


ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Module 7:

XQueryP



ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

XQuery: the successes

- Almost 50 implementations
- The three major databases (Oracle, SQL Server, DB2) implement it
- Used in application servers (Oracle, BEA)
- Open source projects are flourishing
 - Saxon is widely used, same for BerkeleyDB-XML
- Enthusiastic customers
 - not completely satisfied, but that's a good sign...
- Used in a variety of contexts:
 - Temporary data, persistent data, streaming XML data
 - Usage scenarios: data transformation, pure querying, analytics, publishing, etc
- Classes in major universities
- Flourishing research community around XML processing

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

XQuery 1.0: a read only language

- Functional heritage
 - XQuery programs are expressions
 - Expressions can be combined with full generality
 - No real side-effects
- XQuery semantics was carefully crafted to allow optimizations:
 - subexpressions can be evaluated in almost any order
 - lazy evaluation is possible
 - errors are allowed to be non-deterministic
 - And, or are commutative, etc
- Compilers have freedom to do code rewriting

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

XQuery Update expressions

- *Insert, delete, rename*, etc are normal expressions
- They are not fully composable with the rest of the expression language
 - Semantic, not syntactic restrictions
- Distinguish the side-effecting expressions vs. non-side-effecting expressions
 - Only in "control-flow" style expressions (FLWOR, typeswitch, conditionals)
 - Side-effecting functions vs. read-only functions

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Single snapshot program

- No side-effects are visible until the end of an entire XQuery program
 - Not to concurrent XQuery applications
 - Not to the current application either
- Database tradition
- Consequence:
 - Good: old optimization/reorganization of the code is still the same
 - Bad: very hard to write complex applications

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

What are users trying to do with XQuery

- Process XML
- Write complex applications
- Apply computations on input values coming from several data sources, and then output new fragments of XML data as result
- Examples:
 - HealthCare
 - Financial
 - Human resources
 - Business data

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

New use case scenarios

- Implementation of Web Services
- XML Data transformation and integration of heterogeneous data sources (data mashups)
- Processing RSS feeds and other XML message streams
- Coordination of services in an SOA environment
- XML data cleansing or normalization
- Complex manipulations of persistent XML data

Prof. Dr. Peter D. Stoop, peter.stoop@ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

Are such customers happy with XQuery?

- Sure. They like XQuery.
- But are not entirely satisfied.
- Even with the update extension certain pieces of their application logic must still be expressed *outside* the XQuery/XML world.
- They lack support for common programming tasks
- The friction/cost between “inside” and “outside” of an XML world is high
 - Productivity *and* performance

Prof. Dr. Peter D. Stoop, peter.stoop@ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

What are they missing ?

1. The ability to preserve state during computation
All functional programming languages have variable assignment !
2. The ability to “see” the results of their side-effects during the computation
3. The ability to invoke external side-effecting functions (e.g. Web Services) that cannot participate in snapshot semantics
4. The ability to recover (in a controlled way) from dynamic errors
5. The ability to model graphs

Prof. Dr. Peter D. Stoop, peter.stoop@ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

XQueryP overview

- A small language extension
 - More computation can be expressed directly in XQuery
 - Minimize the friction between the inner and outer of an XML world
- A big step towards helping our customers build richer XML applications
 - Supported by Oracle, BEA, DataDirect, etc
 - Proposed to the W3C
- Surprisingly: very small extensions can make XQueryP a good and complete XML processing language

Prof. Dr. Peter D. Stoop, peter.stoop@ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

The XQueryP concrete proposal

1. A well-defined evaluation order for XQuery expressions (“sequential order”)
2. Reduce the granularity of the snapshot to each individual atomic update expression
3. Adds new expressions:
 - Block
 - Set
 - While
 - Break, Continue
4. An error handling facility (try-catch)
5. A way of mapping XQueryP <-> Web Services
6. A way of modeling graphs in XML

The entire package is proposed as an optional feature

Prof. Dr. Peter D. Stoop, peter.stoop@ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

(1) Sequential evaluation order

- Slight modification to existing rules:
 - FLWOR: FLWO clauses are evaluated first; result in a tuple stream; then Return clause is evaluated in order for each tuple. Side-effects made by one row are visible to the subsequent rows.
 - COMMA: subexpressions are evaluated in order
 - (UPDATING) FUNCTION CALL: arguments are evaluated first before body gets evaluated

Required (only) if we add side-effects immediately visible to the program: e.g. variable assignments or single snapshot atomic updates; otherwise semantics not deterministic.

Prof. Dr. Peter D. Stoop, peter.stoop@ethz.ch

(2) Reduce snapshot granularity

- Today update snapshot: entire query
- Change:
 - Every single atomic update expression (insert, delete, rename, replace) is executed and made effective immediately
 - The effects of side-effecting external functions are visible immediately
- Semantics is deterministic because of the sequential evaluation order (point1)

(3) Adding new expressions

- Block expressions
- Assignment expressions
- While expressions
- Break, Continue
- Early return in function bodies

▪ Only under sequential evaluation mode

Block expression

- Syntax:


```
"{" ( BlockDecl ";" )* Expr (";" Expr)* "}"
```

BlockDecl :=
("declare" \$VarName TypeDecl? (":=" ExprSingle) ?)?
(";" \$VarName TypeDecl? (":=" ExprSingle) ?)*
- Semantics:
 - Declare a set of updatable variables, whose scope is only the block expression (in order)
 - Evaluate each expression (in order) and make the effects visible
 - Return the value of the last expression
- Updating if body contains an updating expression
- In the function syntax we change
 - EnclosedExpr /" the body "/ => Block
- We relax the fact the a function cannot update some nodes and return a value

Assignment expression

- Syntax:


```
"set" $VarName ":=" ExprSingle
```
- Semantics:
 - Change the value of the variable
 - Variable has to be external or declared in a block (no let, for or typeswitch)
- Updating expression
- Semantics is deterministic because of the sequential evaluation order

While expression

- Syntax:


```
"while" "(" exprSingle ")" "return" expr
```
- Semantics:
 - Evaluate the test condition
 - If "true" then evaluate the return clause; repeat
 - If "false" return the concatenation of the values returned by all previous evaluations of return
- Syntactic sugar, mostly for convenience
 - Could be written using recursive functions

Break, Continue, Return

- Traditional semantics, nothing surprising
- *Break* (or *continue*) the closest FLWOR or WHILE iteration
- *Return*: early exit from a function body
- Hard(er) to implement in a "database" style evaluation engine
 - Because of the lazy evaluation

Atomic Blocks

- **Syntax:**
"atomic" "{ ... }"
- **Semantics:**
 - If the evaluation of Expr does not raise errors, then result is returned
 - If the evaluation of Expr raises a dynamic error then no partial side-effects are performed (all are rolled back) and the result is the error
- Only the largest atomic scope is effective

Putting everything together: sequential mode

- New setter in the prolog
- **Syntax:**
"declare" "execution" "sequential"
- **Granularity:** query or module
- **What does it mean:**
 - Sequential evaluation mode for expressions
 - Single atomic update snapshot
 - Several new updating expressions (blocks, set, while, break, continue)
- If the query has no side-effects, sequential mode is irrelevant, and traditional optimizations are still applicable

Sequential mode and optimization

- Sequential mode required to ensure deterministic semantics in case of visible side-effects
- In theory, more constraints on evaluation order implies less optimizations
- Decades-old tension between
 - adding side-effects
 - still allowing the optimizations
- **Compromise to be made**
 - Errors might still be allowed to be non-deterministic ?
- The idea that optimization is not possible anymore is certainly not true
 - More complex dataflow analysis and intelligence required to trace when side-effects are being applied, on what data, when two side-effects commute, etc

(4) Error handling

- Errors in XQuery 1.0, Xpath 2.0, XSLT 2.0
 - `fn:error(err:USER0005, "Value out of range", $value)`
- **Design for try-catch**

```
try ( target-expr )
catch ( $name as QName1, $desc, $obj )
return handler-expr1
catch ( $name as QName2, $desc, $obj )
return handler-expr2 . . .
default ( $name, $desc, $obj )
return general-handler-expr
```
- **Example**

```
let $x := expr
return
try ( <a>{ $x }</a> )
catch ( err:XQTY0024 )
return <a> { $x[self::attribute()], $x[fn:not(self::attribute())] } </a>
```

(5) Invoking and coordinating Web Services

- WS are the standard way of *sending and receiving* XML data
- XQuery (and XSLT) are the standard way to *program* the XML processing

XQuery	Web Services
module	service
functions/operations	operations
arguments	ports
values for arguments and Result: XML	value for input and output messages: XML

- XQueryP proposes:
 - A standard way of importing a Web Services into XQuery
 - A standard way of invoking a WS operation as a normal function
 - A standard way of exporting an XQuery module as a Web Service
- Many XQuery implementations already support this. We have to agree on a standard.

Calling Google...

```
import service namespace
ws="http://api.google.com/GoogleSearch.wsdl";

declare execution sequential;
declare variable $result;
declare variable $query;

set $query := mxq:readLine();
set $result := ws:doGoogleSearch(„devkey“, $query, 0,10, fn:true(), "",
fn:false(), "", "UTF-8", "UTF-8");

<results query="{ $query }">
{
  for $url in $result/resultElements/item/URL
  return data($url)
}
</results>
```

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

XQueryP usage scenarios

- XQueryP programs in the browsers
 - We all love Ajax (the results). A pain to program. Really primitive as XML processing goes.
 - Embedding XQueryP in browsers
 - XQueryP code can take input data from WS, RSS streams, directly from databases
 - Automatically change the XHTML of the page
- XQueryP programs in the databases
 - Complex data manipulation executed inside the database
 - Takes advantage of the DB performance, scalability, security, etc
- XQueryP programs in application servers
 - Orchestration of WS calls, together with data extraction for a variety of data sources (applications, databases, files), and XML data transformations
 - XML data mashups

26

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Frequent criticism

- "Programmers do not know how to program declaratively"
 - What about SQL !?
- "If you give users variable assignment, they'll use it (and abuse it)!"
 - Teach them not to, rewrite automatically if they (still) do
- "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)
- "This would require users to learn a new language"
 - Smooth transition, easy integration of pieces written in other languages (thanks WS!)
- "XQuery is too complicated"
 - ?

26

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Automatic XQueryP code generation

- Our goal will only be achieved if the code can be automatically generated, based on:
 - Metadata
 - Higher-level description (UI)
- Better chances with XQueryP then with any other alternative technology
 - The higher level of abstraction, the easier
- Very important open research problem...

27

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

XQueryP's potential impact

Let's imagine the following scenario:

- OracleDB, BerkeleyDB, MySQL, all run XQueryP
- (or more, XQueryP executable can be part of a normal Linux installation..)
- Any HTTP server understands XQueryP, is able to execute it, and responds in XML
- Content managers understand XML/XQuery
- XQueryP runs in the browsers
- XQueryP runs on all kinds of mobile devices
- We manage to optimize it properly (<28ms)
- We find a good UI paradigm for automatically generating XQueryP code

28

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Potential impact, conclusion

- What will happen to the N-tiered architectures ?
- What will happen to client-server architectures ?
- What will happen to the cost of building and changing an application ?
- One step closer to the goal of XML technologies
 - Enable the flow of digital information
 - Automatize the processing the information
 - Decrease the cost of building and running Web applications
 - Transform Web applications in a commodity

29