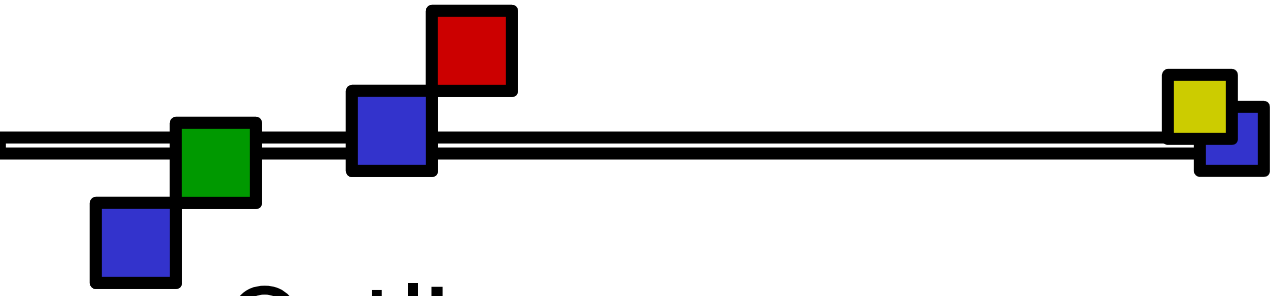
Cluster based Integration of Heterogeneous Biological Databases using the AutoMed toolkit

Michael Maibaum
Lucas Zamboulis
Galia Rimon
Christine Orengo
Nigel Martin
Alexandra Poulovassilis

Department of Biochemistry and Molecular Biology
University College London
School of Computer Science and Information Systems
Birkbeck College, University of London

DILS'05 JULY 22 2005



Outline

- The AutoMed toolkit
  - Architecture and features
- The ID problem
  - Clustering approach adopted
- Data integration framework
- Application to the BioMap warehouse
- The integration process
- Ongoing work

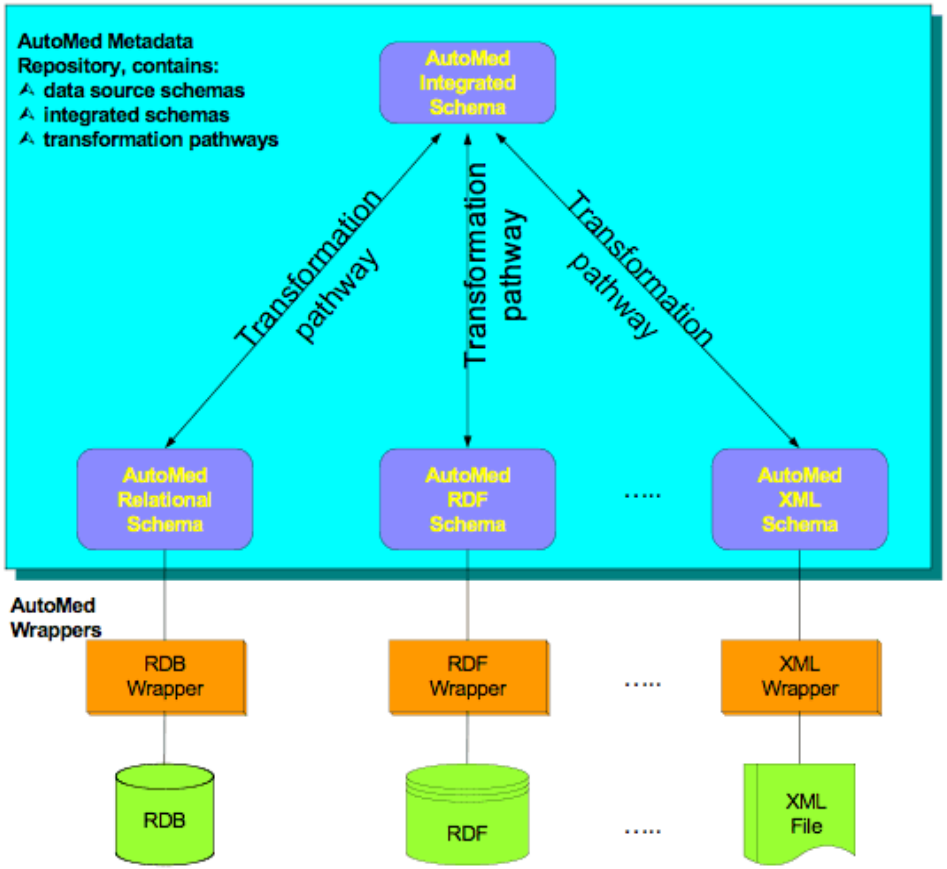
AutoMed

AutoMed is a heterogeneous data transformation and integration system

AutoMed supports

- virtual, materialised and hybrid data integration across
- multiple models

The AutoMed System



The AutoMed toolkit implements the BAV data integration approach

AutoMed repository:

- Model Definitions Repository (MDR)
- Schema Transformation Repository (STR)

AutoMed query language: IQL

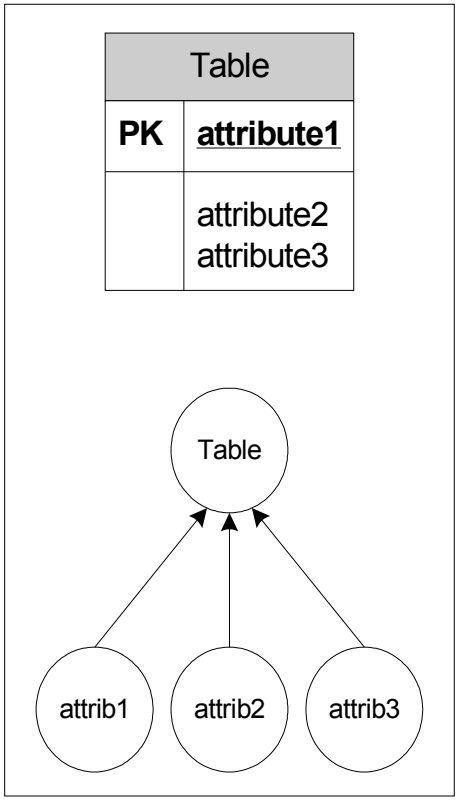
- Higher-level query languages are translated to IQL
- IQL is translated to the query languages of the datasources

AutoMed low-level model

AutoMed supports a low-level Hypergraph Data Model (HDM)

- Constructs are nodes, edges & constraints
- It avoids the semantic mismatches that may occur between constructs of higher-level modelling languages

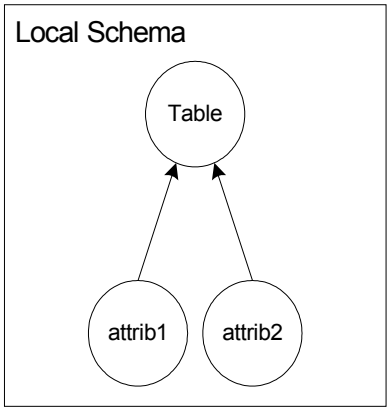
Higher-level modelling languages are specified in terms of the HDM

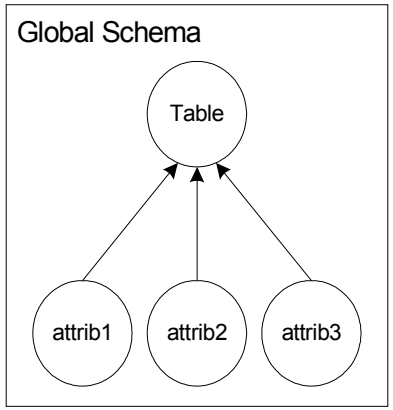


AutoMed transformations

Primitive schema transformations are applied to schema constructs

- add/delete
- rename
- extend/contract





Supply transformations with queries

- add(⟨⟨table,attrib3⟩⟩, q), where q: [{t,(a_1+a_2)}|{t,a_1} ⟨⟨table,attrib1⟩⟩;{t,a_2} ⟨⟨table,attrib2⟩⟩]
- extend(⟨⟨table,attrib3⟩⟩, q_1,q_2)

Example (1/2)

S_g	student(id,name,left#,degree) monitors(*sno*,*id*) staff(sno,sname,dept#)
S_1	ug(id,name,left#,degree,*sno*) tutor(sno,sname)
S_2	phd(id,name,left#,title) supervises(*sno*,*id*) supervisor(sno,sname,dept)

S_1 S_g:

add(⟨⟨monitors⟩⟩ ,q_1)

add(⟨⟨monitors,sno⟩⟩,q_2)

add(⟨⟨monitors,id⟩⟩,q_3)

add(⟨⟨tutor,dept#⟩⟩,q_4)

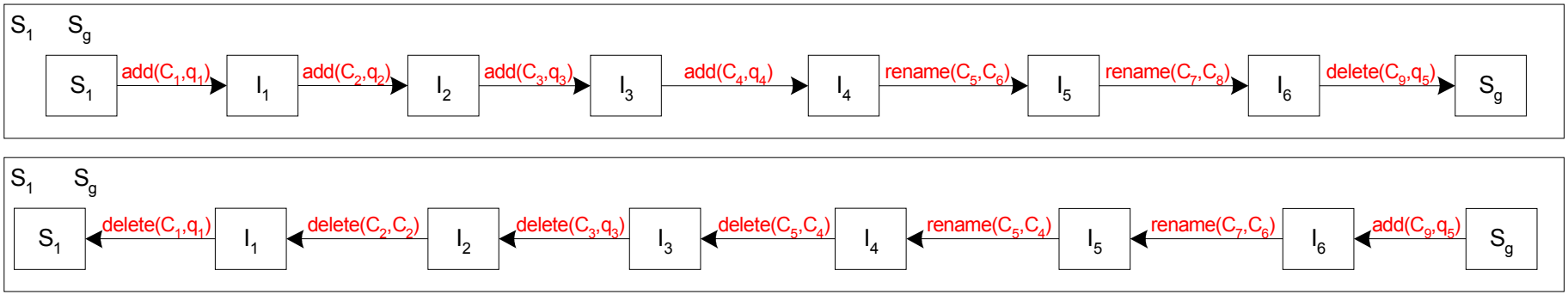
rename(⟨⟨ug⟩⟩,⟨⟨student⟩⟩)

rename(⟨⟨tutor,⟨⟨staff⟩⟩)

delete(⟨⟨student,sno⟩⟩,q_5)

S_2 S_g: can be derived similarly

Example (2/2)



Automatically derivable reverse transformations

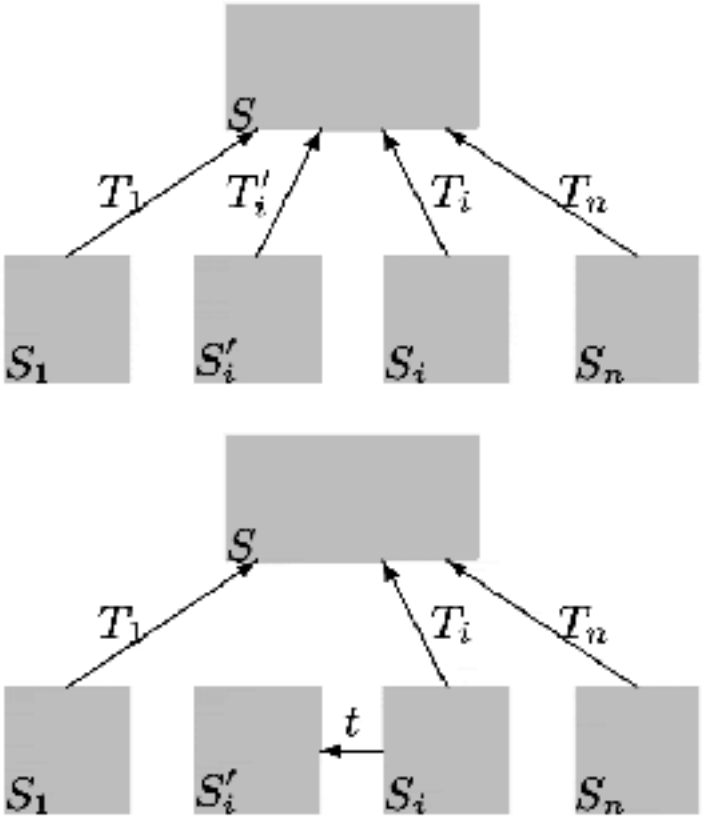
- add(C,q)/extend(C,q_1,q_2) : delete(C,q)/contract(C,q_1,q_2)
- delete/contract : add/extend
- rename(C_1,C_2) : rename(C_2,C_1)

AutoMed's transformation-based approach

- Each primitive transformation has an automatically derivable reverse transformation
- The queries within transformations allow pathways to be used for
  - materialising and incrementally maintaining a materialised global database (and any derived database) in the face of inserts/updates/deletes to data sources
  - lineage tracing
- Pathways can be used to express schema evolution in the face of
  - data source or global database schema changes
  - arising from schema or data model changes or both

Local Schema Evolution Example

Define the evolution of the global or local schema as a schema transformation pathway from the old to the new schema



Clustering to support multiple Ids

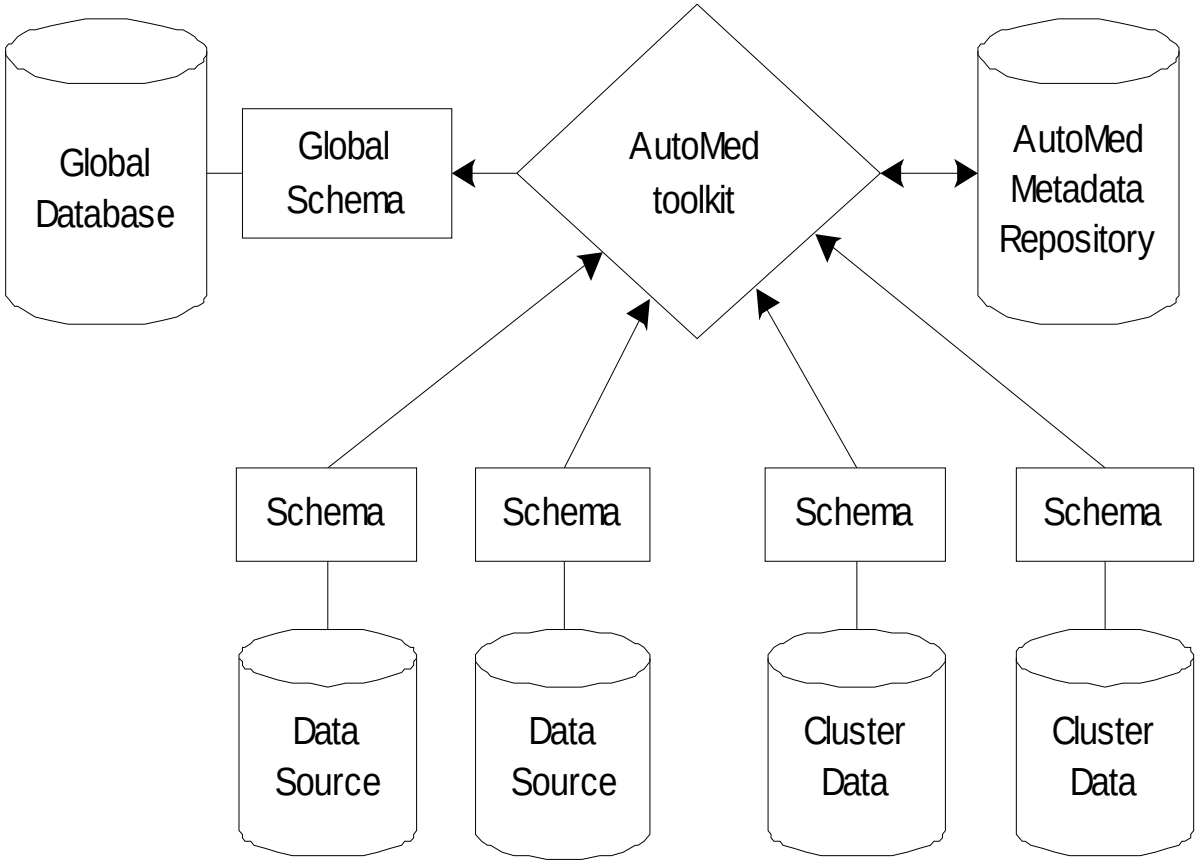
AutoMed supports data transformations, but provides no mechanisms for supporting the equivalence of inconsistent identifiers

Data-based entity clustering provides a general approach to

- integrate any set of logically related entities given an appropriate similarity measure, and hence
- support multiple identifiers

We have used this approach to organise entities hierarchically into nested sets

Data Integration Framework



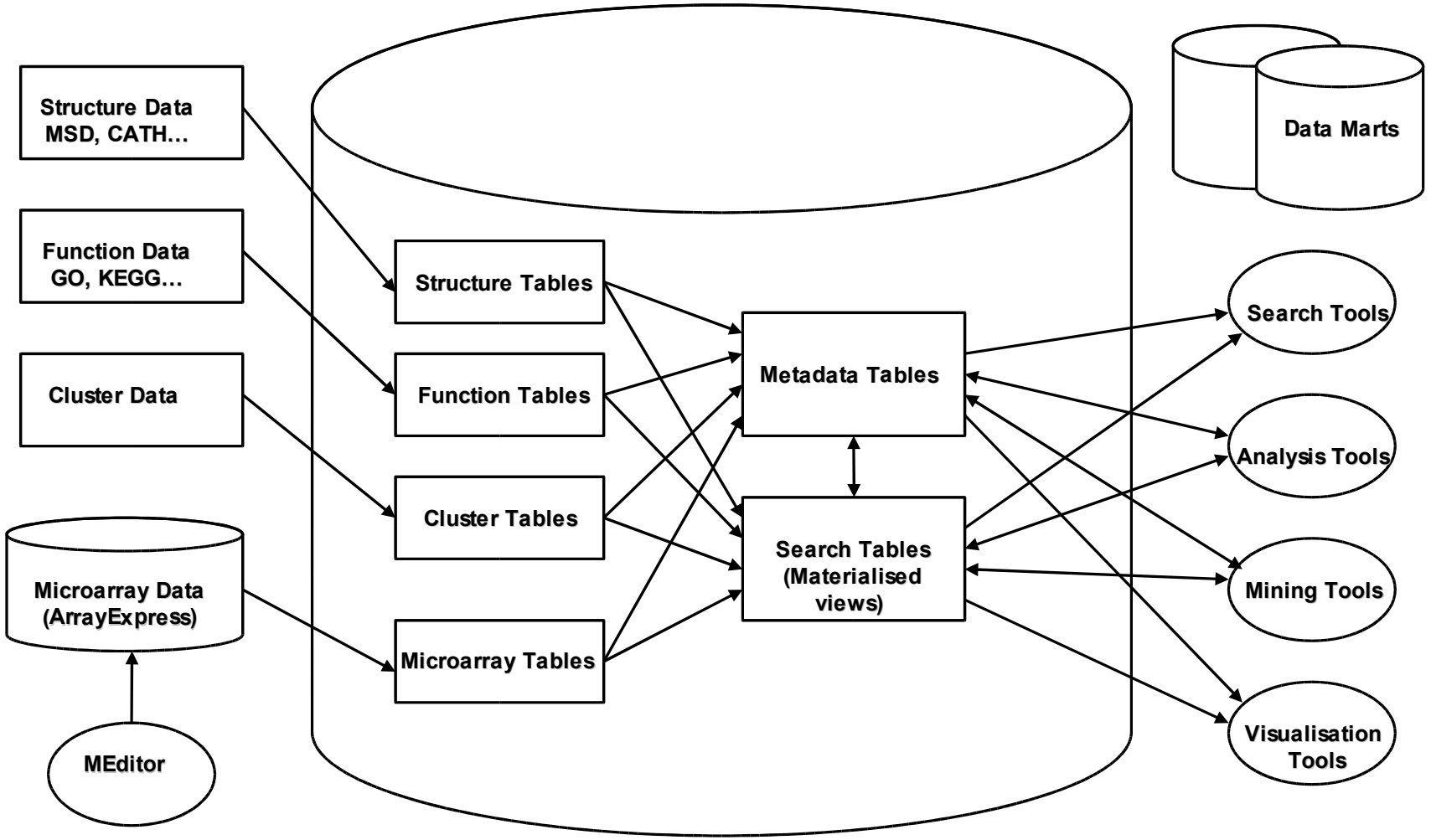
Application to a data warehouse - BioMap

A data warehouse integrating

- microarray gene expression data
- structural data including
  - data from the Macromolecular Structure Database (MSD) from the European Bioinformatics Institute (EBI)
  - CATH structural classification data
- functional data including
  - Gene Ontology
  - KEGG (Gene, Orthology, Genome, Pathway…)

The warehouse will support analysis and mining of gene expression data incorporating the domain knowledge represented within the warehouse

BioMap architecture



Clustering

Gene3D sequence families derived from > 120 complete genomes (NCBI & ENSEMBL) using all vs all BLAST

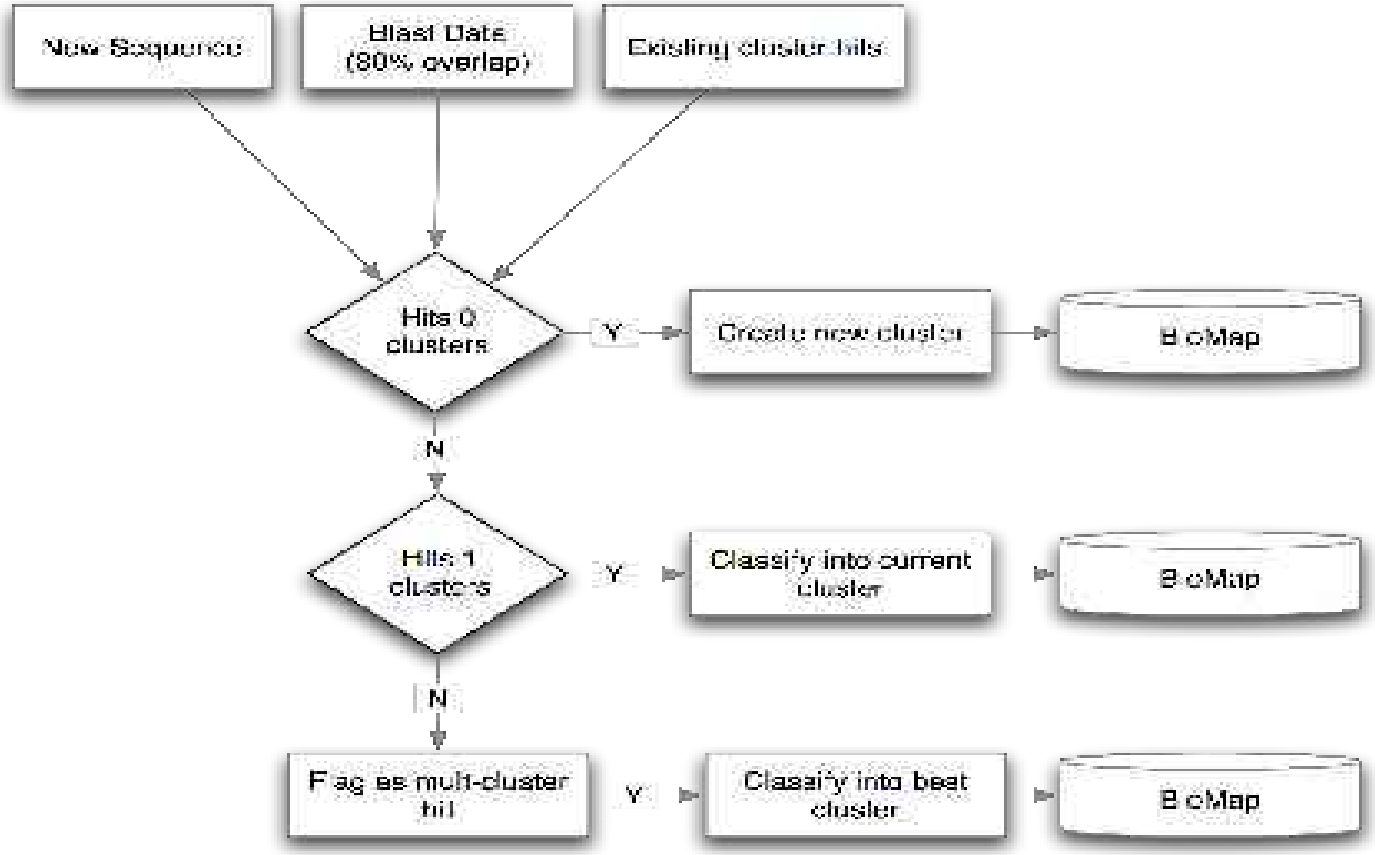
A further > 200 complete genomes downloaded (EBI)

All vs all BLAST performed for existing Gene3D and new sequences. Results filtered with 80% overlap cut off to only retain whole chain matches

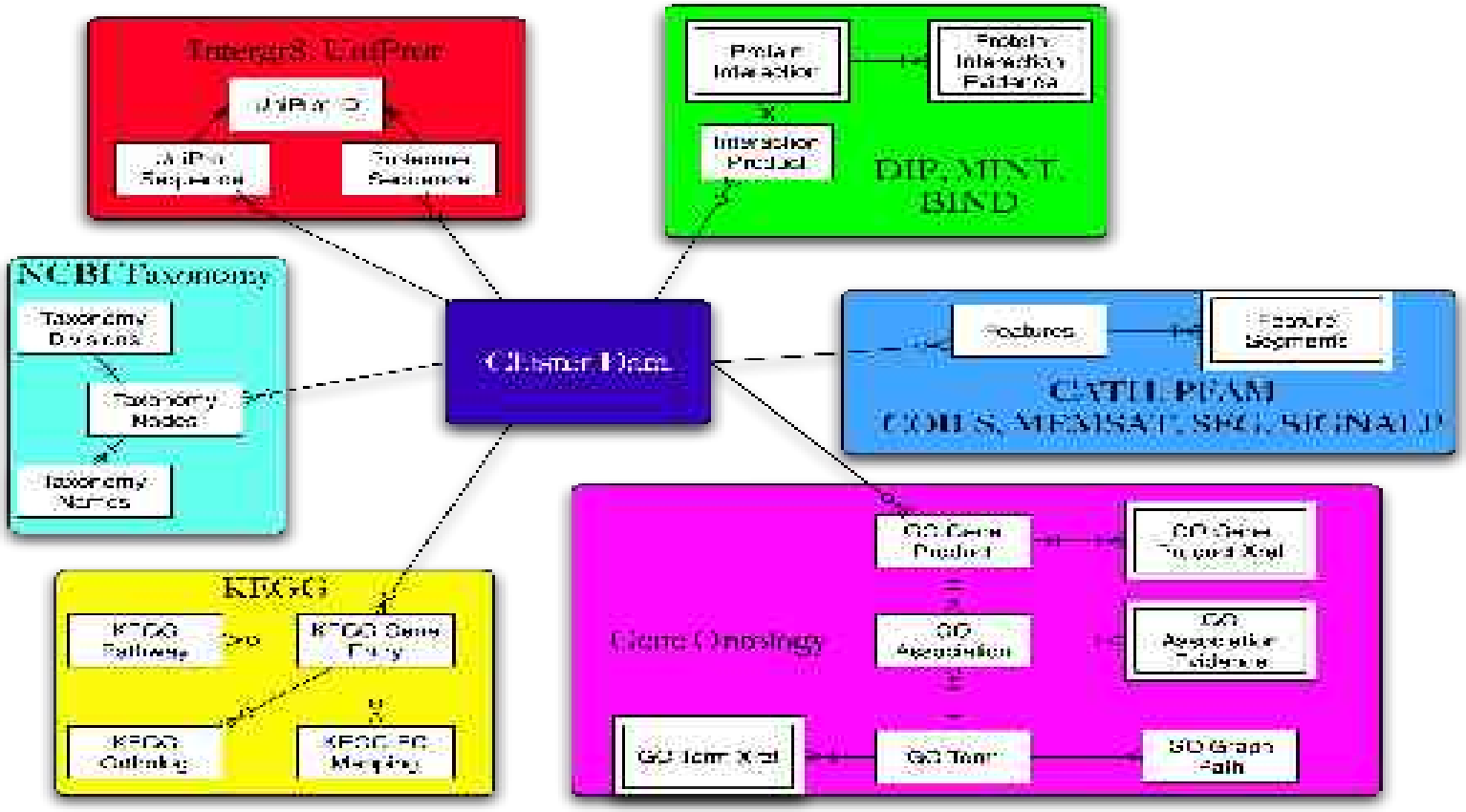
Each new sequence assigned to best hit family, or else new family created

Multi-linkage clustering used to subcluster within protein families with cluster thresholds of 0.3, 0.35, 0.4, 0.5, …, 1

Clustering


New Sequence
Blast Data (80% overlap)
Existing cluster hits
Hits 0 clusters
Y
Create new cluster
BioMap
N
Hits 1 clusters
Y
Classify into current cluster
BioMap
N
Flag as multi-cluster hit
Y
Classify into best cluster
BioMap

Clustering


NCBI Taxonomy
Taxonomy Nodes
Taxonomy Names
KEGG
KEGG Pathway
KEGG Gene
KEGG Ontology
KEGG EC Mapping
Gene Ontology
GO Association
GO Term
GO Graph Path
DIP, MINT, BIND
Protein Interaction
Interaction Protein
Protein Interaction Evidence
CATH, PFAM
Features
Feature Segments

The Integration Process

The integration process consists of the following:

1) Automatic generation of the AutoMed relational schemas corresponding to the data source and cluster data schema.

The Integration Process

The integration process consists of the following:

1) Automatic generation of the AutoMed relational schemas corresponding to the data source and cluster data schema.
2) Automatic generation of the AutoMed relational global schema GS

The Integration Process

The integration process consists of the following:

1) Automatic generation of the AutoMed relational schemas corresponding to the data source and cluster data schema.
2) Automatic generation of the AutoMed relational global schema GS
3) Automatic translation of above schemas into corresponding XMLDSS source/cluster schemas X_i and global schema GX

The Integration Process

The integration process consists of the following:

1) Automatic generation of the AutoMed relational schemas corresponding to the data source and cluster data schema.
2) Automatic generation of the AutoMed relational global schema GS
3) Automatic translation of above schemas into corresponding XMLDSS source/cluster schemas X_i and global schema GX
4) Partial conformance of each X_i to GX by **rename** transformations giving schemas X_i'

The Integration Process

The integration process consists of the following:

1) Automatic generation of the AutoMed relational schemas corresponding to the data source and cluster data schema.
2) Automatic generation of the AutoMed relational global schema GS
3) Automatic translation of above schemas into corresponding XMLDSS source/cluster schemas X_i and global schema GX
4) Partial conformance of each X_i to GX by **rename** transformations giving schemas X_i'
5) Complete conformance of each X_i' to GX by applying automatic XMLDSS schema transformation algorithm giving schemas X_i''

The Integration Process

The integration process consists of the following:

1) Automatic generation of the AutoMed relational schemas corresponding to the data source and cluster data schema.
2) Automatic generation of the AutoMed relational global schema GS
3) Automatic translation of above schemas into corresponding XMLDSS source/cluster schemas X_i and global schema GX
4) Partial conformance of each X_i to GX by **rename** transformations giving schemas X_i'
5) Complete conformance of each X_i' to GX by applying automatic XMLDSS schema transformation algorithm giving schemas X_i''
6) Apply any data cleansing transformations on each X_i'' creating GX_i . The reverse of the transformation generated in Step 3 can be appended to each GX_i to give GS

Use of XMLDSS

XMLDSS used as unifying data model for the intermediate schemas in the above process

Basic characteristics:

- Structure-only representation of XML document
- XML format ease of traversal & manipulation
- Automatically derived from an XML file
- XMLDSS from other schema types (DTD, XML Schema)

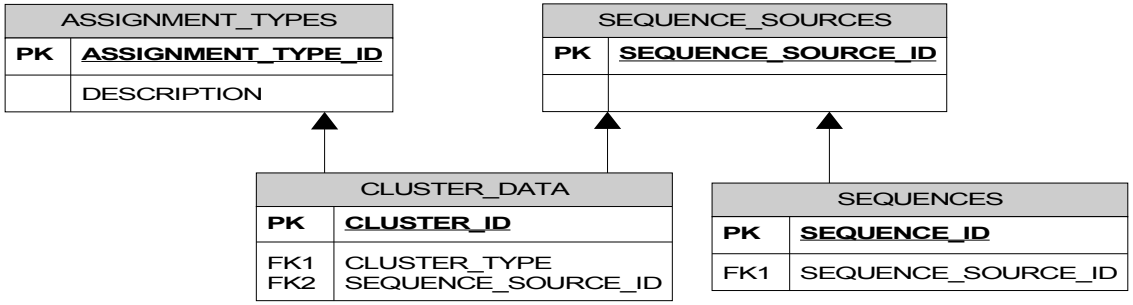
Step 3 – translating AutoMed relational to XMLDSS schemas

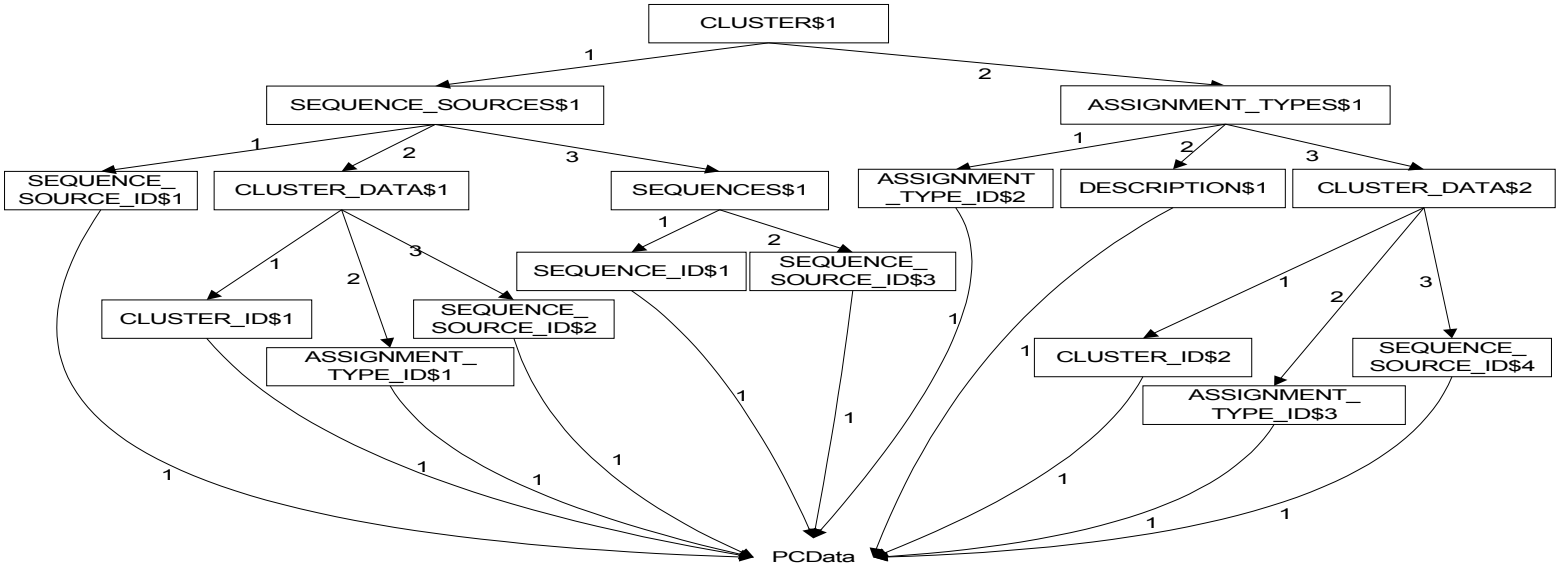
Generate a graph G from the relational schema with a node corresponding to each table and an arc corresponding to each foreign key

Create a set of trees obtained by traversing G from each node with no incoming edges, and convert to a single tree T by adding a generic root

Traverse T to generate the pathway from the relational schema to its corresponding XMLDSS schema

Source schemas fragment

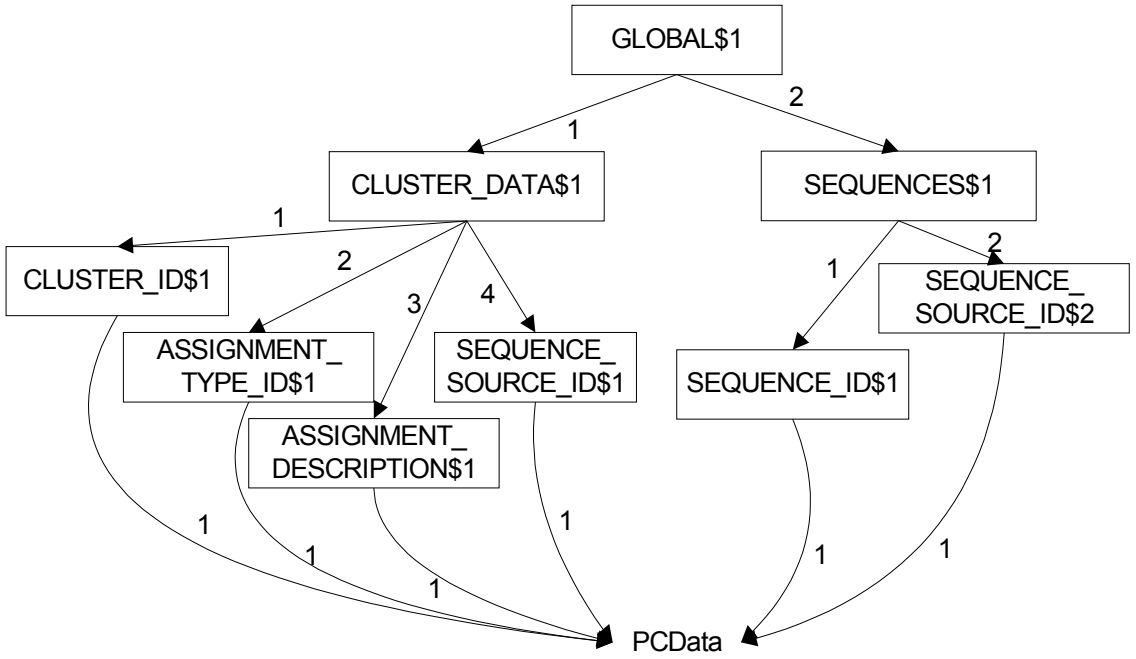




Target Schemas fragment

SEQUENCES	
PK	**SEQUENCE_ID**
	SEQUENCE_SOURCE_ID

CLUSTER_DATA	
PK	**CLUSTER_ID**
	ASSIGNMENT_TYPE_ID ASSIGNMENT_DESCRIPTION SEQUENCE_SOURCE_ID



Step 4 – Schema Matching

Transformation algorithm step 5 assumes that two schema constructs refer to the same real-world concept they have the same name

Manual **rename** transformations to reflect this

```
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>>)
```

Step 5 – Automatic XMLDSS-based integration

Growing phase: traverse the target schema T depth-first and issue an add/extend transformation for every construct that does not exist in the source schema S

Shrinking phase: traverse S depth-first and issue a delete/contract transformation for every construct that does not exist in T resulting in intermediate schema S'

Rename phase: traverse S' depth-first and issue necessary rename transformations to reorder subtrees to create a final schema S_T syntactically identical to T

Transformation pathway fragment

```
add(<<0,GLOBAL$1,CLUSTER_DATA$1>>,
      [{v0,v2}|{v0,v1}<<<GLOBAL$1,SEQUENCE_SOURCES$1>>;
       {v1,v2}<-<<SEQUENCE_SOURCES$1,CLUSTER_DATA$1>>]);
```

Transformation pathway fragment

```
add(<<0,GLOBAL$1,CLUSTER_DATA$1>>,
      [{v0,v2}|{v0,v1}<<<GLOBAL$1,SEQUENCE_SOURCES$1>>;
       {v1,v2}<-<<SEQUENCE_SOURCES$1,CLUSTER_DATA$1>>]);
add(<<0,GLOBAL$1,SEQUENCES$1>>,
      [{v0,v2}|{v0,v1}<-<<GLOBAL$1,SEQUENCE_SOURCES$1>>;
       {v1,v2}<-<<SEQUENCE_SOURCES$1,SEQUENCES$1>>]);
```

Transformation pathway fragment

```
add(<<0,GLOBAL$1,CLUSTER_DATA$1>>,
      [{v0,v2}|{v0,v1}<<<GLOBAL$1,SEQUENCE_SOURCES$1>>;
       {v1,v2}<-<<SEQUENCE_SOURCES$1,CLUSTER_DATA$1>>]);
add(<<0,GLOBAL$1,SEQUENCES$1>>,
      [{v0,v2}|{v0,v1}<-<<GLOBAL$1,SEQUENCE_SOURCES$1>>;
       {v1,v2}<-<<SEQUENCE_SOURCES$1,SEQUENCES$1>>]);
extend(<<0,CLUSTER_DATA$1,ASSIGNMENT_DESCRIPTION$1>>,
      [{v1,v2}|{v0,v1}<-<<ASSIGNMENT_TYPES$1,CLUSTER_DATA$2>>;
       {v0,v2}<-<<ASSIGNMENT_TYPES$1,ASSIGNMENT_DESCRIPTION$1>>);
```

Transformation pathway fragment

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

Step 6 – Data Cleansing

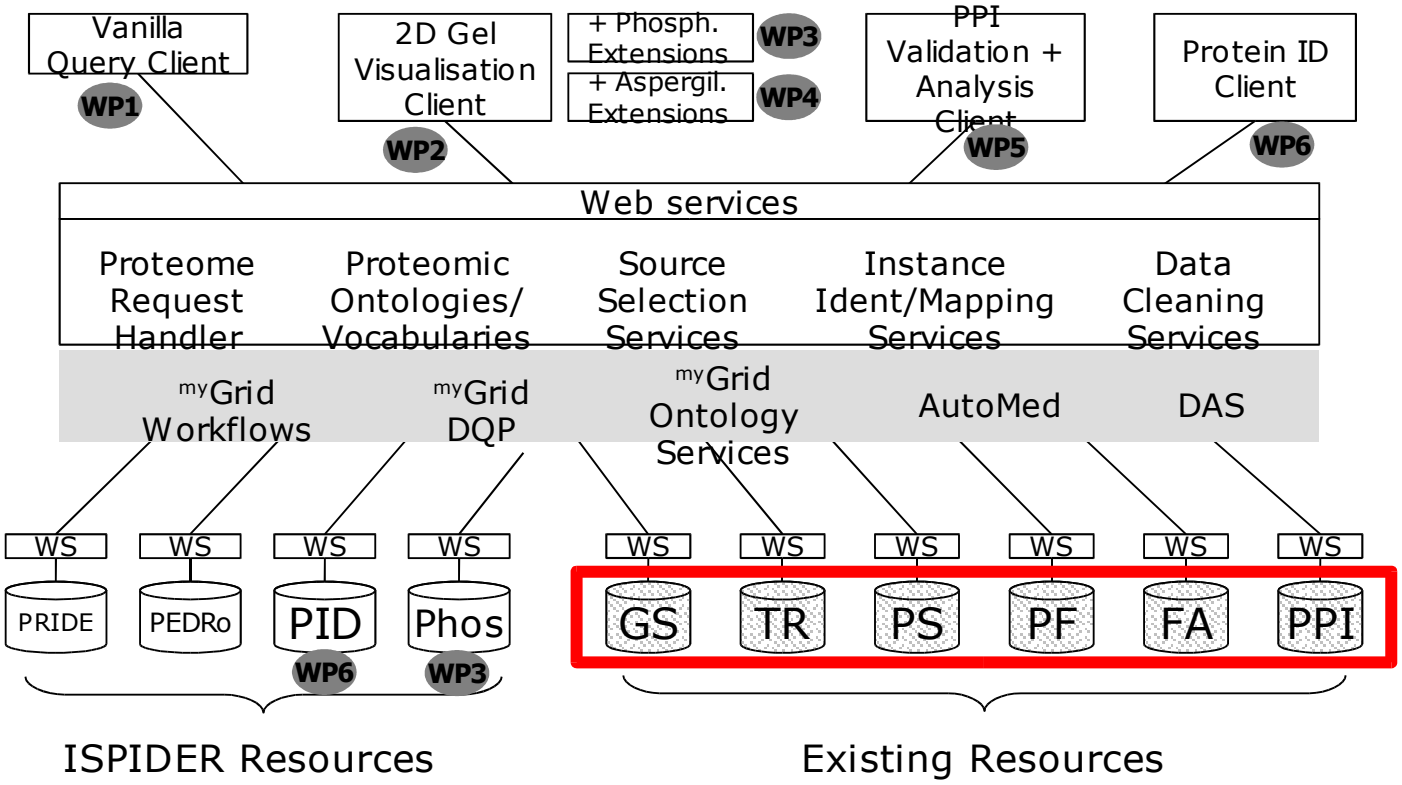
Further transformations may be applied to remove representational heterogeneities at the data level

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

Ongoing Work

- Clustering approach being extended to integrate feature and domain recognition (HMM) approaches to identify attributes of sequences
- Application to warehouse materialisation and maintenance
- Textual data sources
- Grid-based integration of biological data sources - ISpider

Integrated Proteomics Informatics Platform
ISPIDER Architecture



ISPIDER Proteomics Clients

ISPIDER Proteomics Grid Infrastructure Existing E-Science Infrastructure

Public Proteomic Resources

Collaborators and funding

AutoMed – Imperial College London (Comp Sci)

//www.doc.ic.ac.uk/automed/

BioMap – UCL (Comp Sci, Centre for Virology, Institute of Child Health)
EBI, University of Brunel (IS & Comp)

//www.biochem.ucl.ac.uk/bsm/biomap/

ISpider – Univ of Manchester (Biological Sciences, Comp Sci)
EBI, UCL (Comp Sci)

//www.ispider.man.ac.uk/

Funding: BBSRC, EPSRC, Wellcome Trust