Relational Wrapper for Navigation-Driven Lazy Mediator

Michail Petropoulos

Database **L**ab/**CSE D**EPARTMENT
San **D**iego **S**upercomputer **C**enter

University of California
San Diego

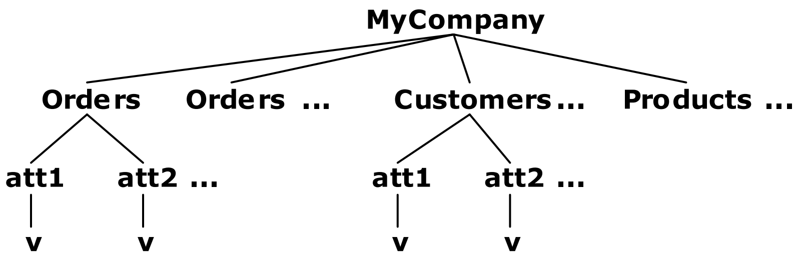
NPACI SDSC

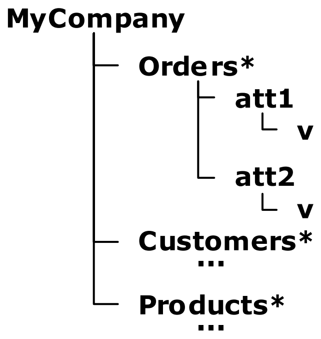
Outline

- Generic X-View (GX-View)
  - Representing a relational database using an XML document
- X-Views
  - How to build a custom view for relational databases
- Architecture of the Relational Wrapper
  - The role of the assistant mediator
- Query Processing
  - Translating XMAS queries to SQL queries
- Future Directions
  - Constructing results for the Lean XML Fragment/LXP Protocol
- MIX Architecture
  - Virtual XML Document/VXD Architecture
- XMAS Algebra
  - Query language for XML data
- Rewriting Rules

Generic X-View (GX-View)

Relational DB as an XML Document

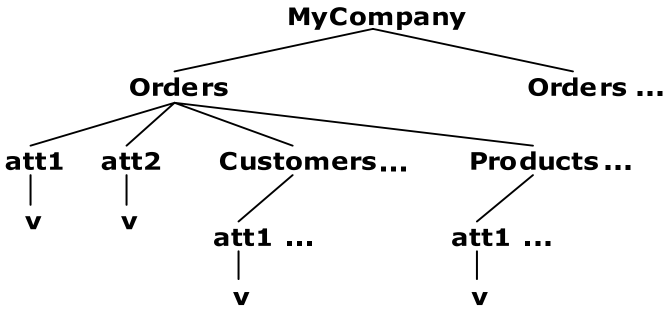


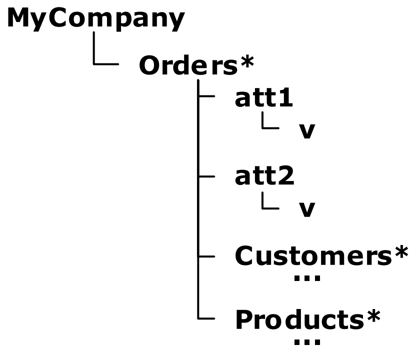


- The root of the XML document is the name of the RDB
- Tuples are represented as subtrees, that contain attributes and values, and are labeled after the the relation they belong to

X-Views

Custom view for Relational DB

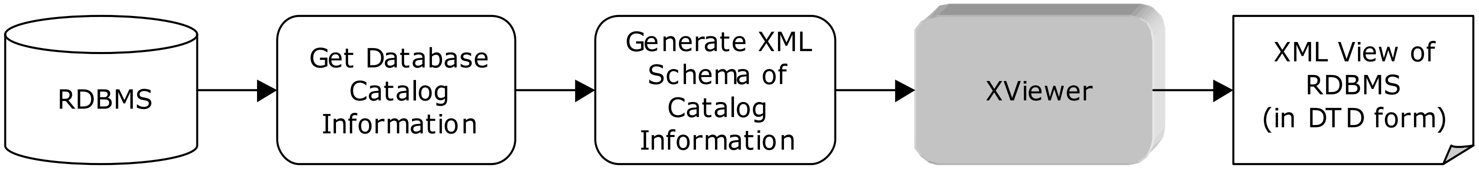




- The mediator engineer would like to choose a custom view of the RDB, other than the generic one, which would make more sense to the mediator’s client and would represent better the semantics of the relational schema
- The client may use the DTD of this XML representation to construct a query via BBQ

X-Views (cont.)

X-Viewer

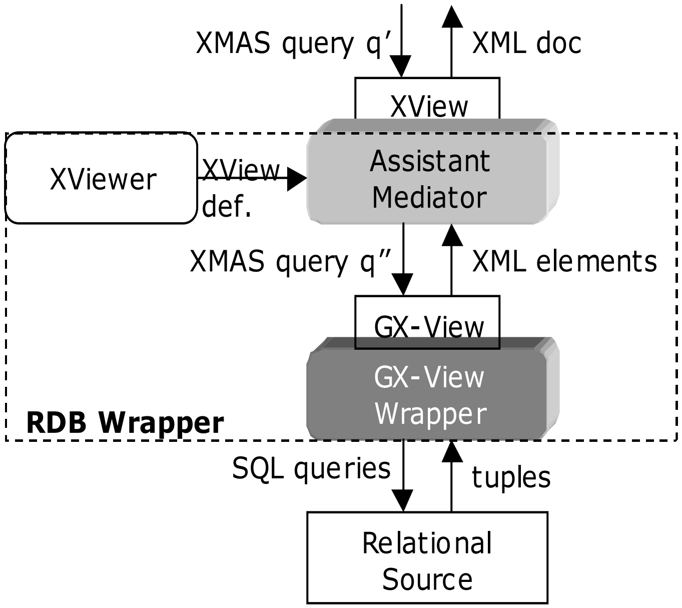

RDBMS
Get Database Catalog Information
Generate XML Schema of Catalog Information
XViewer
XML View of RDBMS (in DTD form)

The X-Viewer is the tool that generates possible XViews (DTDs) to the mediator engineer when the following three methods are applied to the relational schema:

- *MaxIndegree*: The relation which has the highest incoming edges (mostly referenced) is chosen as the starting relation.
- *ZeroIndegree*: The relation which has no incoming edges (many N:M relationships) is chosen as the starting relation
- *User Defined*: The user enters the name of the relation to start with as a parameter

Architecture for RDB Wrapper

The Role of the Assistant Mediator

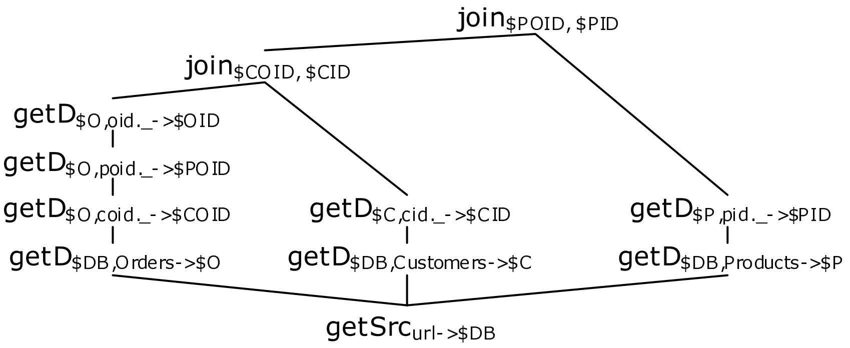


- Modular architecture leverages mediator technology
- GX-View wrapper supports queries against the GXView
- Assistant mediator translates a query *q′* against an XView to a query *q″* against the GXView
- The XView definition is imported to the assistant mediator

Query Processing by GX-View Wrapper

Translating XMAS queries to SQL

- GX-View wrapper accepts only the body part of a XMAS query
- Example: The body of the view definition in XMAS algebraic form for the above XView is the following:



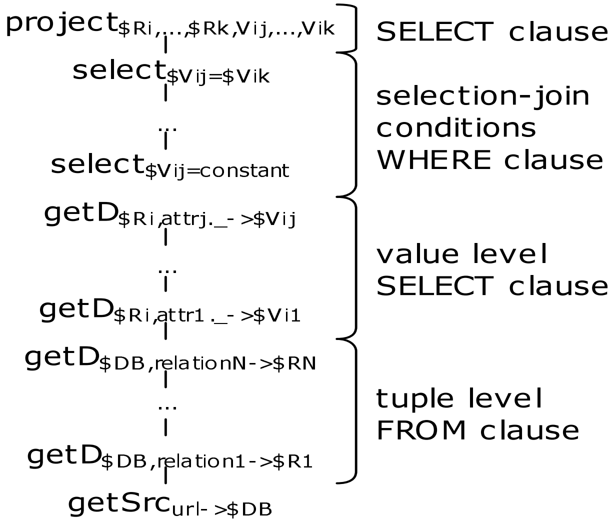
The list of bindings that this body produces are generated from the following SQL query:

```
SELECT *
FROM Orders AS $O, Customers AS $C, Products AS $P
WHERE coid = cid AND poid = pid
```

Query Processing by GX-View Wrapper (cont.)

Translating XMAS queries to SQL

- In general, GX-View wrapper accepts XMAS queries which conform the query pattern below:



$\text{project}_{\$Ri,\ldots,\$Rk,Vij,\ldots,Vik}$

$\text{select}_{\$Vij=\$Vik}$

...

$\text{select}_{\$Vij=constant}$

$\text{getD}_{\$Ri,attrj.\_->\$Vij}$

...

$\text{getD}_{\$Ri,attr1.\_->\$Vi1}$

$\text{getD}_{\$DB,relationN->\$RN}$

...

$\text{getD}_{\$DB,relation1->\$R1}$

$\text{getSrc}_{url->\$DB}$

SELECT clause

selection-join conditions WHERE clause

value level SELECT clause

tuple level FROM clause

The list of bindings for this query pattern can produced by executing the the following SQL query:

SELECT $Ri.*, …, $Rk, Vij, …, Vik

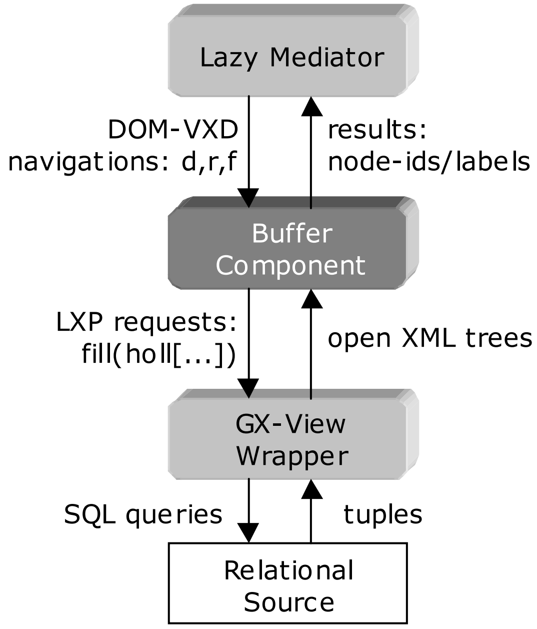
FROM relation1 AS $R1, …, relation2 AS $R2

WHERE Vij = constant AND … AND Vij = Vik

The project operator on top eliminates the (local) variables not needed in the rest of the XMAS query

Future Directions

Constructing Results for the Lean XML Fragment/LXP Protocol



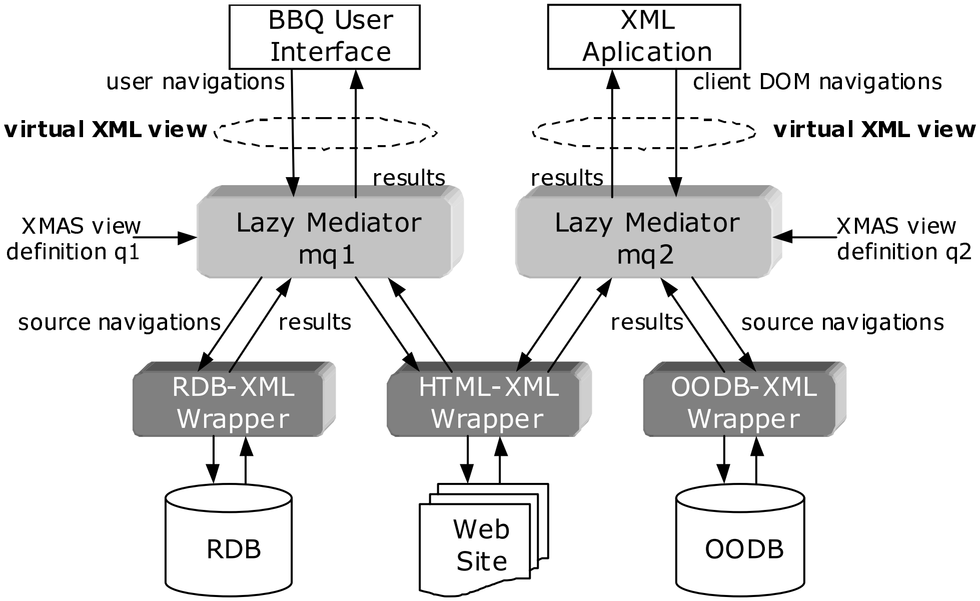
- The lazy mediator uses a buffer component to handle the different granularities of the sources
- The LXP protocol is used to extract (groups of) XML elements from the sources as needed from the DOM commands
- The fill request causes the GX-View wrapper to produce a list of bindings from a predefined number of tuples, and send them to the buffer in XML format

References

- *"Navigation-Driven Evaluation of Virtual Mediated Views"*, B. Ludäscher, Y. Papakonstantinou, P. Velikhov
- *"XML-Based Information Mediation with MIX"*, C. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov
- *"XViews: XML views of relational schemas"*, C. Baru
- *"The Lorel Query Language for Semistructured Data"*, S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Wiener
- *"Optimizing Queries across Diverse Data Sources"*, L. Haas, D. Kossmann, E. Wimmers, J. Yang
- *"A Query Translation Scheme for Rapid Implementation of Wrappers"*, Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, J. Ullman
- *"Object Fusion in Mediator Systems"*, Y.Papakonstantinou, S. Abiteboul, H. Garcia-Molina

MIX Architecture

Virtual XML Document/VXD Architecture



- The mediator composes the client’s query with the view definition and produces an algebraic plan
- Queries are executed against the wrappers

- Client gets back a virtual answer document
- Client uses navigation commands to browse the document
- Client’s navigation commands are translated to source commands in order to retrieve the requested data

XMAS Algebra

Query Language for XML data

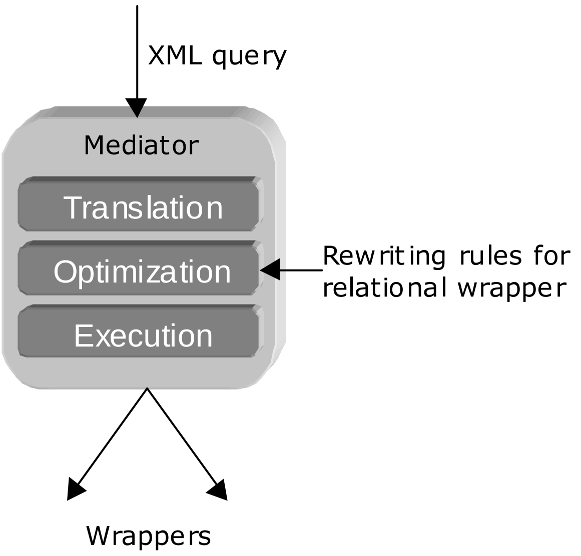
An XML QL operator takes as input a list of bindings(trees) for various element variables and produces a new one

- getDescendants$_{e,re->ch}$: Extracts the children of variable *e* using the general path expression *re* and binds them to variable *ch*
- groupBy$_{\{u1,…,uk\},u->l}$: Groups bindings of variable *u* by bindings of *u1,…,uk* and *l* is the label of the resulting list
- concatenate$_{x,y->z}$: Concatenates the bindings for variables *x* and *y* and binds them to variable *z*
- createElement$_{label,ch->e}$: For each binding of *ch* it outputs a new element labeled *label* and binds it to variable *e*
- orderBy$_{x1,…,xk}$: Orders by bindings of *x1,…,xk*
- Relational select, project, union, join and anti semi-join

Query Processing by Mediator

Rewriting Rules

Mediator's optimizer must be provided with a set of rewriting rules in order to be able to produce a plan that conforms the query pattern supported by the relational wrapper. These rules have the following form:



- Normalization of general path expressions in operators
- Joins involving one source must be expressed as selections
- Operators supported by the wrapper must pass below operators that are not (e.g. *join* under *groupBy*)