

Using AutoMed for XML Data Transformation & Integration

L. Zamboulis - A. Poulouvassilis
{lucas,ap}@dcs.bbk.ac.uk

Outline

- **Integration Approaches**
 - GAV – LAV
 - BAV – AutoMed system
- **XML Data Transformation & Integration**
 - XML DataSource Schema (XMLDSS)
 - Schema restructuring
 - Schema integration
 - Schema materialisation
 - Conclusions – future work

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)

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

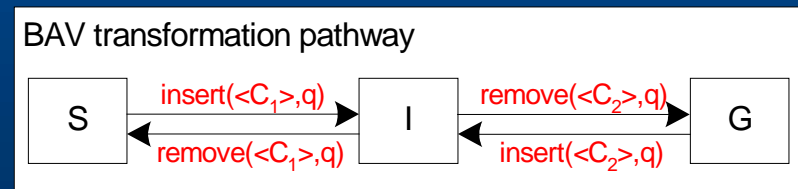
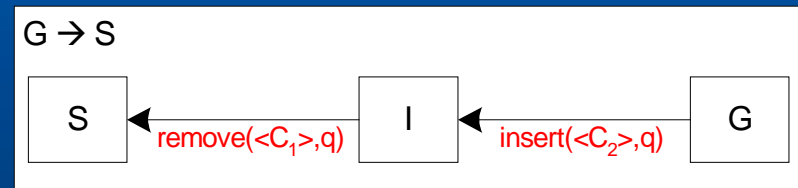
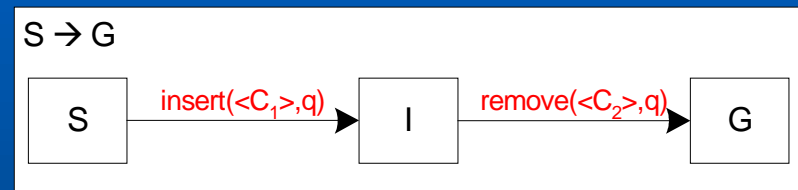
S_1 ug(id, name, left#, degree, sno)
tutor(sno, sname)

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

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

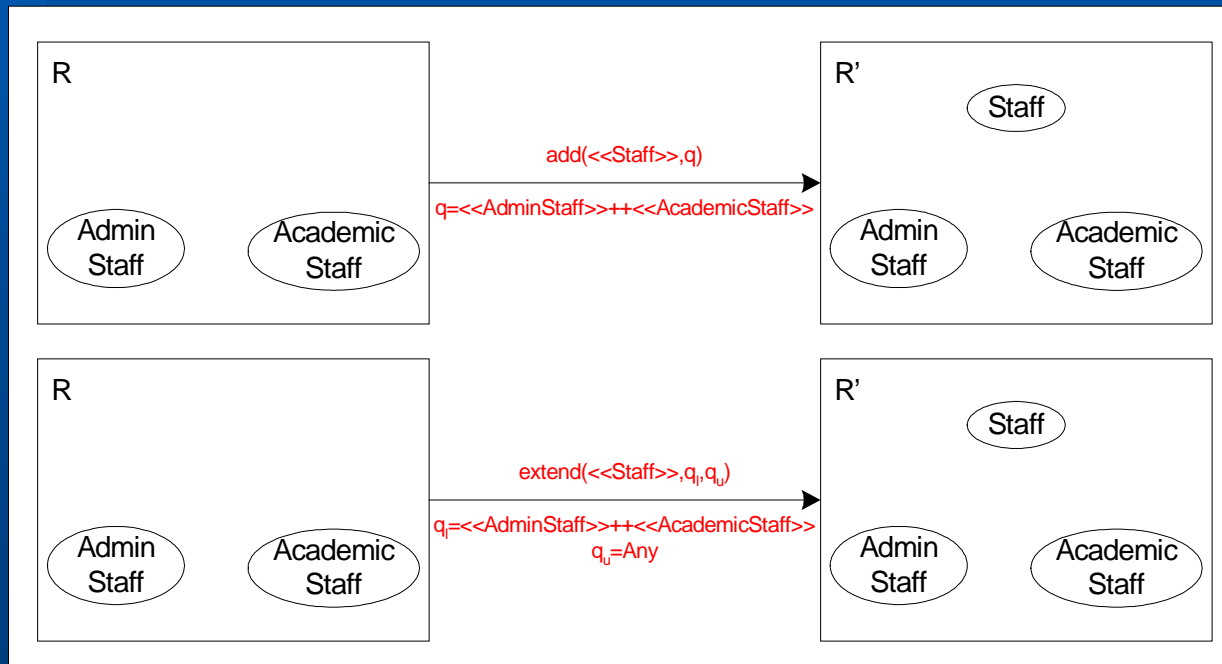
Both-As-View Approach (BAV)

- Schema-based transformation approach
- Automatically derivable reverse transformations
- GAV & LAV derivation from BAV pathways
- Comparison with LAV, GAV and GLAV (DBIS'04)



Both-As-View Approach (BAV)

- add/extend
- delete/contract
- rename

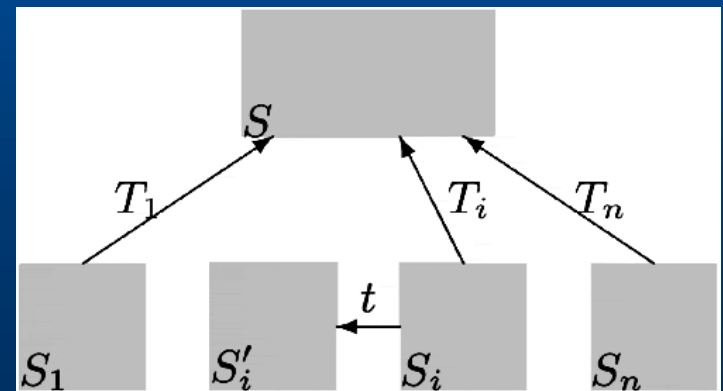
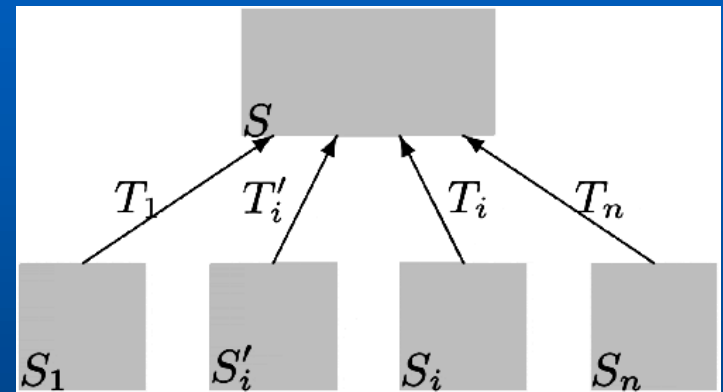


Both-As-View Approach (BAV)

- **Hypergraph-Data-Model (HDM)**
 - Consists of nodes, edges and hyper-edges
 - Low-level to better represent high-level modeling languages

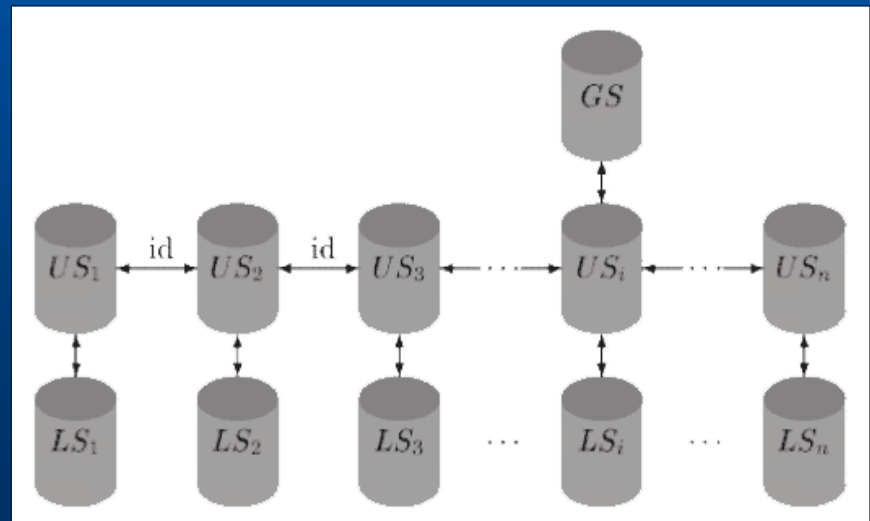
Both-As-View Approach (BAV)

- Readily supports evolution of both global & local schemas (CAiSE'02)
- Data warehousing



Both-As-View Approach (BAV)

- General integration scenario
 - Transform LS_i to US_i , using automatically reversible transformations
 - id transformations
 - Create GS from arbitrary US_i



Overview

- **Objective: restructuring & integration of XML files**
 - Schema matching process assumed
- **Motivation**
 - Interoperability
 - Related work on relational databases
 - Need for XML-specific solutions

Example Applications

- **Web services**
- **XML-enabled applications**
- **XML messaging**
- **P2P & Grid applications**

Problems

- **Same information can be represented in many different ways**
 - Ancestor – descendant \leftrightarrow different branches
 - Elements & attributes not clearly distinguished in XML model
 - Ordering policy

Aims

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

A Schema Type For XML

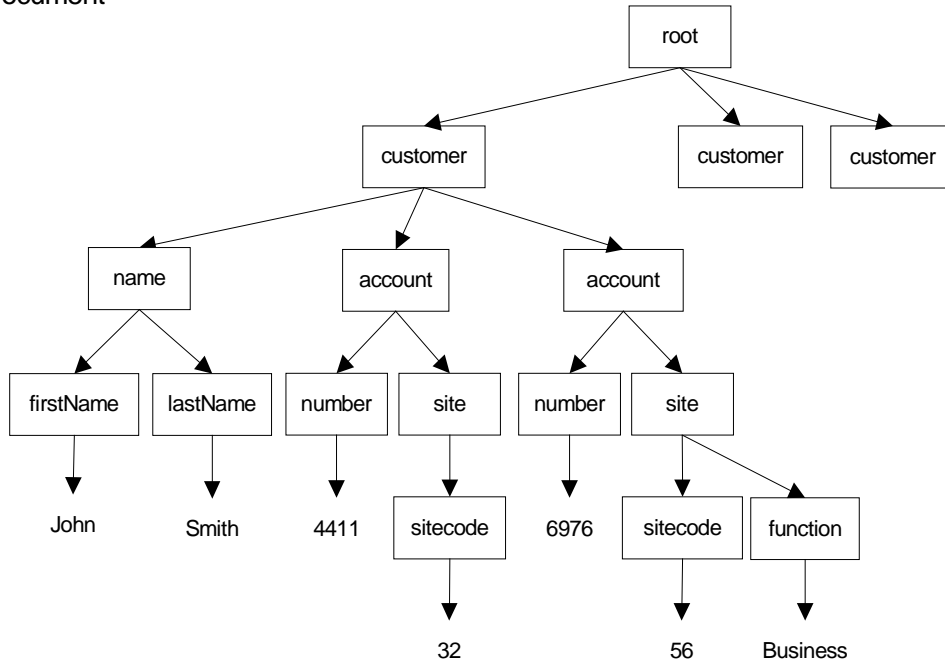
- **DTD**
 - Advantage: wide adoption
 - Disadvantages:
 - Non-XML format
 - Grammar
- **XML Schema**
 - Advantage: XML format
 - Disadvantages:
 - Grammar
 - Unnecessary complexity

XML DataSource Schema (1/3)

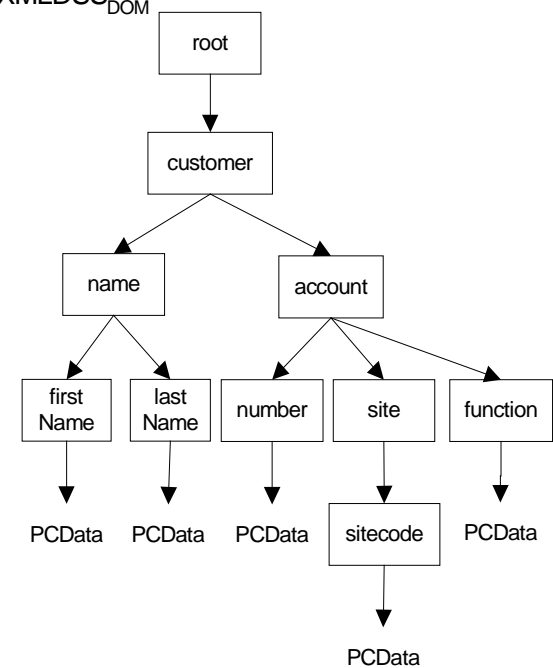
- **Basic characteristics:**
 - **Structure-only representation**
 - **XML format → ease of traversal & manipulation**
 - **Automatically derived from an XML file**
 - **XMLDSS from other schema types (DTD, XML Schema)**

XML DataSource Schema (2/3)

Document



XMLDSS_{DOM}

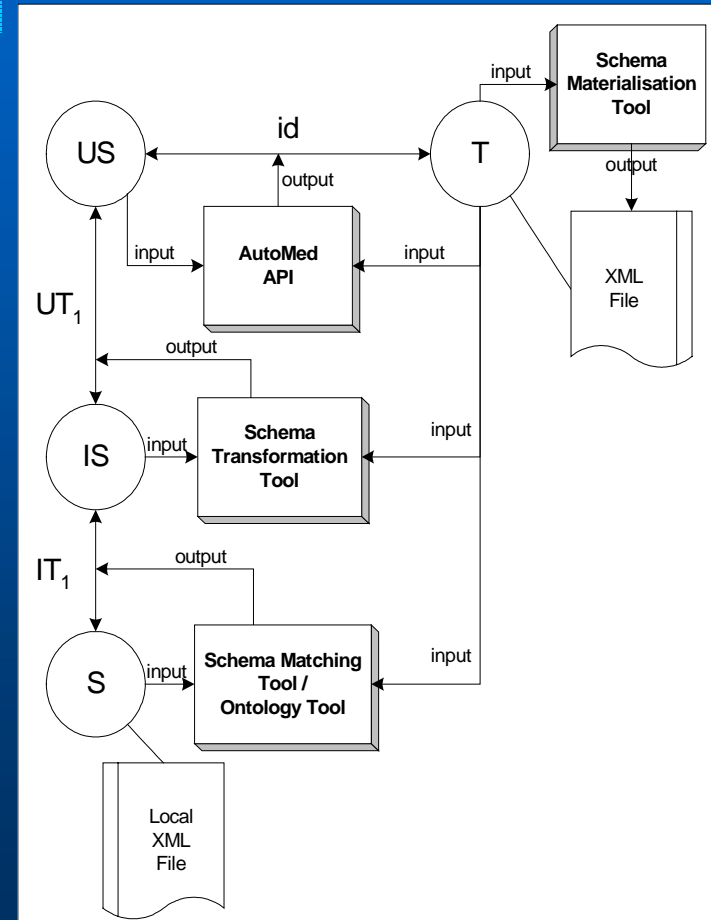


XML DataSource Schema (3/3)

- **DataGuides are rooted directed graphs**
 - XMLDSS more easily traversed & manipulated
 - DataGuides contain cycles → non-XML transformations

Restructuring Scenario

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

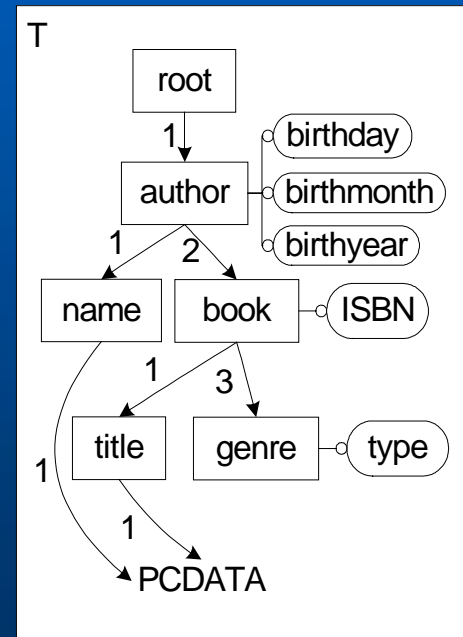
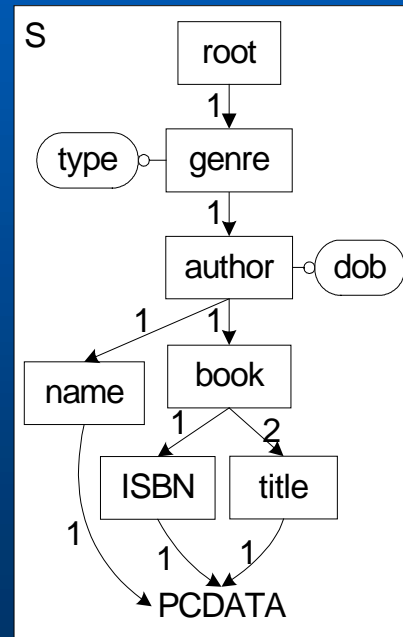


Schema Matching (1/2)

- **Types:**
 - 1-1, 1-n, n-1, n-m
 - Subset, superset, equivalence
- **Use schema matching output to create the intermediate schemas used by the schema restructuring / schema integration algorithms**

Schema Matching (2/2)

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



Schema Restructuring

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

Restructuring Algorithm

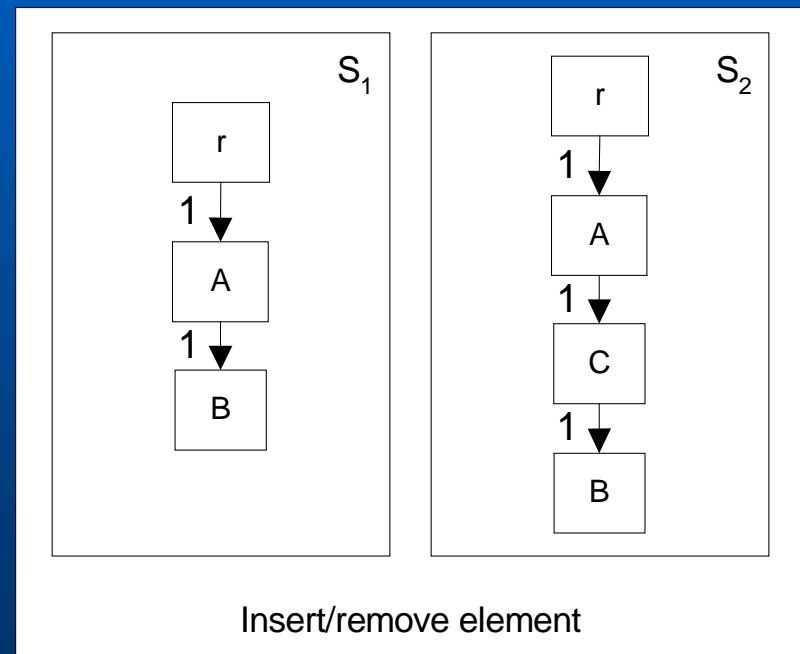
- Growing phase: traverse the target schema and issue an add/extend transformation for every construct that does not exist in the source schema.
- Shrinking phase: traverse the source schema and issue an delete/contract transformation for every construct that does not exist in the target schema.
- Completeness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

Transformation Types

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

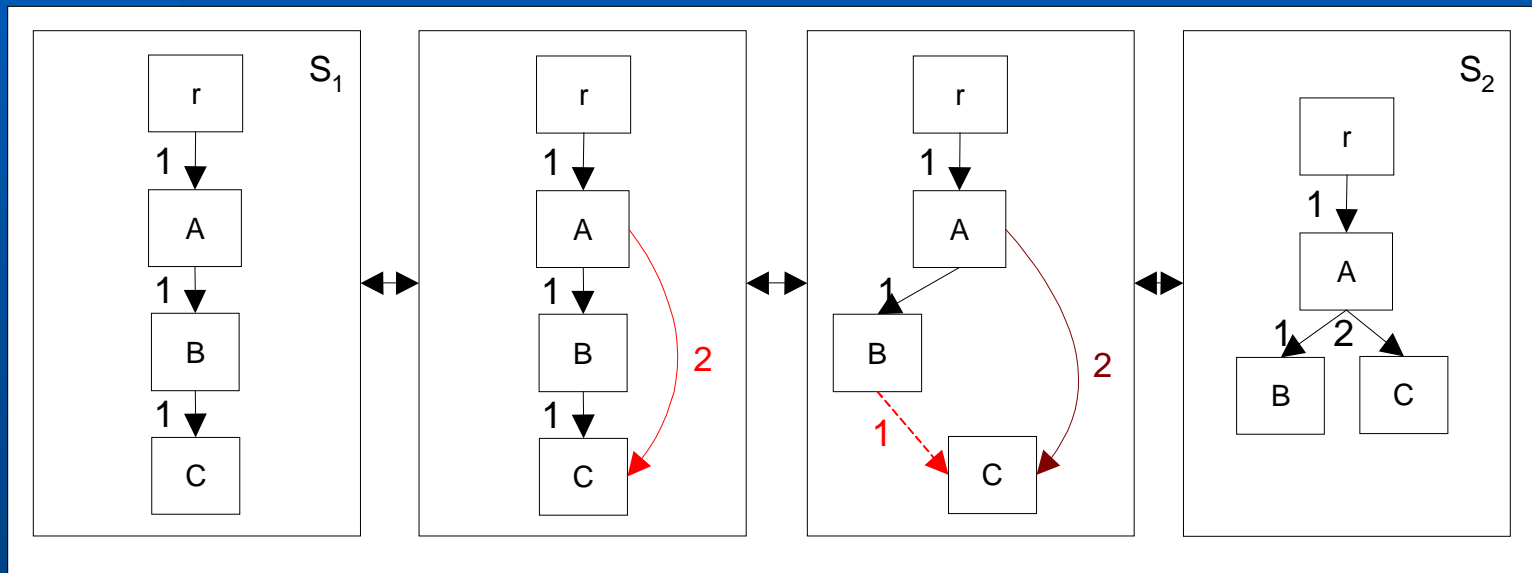
Example 1

- **Insert element C**
 - `ext(<C>,Void,Any)`
 - `ext(<A,C>, Void,Any)`
 - `ext(<C,B>, Void,Any)`
 - `del(<A,B>,q)`
- **Remove element C**
 - `add(<A,B>,q)`
 - `con(<C>, Void,Any)`
 - `con(<C,B>, Void,Any)`
 - `con(<A,C>, Void,Any)`



Example 2

- Insert/remove edge: move operation



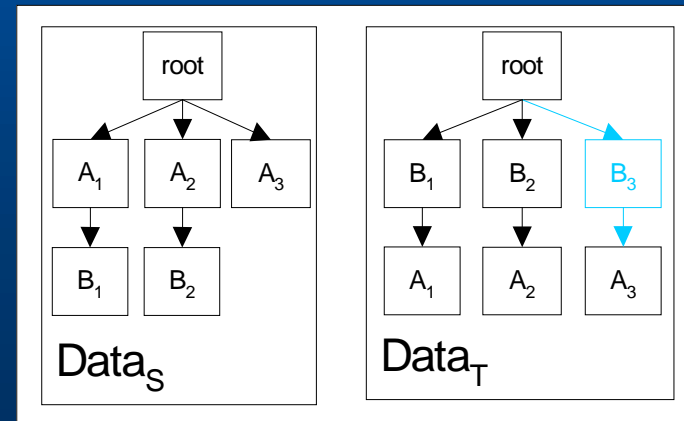
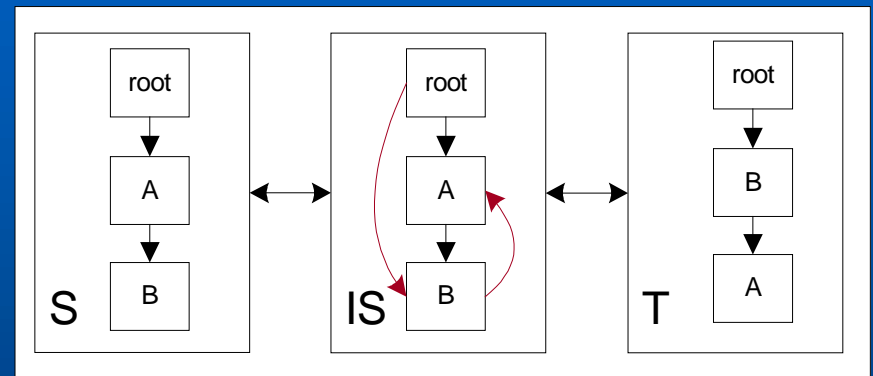
Example 3

- **Move:**

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

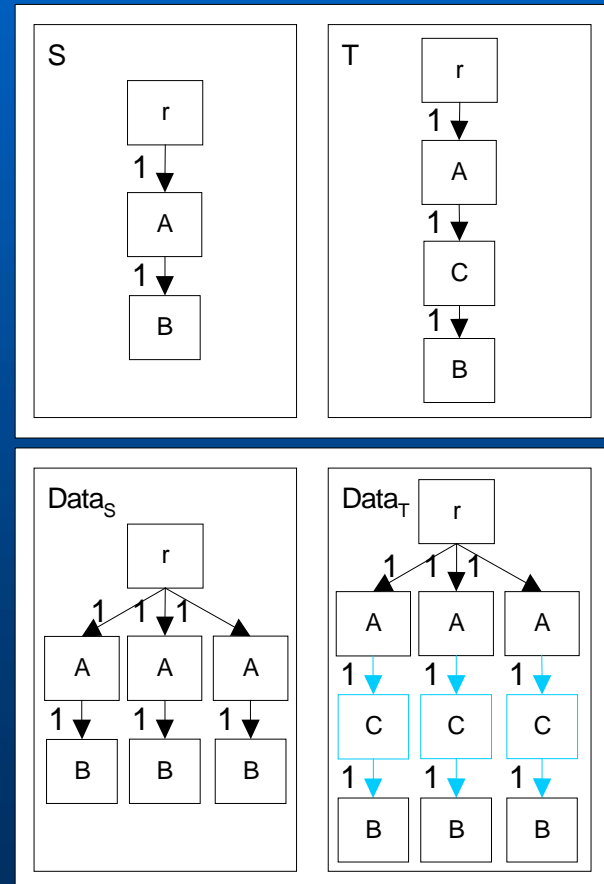
- **Complete:**

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



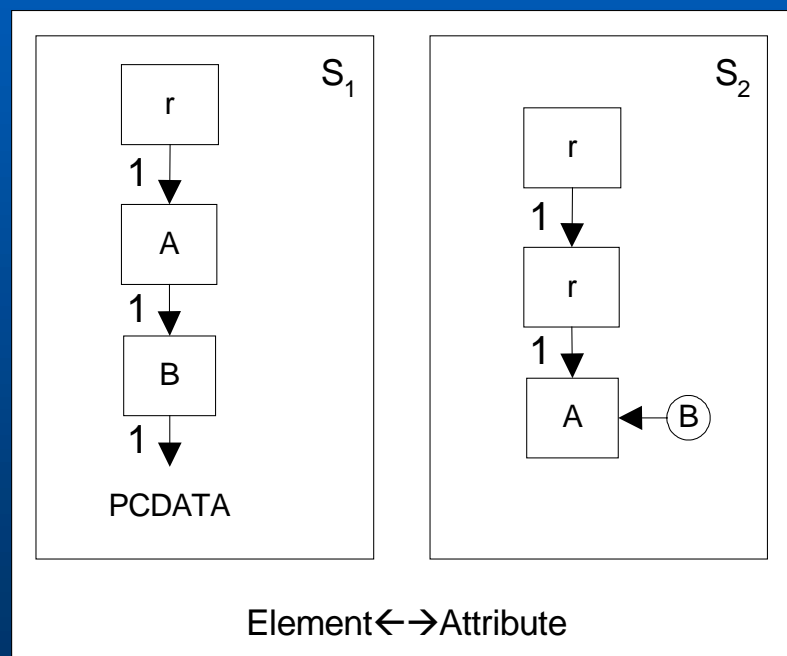
Example 1 - revisited

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



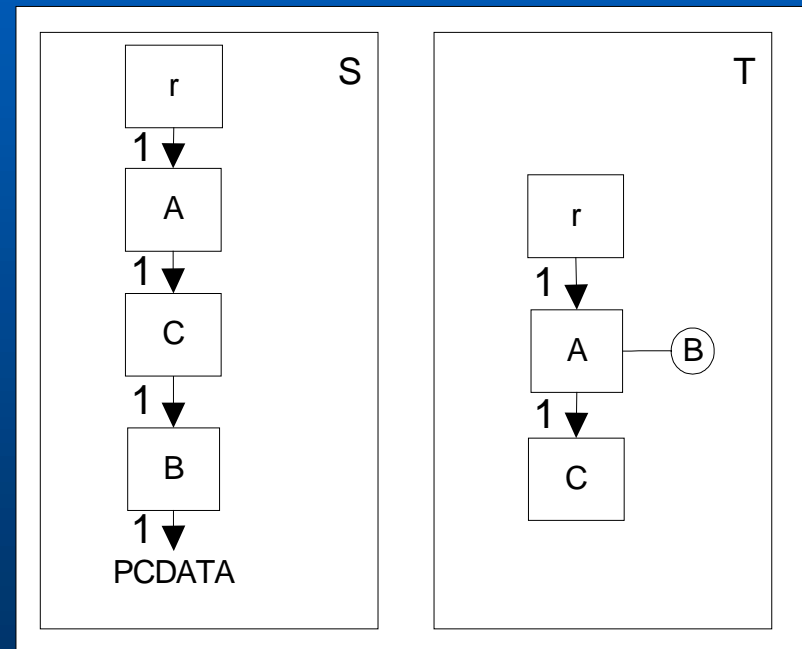
Example 4

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



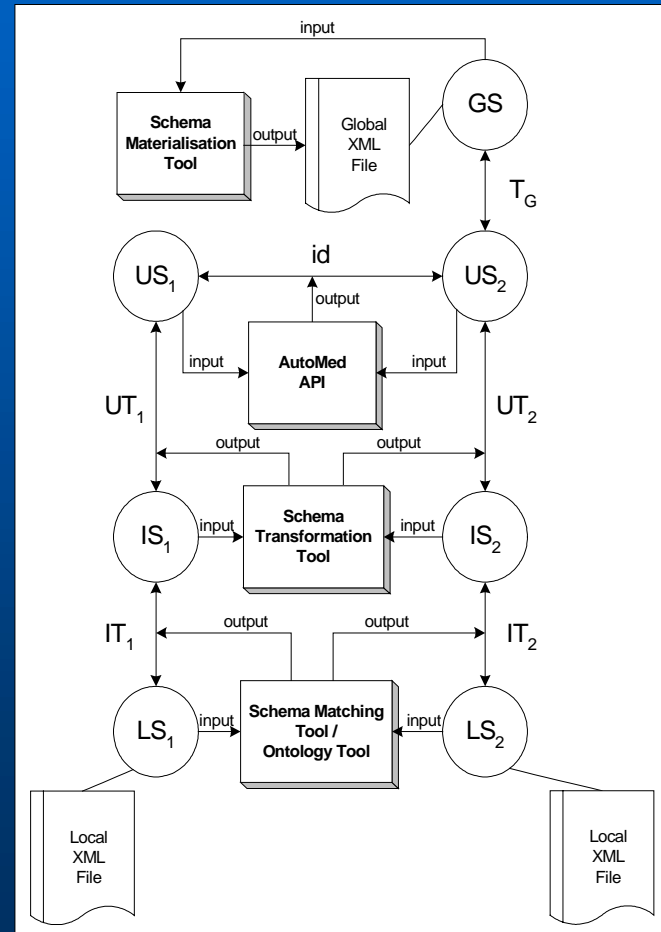
Example 5

- $\text{size}(\text{extent}(\langle B \rangle)) > \text{size}(\text{extent}(\langle A \rangle))$
- Complete $\langle A \rangle$

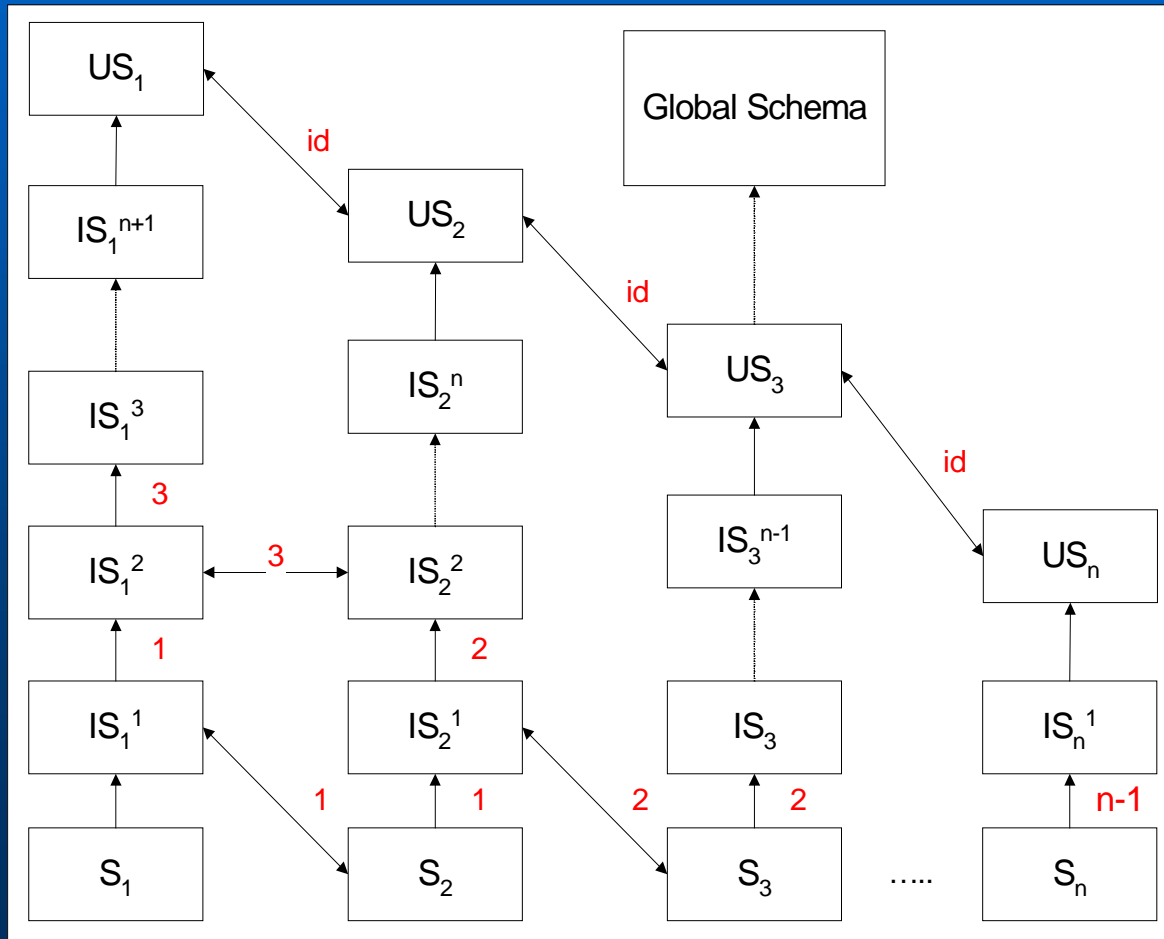


Integration Scenario

- Schema matching phase
- Schema transformation phase
- id phase
- Global schema materialisation

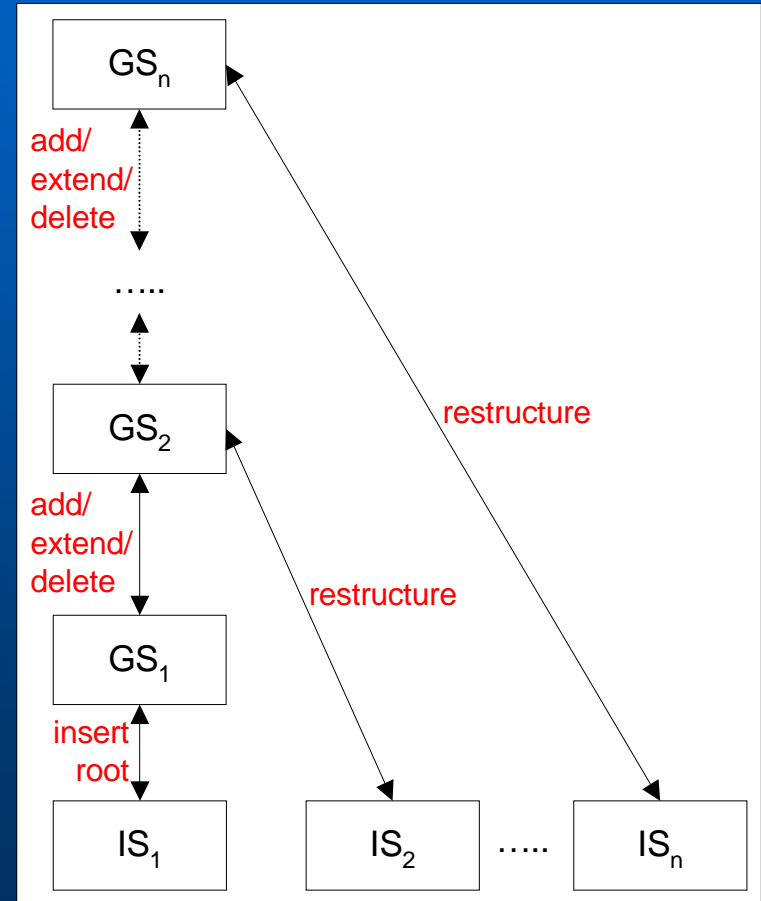


Schema Integration – Type I



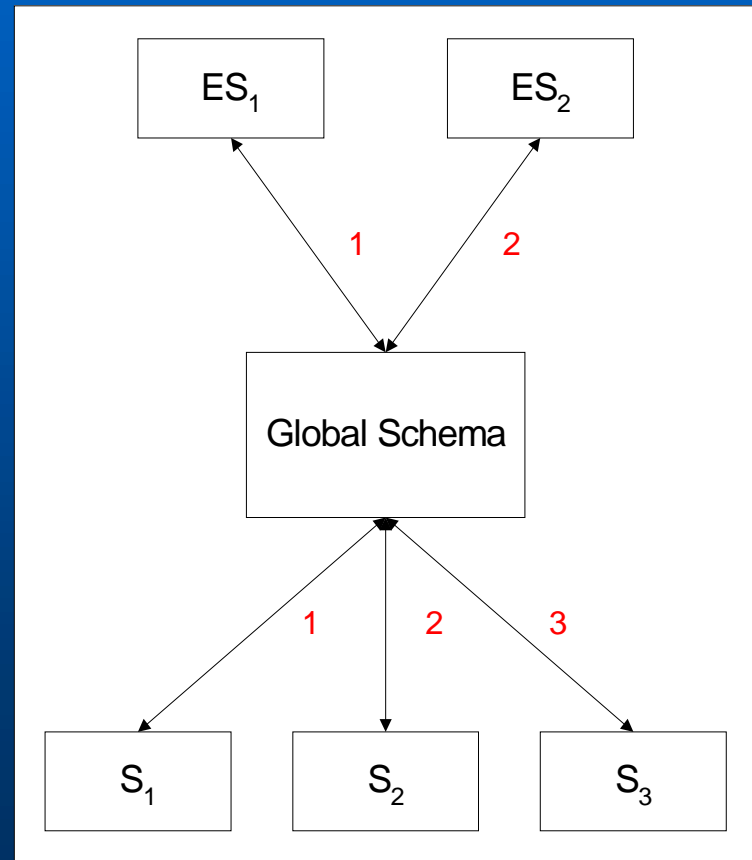
Schema Integration – Type II

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



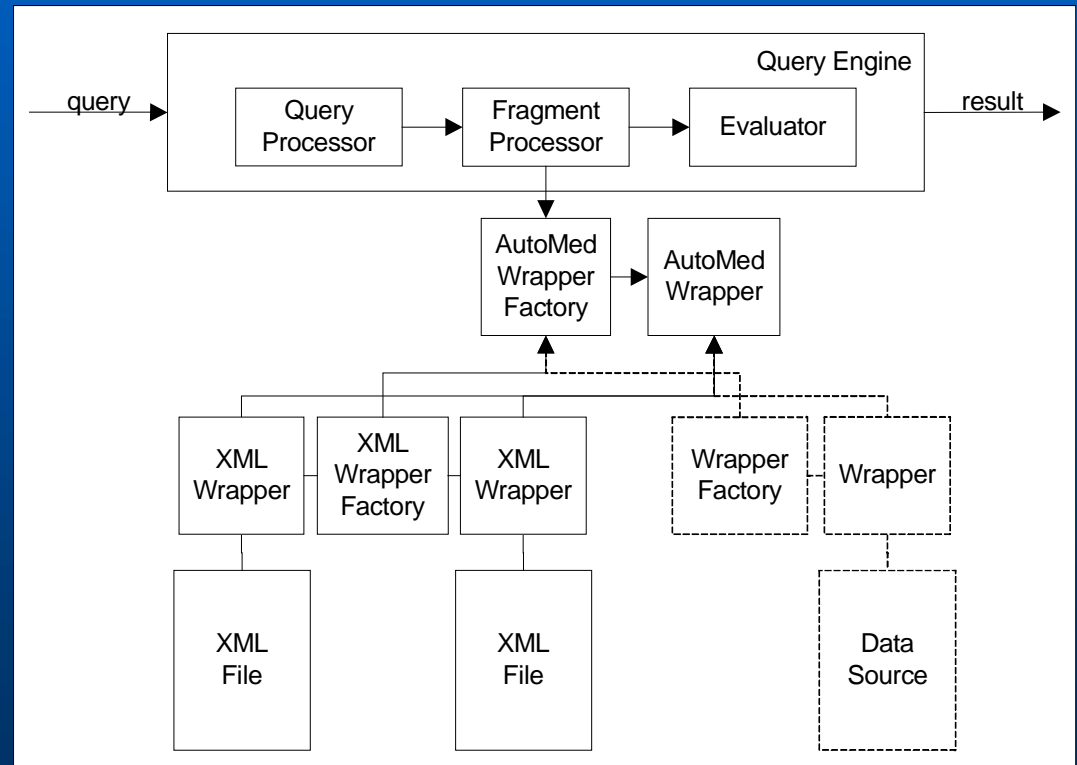
Schema Integration – Type II

- Improve GS as a second step



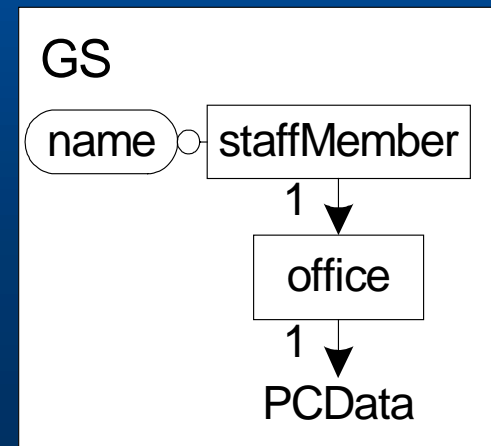
Querying

- IQL query language
- Wrappers:
 - DOM (XPath)
 - SAX
- Wrapper factories contain model specific switches



Materialisation

- **Strategy:**
 - Materialise root and its attributes
 - Consider all edges (e_p, e_c) in a depth-first way
 - Materialise e_c and its attributes
- **Root with multiple instances?**



Conclusions

- **XML specific solution:**
 - element \leftrightarrow attribute transformations
 - move operation
- **No loss of data by synthetically creating missing structure**

Evaluation

- **BIOMAP**

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

- **ISPIDER**

- Pilot Grid for integrative proteomics
- Extend AutoMed for Grid applications

Ongoing Work - Research

- **Include more types of XML data sources:**
 - XML databases
 - Native XML databases
- **Modify restructuring algorithm to accept input**
 - AutoMed's schema matching tool
 - Ontologies (RDFS/OWL)

Future Work

- **GS improvement**
- **Overlapping data identification**
- **Targeted schema evolution**
- **Targeted rematerialisation of GS**
- **Streaming integration/materialisation**

Resources

- <http://www.doc.ic.ac.uk/automed>
 - Publications & technical reports
 - AutoMed distribution