

# *Information Integration with XML*

Chaitan Baru

Richard Marciano

{baru,marciano}@sdsc.edu

Data Intensive Computing Environments  
(DICE) Group

San Diego Supercomputer Center



National Partnership for Advanced Computational Infrastructure

San Diego Supercomputer Center



# *Members of the DICE Group*

- **Staff**
- **Chaitan Baru**
- Amarnath Gupta
- Bertram Ludäscher
- **Richard Marciano**
- Reagan Moore
- Yannis Papakonstantinou
- Arcot Rajasekar
- Wayne Schroeder
- Michael Wan
- Ilya Zaslavsky
- **Students**
- Pratik Mukhopadhyay
- Azra Mulic
- Kevin Munroe
- Paul Nguyen
- Michail Petropolis
- Nicholas Puz
- Pavel Velikhov

# *Tutorial Outline*

- **PART I**
  - Introduction to XML
  - The MIX Project
- **PART II**
  - Storing/retrieving XML documents
  - XML and GIS
  - Open Issues

# *Information Integration with XML*

## *PART I*

- **Introduction to XML**
  - HTML vs. XML
  - XML Industry Initiatives
  - DTD, Schema, Namespaces
  - DOM, SAX
  - XSL, XSLT
  - Tools
- **The MIX Project**

# Introduction to XML

- **SGML (Standard Generalized Markup Language)**
  - ISO Standard, 1986, for data storage & exchange
  - Used in U.S. gvt. & contractors, large manufacturing companies, technical info. Publishers.
  - HTML, a simple application of SGML, 1992 - 1999
  - SGML reference is 600 pages long
- **XML (eXtensible Markup Language)**
  - W3C (World Wide Web Consortium) -- <http://www.w3.org/XML/> recommendation in 1998
  - Simple subset of SGML: “The ASCII of the Web”.
  - XML specification is 26 pages long
- **Canonical XML**
  - equivalence testing of XML documents
- **SML (Simple Markup Language)**
  - No Attributes / No Processing Instructions (PI) / No DTD / No non-character entity-references / No CDATA marked sections / Support for only UTF-8 character encoding /  
No optional features

# Introduction

- **Document & Database communities converge**
  - Document community:
    - more structure is added to documents to:
      - simplify & standardize the transmission of data via documents
      - development of XML
  - Database community:
    - need to represent data with irregular structure:
      - missing data components & multiple occurrences of the same component
      - data is less constrained than in usual relational and object-oriented databases
      - development of the semistructured data model
- **Such data may include both structured and unstructured portions, and is self-describing**

# *What's this I hear about XML?*

## A spectrum of opinions:

- *"XML is the cure for your data exchange, information integration, E-commerce, [x-2-y, U name it] problems"*  
(aka »snake oil«)

## V.S.

- *"XML is nothing but syntax"*
- *"The emperor has new/no clothes"*  
(aka »nothing new under the sun«)

## ... More “Gossip” ...

- XML is just *HTML on steroids*
- XML is just “*dumbed-down*” SGML
- XML - The *Universal* Publishing Format
- XML: Time to *Re-Tool*
- X *Marks the Spot*
- XML, *the mother of all* Web application enablers
- XML *Revolution*

# Many X-cellent(?) Acronyms...

- XML (eXtensible Markup Language)
- XML Namespaces
- XML DTDs, XML Schema
- RDF (Resource Description Framework)
- XSL (Extensible Style Sheet Language)
- XPath (=XSLT  $\cap$  XPointer), XLink
- XQL, XML-QL (XML Query Language)
- XMAS (XML Matching And Structuring language)
- eXcelon, ...

=> XML++ (i.e. += X-tensions) >> just syntax

=> a family of technologies (XML extensions, tools, ... )

# *So what is XML (all about)?*

## Executive Summary:

- XML = HTML – idiosyncrasies (simplified syntax)  
+ user-definable ("semantic") tags

- Separation of data and its presentation

=> simple, very flexible data exchange format:  
semistructured data model

=> new applications:

- Information exchange (B2B), sharing (diglib), integration ("mediation"), archival, ...
- Web site management (XML+XSL stylesheets), ...

# XML Applications & Industry Initiatives

<http://www.oasis-open.org/cover/xml.html#applications>

- **Advertising:** [adXML](#) place an ad onto an ad network or to a single vendor
- **Literature:** [Gutenberg](#) convert the world's great literature into XML
- **Directories:** [dirXML](#) Novell's Directory Services Markup Language ([DSML](#))
- **Web Servers:** [apacheXML](#) parsers, XSL, web publishing
- **Travel:** [openTravel](#) information for airlines, hotels, and car rental places
- **News:** [NewsML](#) creation, transfer and delivery of news
- **Human Resources:** [XML-HR](#) standardization of HR/electronic recruiting XML definitions
- **International Dvt:** [IDML](#) improve the mgt. and exchange of info. for sustainable development
- **Voice:** [VoxML](#) markup language for voice applications
- **Wireless:** [WAP](#) (Wireless Application Protocol) wireless devices on the World Wide Web
- **Weather:** [OMF](#) Weather Observation Markup Format ([simulation](#))
- **Geospatial:** [ANZMETA](#) distributed national directory for land information
- **Banking:** [MBA](#) Mortgage Bankers Association of America --> credit report, loan file, underwriting...
- **Healthcare:** [HL7](#) DTDs for prescriptions, policies & procedures, clinical trials
- **Math:** [MathML](#) (Mathematical Markup Language)
- **Surveys:** [DDI](#) (Data Documentation Initiative) "codebooks" in the social and behavioral sciences

# XML E-commerce Initiatives

- **CommerceNet**
  - **eCo Framework** XML specs. to support interoperability among e-businesses
  - **Commerce One** Common Business Library (CBL): set of business components, docs. In DTD, XDR, SOX
  - **BizTalk** Microsoft spec. based on XML schemas
  - **cXML** (Commerce XML) -- tag-sets for e-procurement into BizTalk
- **Electronic Data Interchange (EDI)**
  - **RosettaNet** Common format for online ordering
  - **FpML** (Financial products Markup Language): sharing of financial data (interest rate & foreign exchange products)
- **Open Buying on the Internet (OBI)**
  - **OBI** high volume b2b purchasing transactions over the Internet (Office Depot, Lockheed, barnesandnoble, AX...
- **E-commerce and XML**
  - **VISA Invoices** The Visa Extensible Markup Language (XML) Invoice Specification provides a comprehensive list of data elements contained in most invoices, including: **Buyer/Supplier, Shipping, Tax, Payment, Currency, Discount, and Line Item Detail.**
- **B2B Integration**
  - **code360** XML-Broker is middleware software that manages XML based transactions
  - **Bluestone XML Suite** Enables to develop and deploy e-commerce, electronic data interchange, application integration and supply chain management applications. Bluestone XML Suite products include: XML-Server, Visual-XML, XML-Contact and XwingML.
  - **webMethods** Provides companies with integrated direct links to buyers and suppliers





## *... And Some Repercussions*

- **Lack of schema/semantics when querying the Web (HTML):**
  - *"find MN State agencies (titles, people, ...)  
where ipc\_workgroup = "Electronic Gov. Services"*
  - *"create a list of people found and (if available) their  
Information Technology interests"*

**=> HTML is inappropriate for**

- data exchange
- automation of information management  
(retrieval, manipulation, integration)

# *XML is Based on Markup*

*Markup indicates  
structure  
and semantics*

```
<information_policy_council>  
  <work_group ID="Data-issues">  
    <member_agencies>  
      <agency>MN Supreme Court</agency>  
      <agency>Land Mgt. Info. Center</agency>  
      <agency>State Archives</agency>  
    </member_agencies>  
    <charter source="DIG-IT"/>  
    <url>http://www.state.mn.us/intergov/data/index.html</url>  
    <title>Data Issues Group-Information Technology</title>  
  </work_group>  
</information_policy_council>
```

*Decoupled from  
presentation*

# Elements and their Content

element name

Element  
Content

```
<information_policy_council>
```

```
<work_group ID="Data-issues">
```

```
<member_agencies>
```

```
<agency>MN Supreme Court</agency>
```

```
<agency>Land Mgt. Info. Center</agency>
```

```
<agency>State Archives</agency>
```

```
</member_agencies>
```

```
<charter source="DIG-IT"/>
```

```
<url>http://www.state.mn.us/intergov/data/index.html</url>
```

```
<title>Data Issues Group-Information Technology</title>
```

```
</work_group>
```

element

Empty  
Element

```
</information_policy_council>
```

Character content

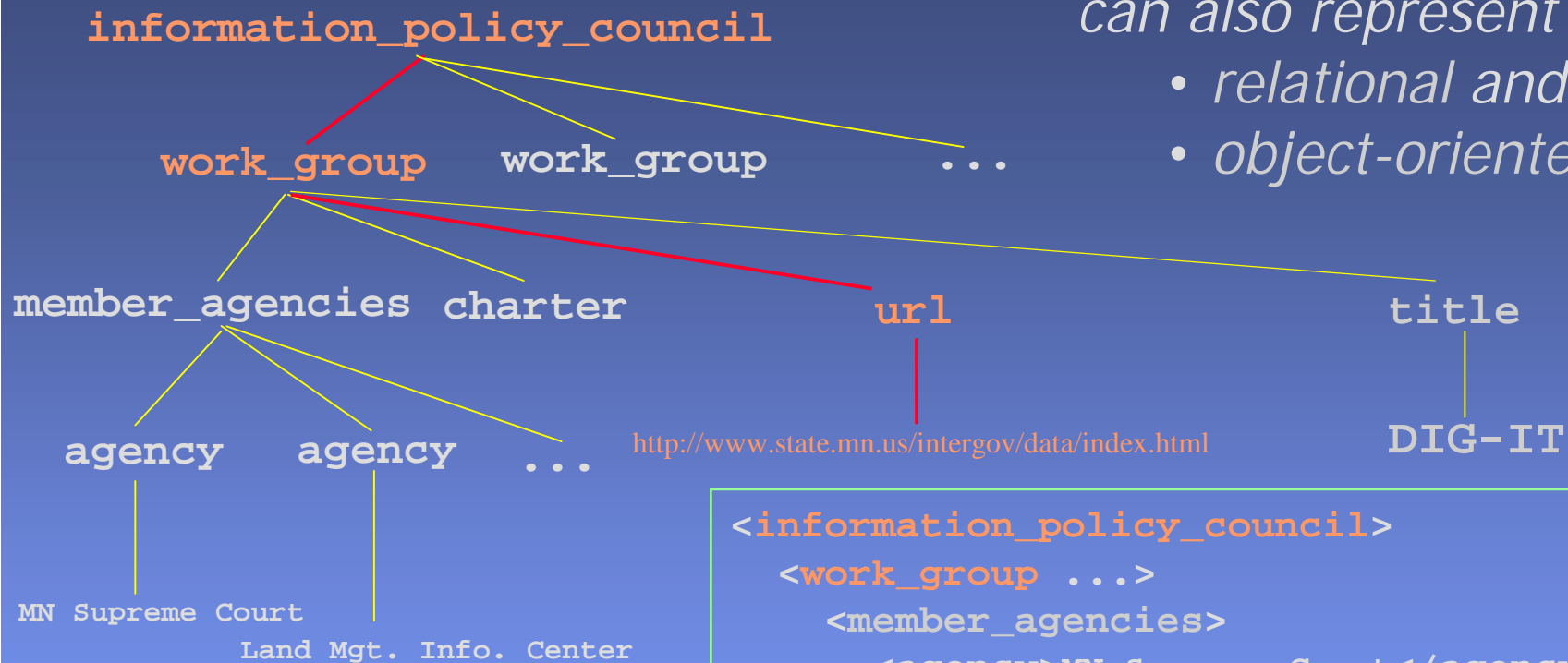
# Element Attributes

Attribute name

Attribute Value

```
<information_policy_council>  
  <work_group ID="Data-issues">  
    <member_agencies>  
      <agency>MN Supreme Court</agency>  
      <agency>Land Mgt. Info. Center</agency>  
      <agency>State Archives</agency>  
    </member_agencies>  
    <charter source="DIG-IT"/>  
    <url>http://www.state.mn.us/intergov/data/index.html</url>  
    <title>Data Issues Group-Information Technology</title>  
  </work_group>  
</information_policy_council>
```

# XML = Labeled Ordered Trees



can also represent

- relational and
- object-oriented data

≈ semistructured data  
≈ labeled trees/graphs

```
<information_policy_council>
  <work_group ...>
    <member_agencies>
      <agency>MN Supreme Court</agency>
      <agency>Land Mgt. Info. Center</agency>
      ...
    </member_agencies>
    <url>http://www.state.mn.us/intergov/data/index.html</url>
    <title>DIG-IT</title>
    ...
  </work_group>
</information_policy_council>
```

# *In Search of the Lost Structure & Semantics*

How do I learn and use the **element structure** of a document?

How do I share **structure and metadata/semantics** with my community?

How to make all this **automatable**?



# Adding Structure and Semantics

- XML Document Type Definitions (**DTDs**):
  - define the structure of "allowed" documents  
(i.e., *valid wrt. a DTD*)
  - $\approx$  database schema
  - PB: absence of data types & use of separate non-XML syntax for DTD

=> improve query formulation, execution, ...
- XML **Schema**
  - defines structure and data types
  - allows developers to build their own libraries of interchanged data types
  - Support **Namespaces** (identify your vocabulary)

# Main Schema Contenders

- **DTDs**- (=> date back to SGML)
- **XML-Data**/XML-Data Reduced (**XML-DR**)
  - Microsoft (used for BizTalk framework). Provides a large set of data types for database and program interchange
- Document Content Description (**DCD**)
  - IBM & Microsoft - Uses ideas from XML-Data and some syntax from W3C, RDF (Resource Description Framework)
- Schema for Object-Oriented XML (**SOX**)
  - Developed by Veo Systems (now CommerceOne). Provides inheritance to XML structures.
- Document Description Markup Language (**DDML**)
  - Once known as Xschema. From XML-dev mailing list (schema language with a subset of DTD)

WHAT TO DO? => **Extensibility**'s XML Authority:

tools for converting among schema formats



# *New Approaches*

- **RELAX** (REgular LAnguage description for XML)
  - Murata Makoto, formerly of Fuji Xerox
  - Standardized by INSTAC XML SWG of Japan. Under the auspices of the Japanese Standard Association(JSA), this committee develops Japanese national standards for XML. Specification for describing XML-based languages
  - A description written in RELAX is called a RELAX grammar. An XML document can be verified against a RELAX grammar.
  - Compared with DTD, RELAX has new features:
    - RELAX grammars are represented in the XML instance syntax
    - RELAX borrows rich data types of XML Schema Part 2
    - RELAX is namespace-aware

# Alternative Approaches

- Schematron

- Rick Jellife, Academia Sinica Computing Centre
- An XML Structure Validation Language using Patterns in Trees (not grammars)
- Processing is based on XSL tools. Validation produces rich reports with error reporting

- Document Structure Description (DSD)

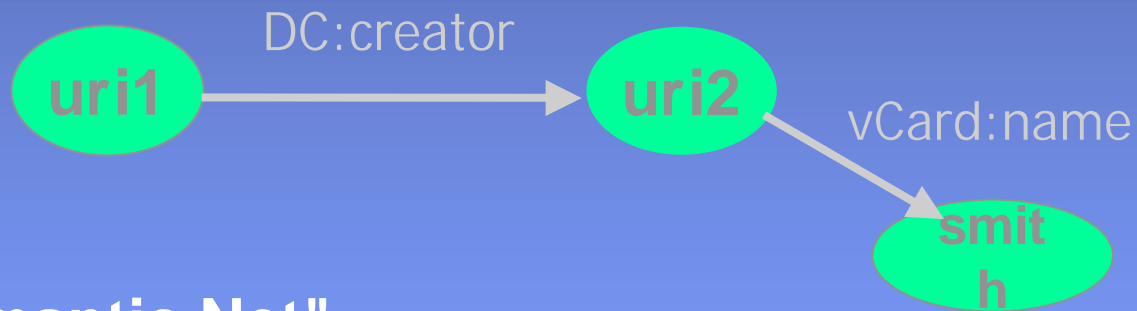
- AT&T Labs and University of Aarhus
- Not based on XSL processing. More like W3C's XML Schemas
- Allows for context-sensitive rules
- Much greater focus on default content for attributes and entities

# *More Ambitious Metadata Efforts*

- **Resource Description Framework (RDF)**
  - Metadata model
  - The designer can describe objects, add properties to define and describe them, and also make complicated statements about the objects (statements about relationships between resources).
  - The specification comes in two sections:
    - Model & Syntax (viewed as directed, labeled graphs)
    - RDF Schemas (using an XML vocabulary)

# Resource Description Framework (RDF)

- Metadata is useful for information retrieval (esp. if no other schema info or semantics is available)
- Idea: representation independent encoding of metadata as triples (**Resource**, PropertyType, **Value**):
  - (**uri1**, DC:creator, **uri2**), (**uri2**, vCard:name, **smith**), ...



- "Semantic Net"

# XML DTDs as Extended CFGs

## XML DTD

```
<!element information_policy_council          work_group*>
<!element work_group                          (member_agencies,charter?,url,title)>
<!element member_agencies                    agency+>
```

## Grammar

information_policy_council	→	work_group*
work_group	→	member_agencies charter? url title
authors	→	agency+

*lhs = element (name)*

*rhs = regular expression over elements + strings (PCDATA)*

# Document Type Definitions (DTDs)

## Define and Constrain Element Names & Structure

```
<!element information_policy_council work_group*>  
<!element work_group (member_agencies,charter?,url,title)>  
<!element member_agencies agency+>  
<!element agency (#PCDATA)>  
<!element charter EMPTY>  
<!element url (#PCDATA)>  
<!element title (#PCDATA)>  
<!attlist charter source ENTITY #REQUIRED>  
<!attlist work_group ID ID>
```

Element Type  
Declaration

Attribute List  
Declaration

# Element Declarations

*member\_agencies* followed by  
optional *charter*,  
followed by *url*,  
followed by *title*

Sequence of 0 or  
more *work\_group*

```
<!element information_policy_council work_group*>  
<!element work_group (member_agencies,charter?,url,title)>  
<!element member_agencies agency+>  
<!element agency (#PCDATA)>
```

Character content

Sequence of 1 or  
more *agency*

```
<!element charter EMPTY>  
<!element url (#PCDATA)>  
<!element title (#PCDATA)>  
<!attlist charter source ENTITY #REQUIRED>  
<!attlist work_group ID ID>
```

# Element Content Declarations

<i>Declaration</i>	<i>Meaning</i>
<element 2>	Exactly one <element 2>
R?	Zero or one instances of R
R*	Zero or more instances of R
R+	One or more instances of R
R <sub>1</sub>  R <sub>2</sub>  ... R <sub>n</sub>	One instance of R <sub>1</sub> or R <sub>2</sub> or ... R <sub>n</sub>
#PCDATA	Character content
EMPTY	Empty element
(#PCDATA e*)*	Mixed Content
ANY	Anything goes

# Attributes

```
<subcommittee ID="DIG-IT-info"> More info on DIG-IT </subcommittee>
<information_policy_council>
  <work_group ID="Data-issues" ROLE="data promotion">
    <member_agencies>
      <agency agencyRef="DIG-IT-info">
        State Archives
      </agency>
    </member_agencies>
    <charter source="DIG-IT" />
    <url>http://www.state.mn.us/intergov/data/index.html</url>
    <related work_groups= "Electronic Gvt. Services" "SNAP" />
  </work_group>
</information_policy_council>
```

Object Identity Attribute

CDATA (character data)

IDREF  
intradocument  
reference

Reference to  
external ENTITY

# Attribute Types

<i>Type</i>	<i>Meaning</i>
ID	Token unique within the document
IDREF	Reference to an ID token
IDREFS	Reference to multiple ID tokens
ENTITY	External entity (image, video, ...)
ENTITIES	External entities
CDATA	Character data
NMTOKEN	Enumerated token
NMTOKENS	Enumerated tokens
More to appear?	More types (eg, DATE) may soon be part of the standard

# *More on Attribute Declarations*

- **Attributes may be**
  - REQUIRED
  - IMPLIED (optional)
  - can have default values
  - default value may be FIXED

# *The Problem with DTDs*

- **Difficult** to write and understand
- **Programmatic** processing of their metadata is difficult
- Not **extensible**
- No support for **namespaces**
- No support for **datatypes**
- No support for **inheritance**

# XML Schema: Example

New types can be derived by **extension** or **restriction**:

```
<element name="Governor" type="personName">
```

```
<type name="personName">
```

```
  <element name="title" minOccurs="0"/>
```

```
  <element name="forename" minOccurs="0" maxOccurs="*/>
```

```
  <element name="surname"/>
```

```
</type>
```

```
<type name="extendedName" source="personName" derivedBy="extension">
```

```
  <element name="generation" minOccurs="0"/>
```

```
</type>
```

```
<type name="simpleName" source="personName" derivedBy="restriction">
```

```
  <restrictions>
```

```
    <element name="title" maxOccurs="0"/>
```

```
    <element name="forename" minOccurs="1" maxOccurs="1"/>
```

```
  </restrictions>
```

```
</type>
```

# W3C Work on XML Schemas

- Structures:
  - Specify **complex element structure** and
  - Set **constraints** on the permitted values of the content of those elements
- Datatypes:
  - Sets forth a standard of **content datatypes** and
  - Sets **rules** for generating new types from them

# Identifying Vocabularies

- My element may not be your **element**:
  - political context:
    - `<body>Jesse Ventura</body>`
    - `<body>Jesse Ventura sticker="don't touch me!"</body>`
  - **automotive** context:
    - `<body>National Automotive Parts Association</body>`
  - SGML/XML context: ....
- ⇒ use XML namespaces to identify the vocabulary

# XML Namespaces

- mechanism for globally unique tag names:

```
<h:html xmlns:xdc="http://www.xml.com/books"
        xmlns:h="http://www.w3.org/HTML/1998/html4">
  <h:head><h:title>Book Review</h:title></h:head>
  ...
  <xdc:bookreview>
    <xdc:title>XML: A Primer</xdc:title>
  ...
</h:html>
```

⇒ mix of different tag vocabularies without confusion

- namespaces only **identify** the vocabulary; additional mechanisms required for **structure** and **meaning** of tags

# Processing XML

- **Parsing**
  - without & with structure
- **API's for XML:**
  - DOM
    - **Document Object Model** for XML
    - Common API for manipulating XML document trees
  - SAX
    - **Simple API** for XML
    - No parse tree: **event-based** XML parsing!

# Parsing

- **Non-validating parser:**
  - checks that XML doc is syntactically well-formed
- **Validating parser:**
  - checks that XML doc is also valid w.r.t. a given DTD
- **Parsing yields tree/object representation:**
  - **Document Object Model** (DOM) API

# DOM Structure Model and API

- hierarchy of Node objects:
  - document, element, attribute, text, comment, ...
- language independent programming **DOM API:**
  - get... first/last child, prev/next sibling, childNodes
  - insertBefore, replace
  - getElementsByTagName
  - ...
- DOM Level 1: core functionality
- DOM Level 2: adds support for
  - namespaces, style sheets, filtering, event model, ranges

# *DOM Summary*

- **Object-Oriented** approach to traverse the XML node tree
- **Automatic processing** of XML docs
- Manipulation & **Updating of XML** on client & server
- **Database interoperability** mechanism
- **Memory-intensive**

# **SAX Event-Based API**

*from "Professional XML"*

- **Pros:**
  - The whole file doesn't need to be loaded into memory
  - Simple
  - Fast
  - Allows you to ignore less interesting data
- **Cons:**
  - Designed for reading XML docs only
  - Not supported in current browsers
  - Complex searches can be hard
  - The DTD is not available

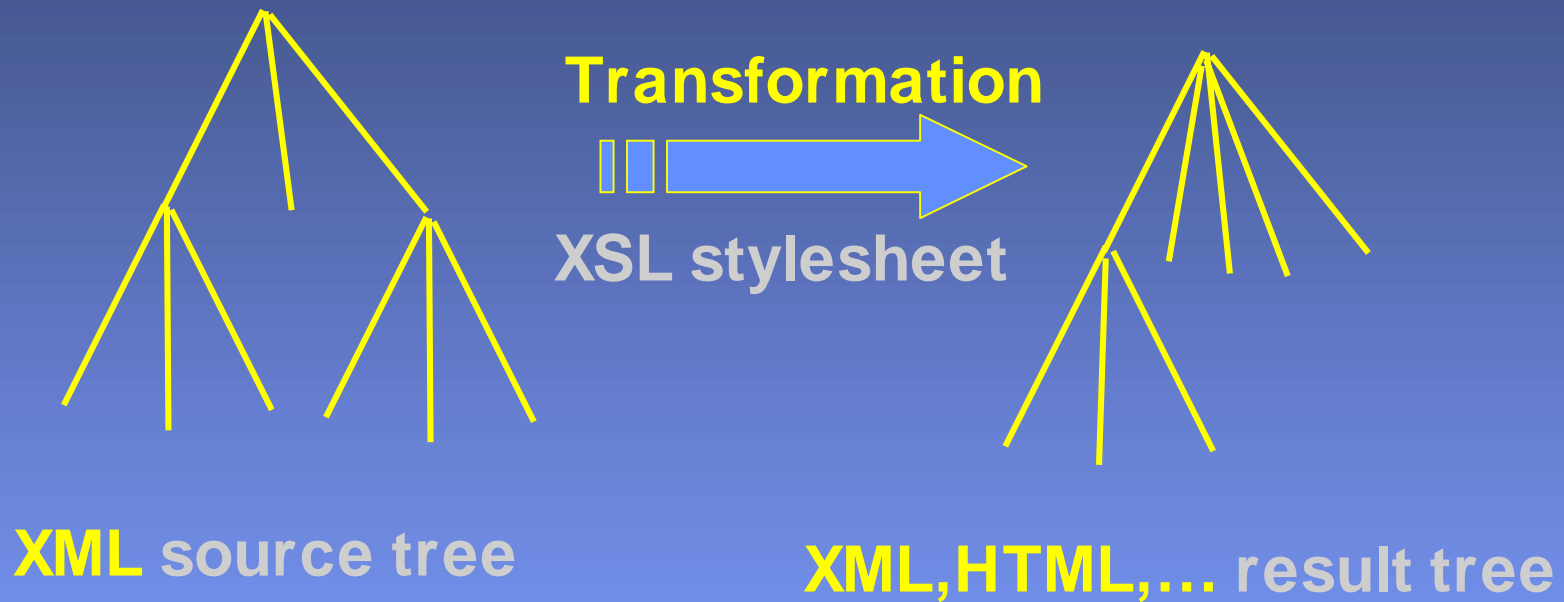
# *Presenting XML: eXtensible Stylesheet Language (XSL)*

- Why Stylesheets?
  - separation of content (XML) from presentation (XSL)
- Why not just CSS for XML?
  - XSL is far more powerful:
    - selecting elements
    - transforming the XML tree
    - content based display (result may depend on data)

# *XSL Overview*

- XSL stylesheets are denoted in XML syntax
- XSL components:
  1. a language for transforming XML documents (XSLT: integral part of the XSL specification)
  2. an XML formatting vocabulary (Formatting Objects: >90% of the formatting properties inherited from CSS)

# *XSLT Processing Model*



# *XSLT Processing Model*

- **XSL stylesheet:** collection of template rules
- **template rule:** (pattern  $\Rightarrow$  template)
- **main steps:**
  - match pattern against source tree
  - instantiate template (replace current node “.” by the template in the result tree)
  - select further nodes for processing
- **control can be**
  - program-driven ("pull": `<xsl:foreach>` ...)
  - data/event-driven ("push": `<xsl:apply-templates>` ...)

**pattern**

## Template Rule: Example

```
<xsl:template match="product">
```

```
  <table>
    <xsl:apply-templates select="sales/domestic"/>
  </table>
  <table>
    <xsl:apply-templates select="sales/foreign"/>
  </table>
```

```
</xsl:template>
```

**template**

- (i) match pattern: **process** `<product>` elements
- (ii) instantiate template: **replace** each a product with two HTML tables
- (iii) select **the** `<product>` grandchildren (“sales/domestic”, “sales/foreign”) for further processing

## Creating the Result Tree...

- Literal result elements: non-XSL elements (e.g., HTML) appear “literally” in the result tree
- Constructing elements:

```
<xsl:element name = "...">  
    attribute & children definition  
</xsl:element>
```

(similar for `xsl:attribute`, `xsl:text`, `xsl:comment`,...)

- Generating text:

```
<xsl:template match="person">  
    <p>  
        <xsl:value-of select="@first-name"/>  
        <xsl:text> </xsl:text>  
        <xsl:value-of select="@surname"/>  
    </p>  
</xsl:template>
```

# Creating the Result Tree...

- Further XSL elements for ...
  - Numbering
    - `<xsl:number value="position()" format="1 ">`
  - Conditions
    - `<xsl:if test="position() mod 2 = 0">`
  - Repetition...

# Creating the Result Tree: Repetition

```
<xsl:template match="/">
  <html>
    <head>
      <title>customers</title>
    </head>
    <body>
      <table>
        <tbody>
          <xsl:for-each select="customers/customer">
            <tr>
              <th>
                <xsl:apply-templates select="name"/>
              </th>
              <xsl:for-each select="order">
                <td>
                  <xsl:apply-templates/>
                </td>
                ...
              </td>
            </tr>
          </xsl:for-each>
        </tbody>
      </table>
    </body>
  </html>
</xsl:template>
```

# Creating the Result Tree: Sorting

```
<xsl:template match="employees">
  <ul>
    <xsl:apply-templates select="employee">
      <xsl:sort select="name/last"/>
      <xsl:sort select="name/first"/>
    </xsl:apply-templates>
  </ul>
</xsl:template>
```

```
<xsl:template match="employee">
  <li>
    <xsl:value-of select="name/first"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="name/last"/>
  </li>
</xsl:template>
```

# XML Tools

- **Products by category:** editors, parsers, converters, search engines, etc.
  - [http://www.garshol.priv.no/download/xmltools/cat\\_ix.html](http://www.garshol.priv.no/download/xmltools/cat_ix.html)
- **XML Dev. Tools:**
  - <http://www.xmlpitstop.com/xmlTools.htm>
- **IBM alphaWorks:**
  - <http://www.alphaworks.ibm.com/>
- **Tools:**
  - <http://www.xml elephant.com/pages/Tools/>
- **Products & Services:** <http://www.xml.com>
  - Document authoring / Website tools / E-Commerce / Data management
- **XML Industry Support:**
  - <http://www.oasis-open.org/cover/xmlSupport.html>
- etc...

# *Additional XML Topics*

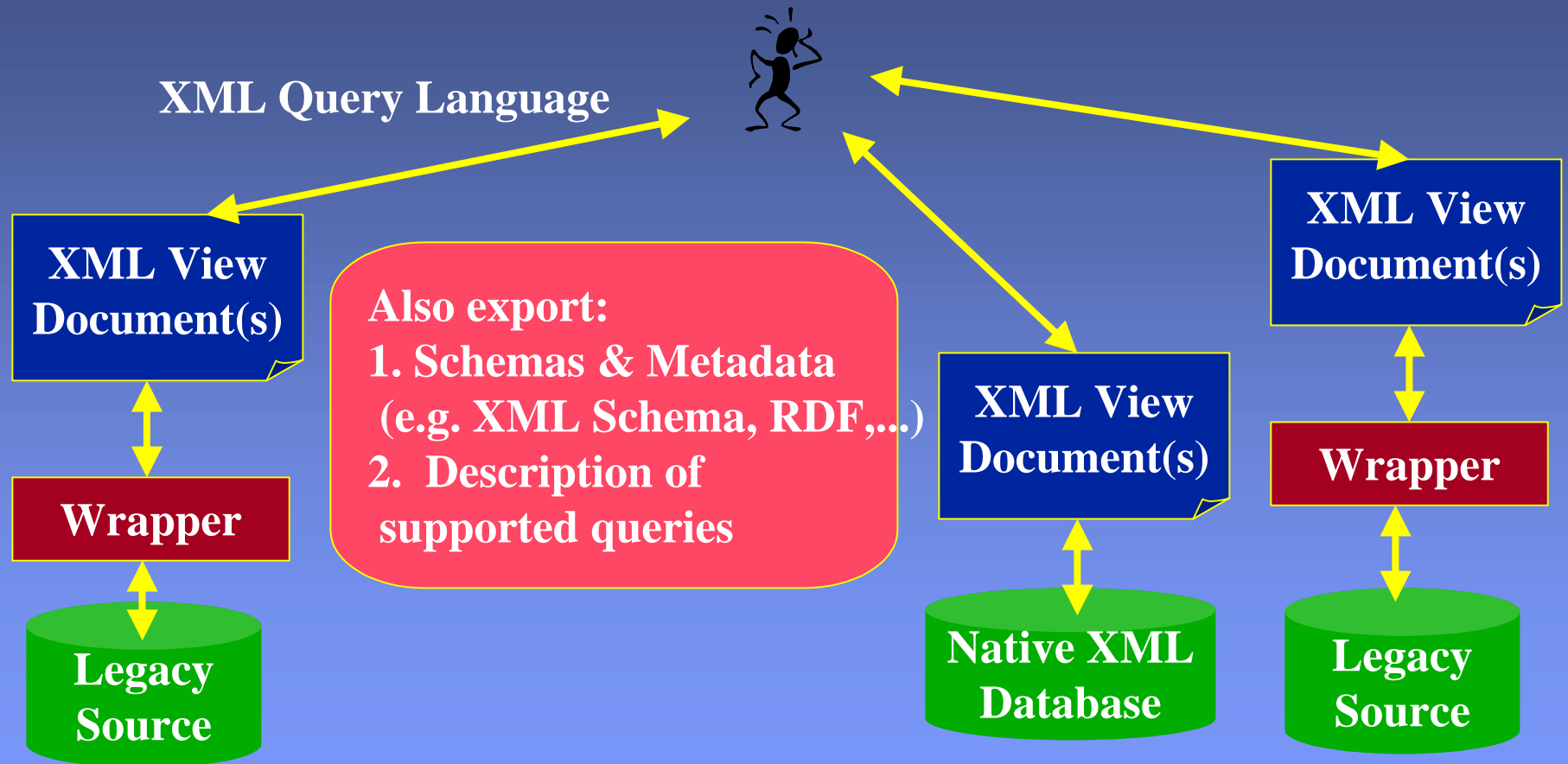
- **XHTML**
- **XML Signature**
  - digital signatures of Web resources
- **XML Fragment Interchange**
  - fragments of XML docs
- **Xlink, Xpointer**
- **etc...**

# *The MIX Project: Mediation of Information using XML*

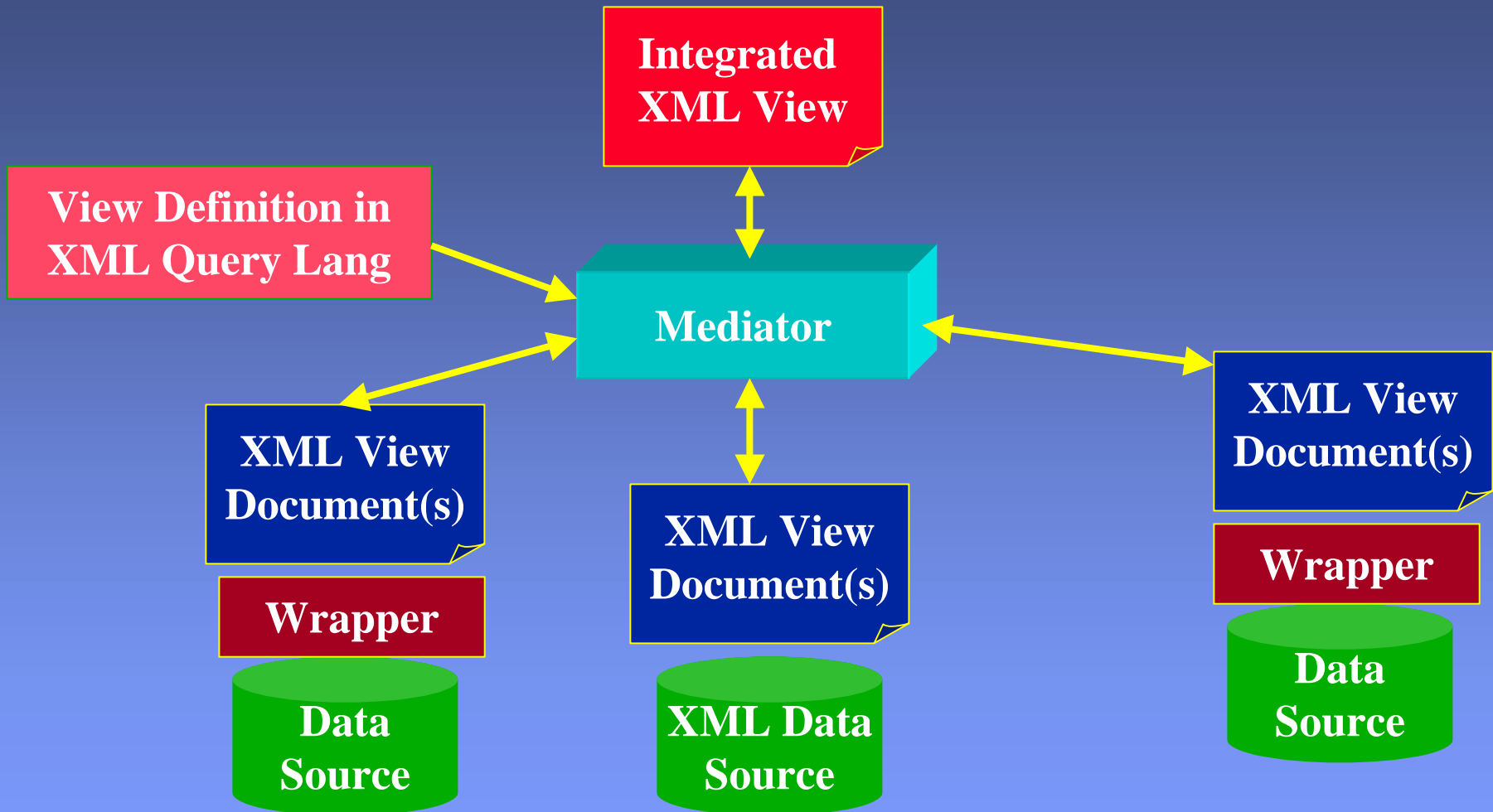
Joint effort between SDSC and the UCSD CSE  
Department

# The MIX View

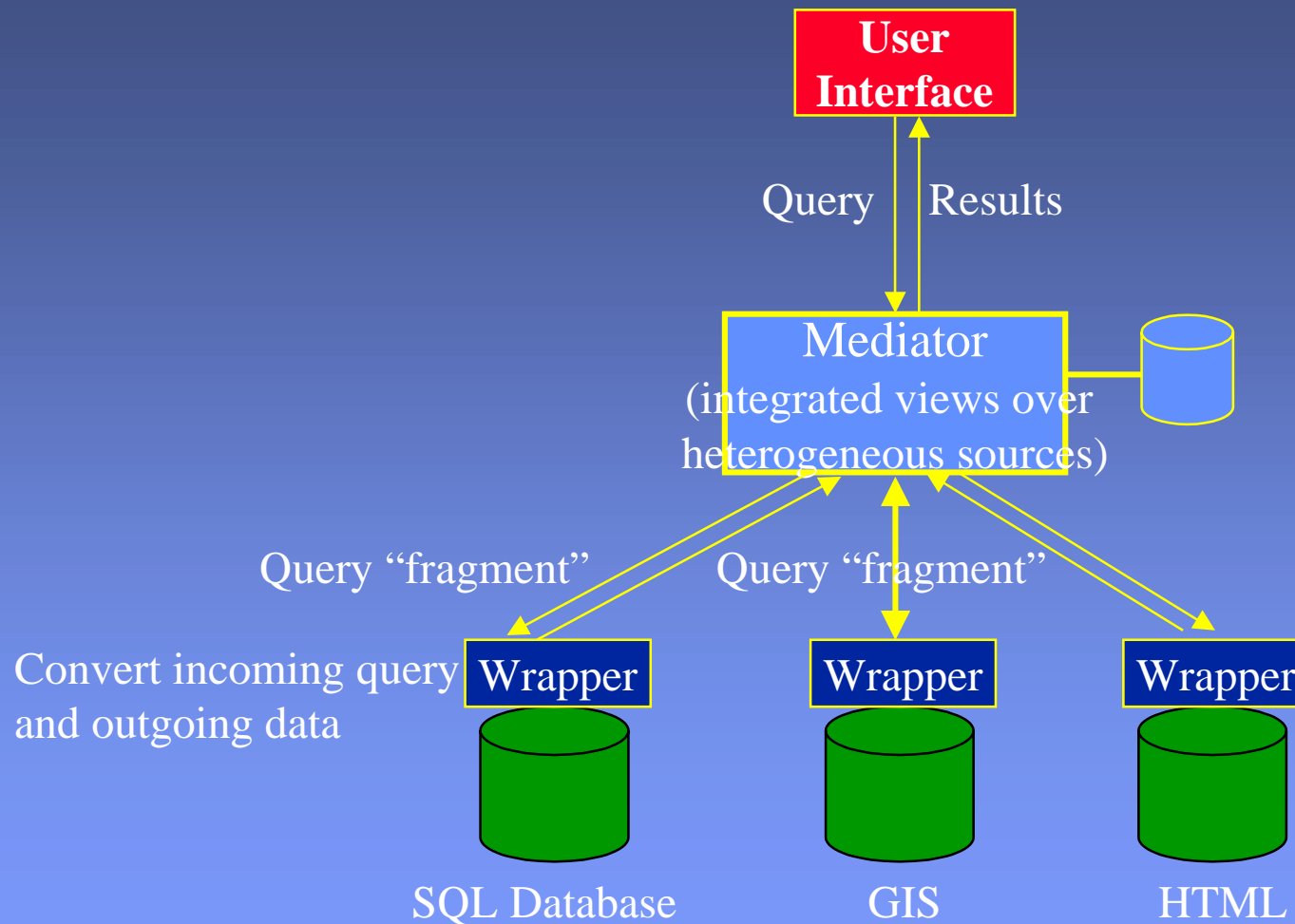
The Web emerges as a distributed database with XML as its data model



# Integrated / Mediated views



# A Typical Mediation Scenario



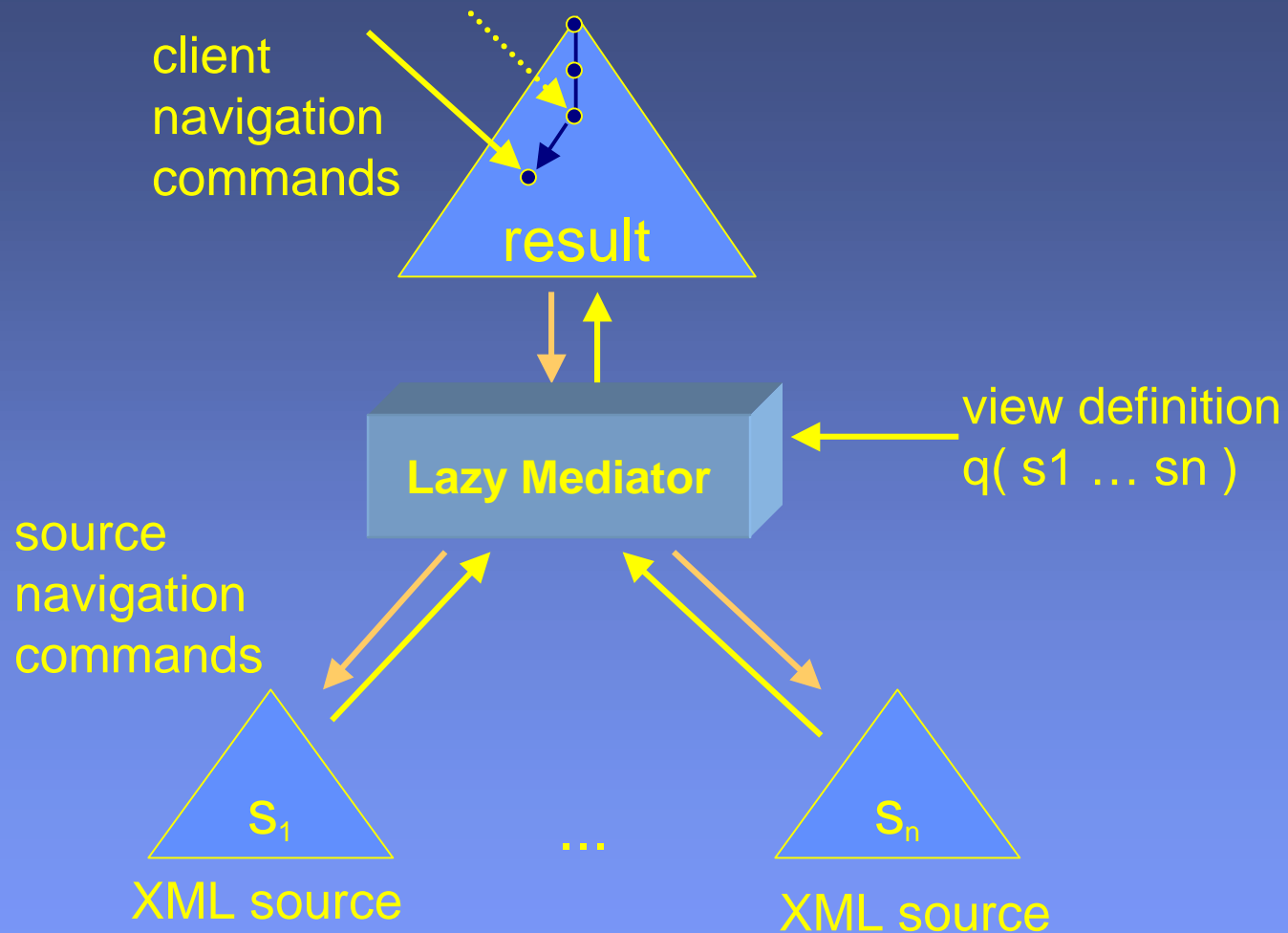
# *MIX Components*

- **MIXm Mediator tool-kit**
  - allows definition of views across multiple resources
  - views are expressed in a declarative query language
  - query engine to execute queries on views
- **XML Matching And Structuring (XMAS) query language**
  - operates on a given set of XML documents to produce a new XML document, using XMAS algebra

## *MIX components...*

- **DOM-VXD: DOM Virtual XML Document extension**
  - a “lazy” implementation of DOM. Supports browsing/navigation of XML documents with a server-side, “compute as you go” model
- **Blended Browsing and Querying (BBQ) interface**
  - supports navigation and querying of XML documents
  - generates XMAS queries on mediator views
  - generates XMAS queries modified by DOM-VXD operations to incrementally evaluate the result set, to support navigation of XML documents

# Navigation driven evaluation



# BBQ Interface

The screenshot displays the BBQ v2.0a9 interface, which is used for editing XML documents. The interface is divided into several panes:

- Data Sources:** A list of XML files including `cameras.xml`, `cameras2.xml`, `junk.xml`, and `junk2.xml`. Buttons for `Refresh` and `Open` are visible.
- packages\_dtd\_1:** A DTD (Document Type Definition) view showing the structure of the XML document. It includes elements like `<packages>`, `<package>`, `<camera>`, `<model>`, `<camera_price>`, `<file_format>`, `<image_storage_capaci>`, `<pixel_resolution>`, `<autofocus_speed>`, and `<lenses>`.
- DTD XML:** A tree view showing the current XML document structure. It includes elements like `Start Here`, `packages`, `package*`, `SEQ`, `camera`, `SEQ`, `model`, `#PCDATA`, `camera_price`, `file_format`, `image_storage_capa`, and `pixel_resolution`.
- Construct Head Here:** A tree view showing the current XML document structure. It includes elements like `cheap_cameras`, `model`, and `#PCDATA(req)`.
- Sample XML:** A text area showing a sample XML document:

```
KODAK DCS 660
KODAK DCS 560
KODAK DCS 620
KODAK DCS 315
```
- Messages:** A text area showing a message: `OK` and `dragGestureRecognized: can't drag SEQ type node`.

# Displaying results in BBQ

The screenshot displays the BBQ v2.0a3 application interface. The top window shows the 'BBQ query composition' tree structure. The middle window shows the 'XML answer document' with XML tags for object titles and image links. The bottom window shows the 'XSL rendered output' as a table of object information.

**BBQ query composition**

- bbq\_root
  - am\_objects
    - am\_object\*
      - AND
        - AID\_\_amico\_identifier
        - OTY\_\_object\_type\*
        - OPP\_\_object\_parts\_pieces\*
        - CLG\_classification\*
        - OTG\_\_object\_title\_name+

**XML answer document**

```

<?xml version="1.0" ?>
- <ans>
- <my_object>
  <my_title>The Battle between the Israelites and the Amalekites</my_title>
  <my_type>Painting</my_type>
  <my_img>AGO_.95-143.TIF</my_img>
</my_object>
- <my_object>
  <my_title>Dynamism of a Dog on a Leash</my_title>
  <my_type>Painting/Oil</my_type>
  <my_img>AKAG.1964.16.tif</my_img>
</my_object>
- <my_object>
  <my_title>Still Life with Ewer, Vessels and Pomegranate</my_title>
  <my_type>Paintings</my_type>
  <my_img>JPGM.00052502.TIF</my_img>
</my_object>
- <my_object>
  <my_title>The Jack Pine</my_title>
  <my_type>Painting</my_type>
  
```

**XSL rendered output**

	OTY: Object Type	OTG+. (OTN,OTT): Title (Type)	CRG*.CRT: Creator Name	...OCT: ...ation ate	RIG.RIL=XID: Related Images
AGO_.95/143	Painting	The Battle between the Israelites and the Amalekites	Luca Giordano.	unknown	<a href="#">AGO_.95-143.TIF</a>
AIC_.1910.238	Mummy Goods	Mummy Case of Paankhenamun (preferred)	Egyptian.	Third Intermediate Period, Dynasty 22, c. 945 - 715 B.C.	<a href="#">AIC_.E22827.TIF</a> <a href="#">AIC_.E15251.TIF</a> <a href="#">AIC_.E31636.TIF</a>
AKAG.1964:16	Painting/Oil	Dynamism of a Dog on a Leash	Giuseppe Penone, Italian, 1912		<a href="#">AKAG.1964.16.tif</a>