

# Cluster based integration of Heterogeneous Biological Databases using the AutoMed toolkit

Michael Maibaum<sup>1</sup>, Lucas Zamboulis<sup>2</sup>, Galia Rimon<sup>2</sup>, Christine Orengo<sup>1</sup>, Nigel Martin<sup>2</sup>, and Alexandra Poulouvasilis<sup>2</sup>

<sup>1</sup> Department of Biochemistry and Molecular Biology, University College London, Gower Street, London WC1E 6BT

<sup>2</sup> 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. There are three major obstacles in such an endeavour: the use of different identifiers for the same biological entities, the diversity of the data models underpinning the biological data, and the requirement to keep the integrated data warehouse current in the face of data and schema changes in the source data sets. In our architecture, entities are categorised into clusters allowing individual biological entities to be annotated with family based data. For example, sequence based clustering enables gene family based annotation of individual sequences.

We use the AutoMed data integration toolkit to store the schemas of the data sources and also the transformations from the source data into the data of the integrated warehouse. These transformations are generated semi-automatically by a process of schema matching and schema restructuring. The transformations 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 in the integration of existing genomic data sets are discussed.

## 1 Introduction

Integrating heterogeneous biological data sets is a key challenge facing the life sciences and database communities. To give one concrete example, following the success of the international genome projects, post-genomic analysis now aims to describe where and when genes are expressed, and how they function in normal and diseased states. The advent of technologies such as gene microarrays has contributed enormously to our ability to address these questions, but they generate vast amounts of data, and interpreting this transcriptomics data to shed light on biological function is a major challenge. At the same time as such transcriptional analysis is booming, considerable data on protein family, function and pathway/process data is accumulating as a result of the structural and

functional genomics initiatives. This data can provide crucial prior knowledge on gene function. The development of successful strategies for mining transcriptomics data will critically depend on integrating the expression data with this protein family/structure/function data.

This paper presents work on a general architecture for integrating biological data sources, and reports our experience on 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).

Our approach focuses on three major problems. First, biological data sources are characterised by the use of large numbers of unstable, inconsistent identifiers for biological entities. Second, such data sets are also 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. Finally, biological data sources are characterised by frequent changes to data and schemas.

Our architecture addresses the first issue of unstable, inconsistent identifiers by using 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. The remaining issues of the heterogeneity and dynamic nature of biological data sources are addressed by exploiting the AutoMed data integration toolkit. The particular strength of AutoMed for this application area is that it supports bi-directional, extensible transformations between data sources, 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.

The paper is organised as follows. Section 2 discusses the problems arising in integration of biological data sources, and previous approaches to solving these problems. Relevant work on AutoMed is introduced, as well as the motivation for our clustering approach. Section 3 presents our data integration framework, while Section 4 reports on our experience applying the 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 Integrating biological data sources

There are three major obstacles to the integration of biological data sets: the use of different identifiers for the same biological entities, the diversity of the data models underpinning the biological data, and the requirement to keep the integrated data current in the face of data and schema changes in the source data sets.

The first issue — use of different identifiers for biological entities — can be a significant problem even within a single biological data source. For example, within the GenBank nucleotide sequence database [7], two distinct types of sequence identification are used. The first, a GI number, identifies a sequence submitted for inclusion in the database — if that submission is updated or changed in any way a new GI number is issued. However, GI numbers are widely misused as an identifier for a particular biological entity (i.e a gene or protein). Furthermore, as many people may sequence the same gene or peptide, there are many GI numbers for a single biological entity. The position is yet further complicated by the fact that GenBank is part of a consortium of sequence databases, each of which has been able to assign its own GI number to a particular sequence.

A more appropriate identifier, an accession number, was introduced to achieve greater consistency. An accession number for a sequence record does not change if the record is modified at the submitter's request: instead, a version number suffix is incremented. However, an original accession number may be associated with a newer accession number if, for example, a submission is made that combines previously submitted sequences.

Given the complexity of identifiers within a data source such as GenBank, it is unsurprising that other data sources reference Genbank sequences in inconsistent ways: in particular, many still use GI numbers rather than accession versions. More generally, a vast range of different identifiers exist for the same or similar biological entities, so even if two data sources are individually maintained correctly, they may well be impossible to map together based on the identifiers in use if they have chosen a different set of reference identifiers to use.

The second obstacle to biological data set integration is the diversity of schemas used, encompassing both the data model and the specific schema within a given data model. Conventional structured relational and object-oriented database models are commonly used to underpin biological databases, but much biological data which has traditionally been held in formatted flat files is now maintained as XML data. Even for data within a particular data model such as the relational model, there is a multitude of possible schema designs, since much biological data, for example sequence data and biochemical pathway data, does not have a single natural representation in data models originally motivated by the needs of business applications with more simply structured data. The multitude of incompatible formats and nomenclatures used by scientists to store and annotate their data adds further complexity.

These issues would be challenging enough with static biological data sources. In reality, new and updated data is generated constantly due to improvements in transcriptomics and proteomics technology, while schemas are themselves constantly evolving to meet the needs of new applications.

Both the life sciences and database communities have developed techniques to help in these three areas. Support for multiple, inconsistent identifiers is not an area which has received significant attention by the database community: the focus has rather been on support for uncertainty and imprecision in non-key values. While the main effort on standardisation in the life sciences community

has been to develop ontologies to deal with inconsistent nomenclature, progress has been made with the standardisation of identifiers. For example, the Life Sciences Identifiers (LSID) initiative [28] is aimed at a standardised scheme for assigning and recognising identifiers for biological entities, while the International Protein Index (IPI) [11] 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 [12]. However, the legacy of very large numbers of inconsistent non-standardised identifiers will remain.

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 [9], K2/Kleisli [13] and Tambis [5]. In practice, the most widely used system is Sequence Retrieval System (SRS) [35]. A recent survey is provided by [18].

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. In either case, a common conceptual data model is normally used: in the materialised case, extract/transform/load routines populate a warehouse whose schema reflects that model, while in the virtual case the mediator software returns data extracted/transformed to reflect the model.

The use of a single conceptual model as the basis for integration gives rise to a number of disadvantages [15]. First, since both the conceptual and individual data source models are likely to be based on high-level data models, there is scope for semantic mismatches between the two with a loss of information as data is transformed between them. Second, the transformation metadata is likely to be tightly coupled with the conceptual model of the target warehouse, making it difficult to modify the transformations should revisions to the conceptual model be required. Finally, if a data source model should change, again it will not be straightforward to modify the transformations.

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 mate-

rialised 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.

Given the problems arising with a data integration approach based on a single high-level conceptual model, and the desirability of supporting both materialised and virtual integration as appropriate, we have chosen to exploit the AutoMed data integration toolkit to support the integration of heterogeneous biological data sources.

## 2.2 The AutoMed Toolkit

AutoMed<sup>3</sup> is a heterogeneous data transformation and integration system which offers the capability to handle virtual, materialised and 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].

An HDM schema consists of a set of nodes, edges and constraints, and each modelling construct of a higher-level modelling language is specified as some combination of HDM nodes, edges and constraints. Instances of modelling constructs within a particular schema are uniquely identified by their *scheme*, enclosed within double chevrons  $\langle\langle \dots \rangle\rangle$ .

For example, in a simple relational model there may be two kinds of modelling construct, *Rel* and *Att*. A *Rel* construct  $\langle\langle R \rangle\rangle$  corresponds to an HDM node labelled *R* while an *Att* construct  $\langle\langle R, a \rangle\rangle$  corresponds to two HDM nodes *R* and *a*, and an edge between them. The extent of a *Rel* construct  $\langle\langle R \rangle\rangle$  is the projection of the relation *R* onto its primary key attributes,  $k_1, \dots, k_n$  say. The extent of each *Att* construct  $\langle\langle R, a \rangle\rangle$  where *a* is an attribute (key or non-key) of *R* is the projection of relation *R* onto  $k_1, \dots, k_n, a$ <sup>4</sup>.

For any modelling language  $\mathcal{M}$  specified in terms of the HDM (via the API of AutoMed's Metadata Repository), AutoMed's data source Wrappers translate data source schemas expressed in  $\mathcal{M}$  into their AutoMed representation. This AutoMed representation is stored in AutoMed's Metadata Repository [1].

AutoMed provides a set of primitive schema transformations that can be applied to schema constructs expressed in  $\mathcal{M}$ . In particular, for every construct of  $\mathcal{M}$  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.

AutoMed schemas can be incrementally transformed by applying to them a sequence of primitive transformations  $t_1, \dots, t_r$ , each primitive transformation adding, deleting or renaming just one schema construct. Thus, AutoMed schemas may contain constructs of more than one modelling language.

---

<sup>3</sup> See <http://www.doc.ic.ac.uk/automed/>

<sup>4</sup> We refer the reader to [25] for an encoding of the full relational data model in the HDM, including the modelling of constraints.

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<sup>5</sup>.

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. Moreover, 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<sup>6</sup>.

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 [1].

The queries present within transformations that add or delete schema constructs mean that each primitive transformation  $t$  has an automatically derivable *reverse transformation*,  $\bar{t}$ . 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.

Thus, AutoMed is a *both-as-view* (BAV) data integration system. As discussed in [27], BAV subsumes the global-as-view (GAV) and local-as-view (LAV) approaches [23], since it is possible to extract a definition of each global schema construct as a view over source schema constructs, and it is also possible to extract definitions of source schema constructs as views over the global schema. We refer the reader to [20] for details of AutoMed’s GAV and LAV view generation algorithms.

BAV is a more expressive data integration approach than than LAV, GAV or indeed GLAV [17, 24] since it allows the expression of mappings in both directions, and since it is not limited on how many source schemas are associated by a mapping. As discussed in [26, 27], a further advantage of BAV over GAV and LAV is that it readily supports the evolution of both global and local schemas, by allowing pathways and schemas to be incrementally modified as opposed to having to be regenerated. A further difference between BAV and GAV, LAV or GLAV is that statements about the relationships between global and local schemas are made at a finer level of detail e.g. at the level of individual attributes as opposed to entire tables.

Figure 1 illustrates the general integration scenario with AutoMed. Each data source is described by a local schema  $LS_i$ . Each  $LS_i$  is first conformed into a

---

<sup>5</sup> IQL is a comprehensions-based functional query language, and we refer the reader to [19] for details of its syntax, semantics and implementation. Such languages subsume query languages such as SQL and OQL in expressiveness [2].

<sup>6</sup> More generally, the *contract* and *extend* transformations can take a pair of queries  $(lq, uq)$  specifying a lower and upper bound on the extent of the new/removed construct, instead of just one lower-bound query as described above. The constant *Void* in place of  $lq$  indicates no information about the lower bound of the extent, while the constant *Any* in place of  $uq$  indicates no information about the upper bound of the extent.

schema  $CS_i$  (which may or may not be expressed in the same modelling language as  $LS_i$ ) by means of a transformation pathway  $T_i$ . Not all of the information within a local schema  $LS_i$  need be transferred into the global schema and this is asserted by means of contract transformation steps within  $T_i$ . Conversely, there may be information within the global schema which is not semantically derivable from  $LS_i$ , and this is asserted by the pathway from  $CS_i$  to a ‘union-schema’  $US_i$  which consists of the necessary extend transformations<sup>7</sup>.

All the union schemas  $US_1, \dots, US_n$  are syntactically identical and this is asserted by creating a pathway of id transformations between each pair  $US_i$  and  $US_{i+1}$ , of the form  $\text{id } US_i : c \text{ } US_{i+1} : c$  for each schema construct  $c$  within  $US_i$  and  $US_{i+1}$  (the transformation pathways containing these id transformations are automatically generated by the AutoMed software). Once this has been done, the extent of each construct  $c$  in any union schema  $US_i$  is equal to the bag-union of the extent of  $c$  in all the union schemas  $US_1, \dots, US_n$ . That is, id is interpreted as bag-union by AutoMed’s view generation functionality. An arbitrary one of the  $US_i$  ( $US$  in Figure 1) can be selected for further transformation into the global schema  $GS$  (by the pathway  $T_u$  in Figure 1).

In a virtual data integration scenario, there is no data associated with any of the schemas apart from the  $LS_i$ . In a data warehousing scenario,  $CS_1, \dots, CS_n$  would be fully materialised and consist of the detailed data of the warehouse. This detailed data would be further augmented with the necessary summary views by the transformations in the pathway  $T_u$ , and these summary views materialised in the global database  $GD$ .

More generally, it is possible to materialise any of the intermediate or global schemas in an AutoMed integration network. It is also possible to use the global schema of one network (which may be materialised or virtual) as a source schema of another integration network, whose global schema may again be either materialised or virtual. In this way, hybrid virtual/materialised integration is possible.

In the context of this paper, all the  $LS_i$  and  $LD_i$  have been extracted from the original data sources and the data in each  $LD_i$  has been cleansed. The data cleansing process can also be expressed using AutoMed transformations — see [15] for details and also Section 4.4 below. See also [15] for a discussion how AutoMed transformations can express both structural and representational changes to schemas and data.

### 2.3 Using AutoMed pathways

We have discussed above how the AutoMed pathways illustrated in Figure 1 can be used to express the data cleansing, transformation and integration processes involved in heterogeneous data integration. The queries within add/extend 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

---

<sup>7</sup> If there are none, then this pathway is empty and  $CS_i$  and  $US_i$  are the same schema

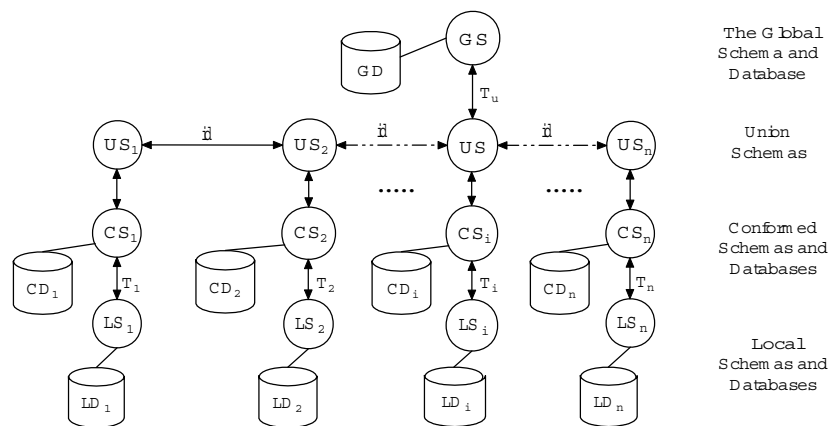


Fig. 1. Data Integration in AutoMed

lineage of data in a materialised global database, or any materialised databases derived from it, to the data sources. We refer the reader to [14, 15] 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. AutoMed pathways can be used to express the schema evolution in all of these cases, allowing the previous transformation, integration and data materialisation effort to be reused. We discuss this further in Section 3.5 below.

## 2.4 XMLDSS Schemas

In this paper, we are concerned with the integration of several relational and XML biological data sources using the AutoMed toolkit. For this particular integration, we have used XML as the unifying data model in which the conformed schemas and union schemas ( $CS_i$  and  $U_i$  in Figure 1) are expressed. We made this choice in order to allow the use of AutoMed’s facilities for automatically transforming and integrating XML data. The global schema  $GS$  is relational in our setting.

The standard schema definition languages for XML are DTD [30] and XML Schema [31], both of which specify the possible structure of an XML document. However, it is the actual structure of a given XML document that is crucial for schema/data integration. Also, it is possible that an XML document has no referenced DTD or XML Schema. The AutoMed toolkit therefore supports a modelling language called *XML DataSource Schema* (XMLDSS) which abstracts only the structure of an XML document.

XMLDSS schemas consist of four kinds of constructs — here, we describe these somewhat informally and refer the reader to [33] for their formal specification in terms of the HDM:

**Element:** Elements,  $e$ , are identified by a scheme  $\langle\langle e \rangle\rangle$ . An XMLDSS element is represented by a node in the HDM.

**Attribute:** Attributes,  $a$ , belonging to elements,  $e$ , are identified by a scheme  $\langle\langle e, a \rangle\rangle$ . They are represented by a node in the HDM, representing the attribute, an edge between this node and the node representing  $e$ , and a cardinality constraint stating that an instance of  $e$  can have at most one instance of  $a$  associated with it, and an instance of  $a$  can be associated with one or more instances of  $e$ .

**NestList:** These are parent-child relationships between two elements  $e_p$  and  $e_c$  and are identified by a scheme  $\langle\langle i, e_p, e_c \rangle\rangle$ , where  $i$  is the position of  $e_c$  within the list of children of  $e_p$ . In the HDM, they are represented by an edge between the nodes representing  $e_p$  and  $e_c$ , and a cardinality constraint that states that each instance of  $e_p$  is associated with zero or more instances of  $e_c$ , and each instance of  $e_c$  is associated with precisely one instance of  $e_p$ <sup>8</sup>.

**PCData:** In any XMLDSS schema there is one construct  $\langle\langle PCData \rangle\rangle$ , representing all the instances of PCDATA within an XML document. To link  $\langle\langle PCData \rangle\rangle$  with an element, we treat it as an element and use the NestList construct.

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.

We refer the reader to [34] for details of the algorithm used for extracting an XMLDSS schema from an XML document. The extent of each **Element** construct  $\langle\langle e \rangle\rangle$  in the XMLDSS schema consists of all the elements with tag  $e$  in the XML document. The extent of each **Attribute** construct  $\langle\langle e, a \rangle\rangle$  consists of all pairs of elements and attributes  $x, y$  such that element  $x$  has tag  $e$  and has an attribute  $a$  with value  $y$ . The extent of each **NestList** construct  $\langle\langle i, p, c \rangle\rangle$  consists of the list of pairs of elements  $x, y$  such that element  $x$  has tag  $p$  and has a child element  $y$  with tag  $c$ .

## 2.5 Clustering for supporting multiple IDs

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 below 40%, even when the sources are nominally describing the same entities.

---

<sup>8</sup> IQL is inherently list-based, and so the ordering of children instances of  $e_c$  under parent instances of  $e_p$  is preserved within the extent of the NestList  $\langle\langle i, e_p, e_c \rangle\rangle$ .

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, 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 into sets of related entities. Having generated sets of related entities, information applicable to each set may be extracted and associated with that set.

Our current implementation of the data integration framework presented below utilises a multi-linkage hierarchical clustering method to create nested sets of entities. Each level of nesting represents an increasing degree of similarity between the entities contained in the set. There is no inherent limitation on the type of clustering or the type or types of similarity measures used to compare entities.

Nesting sets of entities allows an application built on the integrated data source to determine what is an appropriate degree of clustering for that application. 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.

For each level of clustering certain attributes of the entities may only be defined for a subset of members. If it is known that those attributes will be shared amongst similar entities then attribute information may be inferred for remaining members of the set.

### 3 Our Data Integration Framework

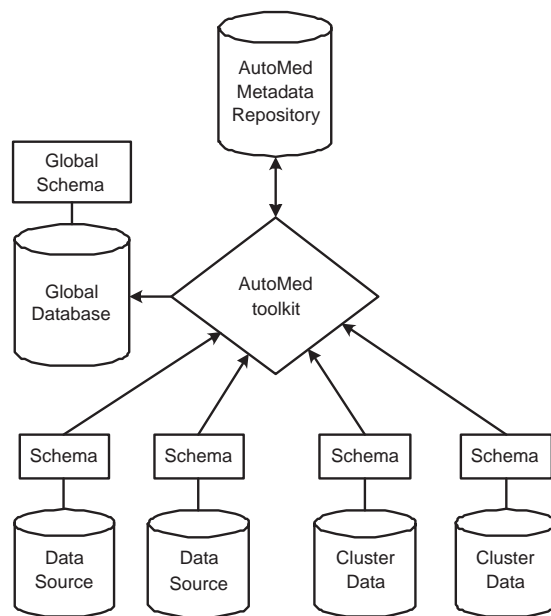
#### 3.1 Architectural Overview

The architecture of our biological data integration framework is illustrated in Figure 2. The **Global Database** can be virtual, partially materialised or completely materialised, depending on the application requirements. 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 2.5 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



**Fig. 2.** Architectural Overview of the Data Integration Framework

the latter case a schema appropriate for the data source can be generated by the appropriate AutoMed wrapper (see [33, 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. As noted in Section 2.1, this is a significant problem with biological data sources. 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, as we describe in Section 3.2 below.

### 3.2 The AutoMed Components

The integration process, which we describe in more detail in Section 4.4 below, consists of the following main steps:

1. Automatic generation of the AutoMed schemas  $LS_1, \dots, LS_n$ , corresponding to the Data Source and Cluster Data Schemas.
2. Similarly, automatic generation of the AutoMed schema  $GS$  corresponding to the Global schema.
3. Automatic translation of  $LS_1, \dots, LS_n$  and  $GS$  into the corresponding XMLDSS schemas  $X_1, \dots, X_n$  and  $GX$  (see Section 4.4).
4. Partial conformance of each schema  $X_i$  to  $GX$  by means of appropriate rename transformations, which ensure that only semantically equivalent schema constructs share the same name, and that all equivalent schema constructs do share the same name (see Section 4.4). This results in a set of new schemas  $X'_1, \dots, X'_n$ .
5. Conformance of each schema  $X'_i$  to  $GX$  by applying an automatic XMLDSS schema transformation algorithm (see Section 4.4) to each pair of schemas  $X'_i, GX$ , creating a set of new schemas  $X''_1, \dots, X''_n$ .
6. Application of any necessary data cleansing transformations on each  $X''_i$ , creating the schemas  $U_1, \dots, U_n$  (see Section 4.4; 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).
7. Automatic generation of the id transformation pathways between each pair  $U_i, U_{i+}$  (as described in Section 2.2 and illustrated in Figure 1).

In Step 3, the translation algorithm (described in Section 4.4) outputs a transformation pathway  $LS_i \rightarrow X_i$  for each  $LS_i$  that is not an XMLDSS schema (i.e. for each  $LS_i$  corresponding to a non-XML data source), and also a transformation pathway  $GS \rightarrow GX$  for the global relational schema  $GS$ . Each of these pathways contains a “growing” phrase, in which add transformations successively insert the XMLDSS constructs derived from the relational constructs, followed by a “shrinking” phrase, in which del transformations successively remove the now redundant relational constructs. For those  $LS_i$  which are XMLDSS schemas,  $X_i$  is identical to  $LS_i$  and the pathway  $LS_i \rightarrow X_i$  is empty.

In Steps 4 - 6, the pathways  $LS_1 \rightarrow X_1, \dots, LS_n \rightarrow X_n$  are extended with further primitive transformations, which are either automatically or manually generated, leading finally to the schemas  $U_1, \dots, U_n$  in Step 6. These  $U_i$  are identical to the global XMLDSS schema  $GX$  from Step 3. The pathway  $T_u$  in Figure 1 consists of reverse of the pathway  $GS \rightarrow GX$  generated in Step 3. The global schema  $GS$  can be virtual, partially materialised or completely materialised, depending on the application requirements.

In the BioMap project (see Section 4.1 below), there are relational and XML data sources to be integrated and XMLDSS has proven to be a good choice for a unifying data model. It has allowed the integration process to be automated for the most part, and to be accomplished with reasonable performance. If in the future a data source that is not relational or XML needs to be integrated, then

the only necessary extension will be an algorithm for translating the schema of the new data source to XMLDSS.

### 3.3 Virtual Integration - Supporting Global Queries

After the integration process has been completed, and the transformation network between the data source schemas  $LS_1, \dots, LS_n$  and the global schema  $GS$  has been set up, queries formulated with respect to  $GS$  can be evaluated. Such a query is submitted to AutoMed's Global Query Processor (see [19]) 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. Whenever a **delete**, **contract** or **rename** transformation is encountered, any occurrences of the removed or renamed scheme are replaced by the query supplied with the transformation. As a result, the original query is turned into a query that can be evaluated on the data sources. 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.

### 3.4 Materialised Integration

The current version of the BioMap warehouse (see Section 4.1 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 [15]).

### 3.5 Schema Evolution

In the longer term, it is possible for either a data source schema or the global schema to evolve. This evolution may be an evolution of the schema, or a change in the data model in which the schema is expressed, or both.

Consider first a schema  $S$  expressed in a modelling language  $\mathcal{M}$ . The evolution of  $S$  to  $S^{new}$ , also expressed in  $\mathcal{M}$ , can be represented as a series of primitive transformations that **rename**, **add**, **extend**, **delete** or **contract** constructs of  $\mathcal{M}$ .

Consider now a schema  $S$  expressed in a modelling language  $\mathcal{M}$  which evolves into an equivalent schema  $S^{new}$  expressed in a modelling language  $\mathcal{M}^{new}$ . This

evolution can be represented by a series of **add** steps defining the constructs of  $S^{new}$  in  $\mathcal{M}^{new}$  in terms of the constructs of  $S$  in  $\mathcal{M}$ . At this stage, the schema contains the constructs of both  $S$  and  $S^{new}$ . A series of **delete** steps can then be specified that remove the constructs of  $S$  (the queries within these **delete** transformations indicate that these are now redundant constructs since they can be derived from the new constructs).

An evolution which is both a change in the schema and in the data model can be expressed as a combination of the previous two kinds of evolution.

It is thus possible to handle schema change and data model change in a uniform way, since both are expressed as a sequence of AutoMed primitive transformations which extend the current transformation network. It is also possible to handle the addition of a new source schema or the removal of an existing source schema as a series of schema changes again expressed as a sequence of AutoMed primitive transformations.

Once the current transformation network has been extended by a new pathway  $S \rightarrow S^{new}$  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 [26, 27, 16] for details of how this can be achieved in both virtual [26, 27] and materialised [16] 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.

## 4 Application of the Framework to Gene Family Based Integration

The architecture described in Section 3 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 Project

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 [29] 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 [21], Gene3D protein sequence database [22], Gene Ontology [10], EBI Macromolecular structure database (MSD) [4] and ten other re-

sources. Gene expression data and metadata are captured and stored in a MAGE-compliant deposition database developed from the ArrayExpress database design [3], before being transformed into a denormalised search database. The MSD database is also itself a denormalised search database transformed from a normalised deposition database. Currently the MSD data source is incrementally updated using techniques developed at the EBI.

## 4.2 The data sources

Thus far, we have taken the CATH, KEGG and Gene3D as a representative but significant subset of BioMap data sources to use for evaluation of our data integration framework. CATH is a locally developed database containing a hierarchical classification of protein domain structures, which clusters proteins at four major levels, Class (C), Architecture (A), Topology (T) and Homologous superfamily (H). KEGG is the Kyoto Encyclopedia of Genes and Genomes, a resource dedicated to cataloguing information about Genes, their products, orthologs, reactions and pathways. Gene3D is a locally developed resource which aims to utilise annotate proteins from complete genomes with structural information. Gene3D also provided the basis of the clustering approach used for BioMap. This subset of data sources has been selected to contain a diverse set of topics, data structures, formatting conventions and sizes to demonstrate the viability of the proposed architecture. The resources describe structural, functional, sequence and ontological information. Each of these source schemas are stored in a relational form using Oracle. Internally each resource uses different formatting conventions so semantically identical cross-references between databases may not match in a string comparison. For example the GI code 'GI:12345' could be stored in some cases as '12345'. The Gene3D contains approximately a million entities, as does KEGG.

## 4.3 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 [8]. Our categorisation method is based on the PFScape protocol [22] which is in turn based on the TRIBE-MCL algorithm [4]. 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. For each sequence a message digest (md5) was calculated to provide a stable identifier for that sequence. This resulted in approximately 800,000 non-redundant sequences. 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 150 Archea, Prokaryotes and Eukaryotes and over 1,000 viruses were downloaded from the NCBI and ENSEMBL. For each sequence an md5 was calculated and the novel sequences were then screened by BLAST against the Gene3D sequences. Both the original and the new 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 by BLAST then a new family was created. Within the protein families multi-linkage clustering was performed based on fraction of sequence identity using cluster levels 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. Overall the comparison took approximately 5 weeks to scan the sequences with BLAST and to parse the data. Approximately 0.75 TB of compressed data was produced.

The use of sequence families to identify clusters or groups of similar or related entities allows a significant improvement in the annotation of the individual members of a group. This approach supports the integration of multiple data sets at a level of similarity appropriate to the type of data being integrated. 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.

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. The results of this process were then stored in a set of relational tables.

#### **4.4 The integration process**

We have integrated six relational data sources with the global schema. These data sources are Gene3D, KEGG\_Gene, KEGG\_Genome, KEGG\_Orthology, CATH, and also a Cluster data source derived from them as discussed above, called CLUSTER. In this case, the global schema was designed by a domain expert, although in general it is possible to automatically create a global XMLDSS schema from a set of local schemas ‘bottom-up’ (see [33] for details). The integration process consists of the steps listed in Section 3.2, and we now consider in more detail Steps 3-6 of this process.

### Translating AutoMed relational schemas to AutoMed XMLDSS schemas.

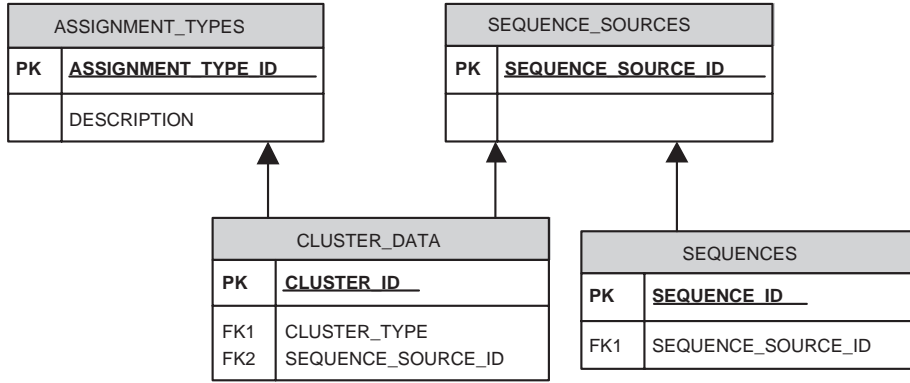
To translate a relational schema into an XMLDSS schema we first generate a graph,  $G$ , from the relational schema. There is a node in  $G$  corresponding to each table in the relational schema. There is an edge from  $R_1$  to  $R_2$  in  $G$  if there is a foreign key in  $R_2$  referencing the primary 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 the `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  $c_i$  do:
    - \* insert  $c_i$  as an `Element` construct
    - \* insert a `NestList` construct from  $t$  to  $c_i$
    - \* 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, Figure 3 illustrates a part of the schema of the CLUSTER data source (where `CLUSTER_TYPE` in `CLUSTER_DATA` references `ASSIGNMENT_TYPE_ID` in `ASSIGNMENT_TYPES`, and the rest of the foreign keys have the same name as the primary keys they reference). Figure 4 illustrates the XMLDSS schema that corresponds to this relational schema. Similarly, Figure 5 illustrates a part of the relational global schema while Figure 6 illustrates the AutoMed XMLDSS schema that corresponds to this.

**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. For this reason, after the XMLDSS schemas are produced, and before the application of the schema transformation algorithm, the necessary rename transformations must be 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



**Fig. 3.** Part of the CLUSTER relational schema

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 4:

```

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 6:

```

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

```

**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

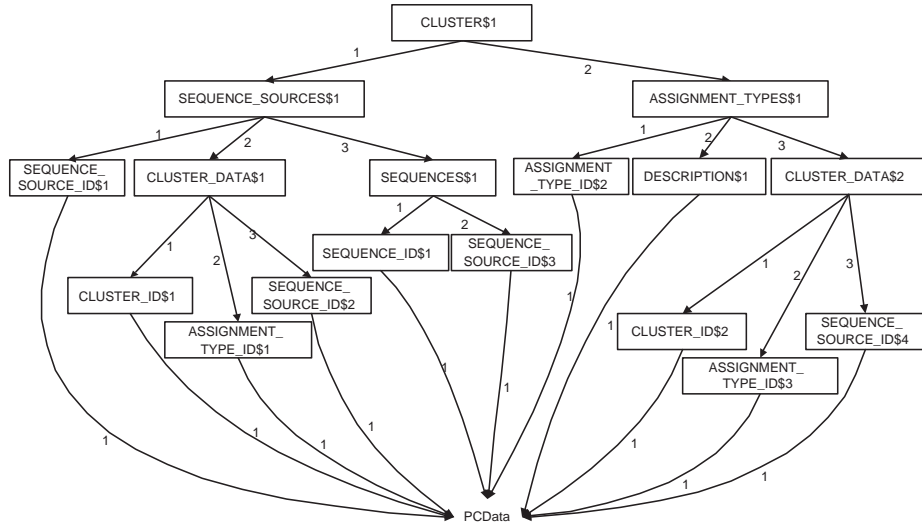


Fig. 4. Part of the CLUSTER XMLDSS schema

CLUSTER_DATA		SEQUENCES	
PK	CLUSTER_ID	PK	SEQUENCE_ID
	ASSIGNMENT_TYPE_ID ASSIGNMENT_DESCRIPTION SEQUENCE_SOURCE_ID		SEQUENCE_SOURCE_ID

Fig. 5. Part of the global relational schema

constructs in order to create the correct ordering of these constructs, resulting in a final schema  $S_T$  that is syntactically identical to the target XMLDSS schema  $T$ .

We refer the reader to [34] for a detailed description of this algorithm. The algorithm has several desirable features. First, as well as the basic insert/remove transformations, it offers an efficient move element/subtree operation, and element-to-attribute and attribute-to-element transformations. Another characteristic of the algorithm is its ability to avoid loss of data, by creating synthetic data to resolve any structural incompatibilities between  $S$  and  $T$ . To further explain this feature, suppose that in schema  $S$  Element construct  $\langle\langle B \rangle\rangle$  is a child Element construct of Element construct  $\langle\langle A \rangle\rangle$ , whereas in  $T$   $\langle\langle A \rangle\rangle$  is the child Element construct of  $\langle\langle B \rangle\rangle$ . If in the data source of  $S$  there are some instances of  $\langle\langle A \rangle\rangle$  that do not have any instances of  $\langle\langle B \rangle\rangle$  as children, then these instances of  $\langle\langle A \rangle\rangle$  would be lost. In such a case, the algorithm can create synthetic instances of  $\langle\langle B \rangle\rangle$  and  $\langle\langle A, B \rangle\rangle$  in the extent of the transformed schema  $S_T$  to avoid loss of any data from  $S$ .

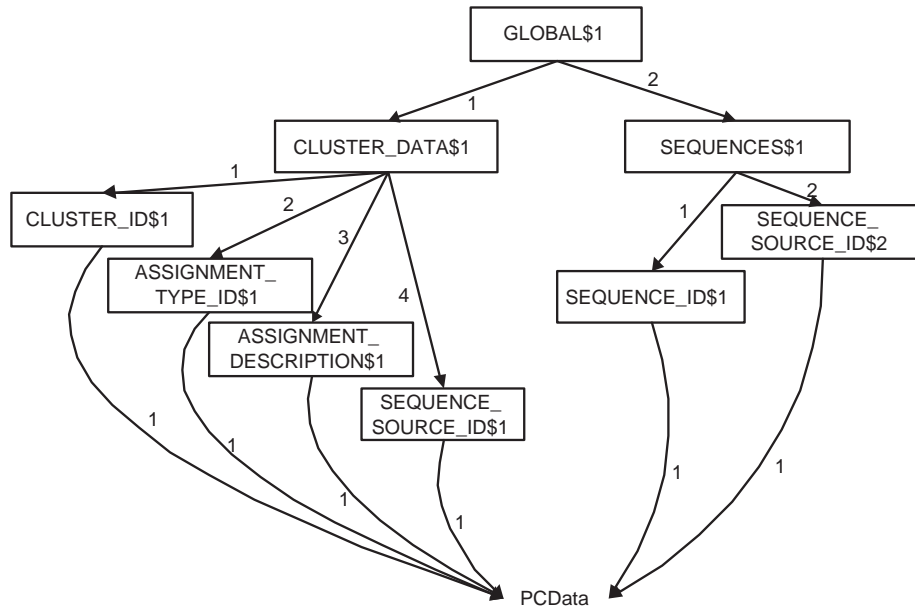


Fig. 6. Part of the global XMLDSS schema

To illustrate the algorithm, we list below a part of the transformation pathway generated to transform the XMLDSS schema in Figure 4 to the XMLDSS schema in Figure 6, after the earlier rename transformations of Section 4.4 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>>,
    [{p0,p2}|{p0,p1}<-<<GLOBAL$1,SEQUENCE_SOURCES$1>>;
     {p1,p2}<-<<SEQUENCE_SOURCES$1,CLUSTER_DATA$1>>]);
add(<<0,GLOBAL$1,SEQUENCES$1>>,
    [{p0,p2}|{p0,p1}<-<<GLOBAL$1,SEQUENCE_SOURCES$1>>;
     {p1,p2}<-<<SEQUENCE_SOURCES$1,SEQUENCES$1>>]);
extend(<<0,CLUSTER_DATA$1,ASSIGNMENT_DESCRIPTION$1>>,
    [{p1,p2}|{p0,p1}<-<<ASSIGNMENT_TYPES$1,CLUSTER_DATA$2>>;
     {p0,p2}<-<<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 Figure 4 are deleted/contracted similarly. A series of rename transformations then follows to create a contiguous ordering of edges beneath a parent element.

**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 3). 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` (see Figure 4). After the partial conformance step (Step 4 in Section 3.2), 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 3.2):

```

add(<<0,ASSIGNMENT_DESCRIPTION$1,PCData>>,
    [{x,stringUpper y} |
     {x,y}<-<<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 data cleansing 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 Future Work

This paper has presented a data integration framework for biological data sources that addresses three particular difficulties encountered in that application area. First, a clustering approach has been developed to support distributed biological data sources with inconsistent identification of biological objects. Second, our framework uses the AutoMed heterogeneous data integration toolkit to support the diversity of data models, schemas and formats which are characteristic of biological data. The particular strength of AutoMed for this application area is that it supports bi-directional, extensible transformations together with support for key data warehousing functions such as data cleansing and lineage tracing. Moreover, it enables both materialised and virtual data integration concurrently within the warehouse, which is particularly valuable for biological data sources which vary enormously in terms of storage overhead and query performance requirements. The data warehouse may be tuned accordingly which is not possible with a wholly materialised or wholly virtual approach. Finally, the use of AutoMed enables support for update of schemas within both the data sources and the warehouse by virtue of the extensibility of AutoMed transformation pathways.

The work we have described in 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 4.1. 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. These manual processes are complex, relying on the identification of changes through SQL level queries, with the required changes to the global database themselves being achieved through SQL insert and delete statements. Use of AutoMed will enable delta changes to be automatically propagated to the global database as well as allowing schema changes to be accommodated. The advantage of the AutoMed approach is even more stark should a source schema change from a relational format to XML. Without AutoMed a complete rewriting of almost all the tools used to transform that resource into the global schema would currently be required. With AutoMed, the new XML schema can be extracted and a transformation pathway generated from it to the original source schema.

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 [32], and these techniques will be applied to textual biological data sources.

A further collaborative project, ISPIDER, is aimed at developing 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 will be pursued on developing these techniques and integrating them with existing Web Service and Grid middleware components for service discovery and metadata management. The ISPIDER project will also extend the range of proteomics data sources which are integrated using the framework.

## 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. Brazma *et al.* ArrayExpress – a public repository for microarray gene expression data at the EBI. *Nucl. Acids Res.*, 31(1):68–71, 2003.
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.A. Benson *et al.* Genbank: update. *Nucl. Acids Res.*, 32 Database Issue:D23–D26, 2004.
8. D.L. Wheeler *et al.* Database resources of the National Center for Biotechnology: update. *Nucl. Acids Res.*, 32 Database Issue:D35–D40, 2004.
9. L.M. Haas *et al.* DiscoveryLink: A system for integrated access to life sciences data sources. *IBM Systems Journal*, 40(20):489–511, 2001.

10. M. Ashburner *et al.* Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genet.*, 25(1):25–29, 2000.
11. P.J. Kersey *et al.* The International Protein Index: An integrated database for proteomics experiments. *Proteomics*, 4(7):1985–1988, 2004.
12. 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.
13. S. Davidson *et al.* K2/Kleisli and GUS: Experiments in integrated access to genomic data sources. *IBM Systems Journal*, 40(20):512–531, 2001.
14. H. Fan and A. Poulouvasilis. 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.
15. H. Fan and A. Poulouvasilis. Using AutoMed metadata in data warehousing environments. In *Proc. DOLAP 2003*, pages 86–93, 2003.
16. H. Fan and A. Poulouvasilis. Schema evolution in data warehousing environments — a schema transformation-based approach. In *To appear in Proc. ER'04*, 2004.
17. M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proc. 16th National Conference on AI*, pages 67–73, 1999.
18. T. Hernandez and S. Kambhampati. Integration of biological sources: Current systems and challenges ahead. *Sigmod Record*, 33(3):51–60, 2004.
19. E. Jasper, A. Poulouvasilis, and L. Zamboulis. Processing IQL queries and migrating data in the AutoMed toolkit. Technical Report 20, AutoMed Project, 2003.
20. E. Jasper, N. Tong, P. McBrien, and A. Poulouvasilis. View generation and optimization in the AutoMed data integration framework. In *Proc. 6th Baltic Conference on Databases and Information Systems*. IOS Press, 2004.
21. 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.
22. 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.
23. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. PODS'02*, pages 233–246, 2002.
24. J. Madhavan and A.Y. Halevy. Composing mappings among data sources. In *Proc. VLDB'03*, pages 572–583, 2003.
25. P. McBrien and A. Poulouvasilis. A uniform approach to inter-model transformations. In *Proc. CAiSE'99*, pages 333–348, 1999.
26. P. McBrien and A. Poulouvasilis. Schema evolution in heterogeneous database architectures, a schema transformation approach. In *Proc. CAiSE'02*, pages 484–499, 2002.
27. P. McBrien and A. Poulouvasilis. Data integration by bi-directional schema transformation rules. In *Proc. ICDE'03*, pages 227–238, 2003.
28. OMG. Life Sciences Identifiers RFP response. <http://www.omg.org/cgi-bin/doc?lifesci/2003-12-02>, 2003.
29. 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.
30. W3C. Guide to the W3C XML Specification (“XMLspec”) DTD, Version 2.1. <http://www.w3.org/XML/1998/06/xmlspec-report-v21.htm#AEN56>, June 1998.

31. W3C. XML Schema Specification. <http://www.w3.org/XML/Schema>, May 2001.
32. D. Williams and A. Poulouvasilis. 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.
33. L. Zamboulis. XML data integration by graph restructuring. In *Proceedings of BNCOD'04*. Springer, July 2004.
34. L. Zamboulis and A. Poulouvasilis. Using AutoMed for XML data transformation and integration. In *Proceedings of DIWeb'04*. Springer, June 2004.
35. E.M. Zdobnov, R. Lopez, R. Apweiler, and T. Etzold. The EBI SRS Server - recent developments. *Bioinformatics*, 18(2):368–373, 2002.