Module 8

General remarks about XQuery

Plan for today

- The semantics of XQuery and the optimization
- XQuery and the static typing
- XQueryX -- the XML syntax of XQuery
- Current limitations of XQuery
- XQuery usage scenarios
- XQuery as a *full (declarative) programming language*
  - XQuery vs. SQL
  - XQuery vs. other programming languages
- Why XQuery has big chances for success
- Plan for the rest of the class

XQuery expressions

XQuery Expr :=Constants | Variable | FunctionCalls | PathExpr |
ComparisonExpr | ArithmeticExpr | LogicExpr |
FLWRExpr | ConditionalExpr | QuantifiedExpr |
TypeSwitchExpr | InstanceofExpr | CastExpr |
UnionExpr | IntersectExceptExpr |
ConstructorExpr | ValidateExpr

Expressions can be nested with full generality !

Functional programming heritage.

A fraction of a real customer XQuery

```
let $wlc := document("tests/ebsample/data/ebSample.xml")
let $ctrlPackage := "foo.pkg"
let $wfPath := "test"

let $tp-list :=
for $tp in $wlc/wlc/trading-partner
return
<trading-partner
        name="{$tp/@name}"
        business-id="{$tp/party-identifier/@business-id}"
        description="{$tp/@description}"
        notes="{$tp/@notes}"
        type="{$tp/@type}"
        email="{$tp/@email}"
        phone="{$tp/@phone}"
        fax="{$tp/@fax}"
        username="{$tp/@user-name}"
```

```
{
    for $tp-ad in $tp/address
    return
      $tp-ad
  }
  {
    for $eps in $wlc/extended-property-set
    where $tp/@extended-property-set-name eq $eps/@name
    return
       $eps
  }
  {
    for $client-cert in $tp/client-certificate
    return
    <client-certificate
          name="{$client-cert/@name}"
    >
    </client-certificate>
  }
```

```
{
    for $server-cert in $tp/server-certificate
    return
    <server-certificate
        name="{$server-cert/@name}"
    >
    </server-certificate>
}
{
    for $sig-cert in $tp/signature-certificate
    return
    <signature-certificate
        name="{$sig-cert/@name}"
    >
    </signature-certificate>
}
{
    for $enc-cert in $tp/encryption-certificate
    return
    <encryption-certificate
        name="{$enc-cert/@name}"
    >
    </encryption-certificate>
}
```

```
{
        for $eb-dc in $tp/delivery-channel
        for $eb-de in $tp/document-exchange
        for $eb-tp in $tp/transport
        where $eb-dc/@document-exchange-name eq $eb-de/@name
              and $eb-dc/@transport-name eq $eb-tp/@name
              and $eb-de/@business-protocol-name eq "ebXML"
        return
        <ebxml-binding
              name="{$eb-dc/@name}"
              business-protocol-name="{$eb-de/@business-protocol-name}"
              business-protocol-version="{$eb-de/@protocol-version}"                \

              is-signature-required="{$eb-dc/@nonrepudiation-of-origin}"
              is-receipt-signature-required="{$eb-dc/@nonrepudiation-of-receipt}"

              signature-certificate-name="{$eb-de/EBXML-binding/@signature-certificate-n}"
              delivery-semantics="{$eb-de/EBXML-binding/@delivery-semantics}"
        {
              if(xf:empty($eb-de/EBXML-binding/@ttl))
              then()
              else attribute persist-duration
                    {concat(($eb-de/EBXML-binding/@ttl div 1000), " seconds")}
```



```
        }
```

```
{
        if( xf:empty($eb-de/EBXML-binding/@retries))
        then ()
        else $eb-de/EBXML-binding/@retries
    }
    {
        if( xf:empty($eb-de/EBXML-binding/@retry-interval))
        then ()
        else attribute retry-interval
            {concat(($eb-de/EBXML-binding/@retry-interval div 1000), " seconds")}
    }

     <transport
        protocol="{$eb-tp/@protocol}"
        protocol-version="{$eb-tp/@protocol-version}"
        endpoint="{$eb-tp/endpoint[1]/@uri}"
      >
        {
```

```
for $ca in $wlc/wlc/collaboration-agreement
                    for $p1 in $ca/party[1]
                    for $p2 in $ca/party[2]
                    for $tp1 in $wlc/wlc/trading-partner
                    for $tp2 in $wlc/wlc/trading-partner
                    where $p1/@delivery-channel-name eq $eb-dc/@name
                    and $tp1/@name eq $p1/@trading-partner-name
                    and $tp2/@name eq $p2/@trading-partner-name
                    or $p2/@delivery-channel-name eq $eb-dc/@name
                    and $tp1/@name eq $p1/@trading-partner-name
                    and $tp2/@name eq $p2/@trading-partner-name
```

```
return
                    if ($p1/@trading-partner-name=$tp/@name)
                    then
                      <authentication
                          client-partner-name="{$tp2/@name}"
                          client-certificate-name="{$tp2/client-certificate/@name}"
                          client-authentication="{
                            if(xf:empty($tp2/client-certificate))
                            then "NONE"
                            else "SSL_CERT_MUTUAL"
                          }"
                          server-certificate-name="{
                              if($tp1/@type="REMOTE")
                              then
                                $tp1/server-certificate/@name
                              else ""
                          }"
                          server-authentication="{
                            if($eb-tp/@protocol="http")
                            then "NONE"
                            else "SSL_CERT"
                          }"
```

```
>
                    </authentication>
                  else
                    <authentication
                        client-partner-name="{$tp1/@name}"
                        client-certificate-name="{$tp1/client-certificate/@name}"
                        client-authentication="{
                          if(xf:empty($tp1/client-certificate))
                          then "NONE"
                          else "SSL_CERT_MUTUAL"
                        }"
                        server-certificate-name="{
                              if($tp2/@type="REMOTE")
                              then $tp2/server-certificate/@name
                              else ""
                        }"
                        server-authentication="{
                          if($eb-tp/@protocol="http")
                          then "NONE"
                          else "SSL_CERT"
                        }"
                    >
                    </authentication>
```

```
 }
            </transport>
          </ebxml-binding>
      }
{-- RosettaNet Binding --}
      {
          for $eb-dc in $tp/delivery-channel
          for $eb-de in $tp/document-exchange
          for $eb-tp in $tp/transport
          where $eb-dc/@document-exchange-name eq $eb-de/@name
                and $eb-dc/@transport-name eq $eb-tp/@name
                and $eb-de/@business-protocol-name eq "RosettaNet"
          return
          <rosettanet-binding
                name="{$eb-dc/@name}"
                business-protocol-name="{$eb-de/@business-protocol-name}"
                business-protocol-version="{$eb-de/@protocol-version}"
```

```
 is-signature-required="{$eb-dc/@nonrepudiation-of-origin}"
                    is-receipt-signature-required="{$eb-dc/@nonrepudiation-of-receipt}"
                    signature-certificate-name="{$eb-de/RosettaNet-binding/@signature-certi\
ficate-name}"
                    encryption-certificate-name="{$eb-de/RosettaNet-binding/@encryption-cer\
tificate-name}"
                    cipher-algorithm="{$eb-de/RosettaNet-binding/@cipher-algorithm}"
                    encryption-level="{
                          if ($eb-de/RosettaNet-binding/@encryption-level = 0)
                          then "NONE"
                          else if($eb-de/RosettaNet-binding/@encryption-level = 1)
                                then "PAYLOAD"
                                else  "ENTIRE_PAYLOAD"
                    }"
      {--            process-timeout="{$eb-de/RosettaNet-binding/@time-out}"  --}

            >
            {
                    if( xf:empty($eb-de/RosettaNet-binding/@retries))
                    then ()
                    else $eb-de/RosettaNet-binding/@retries
            }
```

```
{
                if(xf:empty($eb-de/RosettaNet-binding/@retry-interval))
                then ()
                else attribute retry-interval
                        {concat(($eb-de/RosettaNet-binding/@retry-interval div 1000), "\
seconds")}
            }
            {
                if(xf:empty($eb-de/RosettaNet-binding/@time-out))
                then()
                else attribute process-timeout
                        {concat(($eb-de/RosettaNet-binding/@time-out div 1000), " secon
ds")}

            }
              <transport
                protocol="{$eb-tp/@protocol}"
                protocol-version="{$eb-tp/@protocol-version}"
                endpoint="{$eb-tp/endpoint[1]/@uri}"
              >
                {
```

```
for $ca in $wlc/wlc/collaboration-agreement
                    for $p1 in $ca/party[1]
                    for $p2 in $ca/party[2]
                    for $tp1 in $wlc/wlc/trading-partner
                    for $tp2 in $wlc/wlc/trading-partner
                    where $p1/@delivery-channel-name eq $eb-dc/@name
                    and $tp1/@name eq $p1/@trading-partner-name
                    and $tp2/@name eq $p2/@trading-partner-name
                    or $p2/@delivery-channel-name eq $eb-dc/@name
                    and $tp1/@name eq $p1/@trading-partner-name
                    and $tp2/@name eq $p2/@trading-partner-name

                    return
                        if ($p1/@trading-partner-name=$tp/@name)
                        then
                          <authentication
```

```
<authentication
                    client-partner-name="{$tp2/@name}"
                    client-certificate-name="{$tp2/client-certificate/@name}"
                    client-authentication="{
                      if(xf:empty($tp2/client-certificate))
                      then "NONE"
                      else "SSL_CERT_MUTUAL"
                    }"
                    server-certificate-name="{
                          if($tp1/@type="REMOTE")
                          then
                                $tp1/server-certificate/@name
                          else ""
                    }"
                    server-authentication="{
                      if($eb-tp/@protocol="http")
                      then "NONE"
                      else "SSL_CERT"
                    }"

                  >
                  </authentication>
```

```
else
                    <authentication
                        client-partner-name="{$tp1/@name}"
                        client-certificate-name="{$tp1/client-certificate/@name}"
                        client-authentication="{
                          if(xf:empty($tp1/client-certificate))
                          then "NONE"
                          else "SSL_CERT_MUTUAL"
                        }"
                        server-certificate-name="{
                              if($tp2/@type="REMOTE")
                              then
                                    $tp2/server-certificate/@name
                              else ""
                        }"
                        server-authentication="{
                          if($eb-tp/@protocol="http")
                          then "NONE"
                          else "SSL_CERT"
                        }"
                    >
                    </authentication>
```

```
 }
                </transport>
            </rosettanet-binding>
        }

</trading-partner>

let $sv :=
for $cd in $wlc/wlc/conversation-definition
for $role in $cd/role

where xf:not(xf:empty($role/@wlpi-template) or $role/@wlpi-template="") and
 $cd/@business-protocol-name="ebXML" or $cd/@business-protocol-name="RosettaNet"

    return
      <servicePair>
          <service
              name="{xf:concat($wfPath, $role/@wlpi-template, '.jpd')}"
              description="{$role/@description}"
              note="{$role/@note}"
              service-type="WORKFLOW"
              business-protocol="{xf:upper-case($cd/@business-protocol-name)}"
    >
```

. . . (60 % more to come)

XQuery is *not (only)* a query language

- Declarative programming language
- General purpose XML to XML transformation engine
- Designed for *optimizability*. Primary goal in the design of the language.
- Impact on the semantics of the language.

The XQuery semantics and the optimization (1)

- Trade-off between optimizability (on one side) and complexity, non-determinism and expressive power (on the other side)
  - Query languages are more optimizable but pay a price on the other side
  - Imperative languages lack optimizability but the semantics is simpler, deterministic , and richer
- How can we achieve better performance ?

1. Allow to execute sub-computations in a different order
   - Parallelization, rescheduling

- Possible to use various data access paths
- Allow lazy evaluation
- Allow streaming/pipelining between operations (no materialization of intermediate results)
- Allow various evaluation algorithms for the same logical operation

The XQuery semantics and the optimization (2)

- Allow to execute sub-computations in a different order (e.g. parallelization, rescheduling)
  - XQuery: no real side-effects, errors are non-deterministic
  - (1,2,3, 1 div 0) [1] can return either *1,* or *error*
- Allow lazy evaluation
- Possible to use various data access paths
  - XQuery: and, or commutative, shortcircuiting operations, errors again non-deterministic
  - (1 eq 2) and (1 div 0 eq 2) can return both *false*, or *error*
- Allow streaming/pipelining between operations (no materialization of intermediate results)
- Allow various evaluation algorithms for the same logical operation
  - XQuery : unordered { expr }
  - (unordered { (1,2,3,4) }) [1] => 1, 2, 3 or 4 can be result
  - (unordered { //book[@year=1999]/title })[1]

XQuery type system

- XQuery has a powerful (and complex!) type system
- XQuery types are imported from XML Schemas
- Every XML data model instance has a dynamic type
- Every XQuery expression has a static type
- *Pessimistic* static type inference
  - However, most implementations have an *optimistic* static typing inference

    (1, “foobar”)[1] + 1 => pessimistic static typing error, optimistic no
- Optional feature, few implement it, Galax the correct one
- The goal of the type system is:
  - detect statically errors in the queries
  - infer the type (and or the shape/schema) of the result of valid queries

    #### Type and schemas are not the same thing !!!!

3. ensure statically that the result of a given query is of a given (expected) type if the input dataset is guaranteed to be of a given type

XQuery type system components

- Atomic types
  - *xs:untypedAtomic*
  - All 19 primitive XML Schema types
  - All user defined atomic types
- Empty, None
- Type constructors (simplification!)
  - Elements: *element name {type}*
  - Attributes: *attribute name {type}*
  - Alternation : *type1 | type 2*
  - Sequence: *type1, type2*
  - Repetition: *type**
  - Interleaved product: *type1 & type2*

- $type_1$ *intersect* $type_2$?
- $type_1$ *subtype of* $type_2$?
- $type_1$ *equals* $type_2$?

XQueryX: the XML syntax for XQuery

- Most XML languages (schema, programming, forms, etc) have an XML syntax
- “normal” XQuery doesn’t
  - Has been designed for human programmers
- XQueryX: an alternative, XML-based syntax for Xquery
  - The parsed abstract syntax tree in XML
  - Has an XML Schema for an XQuery program
- Go to the XQueryX specification…

XQueryX: the advantages

- XQuery programs are also data
  - Can be stored, queried, updated, processed in the same way, with the same languages like the rest of the data (remember Lisp ?)
  - Code becomes data
  - Automatic code generation, rewriting
  - We can blend data with code, and with schemas -- all have an XML syntax
- Blurs the distinction between data, metadata, code

Mistakes and limitations of Xquery 1.0

- Mistakes: The *name* !
- Limitations, missing functionality in XQuery 1.0
  - Dynamic namespace generation
  - Better support for group-by and outer-joins
  - Support for references
  - Continuous queries, window queries
  - Better integration with XSLT
  - Integration with Web Services
  - Assertions
  - Error handling: try/catch
  - Scripting extensions
    - Variable assignment
    - Sequential evaluation mode for side-effects
    - Blocks
  - *eval*(XQueryX-fragment)
  - Integration with Semantic Search and ontologies

XQuery Use Case Scenarios (1)

- XML transformation language in Web Services
  - Large and very complex queries
  - Input message + external data sources
  - Small and medium size data sets
  - Transient and streaming data (no indexes)
  - With or without schema validation

Mid-tier

- XML message brokers
  - Simple path expressions, single input message
  - Small data sets
  - Transient and streaming data (no indexes)
  - Mostly non schema validated data

Mid-tier

- Semantic data verification
  - Mostly messages
  - Potentially complex (but small) queries

Mid-tier, server, client

XQuery Usage Scenarios (2)

- Data Integration
  - Complex but smaller queries (FLOWRs, aggregates, constructors)
  - Large, persistent, external data repositories
  - Dynamic data (via Web Services invocations)

Mid-tier, server, client

- Large volumes of blend relational and XML data
  - Structured data with unstructured/semistructured extensions
  - Complex queries
  - Read/write data

Database server

- Large volumes of XML logs and archives
  - Web services, RFIDs, etc
  - Complex queries (statistics, analytics)
  - Mostly read only

Database server

- Large content repositories
  - Large volume of data (books, manuals, etc)
  - With or without schema validation
  - Full text essential, update required

Content server

XQuery Usage Scenarios (3)

- Large volumes of distributed textual data
  - XML search engines
  - High volume of data sources
  - Full text, semantic search crucial

Web

- RSS filtering and aggregation
  - High number of input data channels
  - Data is pushed, not pulled
  - Structure of the data very simple, each item bounded size
  - Aggregators using mostly full-text search

Web

- XML data transformation and integration on mobile devices
  - Small XML messages
  - Transformation or aggregation queries
  - Caching is important
  - Streaming very important

Mobile devices

XQuery usage scenarios (4)

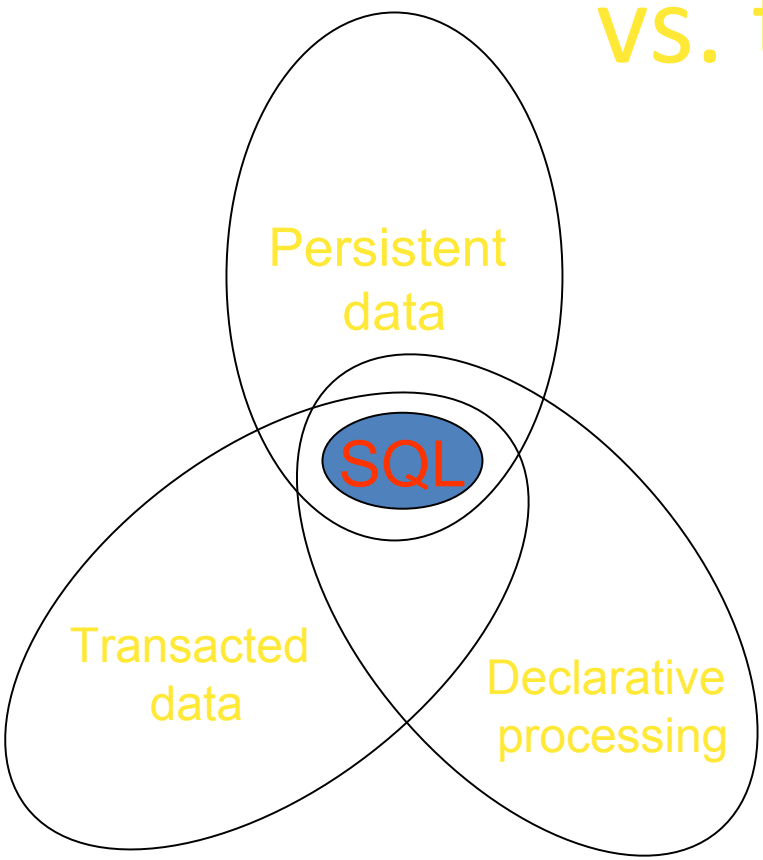
- Content re-purposing
  - E.g. customized books and articles
  - E.g. enterprise customized engineering documentation (product requirements, specs, etc)
- Streamline automatic processing
  - E.g. the creation of the W3C specifications
    - From the *same* XML document we generate automatically the XQuery, Xpath 2.0, Function Libraries specifications, plus the Javacc code that implements the XQuery parser, plus the tests that correctly test the grammar. All those are Xquery views of the same XML document !
- (Ajax-style) dynamic Web pages
  - Xquery is a better way to manipulate the XML of the Web pages then Javascript
- Re-programming the Web /scripting the Web /mashups

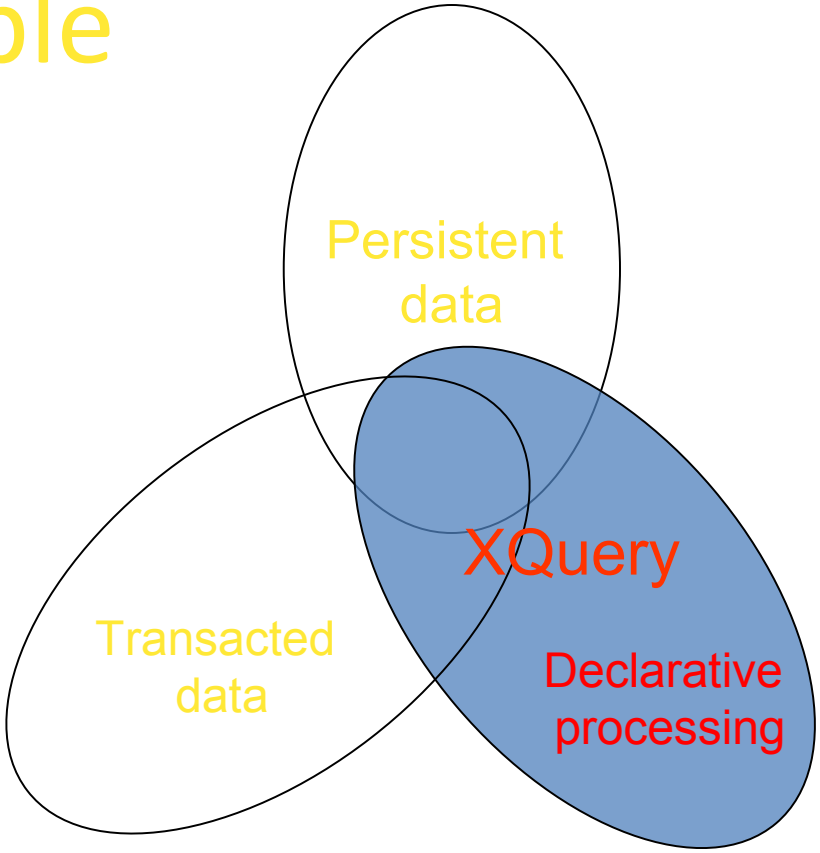
Criteria for XQuery usages

1. Type of queries (e.g. simple, complex, construction-intensive, full text search intensive)
2. Volume of queries
3. Native XML or virtual XML views of other forms of data
4. XML Schema validated data or not
5. Volume of data per query
6. Number of data sources
7. Transient data vs. persistent data
8. Transacted vs. non-transacted data
9. Push data vs. pull data
10. Typed vs. untyped data
11. Read only data vs. updatable data
12. Distributed vs. centralized data sets
13. Data compressed/encrypted or not
14. Target architectures
15. Customer expectation

Each scenario requires <u>different</u> processing techniques.

XQuery vs. SQL: beyond the tree vs. table

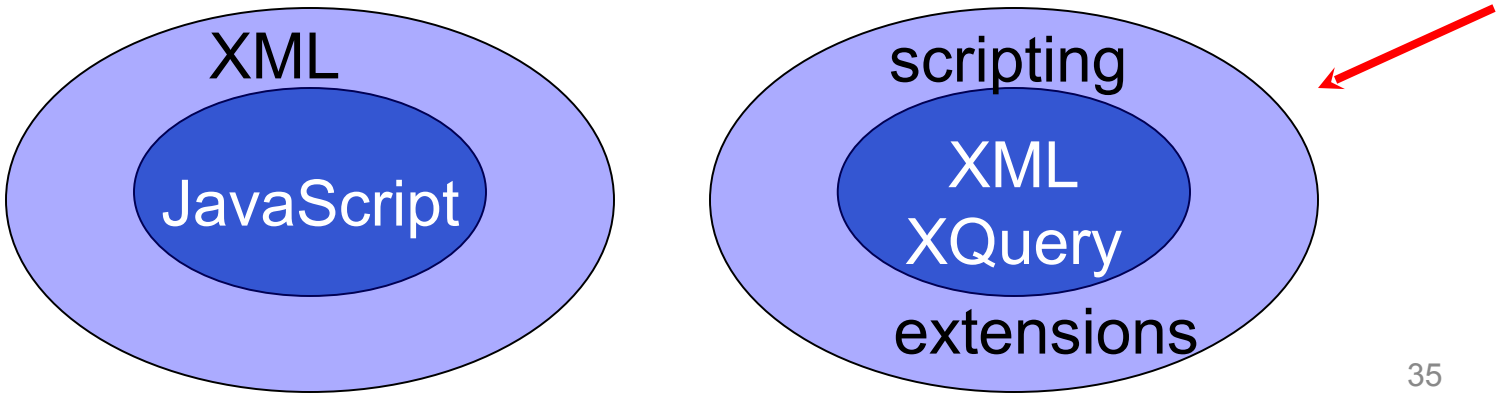




"XQuery: the *XML* replacement for SQL ?"
No, it's more likely that in the *long term* will be the *declarative* replacement for imperative programming languages like Java or C#.

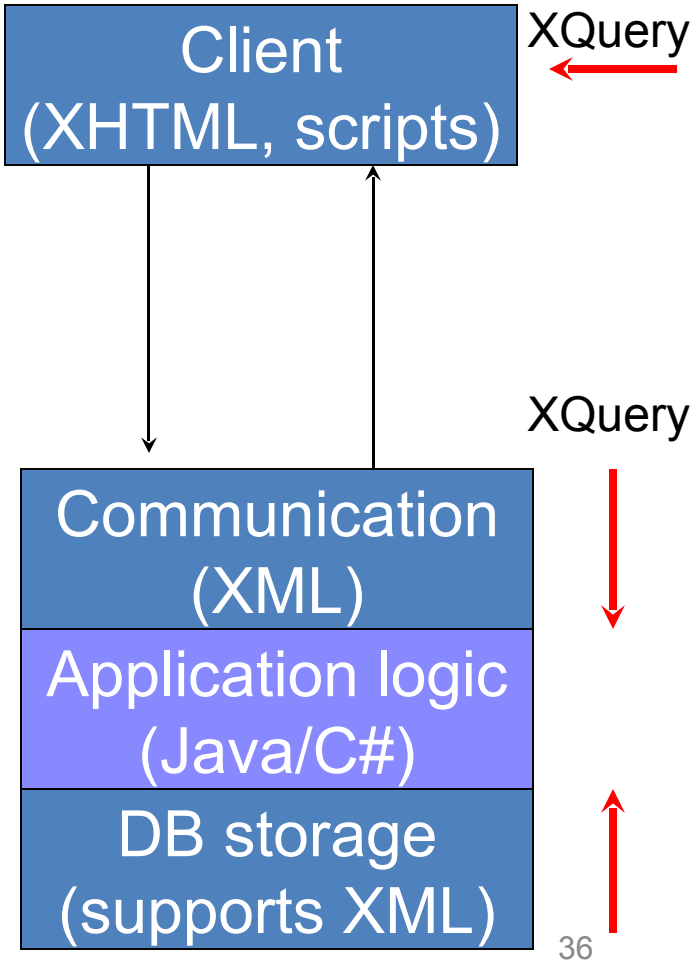
Making XQuery a full XML scripting language (1)

- XQuery is Turing complete, yet "incomplete"
- Users need to write application logic on their data
- *The killer advantages of XML erased by Java, JavaScript, or C#*
- Huge pressure to integrate native XML processing with existing programming languages:
  - C#, EcmaScript, Python, PhP extensions, etc, etc



Making XQuery a full scripting language (2)

- Users are already using XQuery as a scripting language !
- Major missing pieces in XQuery:
  - Order of evaluation has to be deterministic
  - Visible updates/side-effects
  - Variable assignment
  - Error handling



Why will XQuery be successful ?

- Let’s look back at why is XML successful.

Reasons for the overwhelming success of XML

- XML is a general data representation format
- XML is human readable
- XML is machine readable
- XML is internationalized (UNICODE)
- XML is platform independent
- XML is vendor independent
- XML is endorsed by the World Wide Web Consortium
- XML is not a new technology (SGML, HTML)
- XML is not *only* a data representation format, it's a full infrastructure of technologies

REAL reason for the overwhelming success

- Helps companies to *cut costs* in information exchange
  - 1. Avoids the cost of building custom parsers
  - 2. Good quality, low cost parsing software becomes a commodity
  - 3. Minimizes the cost of training
  - 4. Avoids the cost associated with schemas (the evil of all evils :)
- … sometimes at the expense of increased hardware cost due to (bad) parsing performance…
  - But that's OK as long it can be parallelized on cheap machines

The cost of schemas

- Methodology we teach the database students:
  - Gather requirements from the application domain
  - *Design (and agree on) a schema*
  - Write the code (queries + application)
  - Populate the database
  - Execute the code
- Agreeing on schemas is *the most expensive step* in software engineering
  - Plus prohibits the evolution and the customization
- The current information management technology (Java, SQL) doesn't allow us to apply the previous steps in other order, nor to bypass the schema design
- XML does.
- Semi-structured data is the new black in IT industry :-)
  - GoogleBase, etc

Processing XML data

- Huge amount of XML information, and rapidly growing
- We need to "*process*" it
  - Store it efficiently
  - Verify the correctness
  - Filter, search, select
  - Transform, normalize, reshape
  - Join, aggregate
  - Create new data
  - Update the data
  - Take actions based on the existing data
- XQuery has been designed as a solution to the XML processing problem

Alternative solutions to XML processing

- Java, C + APIs (e.g. DOM, SAX, JSR170)
- Perl, PhP, JavaScript + APIs
- Xlinq (MSFT C# extension)
- Code generators
- SQL/XML
- XSLT
- *See Sigmod'06 tutorial on "XML programming techniques"*

Why XQuery *will* be successful

- XQuery helps companies *cut costs on information processing*
  - Avoids the cost of building custom XML processors
  - Improves productivity
  - Good quality, low cost XML processing software (*will)* become a commodity
  - *(Will)* minimize the cost of training
  - (Partially) avoids the cost associated with schemas
  - *(Will)* guarantee best performance

XQuery and the productivity

- Manipulates *only* XML
  - Dealing with two type systems (e.g. Java integer vs. XML integer) is extremely tedious
- Handles *all* XML correctly
  - Typed and untyped, all corner cases of XML (e.g. NS)
- Declarative
  - Smaller amount of code to write
  - Less decisions to make as a programmer
    - streaming, or not, indexes, parallelization
  - Possible to generate automatically

XQuery performance

- Lots of folklore in industry
  - "Everything related to XML *has* to be slow"
  - No, writing *manually optimized* C# or Java over SAX isn't the answer -- *it is not robust to evolutions !!*
- Unfortunately:
  - We don't have benchmarks yet
  - We don't have good XML processing literature yet
- Situation now:
  - In DB, XQuery is executed with the same engines as SQL
  - Good new engines, and improving fast (BEA, Saxon, exist, BerkelyDB, etc)
- XQuery has better chances for good performance then any of the alternatives

XQuery automatic optimization

- Feasible (done in many implementations) in XQuery:
  - Automatic data partitioning, clustering and placement
  - Automatic use of secondary access structures (indexes)
  - Automatic decision about streaming vs. materialization
  - Automatic caching
  - Parallelization of code
  - Program decomposition
  - Program shipping vs. data shipping
  - Rewriting based on assertions
  - Detecting code (in)dependence
  - Rescheduling/reordering of the code
- Impossible (or *much* harder) in Xlinq, Perl, Javascript, etc
- Global dataflow required
  - What operations are executed on each data item
  - What data items will be processed by each operation
- Declarativity of XQuery helps

Frequent criticism of XQuery

- "The performance of XQuery engines isn't acceptable"
  - Why is the alternative any better !?
  - For XQuery, there is hope. For XLinq, PhP, little.
  - Take ideas from both database optimization *and* programming languages compilation, plus innovate
  - Lots of fun research to be done !!!
- "It will never perform as well as if we write the application in Java + SAX"
  - Maybe true today, not sure in near future
  - Optimizing a single XML applications vs. optimizing an XQuery(P) engine (I.e. *all* XML applications)
- "There are no libraries"
  - Let's build some…
  - We will not need the *same* libraries like in Java or C#
    - Different level of abstraction
    - The target applications are different
- "XQuery is too complicated"
  - !?

Frequent criticism (2)

- "Programmers do not know how to program declaratively"
  - What about SQL !?
  - You are the generation who will decide this.
- "This would require users to learn a new language"
  - Smooth transition, easy integration of pieces written in other languages (thanks WS!)

How bad is the bleeding edge ?

- Yes, XQuery is new
- All solutions in XML processing are on the bleeding edge at this point
  - Xlinq is worse in fact
  - XQuery is much older
  - XQuery is subject to open public scrutiny, which insures better quality

Potential impact of XQuery on Web X.0 architectures

- Web 2.0: so much marketing, very little technical substance
- However, we all know that we've reached the limits of Web 1.0
  - User experience -- the Web is becoming annoying
    - Too static, no customization, no push data
  - System builder experience -- building the Web is really expensive, and hard
- We need something new. What ?

Potential impact of XQuery on Web X.0 architectures (cont.)

- Imagine the following scenario:
  - XQuery becomes a full programming language, integrated with Web Services (XQueryP)
  - Good implementations of XQueryP become available in open source, and commodity
  - Databases will implement XQueryP
  - XML repositories will support an HTTP-based simple query protocol (OpenSearch-style, but adapted to XML and XQuery)
  - XQueryP plug-ins in browsers (Ajax ++)
- What will happen to:
  - SQL, Java !?
  - Perl, PhP JavaScript !?
  - Client-server !?
  - Thin clients !?