


ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

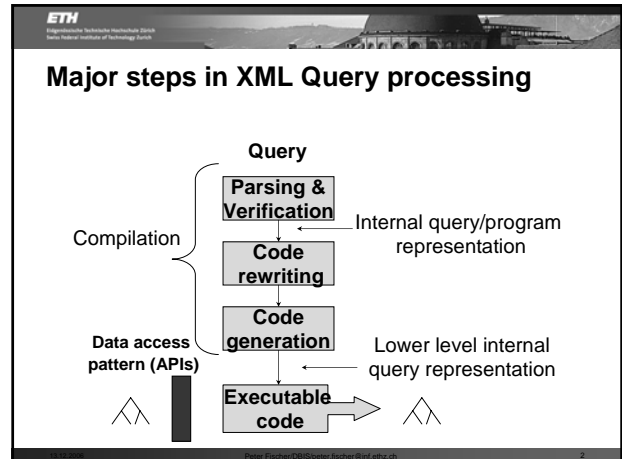
Module 4

Implementation of XQuery

Part 2: Data Storage



13.12.2006



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

Questions to ask for XML data storage

- **What** actions are done with XML data?
- **Where** does the XML data live?
- **How** is the XML data processed?
- **In which** granularity is XML data processed?
- **There is no one fits all solution !?!**
(This is an open research question.)

13.12.2006 Peter Fischer, DB/Systems Group, ETH Zürich

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

What?

- Possible uses of XML data
 - ship (serialize)
 - validate
 - query
 - transform (create new XML data)
 - update
- Example:
 - UNICODE reasonably good to ship XML data
 - UNICODE terrible to query XML data

13.12.2006 Peter Fischer, DB/Systems Group, ETH Zürich

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

Where?

- Possible locations for XML data
 - wire (XML messages)
 - main-memory (intermediate query results)
 - disk (database)
 - (mobile devices)
- Example
 - Compression great for wire and mobile devices
 - Compression not good for main-memory (?)

13.12.2006 Peter Fischer, DB/Systems Group, ETH Zürich

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zürich

How?

- Alternative ways to process XML data
 - materialized, all or nothing
 - streaming (on demand)
 - anything in between
- Examples
 - trees good for materialization
 - trees bad for stream-based processing

13.12.2006 Peter Fischer, DB/Systems Group, ETH Zürich

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Granularity?

- Possible granularities for data processing:
 - documents
 - items (nodes and atomic values)
 - tokens (events)
 - bytes
- Example
 - tokens good for fine granularity (items)
 - tokens bad for whole documents

Peter Fischer, DB/Systems, peter.fischer@inf.ethz.ch 7

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Scenario I: XML Cache

- Cache XHTML pages or results of Web Service calls

ship	yes	wire	yes	materialize	yes
validate	maybe	m.-m.	yes	stream	maybe
query	no	disk	yes	granularity	docs/ items
transform	maybe				
update	no				

Peter Fischer, DB/Systems, peter.fischer@inf.ethz.ch 8

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Scenario II: Message Broker

- Route messages according to simple XPath rules
- Do simple transformations

ship	yes	wire	yes	materialize	no
validate	yes	m.-m.	yes	stream	yes
query	yes	disk	no	granularity	docs
transform	yes				
update	no				

Peter Fischer, DB/Systems, peter.fischer@inf.ethz.ch 9

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Scenario III: XQuery Processor

- apply complex functions
- construct query results

ship	no	wire	yes	materialize	yes
validate	yes	m.-m.	yes	stream	yes
query	yes	disk	maybe	granularity	item
transform	yes				
update	no				

Peter Fischer, DB/Systems, peter.fischer@inf.ethz.ch 10

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Scenario IV: XML Database

- Store and archive XML data

ship	yes	wire	no	materialize	yes
validate	yes	m.-m.	yes	stream	yes
query	yes	disk	yes	granularity	collection ?
transform	yes				
update	yes				

Peter Fischer, DB/Systems, peter.fischer@inf.ethz.ch 11

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Object Stores vs. XML Stores

- Similarities
 - nodes are like objects
 - identifiers to access data
 - support for updates
- Differences
 - XML: tree not graph
 - XML: everything is ordered
 - XML: streaming is essential
 - XML: dual representation (lexical + binary)
 - XML: data is context-sensitive

Peter Fischer, DB/Systems, peter.fischer@inf.ethz.ch 12

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

XML Data Representation Issues

- Data Model Issues
 - InfoSet vs. PSVI vs. **XQuery data model**
- Storage Structures basic Issues
 1. Lexical-based vs. typed-based vs. both
 2. Node identifiers support
 3. Context-sensitive data (namespaces, base-uri)
 - ...
- Storage alternatives:
 - Plain text
 - Trees
 - Arrays of Tokens
 - Binary XML
 - (Relational) tables

Peter Fischer, DB Systems, fischer@inf.ethz.ch 13

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Lexical vs. Type-based

- Data model requires both properties, but allows only one to be stored and compute the other
- Functional dependencies
 - string + type annotation -> value-based
 - value + type annotation -> schema-norm. string
- Example


```
"0001" + xs:integer -> 1
1 + xs:integer -> "1"
```
- Tradeoffs:
 - Space vs. Accuracy
 - Redundancy: cost of updates
 - indexing: restricted applicability

Peter Fischer, DB Systems, fischer@inf.ethz.ch 14

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Node Identifiers Considerations

- XQuery Data Model Requirements
 - identify a node uniquely (implements identity)
 - lives as long as node lives
 - robust to updates
- Identifiers might include additional information
 - Schema/type information
 - Document order
 - Parent/child relationship
 - Ancestor/descendent relationship
 - Document information
- Required for indexes

Peter Fischer, DB Systems, fischer@inf.ethz.ch 15

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Simple Node Identifiers

- Examples:
 - Alternative 1 (data: trees)
 - id of document (integer)
 - pre-order number of node in document (integer)
 - Alternative 2 (data: plain text)
 - file name
 - offset in file
- Encode document ordering (Alternative 1)
 - identity: doc1 = doc2 AND pre1 = pre2
 - order: doc1 < doc2
OR (doc1 = doc2 AND pre1 < pre2)
- Not robust to updates
- Not able to answer more complex queries

Peter Fischer, DB Systems, fischer@inf.ethz.ch 16

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Dewey Order

Tatrinov et al. 2002

- Idea:
 - Generate surrogates for each path
 - 1.2.3 identifies the third child of the second child of the first child of the given root
- Assessment;
 - **good**: order comparison, ancestor/descendent easy
 - **bad**: updates expensive, space overhead
- Improvement: ORDPATH Bit Encoding
 O'Neil et al. 2004 (Microsoft SQL Server)

Peter Fischer, DB Systems, fischer@inf.ethz.ch 17

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Example: Dewey Order

```

graph TD
  1((1 person)) --- 1.1((1.1 name))
  1 --- 1.2((1.2 child))
  1.2 --- 1.2.1((1.2.1 person))
  1.2.1 --- 1.2.1.1((1.2.1.1 name))
  1.2.1 --- 1.2.1.2((1.2.1.2 hobby))
  1.2.1 --- 1.2.1.3((1.2.1.3 hobby))
  
```

Peter Fischer, DB Systems, fischer@inf.ethz.ch 18

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

XML Storage Alternatives

- Plain Text (UNICODE)
- Trees with Random Access
- Binary XML / arrays of events (tokens)
- Tuples (e.g., mapping to RDBMS)

Prof. Fischer, DB Systems, fischer@inf.ethz.ch

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Plain Text

- Use XML standards to encode data
- Advantages:
 - simple, universal
 - indexing possible
- Disadvantages:
 - need to re-parse (re-validate) all the time
 - no compliance with XQuery data model (collections)
 - not an option for XQuery processing

Prof. Fischer, DB Systems, fischer@inf.ethz.ch

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Trees

- XML data model uses tree semantics
 - use Trees/Forests to represent XML instances
 - annotate nodes of tree with data model info
- Example

```

<f1>
  <f2>..</f2> <f3>..</f3>
  <f4> <f7/> <f8>..</f8> </f4>
  <f5/> <f6>..</f6>
</f1>
  
```

Prof. Fischer, DB Systems, fischer@inf.ethz.ch

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Trees

- Advantages
 - natural representation of XML data
 - good support for navigation, updates index built into the data structure
 - compliance with DOM standard interface
- Disadvantages
 - difficult to use in streaming environment
 - difficult to partition
 - high overhead: *mixes indexes and data*
 - index everything
- Example: DOM, *others*
- Lazy trees possible: minimize IOs, able to handle large volumes of data

Prof. Fischer, DB Systems, fischer@inf.ethz.ch

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Trees on disk: Natix

Moerkotte 2000ff

- Each sub-tree is stored in a *record*
- Store records in blocks as in any database
- If record grows beyond size of block: *split*
- Split: establish *proxy nodes* for subtrees
- Technical details:
 - use B-trees to organize space
 - use special concurrency & recovery techniques

Prof. Fischer, DB Systems, fischer@inf.ethz.ch

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Natix

```

<bib>
  <book>
    <title>...</title>
    <author>...</author>
  </book>
</bib>
  
```

Prof. Fischer, DB Systems, fischer@inf.ethz.ch

Binary XML as a flat array of „events“

- Linear representation of XML data
 - pre-order traversal of XML tree
- Node -> array of events (or tokens)
 - tokens carry the data model information
- Advantages
 - good support for stream-based processing
 - low overhead: separate indexes from data
 - logical compliance with SAX standard interface
- Disadvantages
 - difficult to debug, difficult programming model

Example Binary XML as tokens array

```
<?xml version="1.0">
<order id="4711" >
  <date>2003-08-19</date>
  <lineitem xmlns = "www.boo.com" >
  </lineitem>
</order>
```

Translating XML into Event Stream (I)

```
<?xml version="1.0">
<order id="4711" >
  <date>2003-08-19</date>
```

```
BeginDocument()
BeginElement(„order“, „xs:untypedAny“, 1)
BeginAttribute(„id“, „xs:untypedAtomic“, 2)
CharData(„4711“)
EndAttribute()
BeginElement(„date“, „xs:untypedAny“, 3)
Text(„2003-08-19“, 4)
EndElement()
```

Translating XML into Event Stream (II)


```
<lineitem xmlns = „www.boo.com“ >
</lineitem>
</order>
```

```
BeginElement(„www.boo.com:lineitem“,
„xs:untypedAny“, 5)
Namespace(„www.boo.com“, 6)
EndElement()
EndElement()
EndDocument()
```

Storing Event Stream as Token Sequence

```
BeginDocument()
BeginElement(„order“, „xs:untypedAny“, 1)
BeginAttribute(„id“, „xs:untypedAtomic“, 2)
CharData(„4711“)
EndAttribute()
BeginElement(„date“, „xs:untypedAny“, 3)
Text(„2003-08-19“, 4)
EndElement()
BeginElement(„www.boo.com:lineitem“, „xs:untypedAny“, 5)
Namespace(„www.boo.com“, 6)
EndElement()
EndDocument()
```

AttributeToken
(attributeName = "id",
dataType =
xs:untypedAtomic,
nodeID = 2)



Binary XML

- Discussion as part of the W3C
- Processing XML is only *one* of the target goals
- Other goals:
 - Data compression for transmission: WS, mobile
- Open questions today: can we achieve all goals with a single solution ? Will it be disruptive ?
- Data model questions: Infoset or XQuery Data Model ?
- Is streaming a strict requirement or not ?
- More to come in the next months/years.

XML Data represented as tuples

- Motivation: Use an RDBMS infrastructure to store and process the XML data
 - transactions
 - scalability
 - richness and maturity of RDBMS
- Alternative relational storage approaches:
 - Store XML as Blob (text, binary)
 - *Generic shredding of the data (edge, binary, ...)*
 - *Map XML schema to relational schema*
 - Binary (new) XML storage integrated tightly with the relational processor (=> see before, maybe also in a later talk)

Mapping XML to tuples

- External to the relational engine
 - Use when :
 - The structure of the data is relatively simple and fixed
 - The set of queries is known in advance
 - Processing involves hand written SQL queries + procedural logic
 - Frequently used, but not advantageous
 - Very expensive (performance and productivity)
 - Server communication for every single data fetch
 - Very limited solution
- Internally by the relational engine
 - See next slides

XML Example

```

<person, id = 4711>
  <name> Lilly Potter </name>
  <child> <person, id = 314>
    <name> Harry Potter </name>
    <hobby> Quidditch </hobby>
  </child>
</person>
<person, id = 666>
  <name> James Potter </name>
  <child> 314 </child>
</person>

```

```

<person, id = 4711>
  <name> Lilly Potter </name>
  <child> <person, id = 314>
    <name> Harry Potter </name>
  </child>
</person>
<person, id = 666>
  <name> James Potter </name>
  <child> 314 </child>
</person>

```

Edge Approach
(Florescu & Kossmann 99)

Source	Label	Target
0	person	4711
0	person	666
4711	name	v1
4711	child	i314
666	name	v2
666	child	i314

Id	Value
v1	Lilly Potter
v2	James Potter
v3	Harry Potter

Id	Value
v4	12

Binary Approach
Partition Edge Table by Label

Source	Target
0	4711
0	666
i314	314

Source	Target
4711	v1
666	v2
314	v3

Source	Target
4711	i314
666	i314

Source	Target
314	v4

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Tree Encoding

(Grust 2004)

- For every node of tree, keep info
 - pre**: pre-order number
 - size**: number of descendants
 - level**: depth of node in tree
 - kind**: element, attribute, name space, ...
 - prop**: name and type
 - frag**: document id (forests)

Prof. Fischer, DB Systems, fischer@inf.ethz.ch 37

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Example: Tree Encoding

pre	size	level	kind	prop	frag
0	6	0	elem	person	0
1	0	1	attr	id	0
2	0	1	elem	name	0
3	3	1	elem	child	0
...	0
0	3	0	elem	person	1

Prof. Fischer, DB Systems, fischer@inf.ethz.ch 38

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Schema-driven Mapping: DTD -> RDB

Shanmugasundaram et al. 1999

- Idea: Translate DTDs into Relations
 - Element Types -> Tables
 - Attributes -> Columns
 - Nesting (= relationships) -> Tables
 - „Inlining“ reduces fragmentation
- Special treatment for recursive DTDs
- Surrogates as keys of tables
- (Adaptions for XML Schema possible)

Prof. Fischer, DB Systems, fischer@inf.ethz.ch 39

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Example

```

<!ELEMENT book (title, author)>
<!ELEMENT article (title, author*)>
<!ATTLIST book price CDATA>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (firstname, lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ATTLIST author age CDATA>

```

Prof. Fischer, DB Systems, fischer@inf.ethz.ch 40

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Example: Relation „book“

```

<!ELEMENT book (title, author)>
<!ELEMENT article (title, author*)>
<!ATTLIST book price CDATA>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (firstname, lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ATTLIST author age CDATA>

```

- book**(bookID, book.price, book.title, book.author.fname, book.author.lname, book.author.age)
- Inlining possible, since 1:1 relation between book and author

Prof. Fischer, DB Systems, fischer@inf.ethz.ch 41

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Example: Relation „article“

```

<!ELEMENT book (title, author)>
<!ELEMENT article (title, author*)>
<!ATTLIST book price CDATA>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (fname, lname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ATTLIST author age CDATA>

```

- article** (artID, art.title)
- artAuthor** (artAuthorID, artID, art.author.fname, art.author.lname, art.author.age)
- 1:N article:author => separate tables, embed foreign key

Prof. Fischer, DB Systems, fischer@inf.ethz.ch 42

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Example (continued)

- Represent each element as a relation
 - element might be the root of a document

title(titleId, title)
author(authorId, author.age, author.fname, author.lname)
fname(fnameId, fname)
lname(lnameId, lname)

Prof. Dr. Peter F. Schuster, peter.f.schuster@ethz.ch

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Recursive DTDs

```

<!ELEMENT book (author)>
<!ATTLIST book title CDATA>
<!ELEMENT author (book*)>
<!ATTLIST author name CDATA>
  
```

- book**(bookId, book.title, book.author.name)
- author**(authorId, author.name)
- author.book**(author.bookId, authorId, author.book.title)

Prof. Dr. Peter F. Schuster, peter.f.schuster@ethz.ch

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

DTD Normalisation

- Simplify DTDs
 - $(e1, e2)^* \rightarrow e1^*, e2^*$ $(e1, e2)? \rightarrow e1?, e2?$
 - $(e1 | e2) \rightarrow e1?, e2?$ $e1^{**} \rightarrow e1^*$
 - $e1^*? \rightarrow e1^*$ $e1^{??} \rightarrow e1?$
 - $\dots, a^*, \dots, a^*, \dots \rightarrow a^*, \dots$
- Background
 - regular expressions
 - ignore order (not well supported in RDBMS)
 - generalized quantifiers (be less specific)

Prof. Dr. Peter F. Schuster, peter.f.schuster@ethz.ch

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Tradeoffs of tuple mapping variants (I)

- Store XML as a BLOB in relational database
 - index by materializing indexed expressions in separate columns
 - Plus: store XML in parsed and validated form
 - Minus: proprietary solution (blob is a black box)
 - Minus: replicate data for indexing
- Model-driven Shredding
 - Edge, binary approach + alternatives
 - Corresponds to generic APIs for Java
 - Plus: very general; integrates well with relational data
 - Minus: poor performance

Prof. Dr. Peter F. Schuster, peter.f.schuster@ethz.ch

ETH
 Eidgenössische Technische Hochschule Zürich
 Swiss Federal Institute of Technology Zürich

Tradeoffs of tuple mapping variants (II)

- Schema-based Shredding
 - Map XML Schema / DTD to SQL DDL
 - Plus: integrates well with relational data
 - Minus: missing tools, complicated
- SQL / XML
 - Not discussed, maybe after Christmas
 - Extend SQL with XML data type
 - Plus: integrates well with relational data
 - Minus: not clear how it integrates with application, odd „marriage“

Prof. Dr. Peter F. Schuster, peter.f.schuster@ethz.ch