

INFS 740
XML Relational Mapping
(Introduction)

Prof. Alex Brodsky

GMU

What's the Problem

- XML is becoming the standard for
 - Data integration
 - Data exchange on the web (especially B2B)
- *But!* Most data will continue to be stored in relational databases
 - Need some way to convert relational data to XML
- Need to allow existing relational data to be *viewed and queried* as XML

What's the Problem (2)

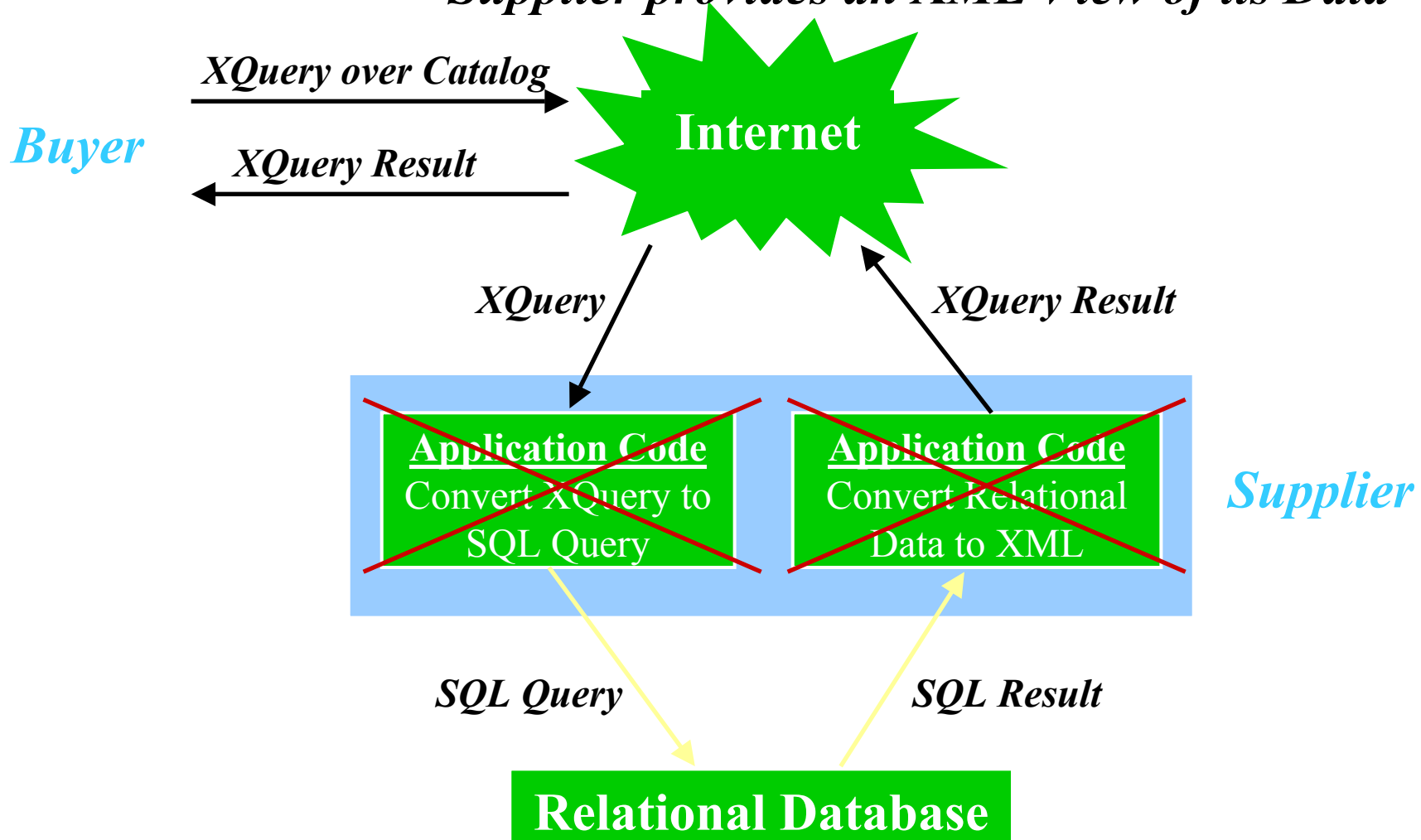
- Flexibility
 - Any XML schema to any Relational schema
- Efficiency
 - Fast access to relational data even though the queries are asked on XML *view*

Three Solutions

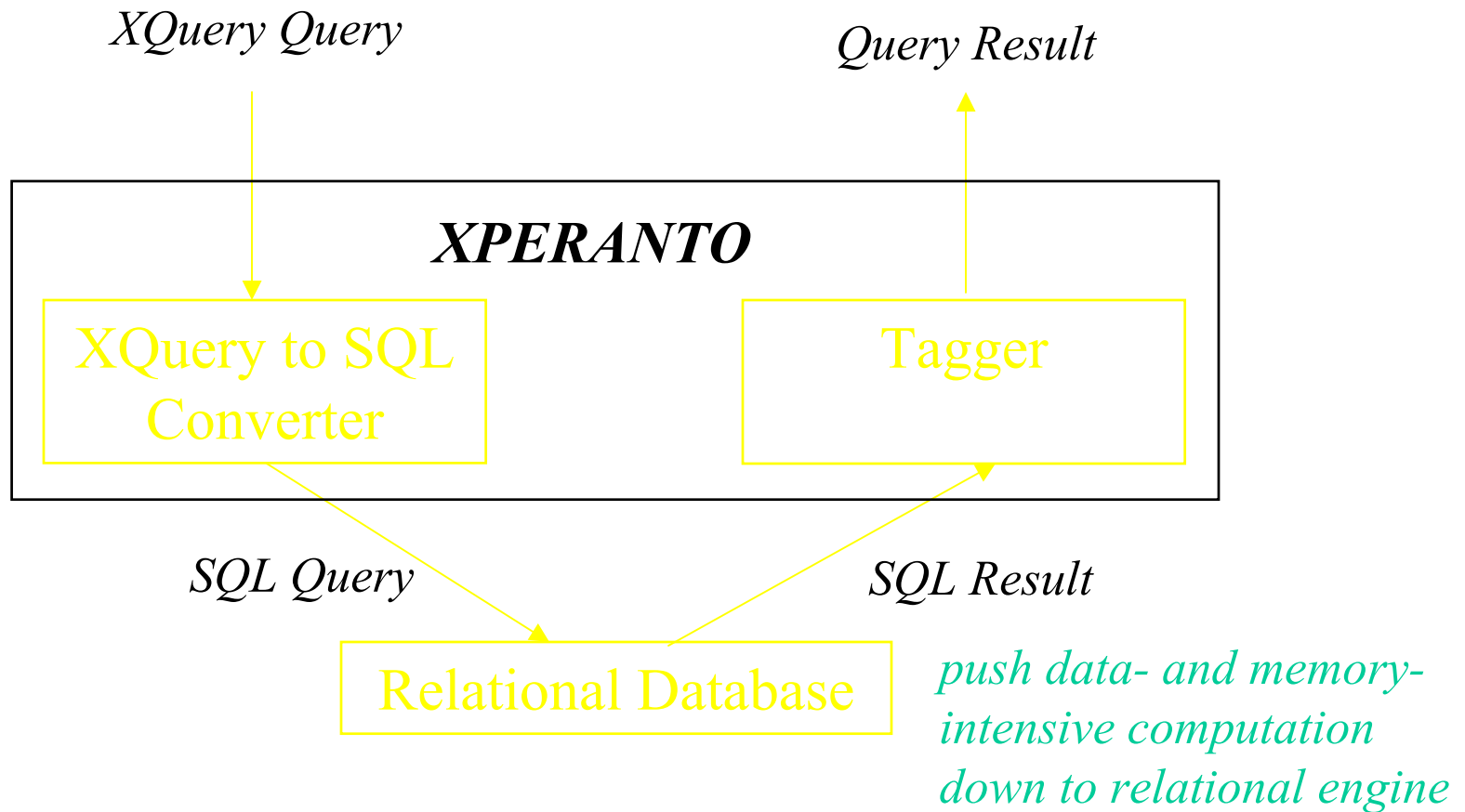
- IBM: Xperanto
- AT&T Labs + others: SilkRoute
- *Microsoft (SQLServer)*

Web Services Example

Supplier provides an XML View of its Data



High-Level Architecture



Example Relational Data

order

id	custname	custnum
10	Smith Construction	7734
9	Western Builders	7725

item

oid	desc	cost
10	generator	8000
10	backhoe	24000

payment

oid	due	amt
10	1/10/01	20000
10	6/10/01	12000

XML View for Partners

```
<order id="10">
  <customer> Smith Construction </customer>
  <items>
    <item description="generator" >
      <cost> 8000 </cost>
    </item>
    <item description="backhoe">
      <cost> 24000 </cost>
    </item>
  </items>
  <payments>
    <payment due="1/10/01">
      <amount> 20000 </amount>
    </payment>
    <payment due="6/10/01">
      <amount> 12000 </amount>
    </payment>
  </payments>
</order>
```

...

Default XML View

```
<db>
  <order>
    <row> <id>10 </id> <custname> Smith Construction </custname> ... </row>
    <row> <id> 9 </id> <custname>Western Builders </custname> ... </row>
  </order>
  <item>
    <row> <oid> 10 </oid> <desc> generator </desc> <cost> 8000 </cost> </row>
    <row> <oid> 10 </oid> <desc> backhoe </desc> <cost> 24000 </cost> </row>
  </item>
  <payment>
    ... similar to <order> and <item>
  </payment>
</db>
```

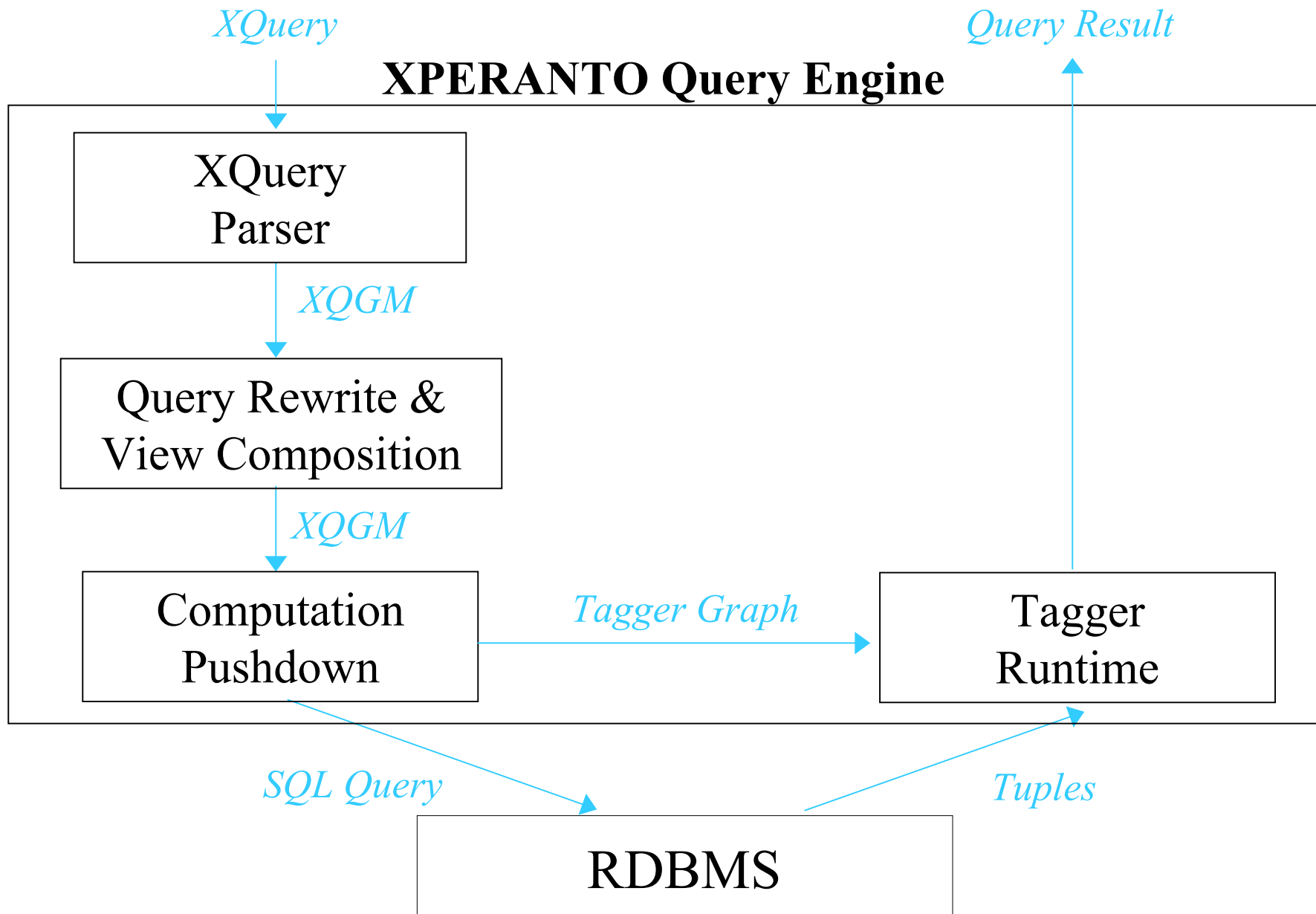
Creating an XPERANTO View

```
create view orders as (  
  for $order in view("default")/order/row  
  return <order id=$order/id>  
    <customer> $order/custname </customer>  
    <items>  
      for $item in view("default")/item/row  
      where $order/id = $item/oid  
      return <item description=$item/desc >  
        <cost> $item/cost/text()</cost>  
      </item>  
    </items>  
    <payments>  
      for $payment in view("default")/payment/row  
      where $order/id = $payment/oid  
      return <payment due=$payment/date>  
        <amount> $payment/amount/text()</amount>  
      </payment>  
    <sortby(@due)>  
  </payments>  
</order>)
```

Allow Partners to Query View

Get all orders of customer 'Smith...'

```
for $order in view("orders")  
where $order/customer/text() like 'Smith%'  
return $order
```



View Composition

- XML views with nesting are **constructed** from flat relational tables
- **Navigational** operations (expressed as XPath) traverse nested elements
- Thus **navigational** operations **undo** the effects of **construction**
- **All** XML navigation can thus be **eliminated**

Benefits of View Composition

- Intermediate XML fragments are eliminated
 - Only the construction of desired XML fragments are computed
- Also enables predicates to be pushed down to relational engine