schema evolution in heterogeneous database architectures, a schema transformation approach. In *Proc. CAiSE'02*, pages 484–499, 2002.
24. P. McBrien and A. Poulovassilis. Data integration by bi-directional schema transformation rules. In *Proc. ICDE'03*, pages 227–238, 2003.
25. OMG. Life Sciences Identifiers RFP response. http://www.omg.org/cgi-bin/doc?lifesci/2003-12-02, 2003.
26. A.J. Shepherd, N.J. Martin, R.G. Johnson, P. Kellam, and C.A. Orengo. PFDB: a generic protein family database integrating the CATH domain structure database with sequence based protein family resources. *Bioinformatics*, 18(12):1666–72, 2002.
27. D. Williams and A. Poulovassilis. Combining data integration with natural language technology for the semantic web. In *Proc. Workshop on Human Language Technology for the Semantic Web and Web Services, ISWC'03*, 2003.
28. L. Zamboulis. XML data integration by graph restructuring. In *Proceedings of BNCOD'04*. Springer, July 2004.
29. L. Zamboulis and A. Poulovassilis. Using AutoMed for XML data transformation and integration. In *Proceedings of DIWeb'04*. Springer, June 2004.
30. E.M. Zdobnov, R. Lopez, R. Apweiler, and T. Etzold. The EBI SRS Server - recent developments. *Bioinformatics*, 18(2):368–373, 2002.