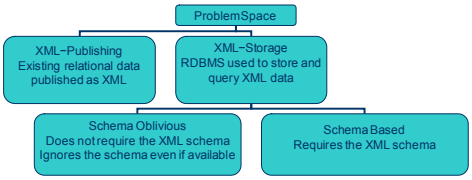
XML-to-SQL Query Translation Literature

The State of the Art and Open Problems

Problem Space



Focus of Published Solutions

		Tree Schema	Recursive Schema	
Simple Queries (path expressions)	XP	lot	XP	some
	XS/SO	lot	XS/SO	lot
	XS/SB	lot	XS/SB	some
Complex Queries	XP	lot	XP	none
	XS/SO	some	XS/SO	some
	XS/SB	none	XS/SB	**none**

Surveyed Techniques

TECHNIQUE	SCENARIO	SUBPROBLEMS SOLVED	CLASS OF XML SCHEMA CONSIDERED	CLASS OF XML QUERIES HANDLED
XPeranto	XP/GAV	VD, QT	tree	XQuery
SilkRoute	XP/GAV	VD, QT	tree	XML-QL
Rolex	XP/GAV	QT	tree	XSLT
[17]	XP/GAV	QT	tree	XSLT
[1]	XP/GAV	VD	recursive	-
Oracle XML DB	XP/GAV, XS/SB	VD, SS, QT	recursive	SQL/XML restricted XPath
SQL Server 2000/ SQLXML	XP/GAV, XS/SB	VD, SS, QT	bounded depth recursive	restricted XPath
DB2 XML Extender	XP/GAV, XS/SB	VD, QT	non-recursive	SQL extensions through UDFs
Agora	XP/LAV	QT	non-recursive	XQuery
MARS	XP/GAV + XP/LAV	QT	non-recursive	XQuery
STORED	XS/SO	SS, QT	all	STORED
Edge	XS/SO	SS, QT	all	pathexpressions
Monet	XS/SO	SS	all	-
Xrel	XS/SO	SS, QT	all	pathexpressions
[35]	XS/SO	SS, QT	all	order-based queries
Dynamic Intervals [7]	XS/SO	QT	all	XQuery
[24,32]	XS/SB	SS	recursive	-
[2, 16, 19,21, 27]	XS/SB	SS	tree	-

XP/GAV: XML Publishing Global-as-view

XP/LAV: XML Publishing Local-as-view

XS/SO: XML Storage schema-oblivious

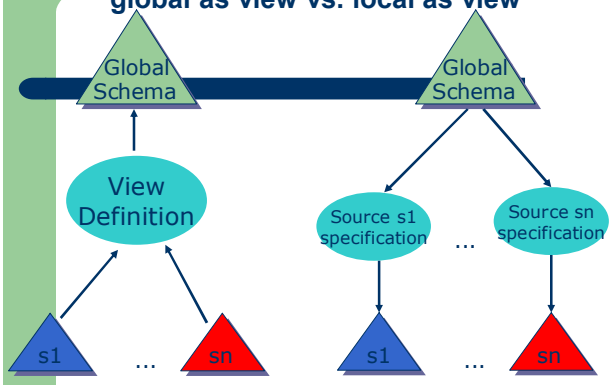
XS/SB: XML Storage, schema-based

QT: Query Translation

VD: View Definition

SS: Storage scheme

global as view vs. local as view



XML Publishing

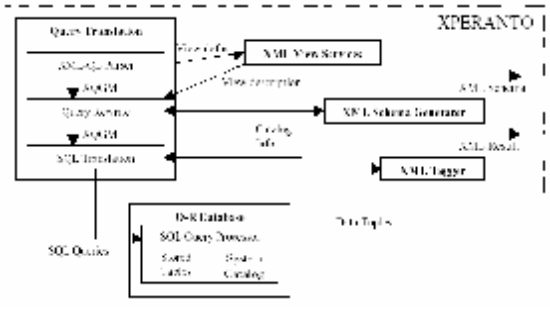
TASKS

- Defining an XML view of relational data.
- Materializing the XML view.
- Evaluating an XML query

Xperanto

- the view denition languages permit definition of tree XML views over the relational data.
- XML view is materialized by pushing down a single “outer union" query into the relational engine
- The XQuery query is converted to an XQGM representa-tion and composed with the view definition

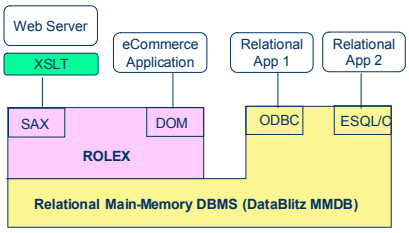
Xperanto-continued



Rolex

- the view definition languages permit definition of tree XML views over the relational data
- a view composition algorithm for composing an XSLT stylesheet with an XML view definition to produce a new XML view
- world view is changed so that a relational system provides a virtual DOM interface to the application.

Rolex- continued



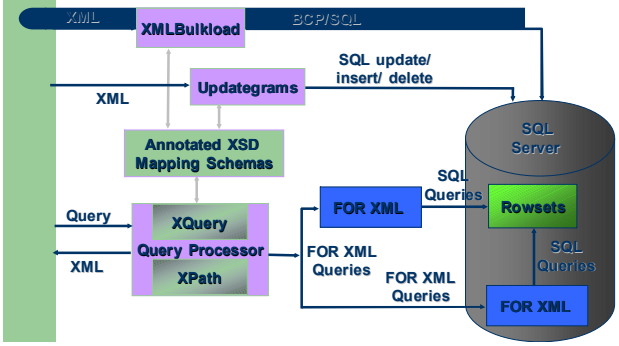
Oracle XML DB

- an annotated XSD XML schema is used to define the XML view
- supports recursive XML views
- implements the majority of the operators incorporated into the forthcoming SQL/XML standard

Microsoft SQL Server 2000/SQLXML

- an annotated XSD XML schema is used to define the XML view
- support for a limited number of depths of recursion using the max-depth annotation.
- supports the evaluation of XPath queries over the annotated XML Schema.
- The XPath query together with the annotated schema is translated into a FOR XML explicit query

Microsoft SQL Server 2000/SQLXML- continued



IBM DB2 XML Extender

- a Document Access Definition (DAD) le is used to define a non-recursive XML view.
- user-defined functions (UDFs) are provided to store and retrieve XML documents in XML columns, as well as to extract XML element or attribute values.

IBM DB2 XML Extender- continued

- A whole XML document is stored in a table column
- The DAD can identify elements and attributes to be indexed (in side tables) for fast access. SQL data type conversion is supported.
- You can retrieve the whole document or specific elements/attributes identified by XPath expressions.
- Validation on input XML documents is supported.
- Appropriate for XML documents that have irregular structure and are not frequently updated.

order_tab

Invoice	Order
356	

```
<?xmlversion="1.0"?>
<!--DOCTYPEOrderSYSTEM "Order.dtd"-->
<Orderkey="1">
 <Partkey="1">
  <Customer>37</Customer>
  <Status>O</Status>
  <Price>131251.81</Price>
  <Date>1996-01-02</Date>
  <Priority>5-LOW</Priority>
  <Clerk>Clerk#000000951</Clerk>
  <ShipPriority>0</ShipPriority>
  <Comment>firstpart</Comment>
 </Part>
</Order>
```

Example:

```
Select db2xml.extractDouble(order,
'/Order/Part/Price)  from order_tab
where invoice = 356
```


Agora

- uses the local-as-view approach (LAV), where the local source's schema are described as views over the global schema.
- describe a generic, virtual relational schema closely modeling the generic structure of an XML document.
- relational schema is then defined as views over this generic, virtual schema.

Agora

- Algorithm is two main steps:
- translating XQuery expressions into SQL on the generic view
- translating generic SQL into specific SQL

Open Problems

- Of the systems presented, only Oracle XML DB supports recursive views at all, and does not support the descendent (//) access.
- Most of the approaches do not support XQuery, or if they do, it's not complete
- performance analysis and the advantages of LAV vs GAV have not been explored

Recursive XML View Schema and Linear Recursion in SQL

- The few papers that touched upon recursion claimed that the linear recursion supported by SQL was not sufficient for the non-linear recursion in DTDs
- the authors provide a counterexample that re-opens the question

Recursive XML View Schema and Linear Recursion in SQL

- non-linear recursive DTD:
  part -> pname, part*
- underlying relational schema has two relations, Part and Subpart with the columns: (partid,pname) and (partid,subpartid)
- **WITH RECURSIVE AllParts(partid,pname,rtolpath) as (**
  select partid,pname,"
  from Part(partid,pname)
  union all
  select P.partid,P.pname,rtolpath+A.partid
  from AllParts A, Subpart S, Part P
  where S.partid = A.partid and S.subpartid = P.partid)
  select * from AllParts
- the core SQL query executes the following linear-recursive Datalog program.
  AllParts(partid,pname) <- Part(partid,pname)
  AllParts(subpartid,subpname) <- AllParts(partid,pname) Subpart(partid,subpartid)
  Part(subpartid,subpname)

Schema-Oblivious XML Storage

goal is to find a relational schema that works for storing XML documents independent of the presence or absence of a schema.

- Relational schema design: which generic relational schema for XML should be used?
- Query translation algorithms: given a decision for the relational schema, how do we translate from XML queries to SQL queries.

Schema Oblivious Storage

- STORED: generates a mapping given a semi-structured database instance. happily stores non-conforming documents into overflow tables
- Edge Approach: XML documents are viewed as graphs, with each edge represented as a tuple in a single table.
- path based approach: all elements of a path are stored in a single relation
- IBM and Oracle: entire XML document is stored as a CLOB data type, bypassing SQL.

Schema Oblivious Storage

1. Id-based: each element is associated with a unique id and the tree structure of the XML document is preserved by maintaining a foreign key to the parent.
2. Interval-based: each element is associated with a region representing the subtree under it.
3. Path-based: each element is associated with a path id representing the root-to-leaf path in addition to an interval-based or id-based representation.

Simple Path Expression Queries

- **interval-based:** evaluating simple path expressions entails performing a range join for each step of the path expression.
- **id-based solutions:** each parent-child(/) steptranslates into an equijoin
- **path-based solutions:** the path id can be used to avoid performing one join per step of the path expression.

Complex XQuery queries

- All approaches save one do not address queries more complex than relatively simple path expressions.
- The paper that does address complex query statements requires the addition of operators to SQL and modification of the relational engine.
- So this area is still open

Schema-based Storage

approaches to storing XML in relational systems that make use of a schema for the XML data in order to choose a good relational schema.

- given an XML schema (or DTD), how should we choose a good relational schema and XML-to-relational mapping.
- having chosen an XML-to-relational mapping, how should we translate XML queries into SQL.