General ranting about XML
(reasoning about updates)

Daniela Florescu
Oracle
dana.florescu@oracle.com

Where do I come from…

- Industry
- Academic background (long time ago..)
- DB person by education
- Part of the XML community since 1996

- I care about *solutions* to help customers better manage their information
- Why care about *updates* !?

Updates

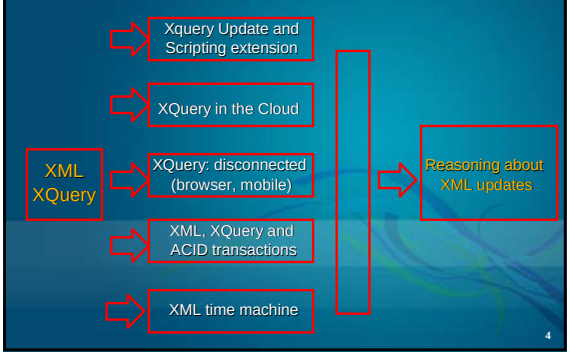
- I do not wake up every morning thinking about updates

But, at least once a week I find myself in a corner thinking:

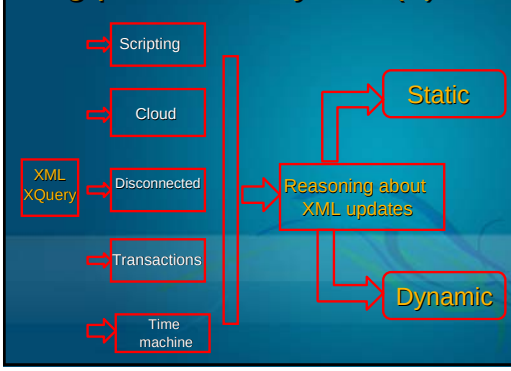
"If I had this algorithm to solve this particular problem about updates, life would be better "

- I arrive to this conclusion from multiple paths and for multiple reasons
- The road from customer solutions to reasoning about updates.

Big picture of my talk (1)



Big picture of my talk (2)

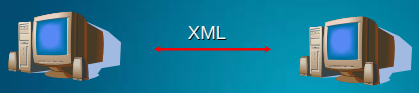


Plan of the talk

- XML and XQuery today: an industrial prospective (15min)
- Problems we address (30 min)
  - XQuery scripting and updates
  - Execution in the cloud
  - Disconnected execution
  - Transactional models
  - XML time machine
- Reasoning about updates (25min)
  - Algorithms we need

XML

- You can see it a little bit everywhere
- The equivalent of "electricity" for modern digital information

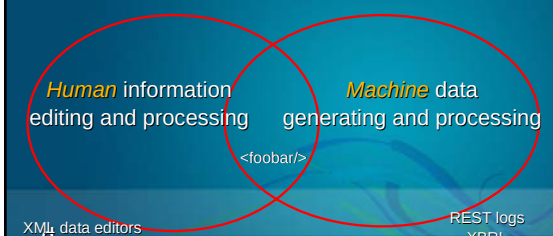


- 400M on Google (SQL 113M, Ruby 11M, RSS 3200M, Java 266M, …)
- In 1996, XML was created as a syntax for information
- In 2010 there is an entire, standalone, technological world associated with it
  - Schemas, XML native programming languages (e.g. XPath, XQuery, XSLT), APIs and protocols (REST), huge amount of REST libraries
- Part of the DNA of computing

Status of XML in 2010

- Used everywhere in industry
- Databases with
  - terabytes of XML data
  - billions of "documents"
- XML databases
  - Open source (eXist)
  - Startups (Marklogic, 28msec)
  - Large Databases (Oracle, DB2)
- XML processors
  - Saxon (1m downloads)
- Patch of *various communities* that rarely talk to each other, do not understand each other and only share the <…/>

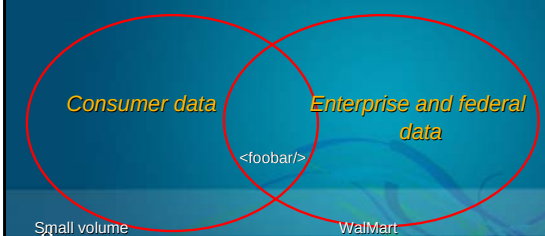
"Human" meets "machine"



XML data editors
Mixed content, XSLT
OpenOfficeXML
XHTML
Browsers

REST logs
XBRL
HL7
NIEM
XQuery

"Consumer" meets "enterprise"



Small volume
AJAX
Simple data
JavaScript
DOM

Google
Facebook
Craiglist

WalMart
Amazon
Boeing
DoD
CreditSuisse

Large volumes
Enormous complexity

What kind of XML can we find (1) ?

- Microsoft Office
- Office 2003 was able to import/export all documents into XML
- Office 2007 models the documents *natively* in XML
- Examples of vocabularies and schemas:
  - WordprocessingML (the XML file format for Word 2003),
  - SpreadsheetML (Excel 2003),
  - FormTemplate XML schemas (InfoPath 2003)
  - DataDiagramingML (Visio 2003)
- Human-generated data

What kind of XML can we find (2) ?

- RSS, Atom
- Content syndication:
  - News tickers
  - Blogs
  - Alerts
- Simple XML format
- Lightweight

XHTML (3)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtm
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" dir="ltr">
  <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <meta name="keywords" content="Main Page,1268,1728,1787,1923,1956,1971 Atlantic hur
hurricane season,2006 World Series" />
<link rel="shortcut icon" href="/favicon.ico" />
<link rel="search" type="application/opensearchdescription+xml" href="/w/opensearch_desc.
<link rel="copyright" href="http://www.gnu.org/copyleft/fdl.html" />
      <title>Main Page - Wikipedia, the free encyclopedia</title>
      <style type="text/css" media="screen,projection">/*<![CDATA[*/ @import "/skins-1.5/
      <link rel="stylesheet" type="text/css" media="print" href="/skins-1.5/common/commo
      <!--[if lt IE 5.5000]><style type="text/css">@import "/skins-1.5/monobook/IE50Fixes
      <!--[if IE 5.5000]><style type="text/css">@import "/skins-1.5/monobook/IE55Fixes.cs
      <!--[if IE 6]><style type="text/css">@import "/skins-1.5/monobook/IE60Fixes.css?22"
      <!--[if IE 7]><style type="text/css">@import "/skins-1.5/monobook/IE70Fixes.css?22"
      <!--[if lt IE 7]><script type="text/javascript" src="/skins-1.5/common/IEFixes.js?22
      <meta http-equiv="imagetoolbar" content="no" /><![endif]-->

      <script type= "text/javascript">
         var skin = "monobook";
         var stylepath = "/skins-1.5";

         var wgArticlePath = "/wiki/$1";
```

What kind of XML can we find (4)?

- REST APIs input/output
  - Data formats: XML or some other format that can be logically seen as XML (e.g. CSV, shapefile)

- WS input/output
- Dynamic data, not static
- Mashups !
- www.programmableweb.com
- Enterprise / consumer

What kind of XML can we find (5) ?

- XBRL
- Goal: facilitate the exchange of business and financial performance information between companies, governments, insurance companies, banks, etc.
- Mandate by law in many countries
- http://en.wikipedia.org/wiki/XBRL

What kind of XML can we find (6) ?

- HealthCareLevel7
- Medical information that is being exchanged between hospitals, patients, doctors, pharmacies and insurance companies
- http://en.wikipedia.org/wiki/HL7

What kind of XML can we find (7) ?

- NIEM
- The National Information Exchange Model, is a partnership of the U.S. Department of Justice and the Department of Homeland Security. It is designed to develop, disseminate and support enterprise-wide information exchange standards and processes that can enable jurisdictions to effectively share critical information in emergency situations, as well as support the day-to-day operations of agencies throughout the nation.

What kind of XML can we find (8)?

- Data.gov
- Goal: The purpose of Data.gov is to increase public access to high value, machine readable datasets generated by the Executive Branch of the Federal Government.

Why XML ?

1. Information can be processed automatically
2. Information is schema independent
3. Can model all kinds of information (documents, structured data, and everything in between)
4. Perfect for information archival

19

XQuery

- General XML information processing language
  - Declarative, functional
  - *Not a "query" language*
- 5M pages on Google
- I am daily dealing with 20K+ lines XQuery programs
- Implementations
  - Oracle, DB2, MarkLogic
  - Standalone (Saxon, Zorba, Xquilla)
  - Open source
  - Cloud (28msec)
  - More then 20 implementations
- *Huge* customer base

20

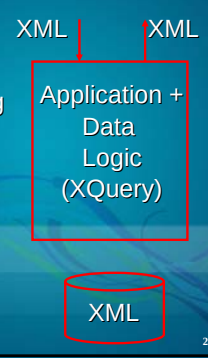
Where can it be used in today's architectures?

- Databases
- Middle tier
  - Information dispatch
  - Transformation
  - Data integration
- Clients
  - Browsers
  - Mobile devices

21

XML/XQuery's real potential

- XML End-To-End architectures
- Standalone programming language for information intensive applications



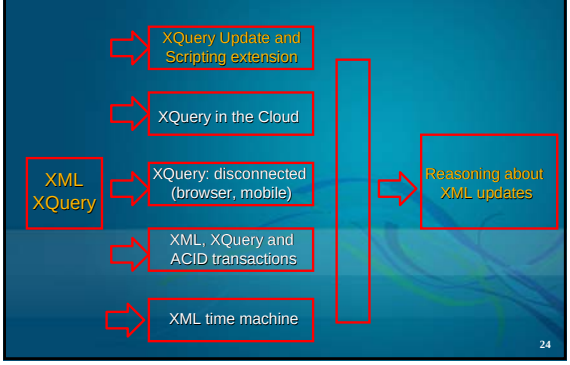
22

Why XQuery ?

- From XML
  - Schema independent
  - Continuity structured data <--> textual data
- Standard
- Declarative
  - Can be optimized, paralellized
  - Can be generated automatically
  - Smaller # lines of code
- Single layer code
- Open source friendly

23

Picture of the talk



24

XQuery Update Extension

- http://www.w3.org/TR/xquery-update-10/
- Language operations (expressions)
  - Insertion of nodes
  - Deletion of nodes
  - Modification of nodes by changing some of the properties (while preserving is node identity)
    - delete *expr*
- Primitive Update Lists (PULs)
  - E.g. insert, delete, rename
    - delete node1234
- Primitive Update Routines
  - revalidate node
- Updating expressions (functions, query, programs) return PULs

XQuery Update Example

```
for $node in $root//abc:*
let $localName := fn:local-name($node),
        $newQName := fn:concat("xyz:", $localName)
return (
    rename node $node as fn:QName("http://xyz/ns", $newQName)
    for $attr in $node/@abc:*
    let $attrLocalName := fn:local-name($attr),
        $attrNewQName := fn:concat("xyz:", $attrLocalName)
    return
        rename node $attr as fn:QName("http://xyz/ns",
          $attrNewQName))
```


XQuery Scripting Extension

- Add the following expressions
  - Apply (";")
  - Variable assignment
  - Block
  - Exit with
  - While
- Expressions
  - Simple (yes, no, no)
  - Updating (no, yes, no)
  - Sequential (yes, no, yes)
- Can:
  - Return a non empty XDM
  - Return a non empty PUL
  - Have side effects
- Both snapshot, and iterative

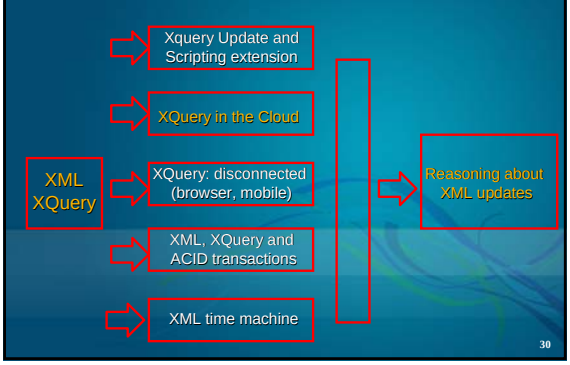
Scripting example

```
declare sequential function validate-and-log($username as xs:string)      as
      xs:boolean
{       declare $log as document-node() := fn:doc("log.xml");
    declare $entry as element() :=          <access-attempt>
      <timestamp>{fn:current-dateTime()}</timestamp>            <user-
      name>{$username}</user-name>            <access-allowed/>
      </access-attempt> ;
 declare $result as xs:boolean;
 if ($username = doc("users.xml")/current-users/user/name )        then (
      replace value of node $entry/access-allowed with "Yes"  ;         $result :=
      true();        )
else (          replace value of node $entry/access-allowed with "No"  ;
      $result := false();        );
insert node $entry as last into $log   ;
fn:put($log, "log.xml");
exit returning $result;
}
```


Compilation, optimization, parallelization of XQuery

- XQuery Updating and Scripting: *not* Datalog, FOL
  - PUL , side-effects, exceptions
- XQuery SE is *not* Java eithe
  - Set oriented updates, deferred PULs
- More closely related to ML
- It is like Oracle's PL-SQL
  - But…PL-SQL is not properly compiled, optimized and parallelized today : 2 independent compilers
- General (functional) programming language with a database-style optimizer
- Extremely interesting research topic
- It's all about static analysis of updating and scripting expressions
- Subsetting of the language is NO, NO, NO

Big picture of my talk



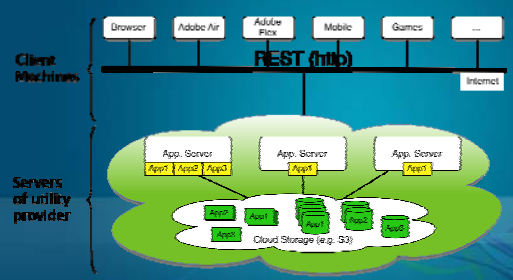
Executing XQuery in the cloud

- XQuery as a programming language reduces the *development cost*
- What about *deployment cost* ?
  - That's the cloud !
  - Let's use Amazon Web Services (AWS)
- The magic is in the glue: XQuery + cloud
  - XML data automatically partitioned, replicated and indexed in the cloud
  - XQuery programs automatically compiled, optimized for parallel execution in the cloud
  - Automatic deployment in the cloud

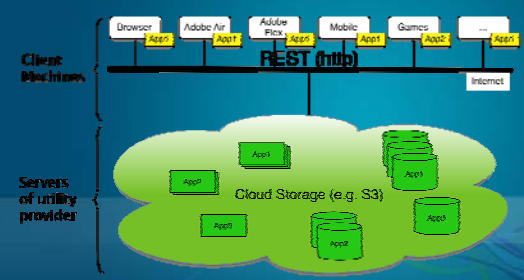
Why the cloud ?

- "Rental car" paradigm for computing resources
  - Outsource the maintenance
- Availability
- Scale up
  - parallelization of computation
- Scale down
  - sharing of resources
- Pay-as-you-go

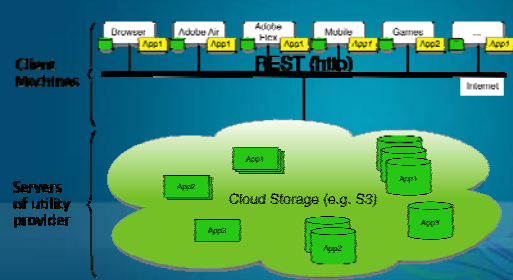
XQuery in the Cloud (connected)



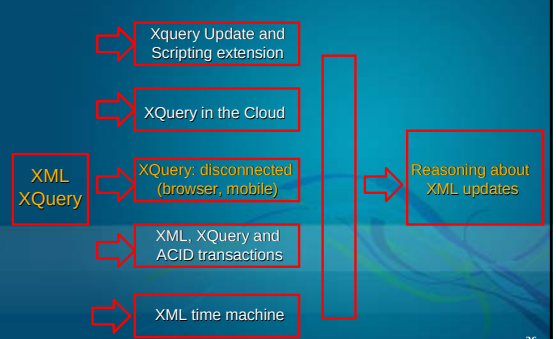
XQuery in the Cloud (no Server)



XQuery in the Cloud (offline)



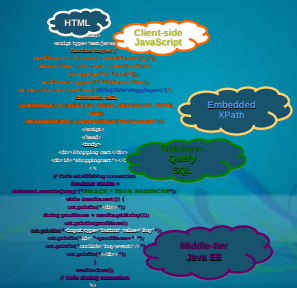
Picture of my talk



Executing XQuery on the client

- Observation
  - XML processing everywhere
- XQuery on the clients
  - Browser (www.xqib.com plug-in)
  - Windows mobile
  - iPhone
- Reasons
  - Technical jungle
  - Data mobility vs. code mobility
  - Executing in disconnected mode

Technology Jungle in the browser



Limited code mobility between layers

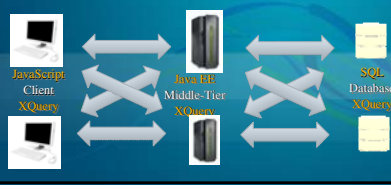
Client
JavaScript

Middle-Tier
Java EE

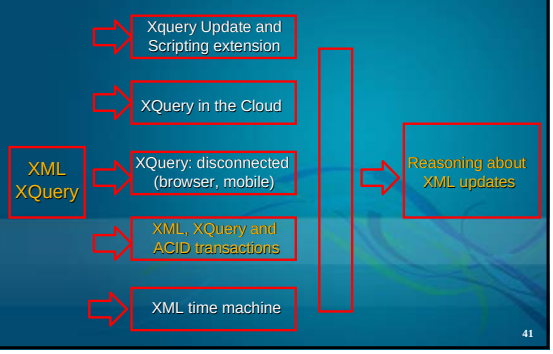
Database
SQL

XQuery…

- ...runs on all three tiers
- ...naturally co-exists with other technologies
  - ☑ Technology jungle
  - ☑ Code mobility



Picture of my talk



Transactional models for XML

- Imagine we use an XML database for collaborative work
  - Google spreadsheets (work in Oracle, SAP)
  - SVN
- ACID transactions !?
  - No way.
- Web + XML+ ACID do not work together
- No locks acceptable while working:
  - Distributed
  - Collaborative fashion
- We need to rethink the transactions (isolation, consistency) models for XML

NOSQL movement

- They have no problems with SQL per se.
- They have problems with:
  - Relational model (too flat) XML OK
  - Schema (too rigid) XML OK
  - *Transactions (not scalable enough) ????*

Data models: state of the art

document-oriented	XML-oriented (semi-structured)	data-oriented
trees	bridge the gap	tables

Consistency models: state of the art

document-oriented	XML-oriented (semi-structured)	data-oriented
SVN, CVS, git	bridge the gap	transactional databases

XQuery now

- **XQuery core**

  Reads from store and outputs a result.
- **XQuery Update**

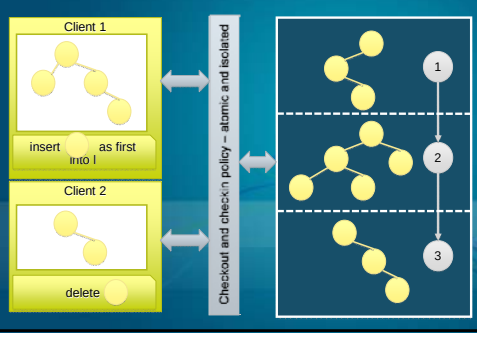
  Reads from store

  Updates are propagated to the store at the end.
- **XQuery Scripting**

  Reads from store, update local copy, writes to store.

No transactions support !!

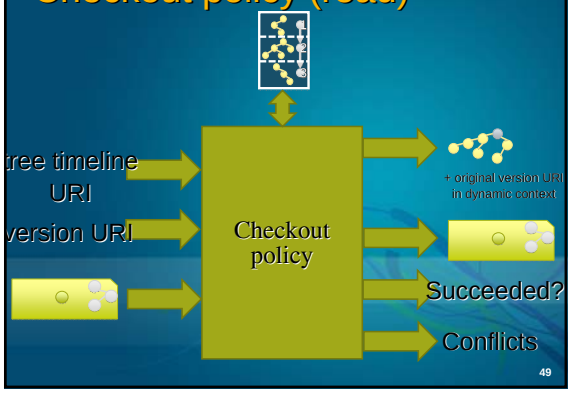
New Processing Model (Checkout, Checkin)



Checkout/checkin policies

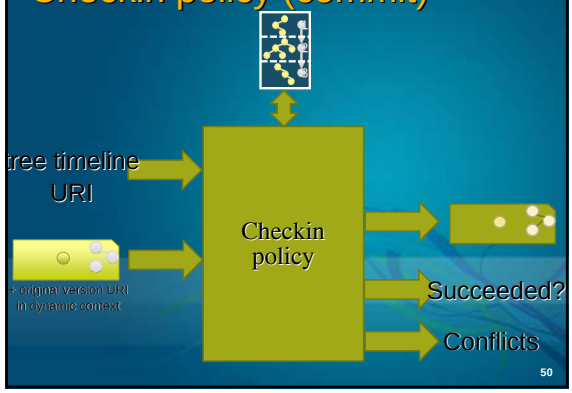
- Black boxes
- Can be chosen by user
- Checkout/checkin can occur anytime
- Both can be explicit or implicit
  - Implicit checkout at first read
  - Implicit checkin at the end of the program
  - Explicit with special instructions
    - vng:checkout($timeline-uri, $version-uri)
    - vng:checkin($timeline-uri)
- Checkout/checkin policy chosen in the prolog.

Checkout policy (read)



49

Checkin policy (commit)



50

Examples of policies

- Database Transaction
  - http://www.example.com/single-checkout: first checkout copies the given version to a local version, subsequent checkouts do nothing.
  - http://www.example.com/conservative-checkin: only apply changes if no concurrent changes have been made, otherwise throw an error.

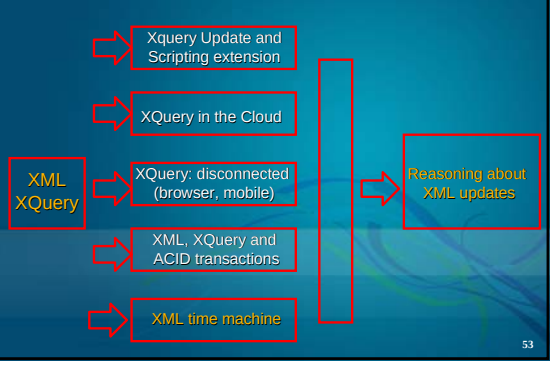
51

Examples of policies

- Document versioning
  - http://www.example.com/merging-checkout: merge given version to local changes
  - http://www.example.com/merging-checkin: merge local changes to changes on the trunk

52

Picture of my talk



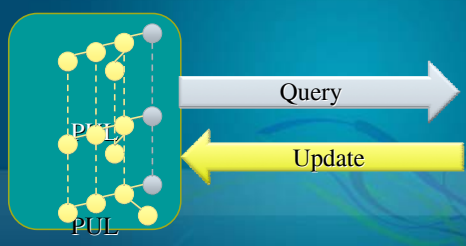
53

XML time machine

- As important as data:
  - Lineage of the data
  - Evolution of the data
- Problem:
  - No temporal support for XML
  - No ability to query the evolution of the data
- Solution
  - XML-aware versioning, extension of XDM
  - Extension of XQuery to query XDM time changes

54

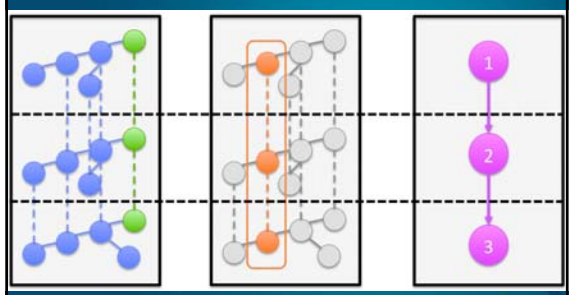
XML+XQuery-based Versioning



XQuery (Core/Update/Scripting)

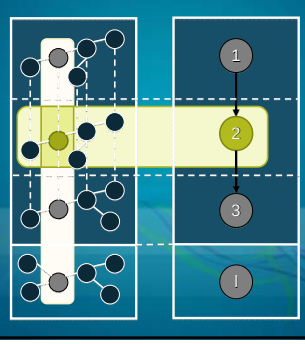
55

Tree Timeline, Node timeline, Version



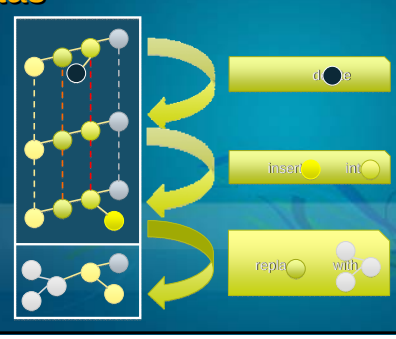
Node Item

- Definition unchanged
- Accessors:
  - dm:reference: URI
  - dm:version: URI
  - and the XDM ones



57

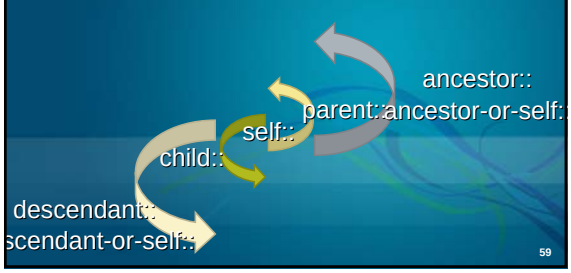
Pending Update Lists as Deltas



58

Navigation in a Tree Timeline

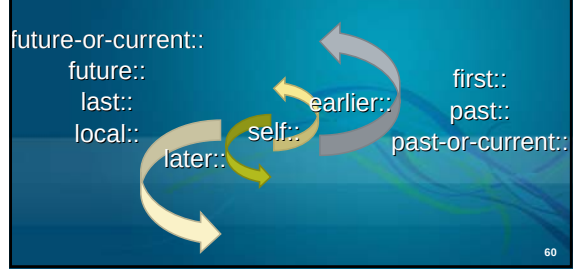
- Existing axes to navigate through space:



59

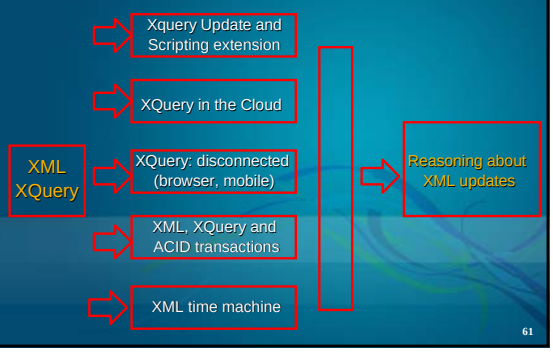
Navigation in a Tree Timeline

- New axes to easily navigate through time:



60

Big picture of my talk



All that requires:

1. Reasoning about updates
   - Static vs. Dynamic
   - Expressions vs. PULs
2. Treating PULs as data
   - Model
   - Serialize
   - Store
   - Index
   - Query

Reasoning about updates

1. Detecting correctness and/or errors
2. Minimization
3. Aggregation
4. Inverse
5. Commutativity
   - ⇒ Static and/or dynamic
   - ⇒ Sufficient conditions good enough

Static vs. dynamic updates

- Updating expressions
  - insert *expr* into *expr*
  - if (*expr*) then delete *expr* else rename *expr* as *expr*
- Primitive Update Lists (PULs)
  - insert <a/> into node1234
  - delete node4567
- Both necessary
  - Static: more info: types, dataflow
  - Dynamic: more concrete info about data

Detection of correctness/errors

- Problem
  - Detect if a given update is consistent, before applying it
  - Detect if the application of updates would result in correct data (schemas, integrity constraints)
- Static
  - Updating expressions
- Dynamic
  - PULs
- Good for:
  - Avoiding costly runtime operations, especially in distributed execution

Minimization

- Problem
  - Given a set of updates, find an *equivalent* set of updates that is "*smaller*" (not necessarily minimal, but smaller)
    - "smaller" := upon execution it would result in a subset of the original PULs
- Static
  - Updating expressions
- Dynamic
  - PULs
- Good for:
  - Decreasing runtime cost (distributed environments)

Aggregation

- Problem
  - Given two updates to be applied successively, find a single equivalent update
  - That's not the union !!
- Dynamic
  - PULs
- Good for:
  - Maintaining deltas in time aware XML
  - Distributed (client-server, cloud) communication of deltas

67

Commutativity with reads

- Problem
  - Given a simple expression and an update expression, find out of they are commutative
- Static
  - simple expression, update expression
- Good for
  - Compilation, parallelization, index maintenance
  - Static data consistency guarantees
- Uses
  - Type information, dataflow analysis, etc

68

Commutativity with updates

- Problem
  - Given two update expressions, find out of they are commutative
- Static
  - update expressions
- Good for:
  - Compilation, parallelization, index maintenance
  - Static data consistency guarantees
  - Optimization of update propagation in distributed environments
- Uses
  - Type information, dataflow analysis, etc

69

Inverse

- Problem
  - Given an update, find the "inverse" update
- Static
  - Updating expression
- Dynamic
  - PULs
- Good for
  - Code rewriting, compilation
  - Storing and indexing time aware XML

70

Treat PULs as data

- Model PULs
  - As XDM
  - For querying
- Serialize PULs
  - As XML
  - For transport
- Store, Index
- Extensions of XQuery to query:
  - Deltas
  - Time evolution of XML versions

71

Reasoning about updates

- We are looking for two "algebras" (operations) for updates
  - Updating expressions
  - PULs
- XQuery subsetting is *not* acceptable
- "Decidable" is not important
- Sufficient conditions good enough
- Efficient algorithms
- Minimize false negatives

72

Conclusion: XML and XQuery

- XML: not a goal in itself
- Opportunity to rethink:
  - Data models (flat vs. nested)
  - Interaction between data/text (data vs. mixed content)
  - Role of schemas (with, without, later)
  - Consistency models and transactions
  - Global IT architectures (3-tiers vs. 1-tier)
  - Deployment models (cloud or not)
- Reasoning about updates
  - A necessary piece of the technical puzzle

73

Concretely

- Please approach the problem differently: XQuery is a functional programming language, not FOL
- Reasoning about updates fundamental
  - Intelligent algorithms
  - Efficient implementations
- Would you like to get involved ?
- Playground: Zorba XQuery engine
  - 100% correct and complete (no toy) portable C++ XQuery engine, open source, good basis for research

Thank you !

74