

# AutoMed

A Heterogeneous Data Integration System



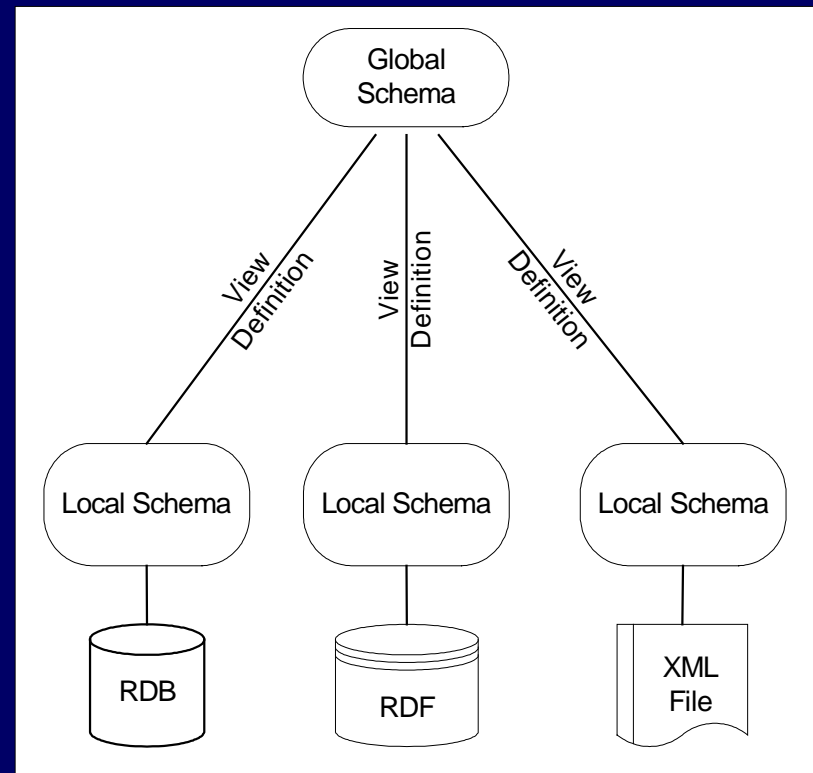


# Outline

- n Both-As-View (BAV) approach
  - n GAV & LAV approaches
  - n BAV approach
  - n Comparison of integration approaches
  - n BAV advantages
- n The AutoMed system
  - n Architecture
  - n Current & future work
  - n Testbeds

# GAV & LAV Approaches

- n Global-As-View (GAV) approach: describe GS constructs with view definitions over  $LS_i$  constructs
- n Local-As-View (LAV) approach: describe  $LS_i$  constructs with view definitions over GS constructs



# Global-As-View Approach (GAV)

$S_g$  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)

n student(id,name,left,degree) =  
[ $\{x,y,z,w\} \mid \langle x,y,z,w,\_ \rangle \in \text{ug} \wedge$   
 $\langle x,\_,\_,\_,\_ \rangle \notin \text{phd} \vee$   
 $\langle x,y,z,w,\_ \rangle \in \text{phd} \wedge$   
 $w = \text{'phd'}$ ]

n monitors(sno,id) =  
[ $\{x,y\} \mid \langle x,\_,\_,\_,\_ \rangle \in \text{ug} \wedge$   
 $\langle x,\_,\_,\_,\_ \rangle \notin \text{phd} \vee$   
 $\langle x,y \rangle \in \text{supervises}$ ]

n staff(sno,sname,dept) =  
[ $\{x,y,z\} \mid \langle x,y,z,w,\_ \rangle \in \text{tutor} \wedge$   
 $\langle x,\_,\_ \rangle \notin \text{supervisor} \vee$   
 $\langle x,y,z \rangle \in \text{supervisor}$ ]

# Local-As-View Approach (LAV)

$S_g$  student(id, name, left#, degree)  
monitors(sno, id)  
staff(sno, sname, dept#)

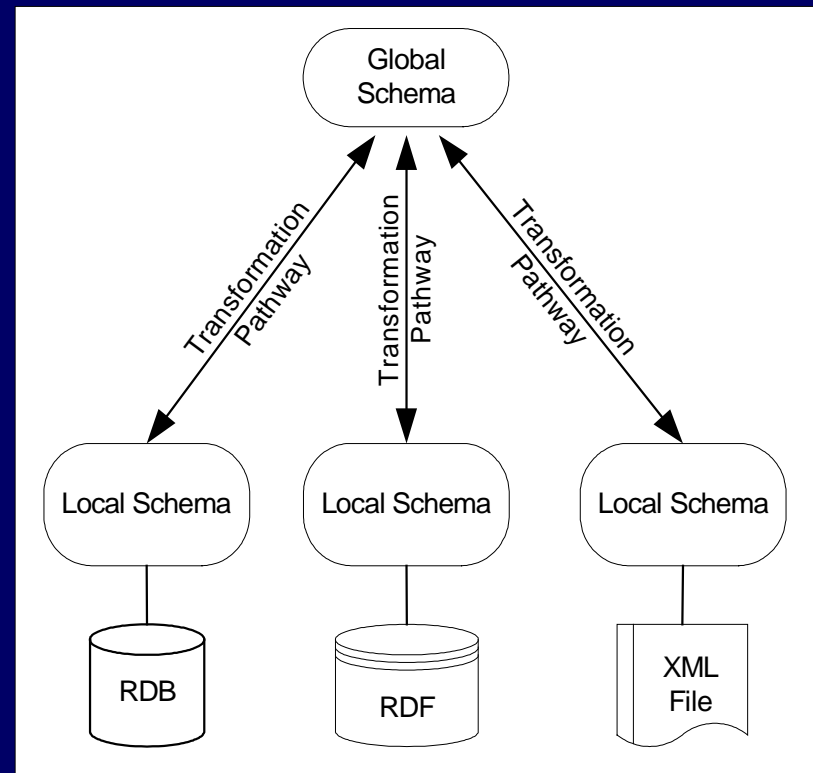
$S_1$  ug(id, name, left#, degree, sno)  
tutor(sno, sname)

$S_2$  phd(id, name, left#, title)  
supervises(sno, id)  
supervisor(sno, sname, dept)

- n tutor(sno, sname) =  
[ $\{x, y\} \mid \langle x, y, \_ \rangle \in \text{staff} \wedge$   
 $\langle x, z \rangle \in \text{monitors} \wedge$   
 $\langle z, \_, \_, w \rangle \in \text{student} \wedge$   
 $w \neq \text{'phd'}$ ]
- n ug(id, name, left, degree, sno) =  
[ $\{x, y, z, w, v\} \mid \langle x, y, z, w \rangle \in \text{student}$   
 $\wedge \langle v, x \rangle \in \text{monitors} \wedge$   
 $w \neq \text{'phd'}$ ]

# Both-As-View (BAV) (1/3)

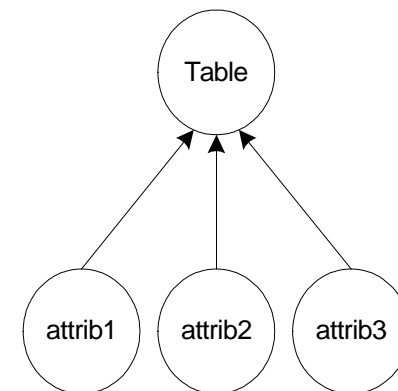
- n Schema transformation approach
- n For each pair  $(LS_i, GS)$ : incrementally modify  $LS_i/GS$  to match  $GS/LS_i$



# Both-As-View (BAV) (2/3)

- n Common Data Model:  
**Hypergraph Data Model (HDM)**
  - n Constructs are nodes, edges & constraints
  - n It avoids the semantic mismatches that may occur between constructs of higher-level modelling languages

Table	
<b>PK</b>	<b><u>attribute1</u></b>
	attribute2 attribute3



# Both-As-View (BAV) (3/3)

n Modify using primitive schema transformations

n add/delete

n rename

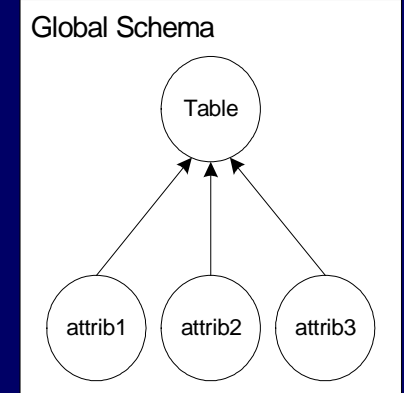
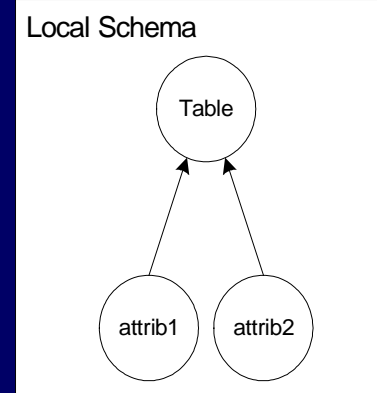
n extend/contract

n Supply transformations with queries

n  $\text{add}(\langle\langle \text{table}, \text{attrib3} \rangle\rangle, q)$ , where  $q$ :

$[\{t, (a_1 + a_2)\} | \{t, a_1\} \mathcal{B} \langle\langle \text{table}, \text{attrib1} \rangle\rangle; \{t, a_2\} \mathcal{B} \langle\langle \text{table}, \text{attrib2} \rangle\rangle]$

n  $\text{extend}(\langle\langle \text{table}, \text{attrib3} \rangle\rangle, q_1, q_2)$



# Example (1/2)

$S_g$  student(id,name,left#,degree)  
monitors(sno,id)  
staff(sno,sname,dept#)

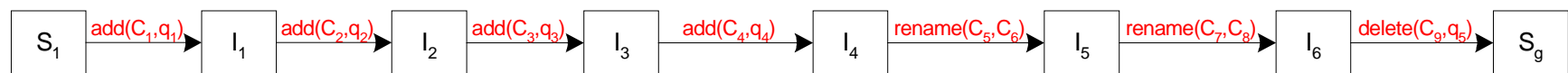
$S_1$  ug(id,name,left#,degree,sno)  
tutor(sno,sname)

$S_2$  phd(id,name,left#,title)  
supervises(sno,id)  
supervisor(sno,sname,dept)

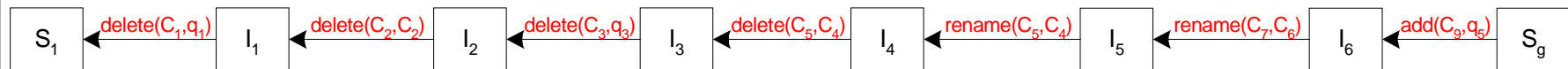
- n  $S_1 \rightarrow S_g$ :
  - n add(⟨⟨monitors⟩⟩,  $q_1$ )
  - n add(⟨⟨monitors, sno⟩⟩,  $q_2$ )
  - n add(⟨⟨monitors, id⟩⟩,  $q_3$ )
  - n add(⟨⟨tutor, dept#⟩⟩,  $q_4$ )
  - n rename(⟨⟨ug⟩⟩, ⟨⟨student⟩⟩)
  - n rename(⟨⟨tutor⟩⟩, ⟨⟨staff⟩⟩)
  - n delete(⟨⟨student, sno⟩⟩,  $q_5$ )
- n  $S_2 \rightarrow S_g$ : can be derived similarly

# Example (2/2)

$S_1 \dot{\rightarrow} S_g$



$S_1 \dot{\leftarrow} S_g$



n Automatically derivable reverse transformations

- n  $\text{add}(C,q)/\text{extend}(C,q_1,q_2) : \text{delete}(C,q)/\text{contract}(C,q_1,q_2)$
- n  $\text{delete}/\text{contract} : \text{add}/\text{extend}$
- n  $\text{rename}(C_1,C_2) : \text{rename}(C_2,C_1)$



# BAV vs. LAV, GAV & GLAV

- n BAV approach subsumes other integration approaches:
  - n Can be used to derive GAV & LAV view definitions (ICDE'03)
  - n Comparison with GAV, LAV & GLAV in DBIS'04

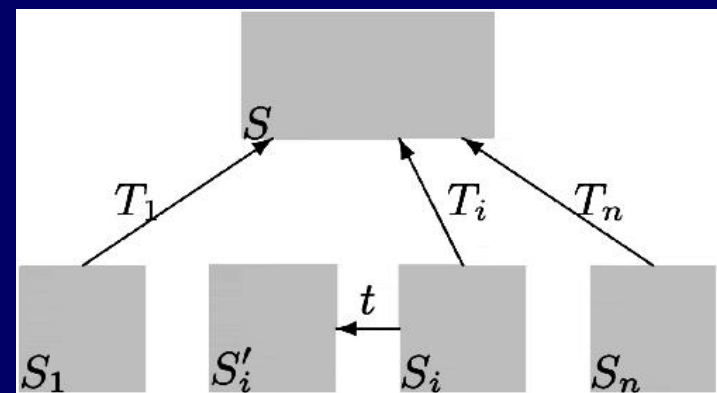
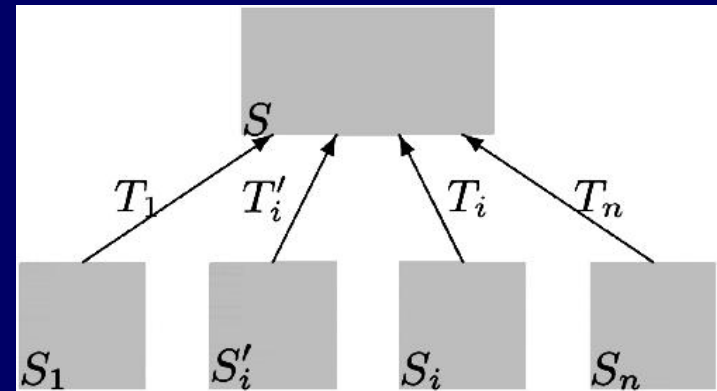


# Schema Evolution

- n In GAV & LAV view definitions have to be regenerated
- n The BAV approach readily supports the **evolution** of both **local** and **global** schemas
- n In particular (CAiSE'02 & ICDE'03 papers):
  - n if the evolved schema is **semantically equivalent** to the original schema, schema evolution is automatic
  - n if the evolved schema is a **contraction** of the original schema, schema evolution is automatic
  - n if the evolved schema is an **extension** of the original schema, then domain knowledge may be required (but again the pathway can be evolved rather than regenerated)

# Local Schema Evolution Example

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





# Types Of Integration

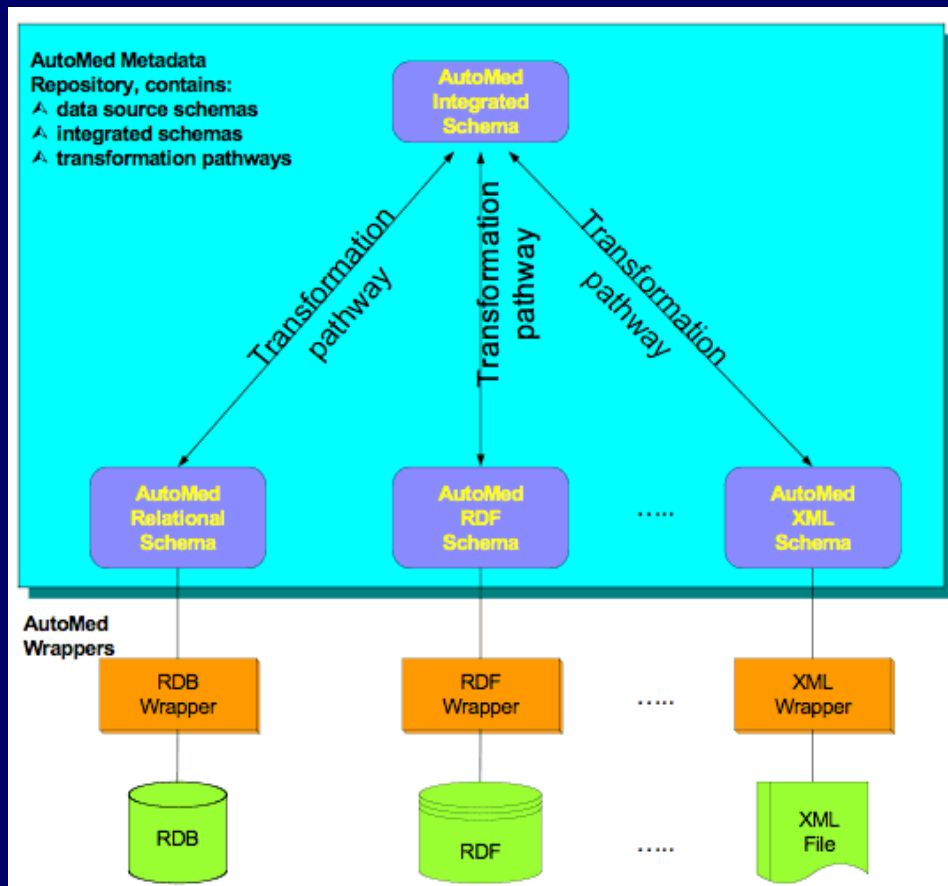
- n Virtual integration
- n Materialised integration
- n Hybrid integration



# Outline

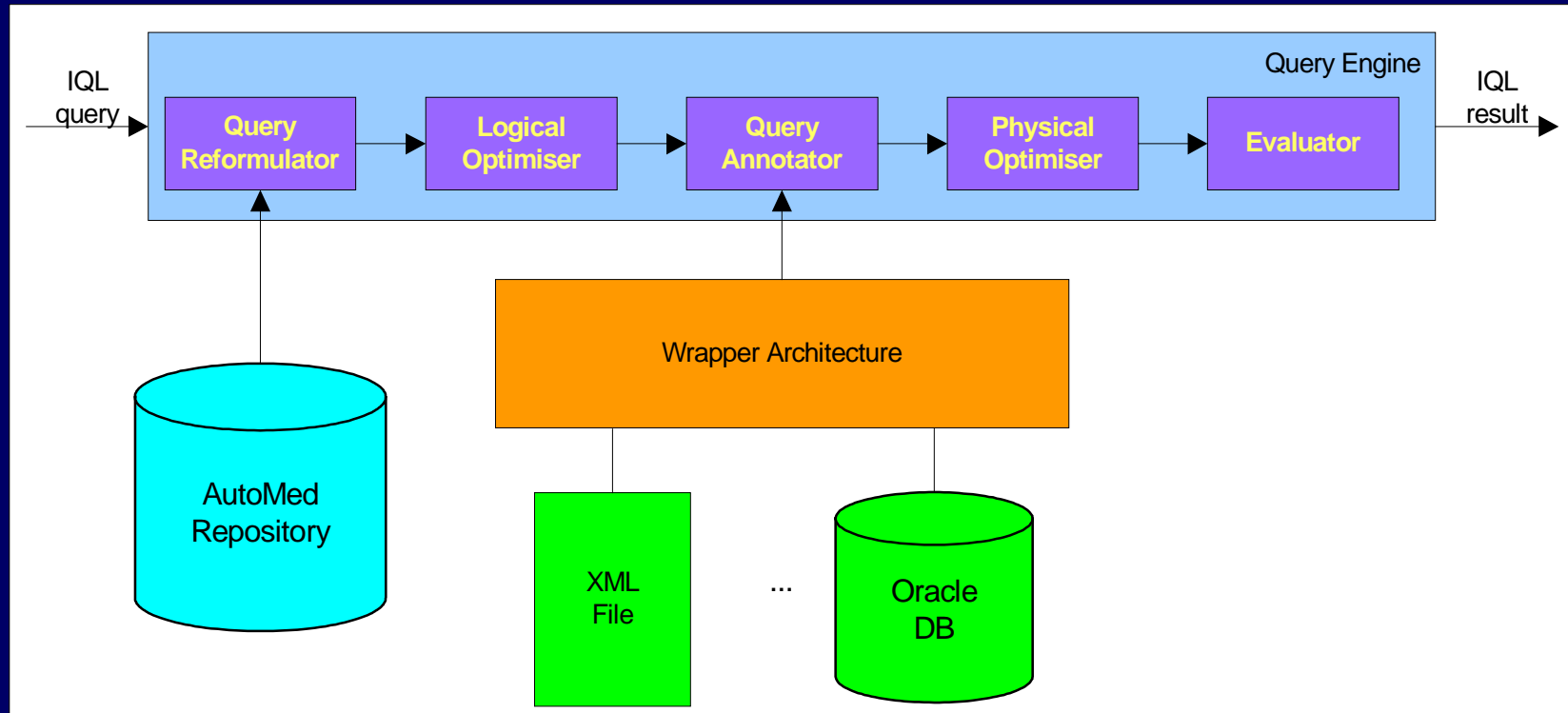
- n Both-As-View (BAV) approach
  - n GAV & LAV approaches
  - n BAV approach
  - n Comparison with GAV, LAV & GLAV
  - n BAV advantages
- n The AutoMed system
  - n Architecture
  - n Current & future work
  - n Testbeds

# The AutoMed System

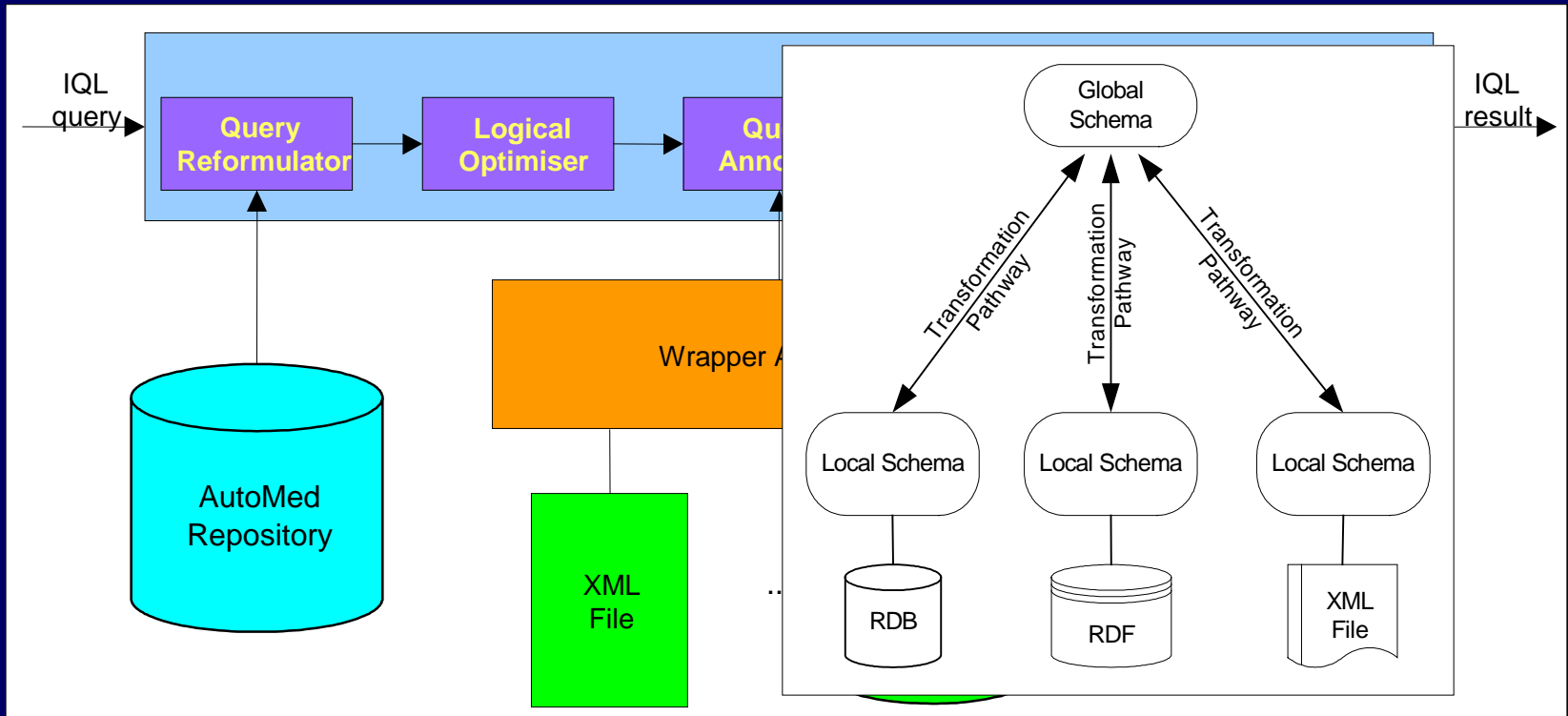


- n The AutoMed toolkit implements the BAV data integration approach
- n AutoMed repository:
  - n Model Definitions Repository (MDR)
  - n Schema Transformation Repository (STR)
- n AutoMed query language: IQL
  - n Higher-level query languages are translated to IQL
  - n IQL is translated to the query languages of the datasources

# Query Engine



# Query Engine





# Wrappers

## n Current

- n Relational (Oracle, PostgreSQL, SQLServer)
- n XML documents (DOM & SAX)
- n YATTA
- n RDF

## n Near future

- n Object-oriented (ODMG 3.0 compliant)
- n Native XML Databases (Xindice, Sedna)
- n RDF Schema Specific DataBase (RSSDB)



# Testbeds

## n BioMap

- n <http://www.biochem.ucl.ac.uk/bsm/biomap>
- n Data warehouse containing diverse biological data
- n AutoMed used for the creation and maintenance of the data warehouse

## n ISPIDER

- n <http://www.ispider.man.ac.uk>
- n Develop a Grid architecture for sharing data from various biological data sources (such as BioMap)
- n Extend AutoMed system with Grid services



# Development/Research Areas

- n Query engine
  - n Query processing & optimisation
  - n Query language translation
- n Tools
  - n Data Warehousing – data lineage
  - n Automatic schema matching (data mining)
  - n Automatic integration of XML data sources
  - n Unstructured/semi-structured data
  - n Transformation pathway optimisation
  - n Visualisation tool
- n Grid/P2P architecture



# Project Information

- n Homepage: <http://www.doc.ic.ac.uk/automed>
- n Technical details
  - n Papers
  - n Technical reports
- n Software
  - n AutoMed releases
  - n Documentation



# Project Members

- n Birkbeck College

- n Alexandra Poulouvassilis (P.I.)
- n Hao Fan
- n Dean Williams
- n Lucas Zamboulis

- n Past members

- n Tanvir Amed Faqueer
- n Edgar Jasper
- n Dimitri Theodoratos

- n Imperial College

- n Peter McBrien (P.I.)
- n Mike Boyd
- n Sasivimol Kittivoravitkul
- n Nikolaos Rizopoulos
- n Nerissa Tong

- n Past members

- n Siegfried Hodgson
- n Charalambos Lazanitis



# XML Data Transformation & Integration



Lucas Zamboulis, Alexandra Poulouvasilis  
{lucas,ap}@dcs.bbk.ac.uk



# Overview

- n Objective: restructuring & integration of XML files
- n Motivation
  - n Interoperability
  - n Related work on relational databases
  - n Need for XML-specific solutions



# Outline

- n Semantic Heterogeneity
  - n Schema Matching
  - n Ontologies
- n Structural Heterogeneity
  - n XML schema type in AutoMed
  - n Schema transformation
  - n Schema integration



# Semantic Heterogeneity

- n Problem definition
- n Schema Matching
  - n Data mining
  - n Neural networks
  - n Machine learning (LSD)
- n Ontologies (RDFS/OWL)



# Schema Matching (1/2)

- n Types:

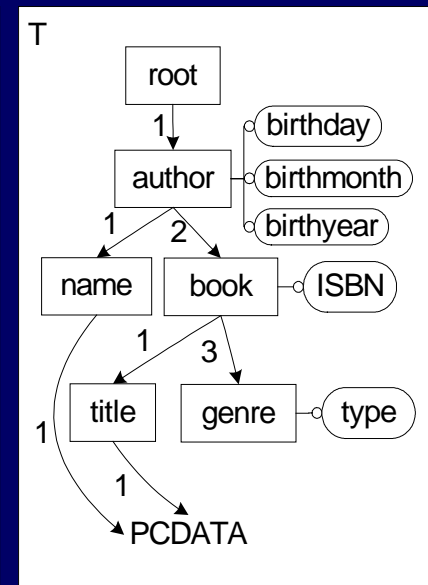
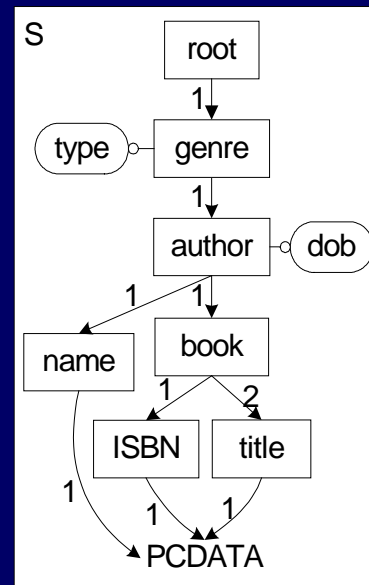
- n 1-1, 1-n, n-1, n-m

- n Subset, superset, equivalence

- n Use schema matching output to create the intermediate schemas used by the schema restructuring / schema integration algorithms

# Schema Matching (2/2)

- n Necessary transformations:
  - n add attributes day, month, year in S
  - n delete attribute dob from S
- n The reverse transformation pathway describes a  $n-1$  match





# Structural Heterogeneity

- n Problem: Same information can be represented in many different ways
  - n Ancestor – descendant  $\beta \rightarrow$  different branches
  - n Elements & attributes not clearly distinguished in XML model
  - n Ordering policy



# Aims

- n XML-specific solution:
  - n Insert-remove-rename operations on elements, attributes, edges
  - n Efficient 'move' (node/subtree) operation
  - n Element-to-attribute, attribute-to-element transformations
- n Avoid loss of data due to structural incompatibilities
- n Automation



# A Schema Type For XML

## n DTD

- n Advantage: wide adoption

- n Disadvantages:

  - n Non-XML format

  - n Grammar

## n XML Schema

- n Advantage: XML format

- n Disadvantages:

  - n Grammar

  - n Unnecessary complexity

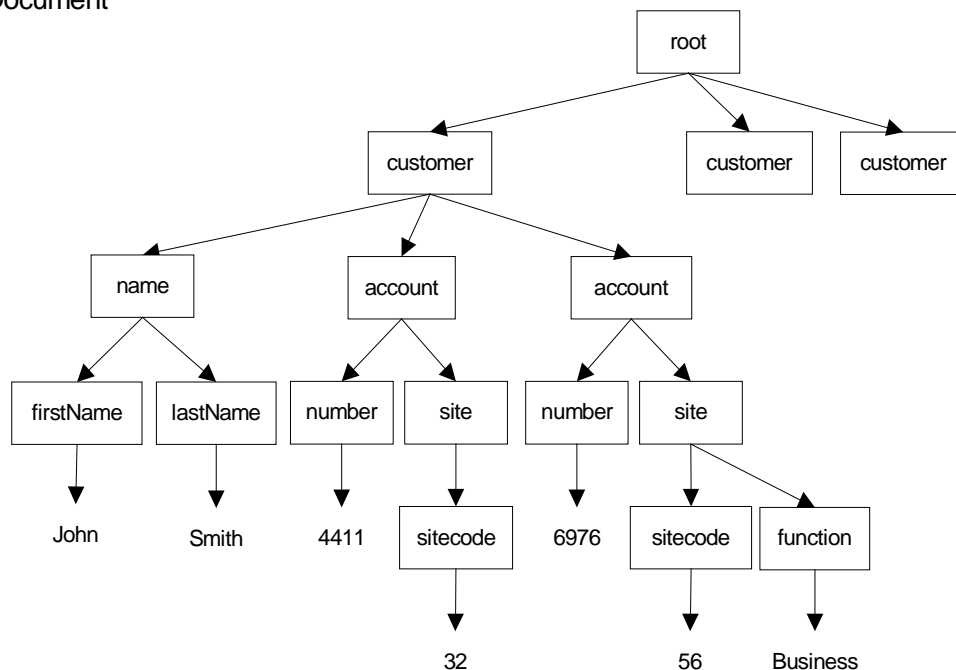


# XML DataSource Schema (1/3)

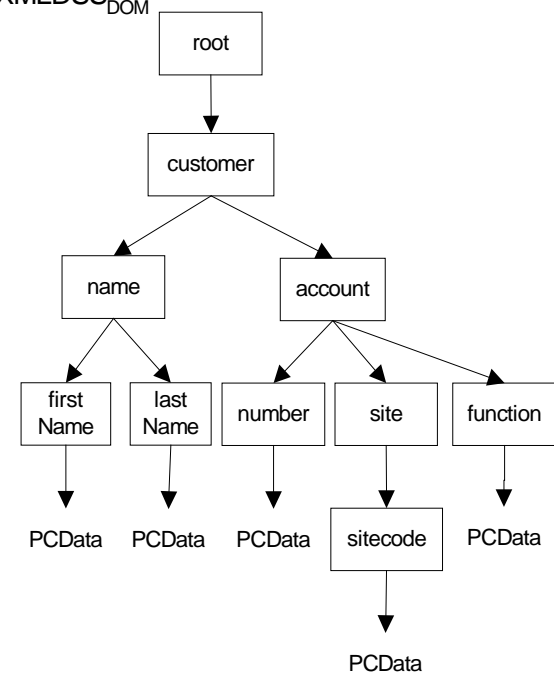
- n Basic characteristics:
  - n Structure-only representation
  - n XML format  $\Rightarrow$  ease of traversal & manipulation
  - n Automatically derived from an XML file
  - n XMLDSS from other schema types (DTD, XML Schema)

# XML DataSource Schema (2/3)

Document



XMLDSS<sub>DOM</sub>





# XML DataSource Schema (3/3)

- n XMLDSS is being extended
  - n Structural summary → schema type (persistence, describe multiple documents)
  - n Constraints
    - n Primary/foreign keys
    - n Cardinality
    - n Ordering
- n If present, translate DTD/XML Schema to XMLDSS

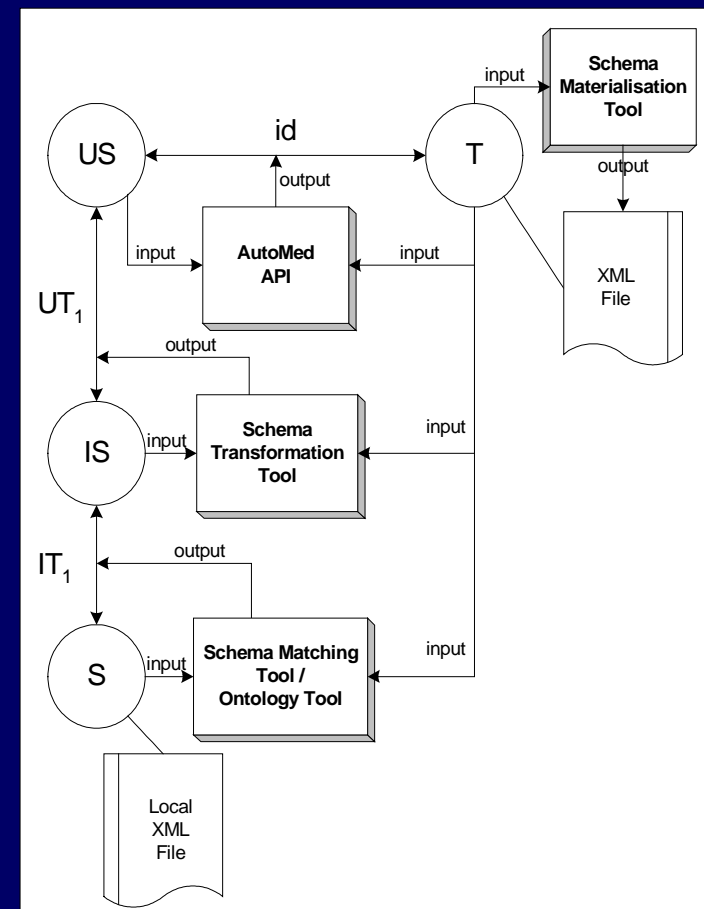


# Schema Transformation (1/2)

- n Target schema  $T$  given
- n Source schema  $S$  is transformed to match the structure of  $T$

# Schema Transformation (2/2)

- n Schema matching phase
- n Schema transformation phase
- n id phase
- n Target schema materialisation





# Algorithm

- n Growing phase: traverse the target schema and issue an add/extend transformation for every construct that does not exist in the source schema.
- n Shrinking phase: traverse the source schema and issue an delete/contract transformation for every construct that does not exist in the target schema.
- n Completeness of algorithm



# Transformation Types

- n AutoMed primitive transformations:
  - n add/extend
  - n delete/contract
  - n rename
- n Schema level:
  - n Insert, remove or rename schema constructs
  - n Move element/subtree
  - n Element  $\beta$  à attribute

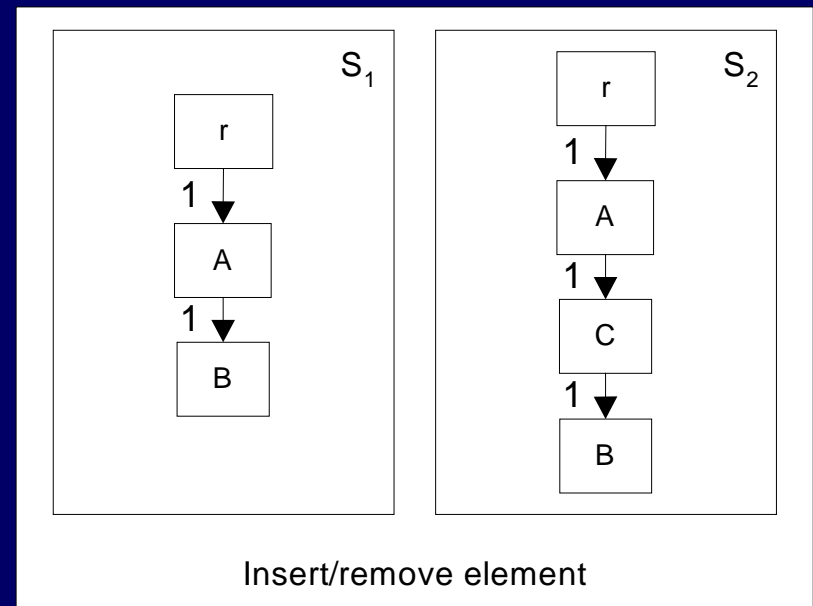
# Example 1

## n Insert element C

- n  $\text{ext}(\langle C \rangle, \text{Void}, \text{Any})$
- n  $\text{ext}(\langle A, C \rangle, \text{Void}, \text{Any})$
- n  $\text{ext}(\langle C, B \rangle, \text{Void}, \text{Any})$
- n  $\text{del}(\langle A, B \rangle, q)$

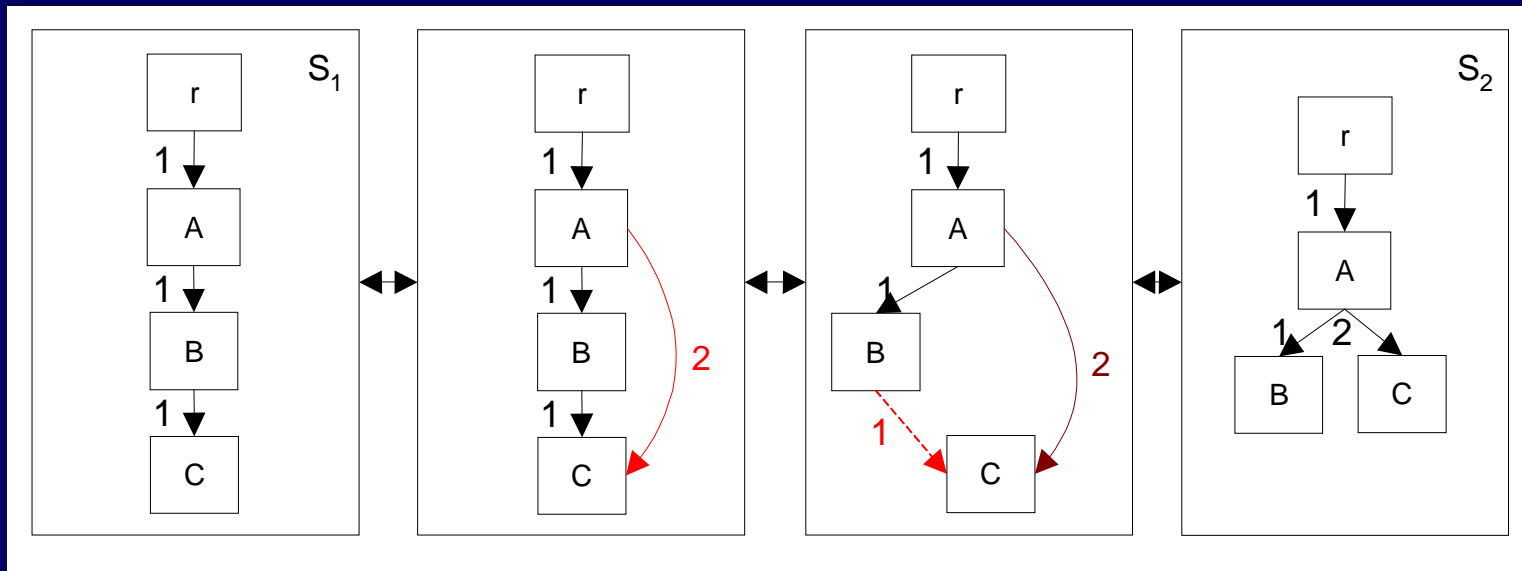
## n Remove element C

- n  $\text{add}(\langle A, B \rangle, q)$
- n  $\text{con}(\langle C \rangle, \text{Void}, \text{Any})$
- n  $\text{con}(\langle C, B \rangle, \text{Void}, \text{Any})$
- n  $\text{con}(\langle A, C \rangle, \text{Void}, \text{Any})$



# Example 2

$n$  Insert/remove edge: move operation



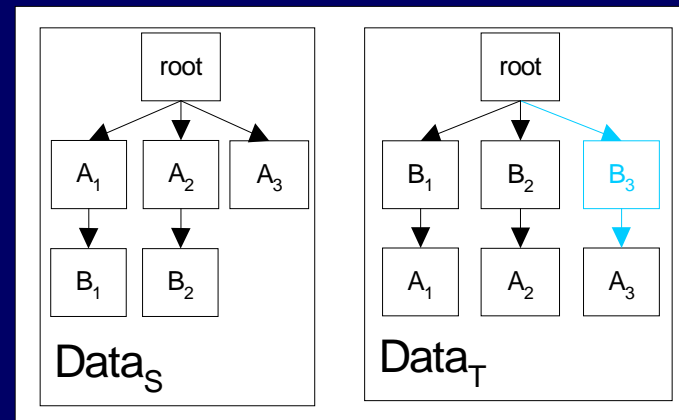
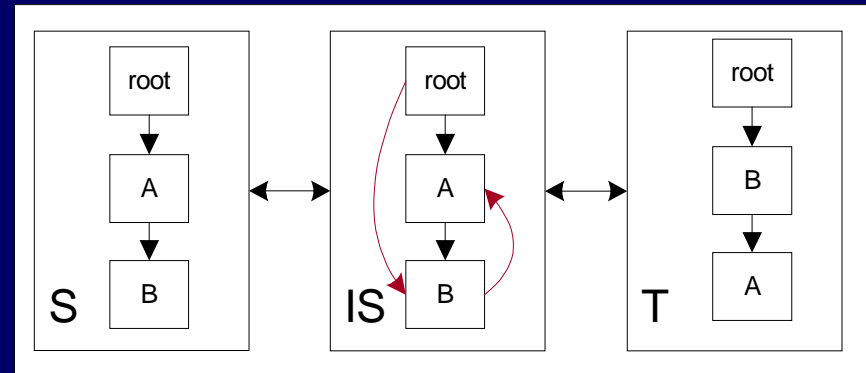
# Example 3

## n Move:

- n add( $\langle \text{root}, B \rangle, q_3$ )
- n add( $\langle B, A \rangle, [\{b, a\} | \{a, b\} \mathcal{B} \langle A, B \rangle]$ )
- n delete( $\langle A, B \rangle$ )
- $[\{a, b\} | \{b, a\} \mathcal{B} \langle B, A \rangle]$

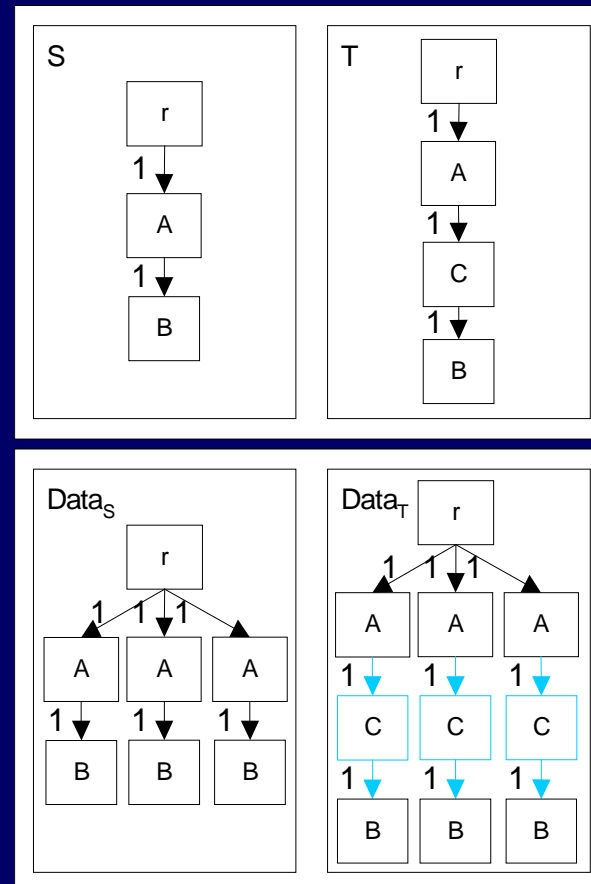
## n Complete:

- n add( $\langle B' \rangle, \langle B \rangle ++ q_1$ )
- n add( $\langle A, B' \rangle, \langle A, B \rangle ++ q_2$ )
- n delete( $\langle A, B \rangle, \langle A, B' \rangle$ )
- n delete( $\langle B \rangle, \langle B' \rangle$ )
- n rename( $\langle B' \rangle, \langle B \rangle$ )



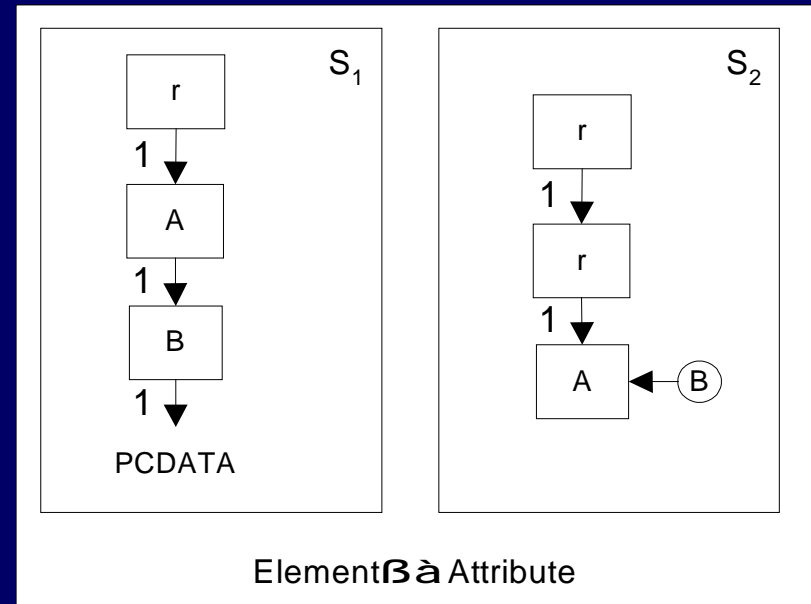
# Example 1 - revisited

- n Actually, this can also be treated with an add/delete transformation

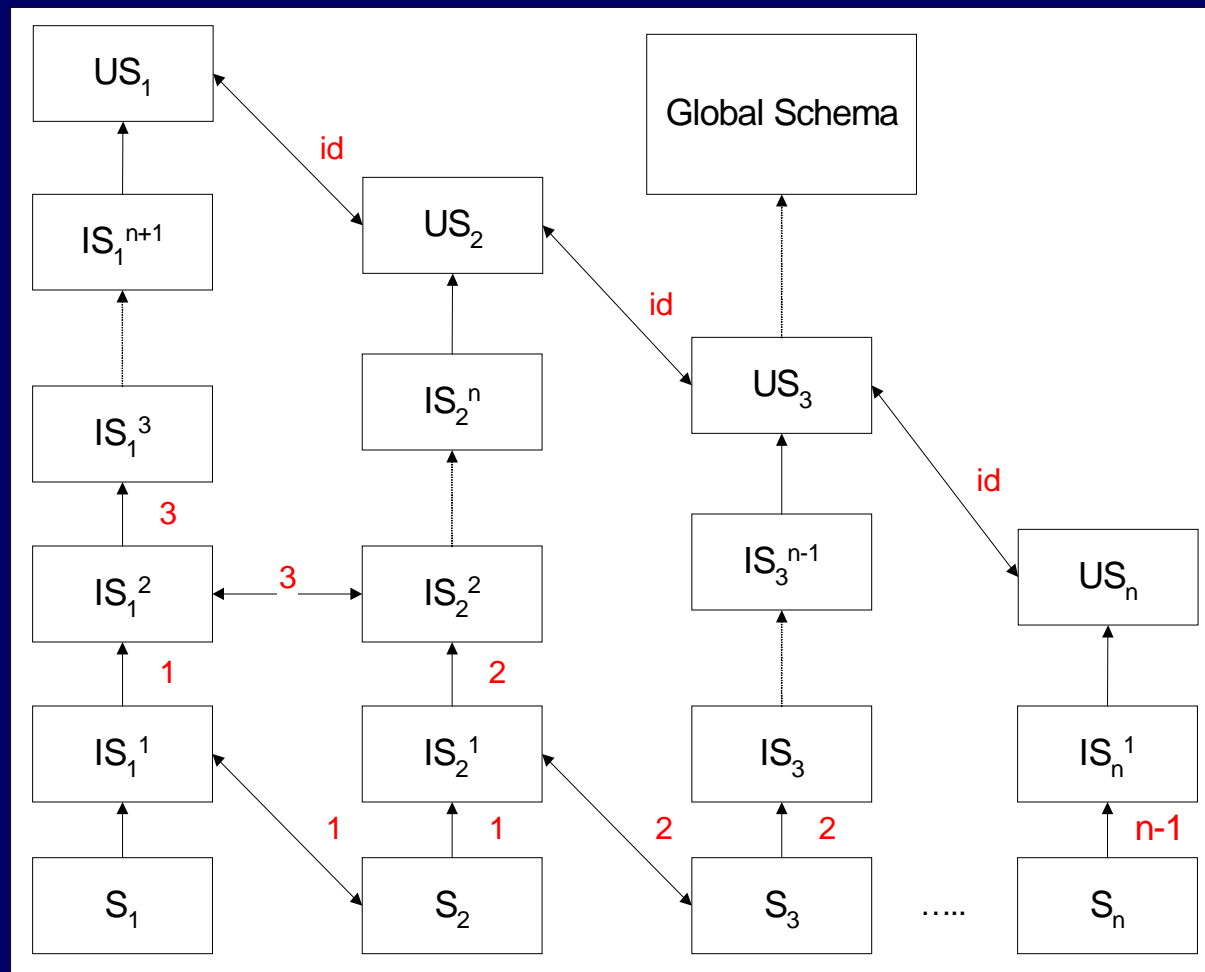


# Example 4

- n Element-to-attribute transformation
  - n insert(<A,A:B>,q)
  - n remove(<A,B>,q)
  - n remove(<B,PCDATA>,q)
  - n remove(<B>,q)
- n Attribute-to-element transformation
  - n insert(<B>,q)
  - n insert(<A,B>,q)
  - n insert(<B,PCDATA>,q)
  - n remove(<A,A:B>,q)

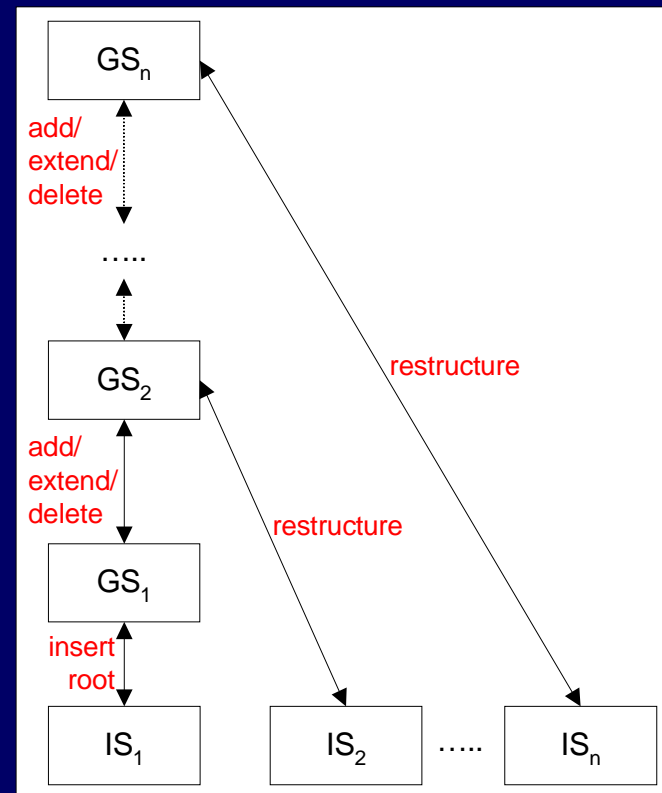


# Schema Integration – Type I



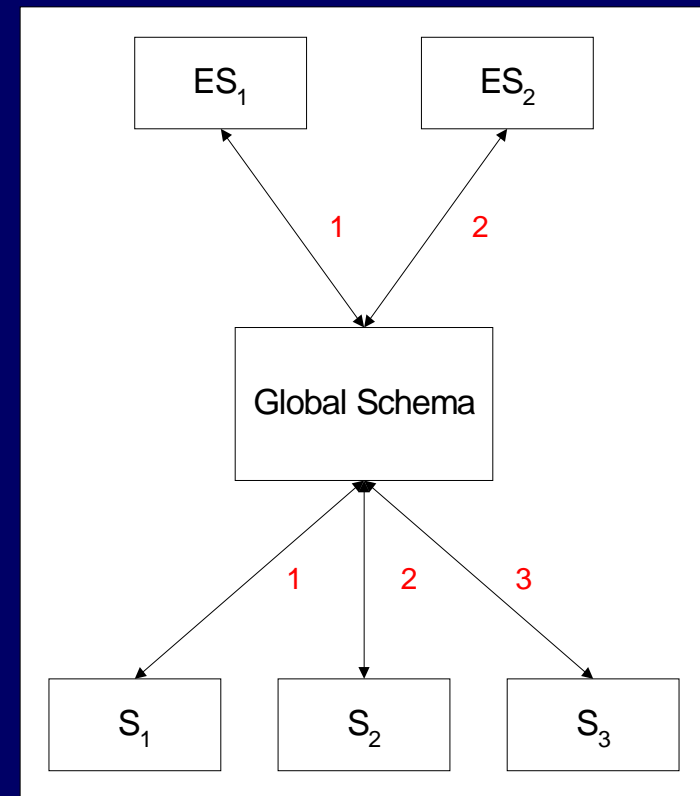
# Schema Integration – Type II

- n Type I integration performs two tasks at once:
  - n schema integration
  - n schema improvement
- n Type II:
  - n Augment with missing constructs
  - n Remove redundant constructs



# Schema Integration – Type II

- n Improve GS as a second step





# Materialisation

## n Strategy:

- n Materialise root and its attributes
- n Consider all edges  $(e_p, e_c)$  in a depth-first way
- n Materialise  $e_c$  and its attributes



# Conclusions

- n XML specific solution:
  - n element  $\beta$   $\rightarrow$  attribute transformations
  - n move operation
- n No loss of data by synthetically creating missing structure



# Evaluation

## n BIOMAP

- n Integration of biological data sources
- n Relational databases, XML documents, XML databases



# Future Work

## n Short-term

- n Use ontologies for resolving semantic heterogeneity
- n Extend XMLDSS
- n Native XML Databases (Xindice, Sedna)
- n XML-Enabled Databases (Oracle)

## n Long-term

- n Schema integration:
  - n GS improvement (type II)
  - n Overlapping data identification
  - n Targeted rematerialisation of GS
- n Schema evolution