Data Access and Integration in the ISPIDER Proteomics Grid

Lucas Zamboulis[1,2], Hao Fan[1,2,⋆], Khalid Belhajjame[3], Jennifer Siepen[3], Andrew Jones[3], Nigel Martin[1], Alexandra Poulovassilis[1], Simon Hubbard[3], Suzanne M. Embury[4], and Norman W. Paton[4]

[1] School of Computer Science and Information Systems, Birkbeck, Univ. of London
[2] Department of Biochemistry and Molecular Biology, University College London
[3] Faculty of Life Sciences, University of Manchester
[4] School of Computer Science, University of Manchester

Abstract. Grid computing has great potential for supporting the integration of complex, fast changing biological data repositories to enable distributed data analysis. One scenario where Grid computing has such potential is provided by proteomics resources which are rapidly being developed with the emergence of affordable, reliable methods to study the proteome. The protein identifications arising from these methods derive from multiple repositories which need to be integrated to enable uniform access to them. A number of technologies exist which enable these resources to be accessed in a Grid environment, but the independent development of these resources means that significant data integration challenges, such as heterogeneity and schema evolution, have to be met. This paper presents an architecture which supports the combined use of Grid data access (OGSA-DAI), Grid distributed querying (OGSA-DQP) and data integration (AutoMed) software tools to support distributed data analysis. We discuss the application of this architecture for the integration of several autonomous proteomics data resources.

1 Introduction

Grid computing technologies are becoming established which enable distributed computational and data resources to be accessed in a service-based environment. In the life sciences, these technologies offer the possibility of analysis of complex distributed post-genomic resources. To support transparent access, however, such heterogeneous resources need to be integrated rather than simply accessed in a distributed fashion. This paper presents an architecture for such integration and discusses the application of this architecture for the integration of several autonomous proteomics resources.

Proteomics is the study of the protein complement of the genome. It is a rapidly expanding group of technologies adopted by laboratories around the world as it is an essential component of any comprehensive functional genomics

⋆ Currently at International School of Software, Wuhan University, China study targeted at the elucidation of biological function. This popularity stems from the increased availability and affordability of reliable methods to study the proteome, as well as the ever growing numbers of tertiary structures and genome sequences emerging from structural genomics and sequencing projects.

The *In Silico Proteome Integrated Data Environment Resource* (ISPIDER) project[5] aims to develop an integrated platform of proteome-related resources, using existing standards from proteomics, bioinformatics and e-Science. The integration of such resources would be extremely beneficial for a number of reasons. First, having access to more data leads to more reliable analyses; for example, performing protein identifications over an integrated resource would reduce the chances of false negatives. Second, bringing together resources containing different but closely related data increases the breadth of information the biologist has access to. Furthermore, the integration of these resources, as opposed to merely providing a common interface for accessing them, enables data from a range of experiments, tissues, or different cell states to be brought together in a form which may be analysed by a biologist in spite of the widely varying coverage and underlying technology of each resource.

In this paper we present an architecture which supports the combined use of Grid data access (OGSA-DAI), Grid distributed querying (OGSA-DQP) and data integration (AutoMed) software tools, together with initial results from the integration of three distributed, autonomous proteomics resources, namely gpmDB[6], Pedro[7] and PepSeeker[8]. The emergence of databases on experimental proteomics, capturing data from experiments on protein separation and identification, is very recent and we know of no previous work that combines data access, distributed querying and data integration of multiple proteomics databases as described here.

Paper outline: Section 2 gives an overview of the OGSA-DAI, OGSA-DQP and AutoMed technologies and introduces the three proteomics resources we have integrated. Section 3 discusses the development of the global schema integrating the proteomics resources within the ISPIDER project, Section 4 presents our new architecture, Section 5 discusses related work and Section 6 gives our conclusions and directions of further work.

2 Background

2.1 OGSA-DAI and OGSA-DQP

OGSA-DAI (Open Grid Services Architecture - Data Access and Integration) is an open-source, extendable middleware product exposing data resources on Grids via web services [2]. OGSA-DAI[9] supports both relational (MySQL, DB2,

[5] See http://www.ispider.man.ac.uk
[6] See http://gpmdb.thegpm.org
[7] See http://pedrodb.man.ac.uk:8080/pedrodb
[8] See http://nwsr.smith.man.ac.uk/pepseeker
[9] See http://www.ogsadai.org.uk/

SQL Server, Oracle, PostgreSQL), XML (Xindice, plans for eXist) and text data sources. It provides a uniform request format for a number of operations on data sources, including querying/updating, data transformation (XSLT), compression (ZIP/GZIP), and data delivery (FTP/SOAP).

OGSA-DQP (Open Grid Services Architecture - Distributed Query Processor) is a service-based distributed query processor [1], offering parallelism to support efficient querying of OGSA-DAI resources available in a grid environment. OGSA-DQP[10] offers two services, the Grid Distributed Query Service (GDQS) or Coordinator, and the Query Evaluation Service (QES) or Evaluator. The Coordinator uses resource metadata and computational resource information to compile, optimise, partition and schedule distributed query execution plans over multiple execution nodes in the Grid. The distributed evaluator services execute query plans generated by the Coordinator. Each Evaluator evaluates a partition of the query execution plan assigned to it by a coordinator. A set of Evaluators participating in a query form a tree through which data flows from leaf Evaluators which interact with Grid data services, up the tree to reach its destination.

The following steps are needed for a client to set up a connection with OGSA-DQP and execute queries over OGSA-DAI resources. First, the client configures an appropriate GDQS data service resource. As a result of this process, the schemas of the resources are imported and the client is able to access one or more of the databases whose schemas have been referenced within a single query. The client then submits a Perform Document to OGSA-DQP containing an OQL [5] query. The Polar* [21] compiler parses, optimises and schedules the query. The query is partitioned, and each partition is sent to a different Evaluator. The Evaluators then interact with the OGSA-DAI resources and with each other, and send their results back to the GDQS, and, finally, the client.

2.2 AutoMed

AutoMed[11] 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. It supports a low-level *hypergraph-based data model (HDM)* and provides facilities for specifying higher-level modelling languages in terms of this HDM. 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. For any modelling language $\mathcal{M}$ specified in this way (via the API of AutoMed's Model Definitions Repository), 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.

[10] See http://www.ogsadai.org.uk/about/ogsa-dqp/
[11] See http://www.doc.ic.ac.uk/automed

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). A sequence of primitive transformations from one schema S_1 to another schema S_2 is termed a *pathway* from S_1 to S_2. All source, intermediate, and integrated schemas, and the pathways between them, are stored in AutoMed's Schemas & Transformations Repository.

Each `add` and `delete` transformation is accompanied by a query specifying the extent of the added or deleted construct in terms of the rest of the constructs in the schema. This query is expressed in a functional query language, IQL, and we will see some examples of IQL queries in Section 4.2. Also available are `extend` and `contract` primitive transformations which behave in the same way as `add` and `delete` except that they state that the extent of the new/removed construct cannot be precisely derived from the other constructs present in the schema. More specifically, each `extend` and `contract` transformation takes a pair of queries that specify a lower and an upper bound on the extent of the construct. The lower bound may be `Void` and the upper bound may be `Any`, which respectively indicate no known information about the lower or upper bound of the extent of the new construct.

The queries supplied with primitive transformations can be used to translate queries or data along a transformation pathway — we refer the reader to [15, 14] for details. The queries supplied with primitive transformations also provide the necessary information for these transformations to be automatically *reversible*, in that each `add`/`extend` transformation is reversed by a `delete`/`contract` transformation with the same arguments, while each `rename` is reversed by a `rename` with the two arguments swapped.

As discussed in [15], this means that AutoMed is a *both-as-view (BAV)* data integration system: the `add`/`extend` steps in a transformation pathway correspond to Global-As-View (GAV) rules as they incrementally define target schema constructs in terms of source schema constructs; while the `delete` and `contract` steps correspond to Local-As-View (LAV) rules since they define source schema constructs in terms of target schema constructs. An in-depth comparison of BAV with other data integration approaches can be found in [15, 14].

2.3 The Proteomics Resources

Thus far we have integrated three autonomous proteomics resources, all of which contain information on protein/peptide identification:

The Proteome Experimental Data RepOsitory (PEDRo [9]) provides access to a collection of descriptions of experimental data sets in proteomics. PEDRo was one of the first databases used for storing proteomics experimental data. It has also been used as a format for exchanging proteomics data, and in this respect has influenced the standardisation activities of the Proteomics Standards Initiative (PSI[12]).

[12] See http://psidev.sourceforge.net

The Global Proteome Machine Database (gpmDB [6]) is a publicly available database with over 2,200,000 proteins and almost 470,000 unique peptide identifications. The resource was initially designed to assist in the validation of peptide MS/MS spectra and protein coverage patterns, where patterns in previous assignments could be used to allow some measure of confidence to be assigned to new identifications. Although the gpmDB is restricted to minimal information relating to the protein/peptide identification, it provides access to a wealth of interesting and useful peptide identifications from a range of different laboratories and instruments.

PepSeeker [16] is a database developed as part of the ISPIDER project and is targeted directly at the identification stage of the proteomics pipeline. The database captures the identification allied to the peptide sequence data, coupled to the underlying ion series and as a result it is a comprehensive resource of peptide/protein identifications. The repository currently holds over 50,000 proteins and 50,000 unique peptide identifications.

3 The Proteomics Grid Application

3.1 The ISPIDER Project

Experimental proteomics is an essential component for the elucidation of protein biological functions. It involves the study of a set of proteins produced by an organism with the aim of understanding their behaviour under a variety of experimental conditions and environments. The development of new technologies for protein separation, such as 2D-SDS-PAGE (PolyAcrylamide Gel Electrophoresis), High Performance Liquid Chromatography (HPLC) and Capillary Electrophoresis, together with the availability of public accessible protein sequence databases, has enabled scientists to conduct many interesting proteomics experiments on a daily basis. Also, thanks to techniques such as Multi-Dimensional Protein Identification Technology (MudPIT), a single proteomics experiment may identify hundreds of proteins and, as a result, produce a large amount of valuable biological data.

There is a growing number of resources that offer a range of approaches for the capture, storage and dissemination of proteomic experimental data, reflecting the fact that proteomics has now come of age in the post-genomic era and is delivering large, complex datasets which are rich in information. While the existence of such databases opens up many possibilities for the proteomics community, there is still a need for a support for integrating proteomics data, and tools for constructing proteomics-specific experiments.

The aim of the ISPIDER project is to build on state-of-the-art technologies for e-science and data integration in order to provide an environment for integrating proteomics data, constructing and executing analyses over such data, and a library of proteomics-aware components that can act as building blocks for such analyses. The project is Grid-enabling existing proteomics data resources, creating new resources, producing middleware technologies for the integration of these resources — including tools for data integration, workflows and data analysis — and producing visualisation and other types of client for biologist end users.

3.2 Developing the Global Schema

One of the key questions that arose when we started the integration task, was the scope of the global schema. One choice would be a global schema targeted to answering a specific class of proteomics questions e.g. protein-specific questions. A typical query that could be issued in such a case would be *give me the list of proteins that have been identified so far by the proteomics experiments.* Opting for this choice implies a limited usage of the integrated databases. For example, the user will probably not be able to have information on the peptide masses that have been used as inputs to the identification. An alternative choice would be a global schema that is a 'union' schema, integrating the full schemas of the participant databases. Building such a schema together with specifying the mappings between its constructs and the constructs of the participant schemas may, however, turn out to be a complex and lengthy process.

The option we chose is therefore a trade-off between these two alternatives, and the global schema is a subset of the union of the participant schemas. This global schema captures enough information for answering common proteomics questions, particularly queries involving analysis of the results of the proteomics experiments. The scope of this results analysis ranges from the software used for the identification, the peptides produced by the digestion of the proteins, the protein database against which the candidate proteins are compared, to the score and description of the identified proteins. Our choice of results analysis as the scope of the global schema has also been motivated by the fact that it represents the area of overlap between the three databases being integrated, thus allowing the user to pose queries that combine and compare results analysis from the three databases.

Figure 1 gives a UML class diagram of the global schema (the PRIDE[13] data resource mentioned in the figure has not yet been integrated with the other three resources and this is an area of ongoing work). In a protein identification pipeline (a common type of proteomics experiment), a protein is identified using a mass spectrometer which determines the mass-to-charge ratio of the protein ions. The *ms_level* of *Spectrum* describes how many rounds of Mass Spectrometry (MS) have been performed, for example a common and powerful MS technique is tandem MS, in which two rounds of MS are performed. The first round of MS produces a spectra of the precursor ions, predefined selections (determined by *mz_range_start* and *mz_range_end*) of the precursor ions then undergo a second MS round to produce a number of product ion spectra. The relationship between each product spectra and its respective precursor spectra is captured by the *Precursor* association. Individual peaks in each of the precursor spectra are described by

[13] See http://www.ebi.ac.uk/pride/

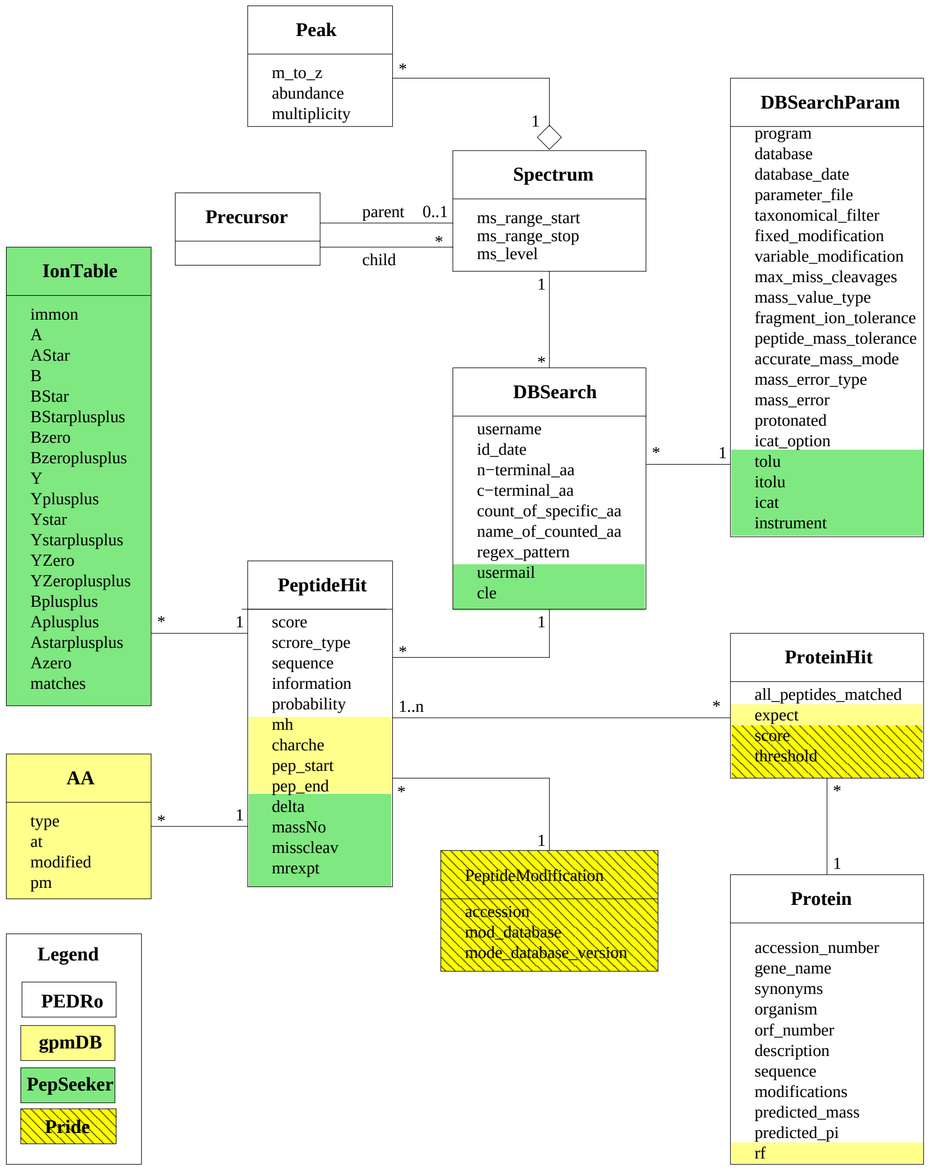

Fig. 1. The Global Schema — colouring denotes the origin of the attributes.

the mass-to-charge ratio (*m_to_z*), the peak height (*m_to_z*) and the istopic pattern around the main peak (*multiplicity*).

The next step in the protein identification pipeline then involves submission of the ion spectra (described by *Spectrum*) to an identification tool such as

Mascot[14] [18] or Imprint[15]. The classes *DBSearch* and *DBSearchParam* capture information about who did the identification, when they did it, what program they used, what database was searched, etc.

In tandem MS, several peptide hits are often generated in the identification process. A *PeptideHit* is linked to *IonTable* and *AA*. *IonTable* provides information on ions matching peptide ion fragments. *AA* describes how specific amino acid residues in a Peptide are modified (usually chemical modifications), *modified*, and indicates whether the residue was determined to be a point mutation, *pm*. *ProteinHit* represents the proteins against which all or some of the peptides have been aligned, and links to some information about the protein itself. A *Protein* is characterized by a textual description of the protein, an accession number, the predicted mass of the protein, its amino-acid sequence, any common *in vivo* modifications, the organism in which it is to be found, the open reading frame number, *orf_number*, and the reading frame, *rf*.

To build the above global schema, we adopted an incremental approach. We began with the PEDRo schema, specifically the section of its schema that captures peptide/protein identifications. This was for two reasons. First, the results analysis in the PEDRo schema has significant overlaps with the schemas of the other databases and covers most of our target global schema. Second, the PEDRo schema captures more information compared to the other databases, and thus allows for a more detailed view of the results analysis. For example, in PEDRo, the protein is characterized by the accession number, the synonyms, the organism that was the source of the protein and the sequence of the protein, in addition to other information. In contrast, a protein in PepSeeker, for instance, is simply described by its accession number and name.

Given this initial global schema, we then derived the correspondences between the classes and attributes of gpmDB and PepSeeker with this schema. The limited schema documentation and sometimes cryptic attribute naming of those resources meant interviews with the database providers were needed to identify precisely the meaning of every attribute in the schemas.

The global schema was then incrementally expanded by additional classes and attributes that were captured in those databases and not already in the global schema. This mainly consisted of adding the information about the ions associated to the peptides and the modifications they undergo. For example, from PepSeeker, we added the entity *IonTable* which provides information on the ions matching peptide ion fragments. The schemas of gpmDB and PepSeeker are relatively disjoint, with respect to the set of fields that have been added to the global schema, with few exceptions such as the attributes *pep_start* and *pep_end* of the class *PeptideHit* which exist in both PepSeeker and gpmDB schemas.

To identify the instances of the global schema entities, we chose to use life science identifiers *LSID*s[16]. *LSID* is a Life Sciences Research Uniform Resource

[14] See http://www.matrixscience.com/search_form_select.html
[15] Imprint is an in-house software tool for Peptide Mass Fingerprinting (PMF), which involves only a single round of MS
[16] See http://www.omg.org/technology/documents/formal/life_sciences.htm

Name (URN) specification which provides a standardized naming schema for biological entities in the life sciences domain. The three databases use integers to identify their entity instances, and the usage of LSIDs in the (virtual) global database allowed us to overcome the problem of identifier conflict. For example, the LSID *URN:LSID:ispider.man.ac.uk:pedro.protein:99* refers to the protein identified by the number *99* in the Pedro database, where *ispider.man.ac.uk* denotes the authority that issued the LSID[17].

4 System Architecture

While OGSA-DAI supports access of data resources in a Grid and OGSA-DQP supports distributed querying of such resources and location transparency, these technologies do not support schema transformation and schema integration. Thus, if applications require heterogeneous Grid-based data to be transformed and integrated, the onus is on the application to encode the necessary transformation/integration logic. This may impact on the robustness and maintainability of applications, and hence the use of data integration middleware that abstracts out this functionality from applications is advantageous because it enables applications to access resources as one virtual integrated resource, notwithstanding the varying formats and data models used by those autonomous resources. To our knowledge, there is currently no such Grid-enabled middleware, and hence our decision to combine OGSA-DAI/DQP with AutoMed into an architecture that enables both transformation and integration of Grid-based data and distributed query processing over the Grid resources. The main advantage of using AutoMed rather than a LAV or GAV-based data integration system is that it readily supports the evolution of both source and integrated schemas by allowing transformation pathways to be extended — this means that the entire integration process does not have to be repeated, and the schemas and pathways can instead be 'repaired'.

Figure 2 illustrates the architecture we have developed. Data sources are exposed using OGSA-DAI grid services. The AutoMed-DAI wrapper imports schema information from any data source, via OGSA-DAI, into the AutoMed Metadata Repository. Thereafter, AutoMed's schema transformation/integration functionality can be used to create one or more virtual global schemas, together with the transformation pathways between these and the AutoMed representations of the data source schemas. Queries posed on a virtual global schema can be submitted to AutoMed's Query Processor, and this interacts with OGSA-DQP via an AutoMed-DQP wrapper to evaluate these queries. OGSA-DQP itself interacts with the data sources via the OGSA-DAI services.

In the remainder of this section we present the major components of this architecture in greater detail: the mechanisms for enabling data access and integration; how queries posed on a virtual global schema are processed by AutoMed's Query Processor and OGSA-DQP; and the AutoMed-DQP wrapper.

[17] Note the LSID key attributes are not listed in the UML class diagram in Figure 1.

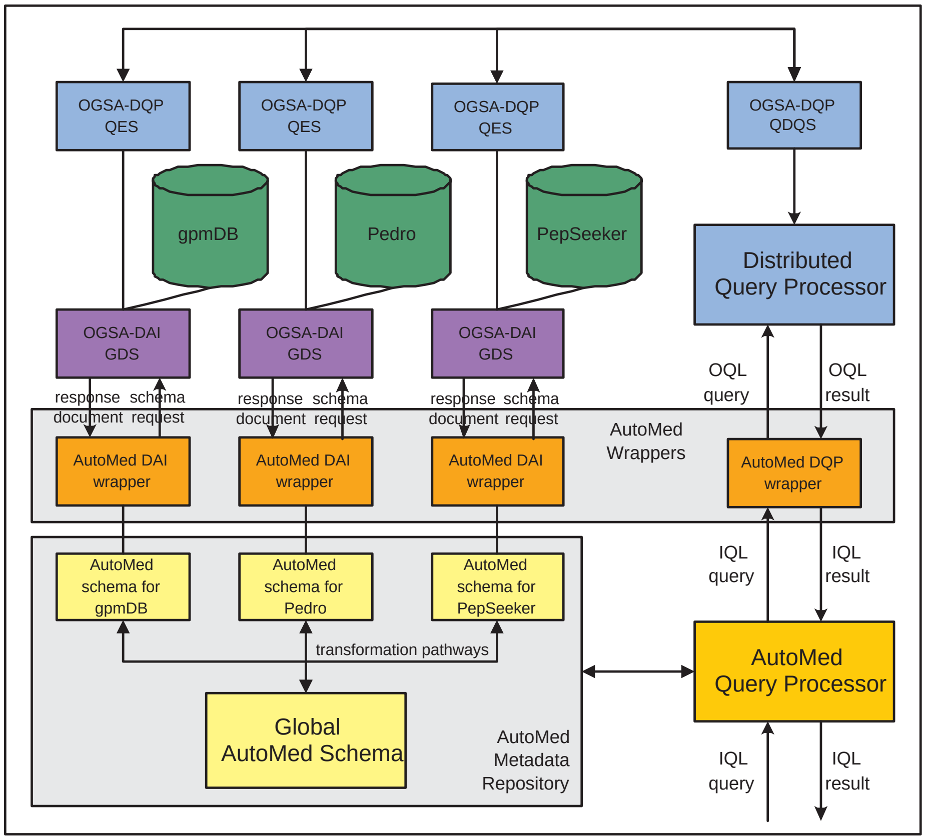

Fig. 2. The AutoMed, OGSA-DAI and OGSA-DQP architecture.

4.1 Data Access and Integration

We assume that each data source is made accessible as a grid data resource using OGSA-DAI's Grid Data Service (GDS). To 'import' a data source schema into AutoMed, we have developed the *AutoMed-DAI Wrapper*. This sends a schema request document to the GDS, which returns an XML Response Document containing the schema metadata of the data source. The AutoMed-DAI wrapper uses this information to create the corresponding AutoMed schema in the AutoMed Metada Repository.

These AutoMed data source schemas can now be incrementally transformed and integrated into one or more global virtual schemas, using the API of AutoMed's Schemas & Transformations Repository to issue transformation steps and to create intermediate and final virtual schemas within this repository — we give some examples of transformations below.

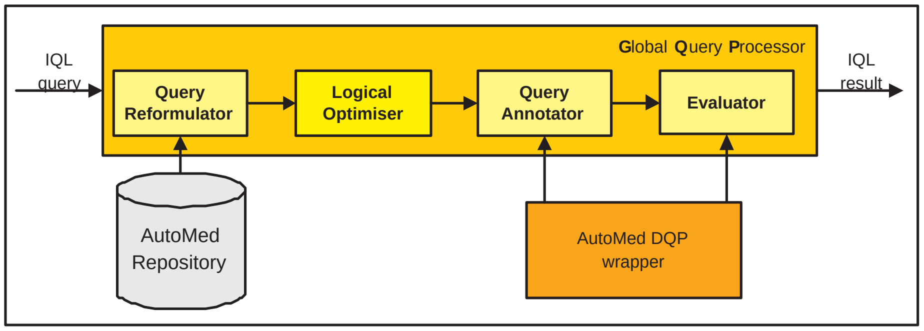

Fig. 3. The AutoMed Query Processor.

4.2 Query Processing

After the integration of the data sources, the user is able to submit to AutoMed's Query Processor (AQP) a query, Q, to be evaluated with respect to a virtual global schema. Q is expressed in AutoMed's own query language, IQL[18].

For example, the following query, Q^1, retrieves all identifications for the protein with accession number `ENSP00000339074`. This query would allow biologists studying this protein to find out more about the kinds of environments in which it has been seen by other scientists. Here, `<<Protein,accession_number>>` denotes the projection of the Protein (virtual) relation onto its primary key attribute (LSID) plus its `accession_number` attribute:

```
[lsid|{lsid,an}<-<<Protein,accession_number>>;an='ENSP00000339074']
```

As a second example, the following query, Q^2, retrieves all protein identifications that match a given peptide. Such a query would allow a scientist working with a protein sequence to ask whether peptide `ATLITFLCDR` has been seen before in other proteomics experiments:

```
[{an,lsid3}|{lsid1,seq}<-<<PeptideHit,sequence>>; seq = 'ATLITFLCDR';
            {lsid2,pr}<-<<ProteinHit,protein>>;
            {lsid3,an}<-<<Protein,accession_number>>; pr = lsid3;
            {pepID,protID}<-<<PeptideHitToProteinHit_mm>>;
            lsid1 = pepID; lsid2 = protID]
```

Figure 3 shows the major components of the AQP and we now consider each of them in turn. Since the initial query, Q, is expressed over a global schema, it references only global schema constructs and needs to be transformed into a query expressed over the data source schemas before it can be evaluated. This is accomplished by the *Query Reformulator* component of the AQP which traverses

[18] IQL is a comprehensions-based language and we refer the reader to [12] for details of its syntax, semantics and implementation. Such languages subsume query languages such as SQL-92 and OQL in expressiveness [4].

the schema transformation pathways from the global schema down to the data source AutoMed schemas, and uses the query within each transformation step to incrementally reformulate Q until finally an equivalent query Q_{ref} results which references only schema constructs within the data source schemas.

For example, the following transformation steps within the transformation pathways integrating respectively the PepSeeker, PEDRo and gpmDB schemas are of relevance to reformulating Q^1. Here, `id2lsid` is an IQL function that generates the global LSID identifiers[19]. In the first step, the relation `Protein` of the global schema is populated from the `proteinhit` relation of PepSeeker; since the latter may contain multiple occurrences of any given protein, the IQL function `distinct` is used to remove duplicates:

```
 add(<<Protein,accession_num>>,
     [{id2lsid [‘pepseeker.proteinhit:’,toString d],x}|
            {d,x}<-(distinct [{k,x}|{k,x}<-<<proteinhit,ProteinID>>])])
...
 add(<<Protein,accession_num>>,[{id2lsid [‘pedro.protein:’,toString d],x}|
                                         {d,x}<-<<protein,accession_num>>])
...
 add(<<Protein,accession_num>>,
       [{id2lsid [‘gpmdb.proseq:’,toString d],x}|{d,x}<-<<proseq,label>>])
```

Q^1 is reformulated to Q^1_{ref} below using the queries appearing within the above transformation steps:

```
[lsid|{lsid,an}<-([{id2lsid [‘pepseeker.proteinhit:’,toString d],x}|
             {d,x}<-(distinct [{k,x}|{k,x}<-<<proteinhit,ProteinID>>])]
   ++ [{id2lsid [‘pedro.protein:’,toString d],x}|
                                        {d,x}<-<<protein,accession_num>>]
   ++ [{id2lsid [‘gpmdb.proseq:’,toString d],x}|{d,x}<-<<proseq,label>>]);
                         an = ‘ENSP00000339074’]
```

Q^2 makes use of the many-to-many relationship between the `ProteinHit` and `PeptideHit` relations of the global schema for reformulation. The following transformation steps are of relevance to answering Q^2 with respect to the PEDRo schema (we do not list the transformations relevant to the other schemas or the reformulated query Q^2_{ref} itself, due to space limitations)[20]:

```
 add(<<PeptideHit,sequence>>,[{id2lsid [‘pedro.peptidehit:’,toString d],
                                   x}|{{d,e},x}<-<<peptidehit,sequence>>])
 ...
 add(<<ProteinHit,protein>>,[{id2lsid [‘pedro.proteinhit:’,toString d],
                              id2lsid [‘pedro.protein:’,x]}|
                                {d,x}<-<<proteinhit,protein>>])
```

[19] It takes as input two string arguments, concatenates them, and prefixes the result by ‘URN:LSID:ispider.man.ac.uk:’

[20] In the first step, `peptidehit` has a composite key whose first attribute is used to generate the LSID, the second attribute being a foreign key to another table.

```
...
add(<<Protein,accession_number>>,[{id2lsid [‘pedro.protein:’,toString d],
                                   x}|{d,x}<-<<protein,accession_num>>])
...
add(<<PeptideHitToProteinHit_mm>>,
       [{k1,k2}|{k1,x}<-[{id2lsid [‘pedro.peptidehit:’,toString d],x}|
                                    {{d,e},x}<-<<peptidehit,db_search>>];
                {k2,y}<-[{id2lsid [‘pedro.proteinhit:’,toString d],x}|
                                        {d,x}<-<<proteinhit,db_search>>];
                x = y])
```

A reformulated query, Q_{ref}, is next processed by the *Logical Optimiser* component which simplifies Q_{ref} by applying a number of algebraic optimisations. In our context here, one goal of this component is to simplify Q_{ref} is to create the largest possible subqueries that can be pushed down to OGSA-DQP for evaluation, so as to make maximum usage of the data sources' own query capabilities and minimise the resource consumption of AutoMed's Evaluator.

The optimised query, Q_{opt}, is still expressed in IQL and needs to be translated into OQL, the query language supported by OGSA-DQP. We have developed an *AutoMed-DQP Wrapper*, see below, for translating (a subset of) IQL into OQL.

The *Query Annotator* component interacts with the AutoMed-DQP wrapper to identify maximal subqueries translatable by that wrapper and to instantiate wrapper objects within Q_{opt}. The resulting query, Q_{annot}, is finally sent to AutoMed's Evaluator for evaluation. This makes calls to OGSA-DQP to compute the results of the subqueries specified by the Query Annotator, and undertakes any further necessary post-processing of these results.

4.3 The AutoMed-DQP wrapper

The AutoMed-DQP wrapper undertakes two tasks. First, it needs to inform the AutoMed Query Processor of the subset of IQL it is capable of translating into OQL. As with all other AutoMed wrappers, we have developed a BNF grammar specification from which a parser for the relevant subset of IQL is automatically generated The AutoMed-DQP wrapper translates IQL comprehensions with one level of nesting (in accordance with the OQL queries supported by OGSA-DQP).

The AutoMed-DQP wrapper is also responsible for making interactions with OGSA-DQP transparent to the remainder of the AutoMed infrastructure. On receiving an IQL query, the wrapper first translates it into the equivalent OQL query. The OQL query is then sent to OGSA-DQP for evaluation. The reply from OGSA-DQP is in the form of an XML Response Document containing the query results. The AutoMed-DQP wrapper translates this document into the IQL type system, and returns the result to AutoMed's Evaluator component.

5 Related Work

The importance of data integration in the life sciences has resulted in diverse technical approaches being followed. In many of these there is little support provided by the infrastructure for resolving schematic heterogeneities. For example, workflow systems enable requests to be formed that both access data resources and invoke analyses on the values retrieved (e.g. [3, 17]). However, many bioinformatics web services take and return formatted strings that require custom transformation operations to be developed for converting data between formats. Perhaps the most widely used data integration system in bioinformatics is SRS [22]. However, like workflow systems, it principally supports storage and access to entries from data resources that started out as formatted textual documents, and the principal mode of access involves navigation between these documents, rather than querying an integrated schema. As such, both of the above approaches make visible the sources from which data is derived and preserve at least some aspects of the source data format.

In approaches building on distributed database technology, there is a tendency to construct views over the underlying data resources, thus hiding schematic heterogeneities from users. In distributed query processing systems that have been designed for or used in bioinformatics, such as DiscoveryLink [11] or Kleisli [7], existing databases or file-based resources are wrapped, and views can be constructed over the wrapped sources using the GAV approach. As such, declarative techniques can be used to provide a more uniform representation of the data in a domain, although with the maintenance challenges widely associated with GAV. There have also been attempts to support querying over domain models expressed as biological ontologies, as in Tambis [10], but again the global schema either directly reflects the structure of the underlying resources or defines the global model using GAV.

Where data is to be subject to intensive integrated analysis, the warehousing approach has also been popular in bioinformatics (e.g. [8, 20]). However, the population and maintenance of a centralised warehouse is often laborious, due to inconsistencies between different data sources, naming schemes, etc. However, where data is obtained from databases, these can use queries to populate the warehouse model, as in GAV, or make use of technologies such as AutoMed, as in BioMap [13].

In proteomics, although there are many resources that integrate data about proteins (e.g. [19]), the emergence of databases on experimental proteomics, capturing data from experiments on protein separation and identification, is very recent. As such, we know of no previous work that seeks to support access to multiple proteomics databases, as described here. Furthermore, as the schemas of the databases to be integrated overlap significantly, fine-grained resolution of schema conflicts is crucial to the provision of an effective integration strategy. In essence, in the approach described in this paper, OGSA-DQP provides a query-oriented middleware analogous to that provided by DiscoveryLink or Kleisli, and AutoMed is used to resolve the heterogeneities in the schemas of the sources. We are not aware of a similar approach elsewhere in the life sciences.

6 Conclusions

We have presented an architecture combining Grid data querying (OGSA-DAI/DQP) and data integration (AutoMed) software tools which enables distributed query processing together with the resolution of semantic heterogeneity over autonomous data resources. We have presented results within the ISPIDER project of integrating autonomous proteomics resources reflecting varying formats and data models. From a biology viewpoint, the final ISPIDER platform will provide researchers with more information than any of the resources alone, so allowing them to perform analyses that were previously prohibitively difficult or impossible. This integration process both builds on and provides impetus to the development of data standards in the proteomics and proteomics-related domains.

Additional global schemas may be created as resources holding information relevant to, but disjoint from, the initial global schema are integrated within the ISPIDER platform. To enable querying across such schemas, a global 'super-schema' could then be created. This methodology exemplifies the flexibility and scalability of AutoMed's transformation-based approach which also provides the basis for materialised as well as virtual data integration and tracking data provenance. These facilities too are being pursued within the ISPIDER project.

Beyond data integration, the ISPIDER data sources offer a number of web services to the outside world, performing tasks ranging from simple data retrieval, to significantly more complex operations. We are using Taverna[21], part of the ^{my}Grid[22] middleware, to enable users to construct complex analysis workflows from the available web services. We are currently investigating the interoperation of AutoMed with Taverna for integrating heterogeneous web services.

We are currently evaluating our system in terms of query processing. We are considering extensions to the *LogicalOptimiser* component of the AQP as well as extensions to the subset of OQL supported by OGSA-DQP; this will enable the translation of larger IQL queries into OQL, offering a notable performance boost.

Acknowledgements. The work presented in this paper was funded by a grant from the BBSRC. We are also grateful to Steven Lynden for his help with the OGSA-DQP system, Thomas McLaughlin and Julian Selley for their help in defining the correspondences between the database schemas, and Robert Stevens and Carole Goble for their comments and advice.

References

1. M. N. Alpdemir, A. Mukherjee, N.W. Paton, P.Watson, A. A. Fernandes, A. Gounaris, and J. Smith. Service-based distributed querying on the Grid. In *Proc. of the 1st Int. Conf. on Service Oriented Computing*, pages 467–482, 2003.
2. M. Antonioletti et al. The design and implementation of grid database services in OGSA-DAI. *Concurrency - Practice and Experience*, 17(2-4):357–376, 2005.

[21] See http://taverna.sourceforge.net
[22] See http://www.mygrid.org.uk

3. S. Bowers and B. Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In *DILS*, pages 1–16. Springer, 2004.
4. P. Buneman, L. Libkin, D. Suciu, V. Tannen, and L. Wong. Comprehension syntax. *SIGMOD Record*, 23(1):87–96, 1994.
5. R. G. G. Cattell and D. K. Barry. *The Object Database Standard: ODMG 3.0*. Morgan Kaufmann, 2000.
6. R. Craig, J. P. Cortens, and R. C. Beavis. Open source system for analyzing, validating, and storing protein identification data. *Journal of Proteome Research*, 3(6), 2004.
7. S.B. Davidson, C. Overton, V. Tannen, and L. Wong. BioKleisli: A Digital Library for Biomedical Researchers. *Journal of Digital Libraries*, 1(1):36–53, Nov 1997.
8. S. Durinck, Y. Moreau, A. Kasprzyk, S. Davis, B. De Moor, A. Brazma, and W. Huber. Biomart and bioconductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics*, 21(16):3439–3440, 2005.
9. K. Garwood et al. Pedro: A database for storing, searching and disseminating experimental proteomics data. *BMC Genomics*, 5, 2004.
10. C. A. Goble, R. Stevens, G. Ng, S. Bechhofer, N. W. Paton, P. G. Baker, M. Peim, and A. Brass. Transparent access to multiple bioinformatics information sources. *IBM Systems Journal*, 40(2):532–551, 2001.
11. L. M. Haas, P. M. Schwarz, P. Kodali, E. Kotlar, J. E. Rice, and W. C. Swope. Discoverylink: A system for integrated access to life sciences data sources. *IBM Systems Journal*, 40(2):489–511, 2001.
12. E. Jasper, A. Poulovassilis, and L. Zamboulis. Processing IQL queries and migrating data in the AutoMed toolkit. AutoMed Tech. Rep. 20, June 2003.
13. M. Maibaum, L. Zamboulis, G. Rimon, N. Martin, and A. Poulovassilis. Cluster based integration of heterogeneous biological databases using the AutoMed toolkit. In *Proc. Data Integration for the Life Sciences 2005 (DILS'05)*, pages 191–207.
14. P. McBrien and A.Poulovassilis. Defining peer-to-peer data integration using both as view rules. In *Proc. Workshop on Databases, Information Systems and Peer-to-Peer Computing (at VLDB'03), Berlin*, 2003.
15. P. McBrien and A. Poulovassilis. Data integration by bi-directional schema transformation rules. In *Proc. ICDE'03*, pages 227–238, 2003.
16. T. McLaughlin, J. A. Siepen, J. Selley, J. A. Lynch, K. W. Lau, H. Yin, S. J. Gaskell, and S. J. Hubbard. Pepseeker: a database of proteome peptide identifications for investigating fragmentation patterns. *Nucleic Acids Research*, 34, 2006.
17. T. M. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, R. M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.
18. D.N. Perkins, D.J. Pappin, D.M. Creasy, and J.S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20(18), 1999.
19. M. Pruess, P. Kersey, and R. Apweiler. The integr8 project - a resource for genomic and proteomic data. *In Silico Biology*, 5, 2004.
20. S.P. Shah, Y. Huang, Y. Xu, M.M.S. Yuen, J. Ling, and B.F.F. Ouellette. Atlas – a data warehouse for integrative bioinformatics. *BMC Bioinformatics*, 6(81), 2005.
21. J. Smith, A. Gounaris, P. Watson, N. W. Paton, A. A. A. Fernandes, and R. Sakellariou. Distributed query processing on the Grid. In *Proc. Grid Computing*, pages 279–290, 2002.
22. E. M. Zdobnov, R. Lopez, R. Apweiler, and T. Etzold. The EBI SRS server-recent developments. *Bioinformatics*, 18(2):368–373, 2002.