

DBS2: Exkursus XQuery and XML-Databases

Jan Sievers
Jens Hündling
Lars Trieloff

Motivation

- XML ubiquitous data exchange format
- Can be used to present Object data, relational data, semi-structured data
- How to store XML data?

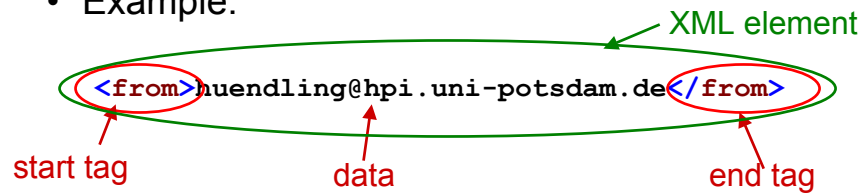
XML in a Nutshell

What is XML

- XML = Extensible Markup Language
 - designed to describe data
 - tags not predefined in XML
 - You must **define your own tags**.
- XML documents **can** describe data by
 - using DTD (**Document Type Definition**) or
 - using **XML Schema**
- XML + DTD or XML Schema
 - designed to be **self-descriptive**

Quick start in XML

- Important
 - set of rules for creating **tags**
 - contents: elements, attributes, (text) data
- XML **documents** to describe data
- NOT a programming language
- Example:



Application Areas for XML

- DB Technology
 - How to store and retrieve XML documents ?
 - Use relational DBS or build native XML DBS ?
- Business-To-Business (B2B) Integration
 - communication between companies
 - now: Information Integration based on XML
- Many other application areas
 - Scalable Vector Graphics (SVG)
 - XHTML

XML terminology

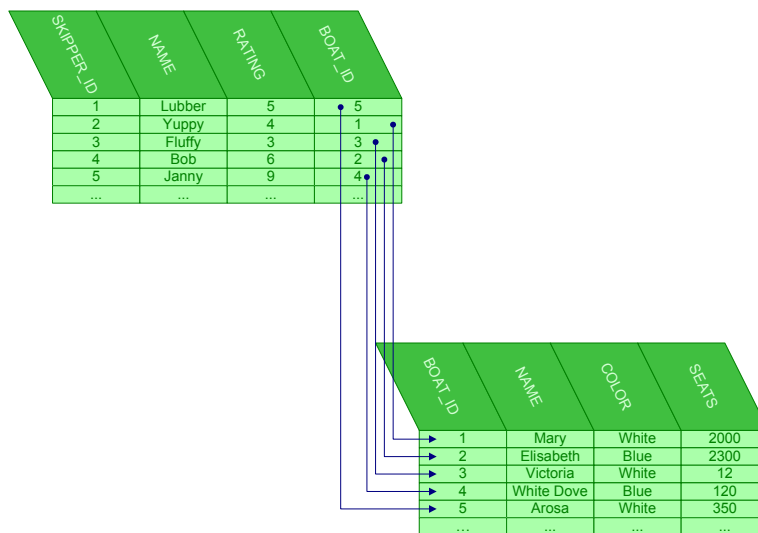
- XML element
 - start tag: `<subject>`
 - end tag: `</subject>`
 - `<subject>text data</subject>`
 - case sensitive: `<subject>` `<Subject>` `<SUBJECT>` different tags
- Attributes
 - Name-Value Pairs inside start tag (0 to n)
`<body Content-Type="text/plain" charset="iso-8859-1">`
 - equivalent
`<body Content-Type='text/plain' charset='iso-8859-1'>`

Name Conflicts

- element names are not globally unique
- conflicts will occur in some scenarios, e.g.
 - two different documents
 - using same names for elements
 - describing two different types of elements
- Solution: XML Namespaces
 - XML Element and Attribute Names are QNames
 - QName: Local Name and Namespace URI

XML Database Basics

Relations vs. Trees



Relations vs. Trees

SKIPPER_ID	NAME	RATING	BOAT_ID
1	Lubber	5	5
2	Yuppy	4	1
3	Fluffy	3	3
4	Bob	6	2
5	Janny	9	4
...

```

1: <?xml version="1.0"?>
2: <ship_company>
3:   <skipper rating="5">
4:     <name>Lubber</name>
5:     <ship color="White" seats="350">
6:       <name>Arosa</name>
7:     </ship>
8:   </skipper>
9:   <skipper rating="4">
10:    <name>Yuppy</name>
11:    <ship color="White" seats="2000">
12:      <name>Mary</name>
13:    </ship>
14:  </skipper>
15:  <skipper rating="3">
16:    <name>Fluffy</name>
17:    <ship color="White" seats="12">
18:      <name>Victoria</name>
19:    </ship>
20:  </skipper>
21:  <skipper rating="6">
22:    <name>Bob</name>
23:    <ship color="Blue" seats="2300">
24:      <name>Elisabeth</name>
25:    </ship>
26:  </skipper>
27:  <skipper rating="9">
28:    <name>Janny</name>
29:    <ship color="Blue" seats="120">
30:      <name>White Dove</name>
31:    </ship>
32:  </skipper>
33: </ship_company>
  
```

BOAT_ID	NAME	COLOR	SEATS
1	Mary	White	2000
2	Elisabeth	Blue	2300
3	Victoria	White	12
4	White Dove	Blue	120
5	Arosa	White	350
...

RDBMS vs. Native XMLDBMS

- Data is organized in Databases, Tables, Rows, Columns
- Organization through Foreign-Key-Relationship
- Tuples in Relations are unordered
- Data is organized in Collections, Documents, Elements, Attributes
- Organization through Hierarchy
- Nodes in XML Tree are ordered

Conclusion

- Different Data Structure
- SQL is not sufficient
- We need a new **query** language
 - Should work with all kinds of XML data
- DML is not sufficient
- We need a new **update** language
 - With support for XML data

XQuery: An XML query language

XQuery Basics

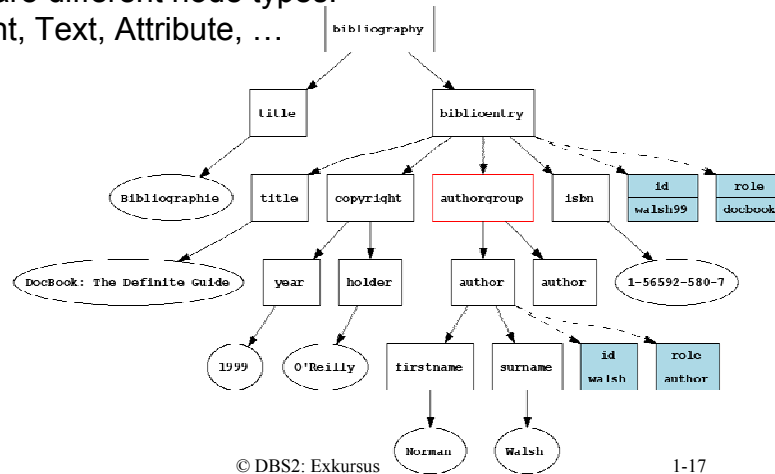
- XQuery is a declarative, functional language
- It provides
 - Navigation in XML document
 - Specify the documents to read from
 - Construction of new elements and attributes for the result
 - Binding and iteration of variables
 - Ordering results
 - Providing a function library
 - Validation of XML

Navigation – Path expressions

- Core functionality of XPath 2.0
 - subset of XQuery
- Select parts of the XML document

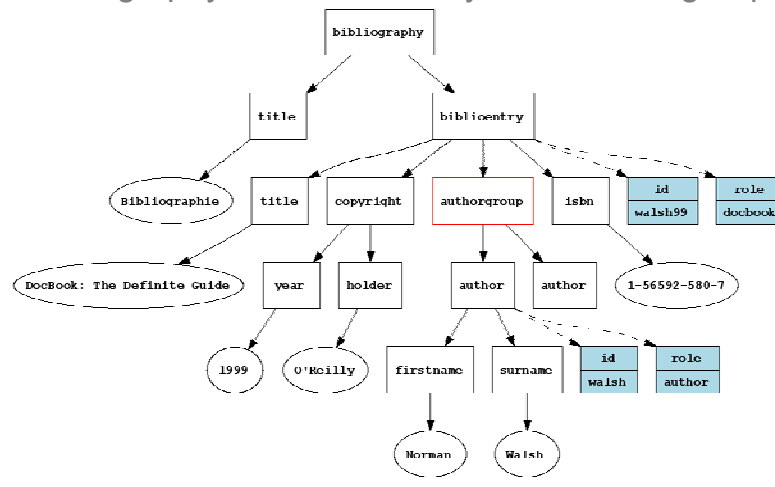
An XML Document is a Tree

- XML Documents are trees of nodes
- There are different node types:
Element, Text, Attribute, ...



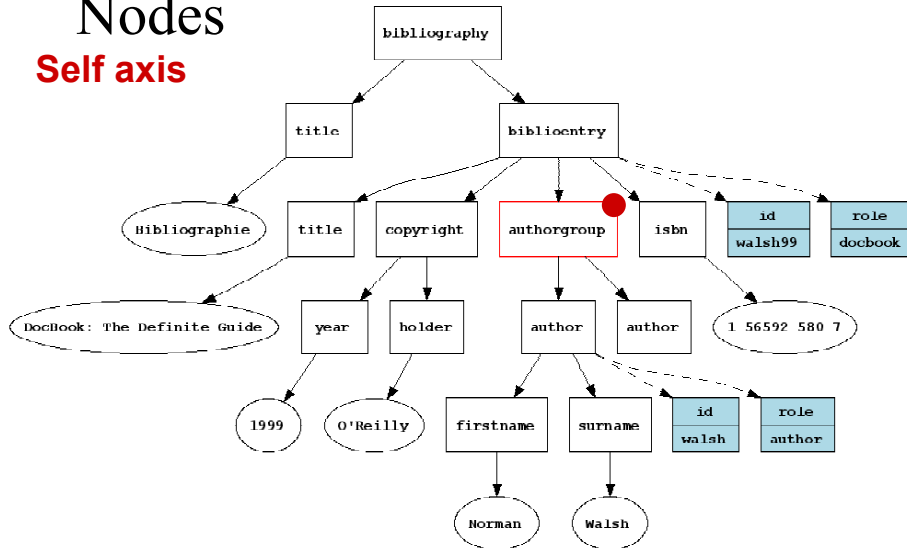
Example: Path Expression

- `/bibliography/biblioentry/authorgroup`
- `/child::bibliography/child::biblioentry/child::authorgroup`



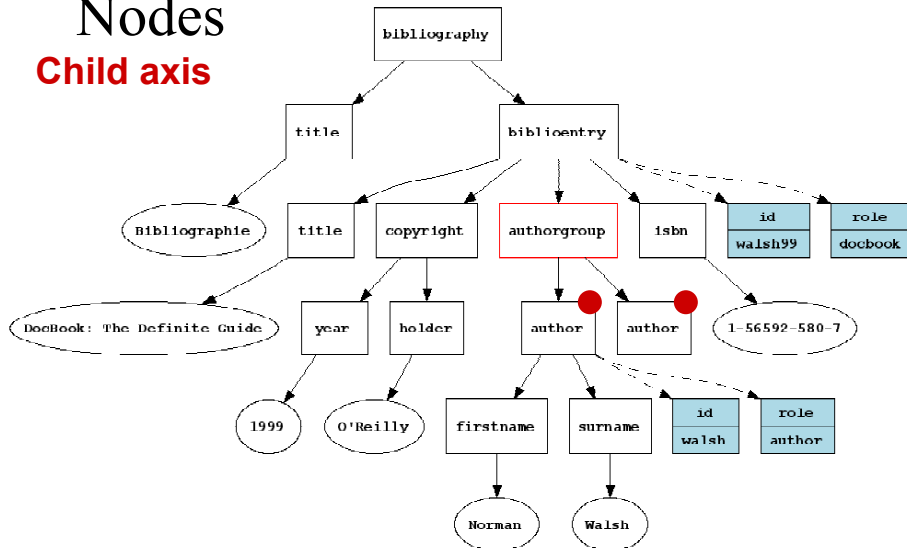
Axes: Relationships between Nodes

Self axis



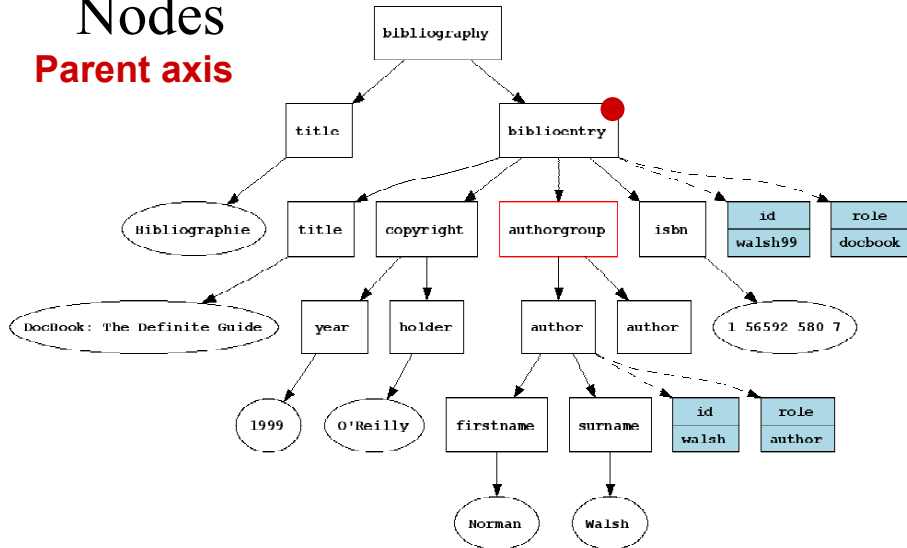
Axes: Relationships between Nodes

Child axis



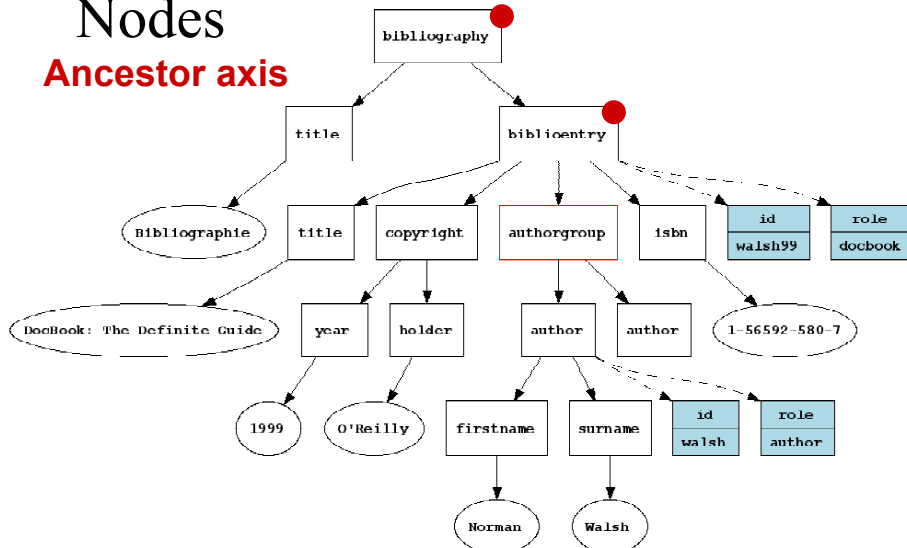
Axes: Relationships between Nodes

Parent axis



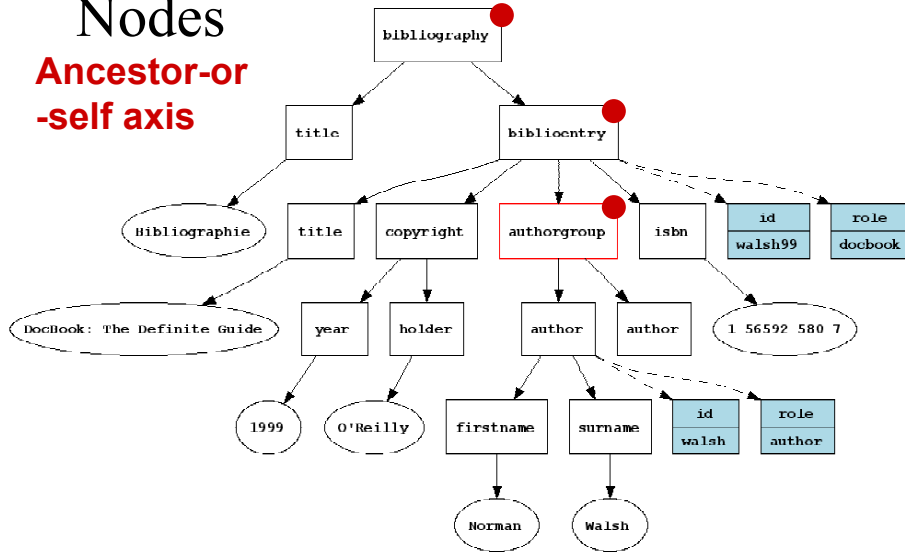
Axes: Relationships between Nodes

Ancestor axis



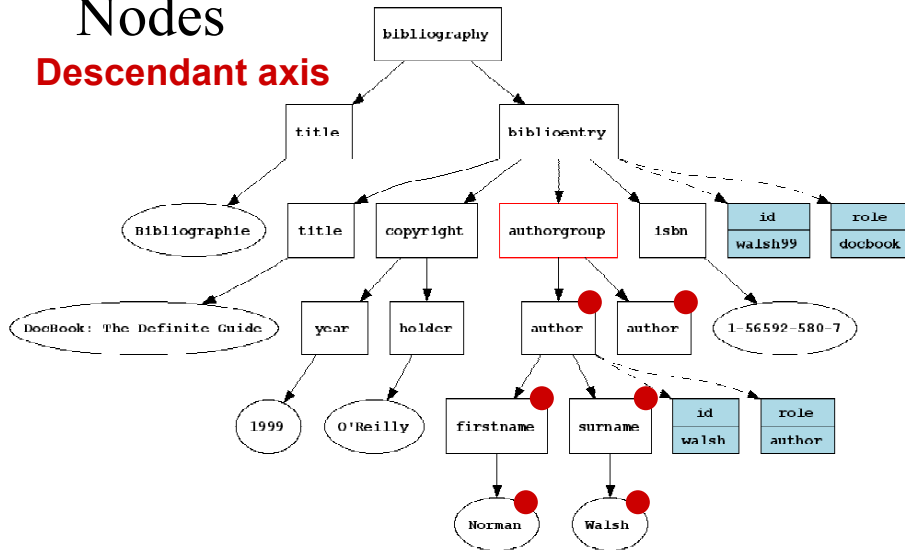
Axes: Relationships between Nodes

Ancestor-or-self axis



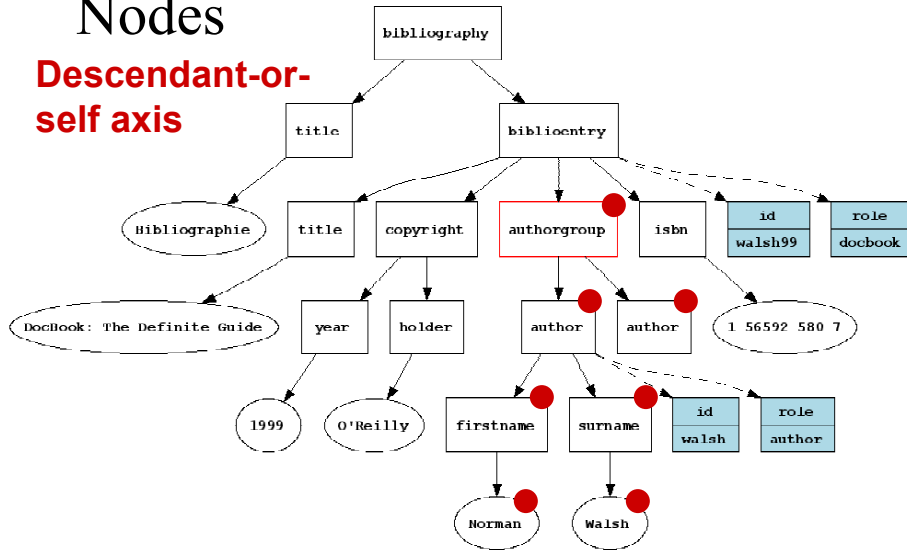
Axes: Relationships between Nodes

Descendant axis



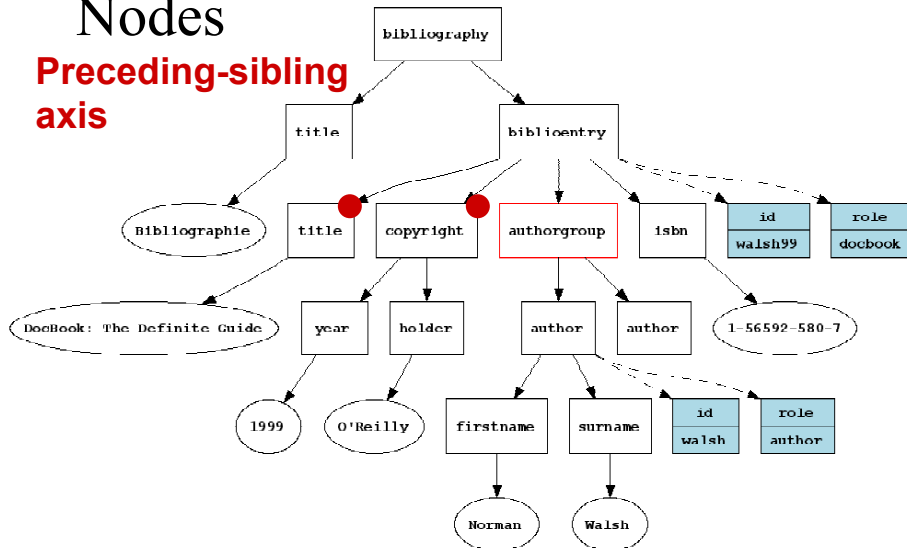
Axes: Relationships between Nodes

Descendant-or-self axis



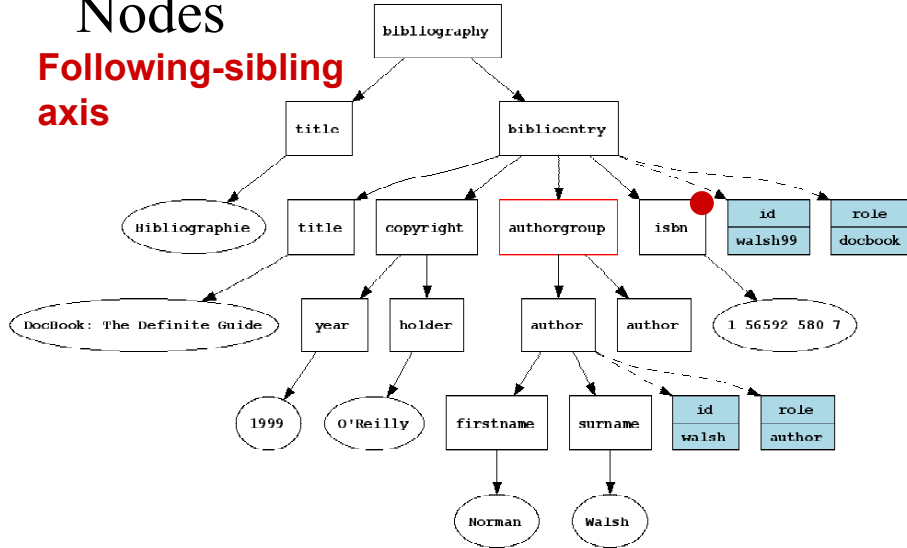
Axes: Relationships between Nodes

Preceding-sibling axis



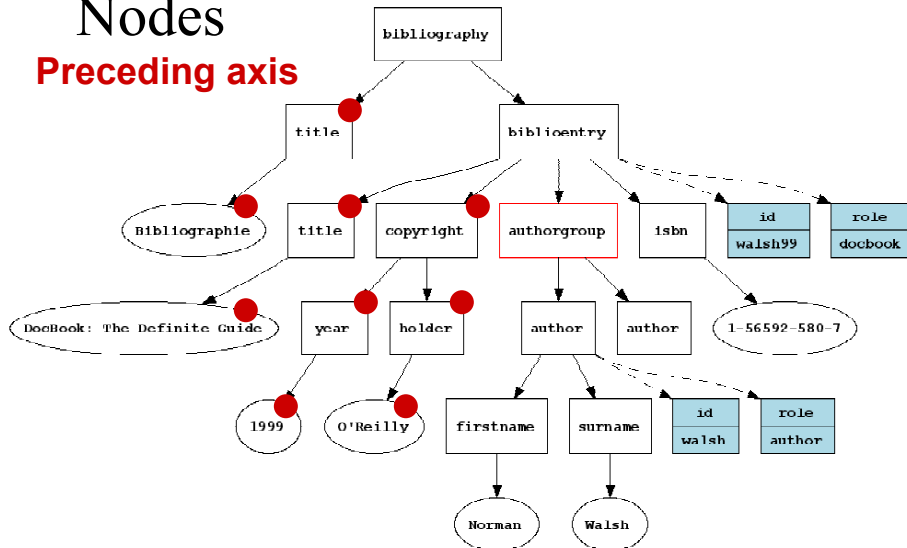
Axes: Relationships between Nodes

Following-sibling axis



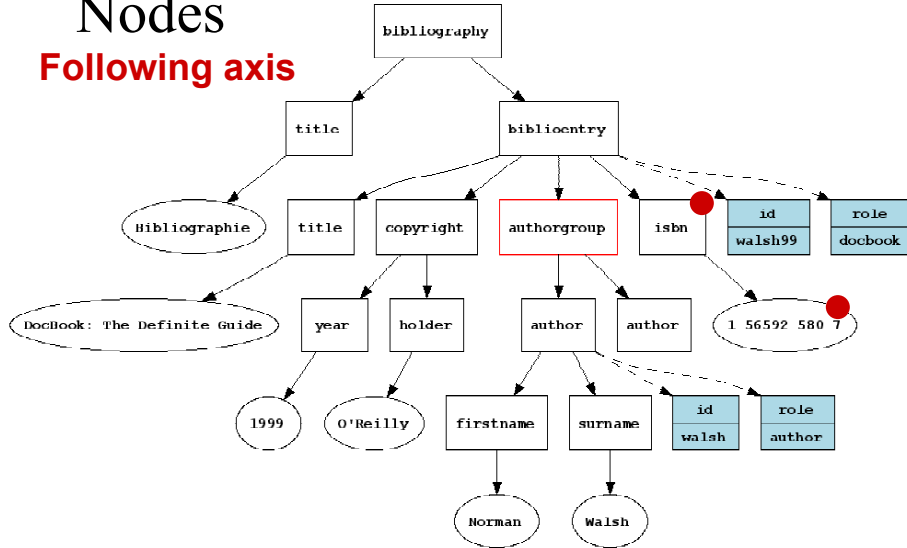
Axes: Relationships between Nodes

Preceding axis



Axes: Relationships between Nodes

Following axis

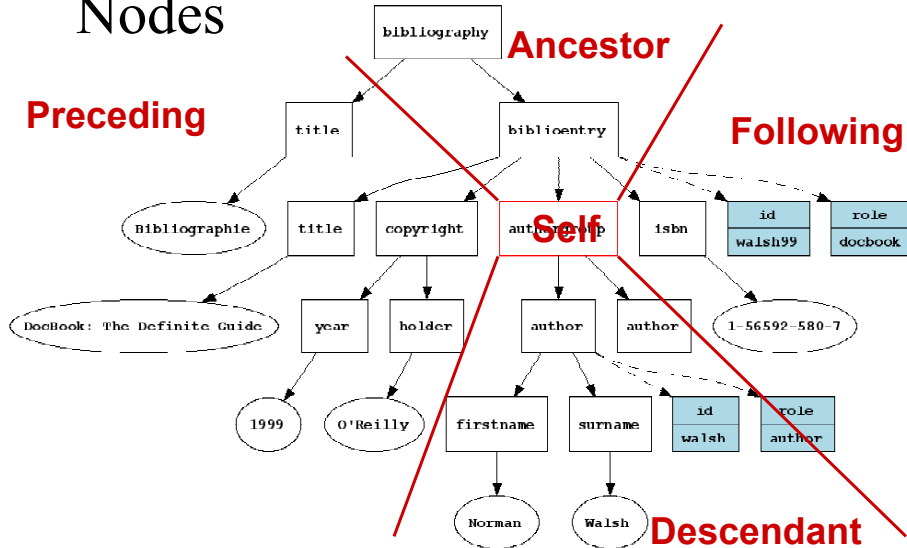


Axes: Relationships between Nodes

Preceding

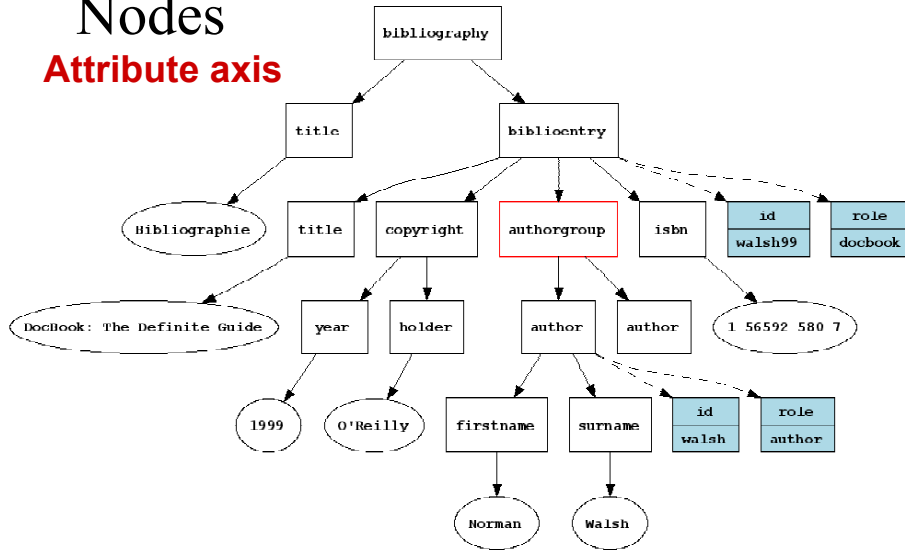
Ancestor

Following



Axes: Relationships between Nodes

Attribute axis



Shortcuts for Axes

Axis	Meaning	Abbreviation
ancestor	All ancestors of the context node (incl. parent)	none
ancestor-or-self	The context node and all its ancestors	none
attribute	All attributes of the context node	root/@attr
child	All children of the context node (does not include attributes)	root/foo
descendant	All descendants of the context node (does not include attributes)	root//foo
descendant-or-self	The context node and all its descendants (does not include attributes)	none
following	All nodes in the document following the context node in document order	none
following-sibling	All following siblings of the context node	none
parent	Parent node of the context node	root/foo/..
preceding	All nodes in the document preceding the context node in document order	none
preceding-sibling	The context node and all its preceding siblings	none
self	The context node	root/.

Node Tests

- Kind Tests
 - `element()`
 - `text()`
 - `node()`
 - Name Tests
 - `bar`
 - `foo:bar`
 - `*:bar`
 - `foo:*`
 - Examples
 - `/bar/child::element()`
 - `//surname/text()`
 - `/node()`
- Namespace:
`xhtml='http://w3.org/xhtml'`
- `/descendant::xhtml:img`
 - `/descendant::*:img`

Predicates

- Further reduction of the result sequence
 - `//skipper[@rating > 3]`
 - `//skipper[competence]`
 - `//skipper[competence/text() != "none"]`
 - `//skipper[@rating > 3][name = 'Yuppy']`
- Numeric predicates
 - `//ship[3]`
 - `//ship[last()]`
- Logical operations
 - `//skipper[@rating="1" or @rating="2"]`

Example

```
/ship_company/skipper[@rating < 3]/  
  ship[name = "Mary" or  
        name = "Victoria" or  
        name = "Elisabeth"]  
  [@seats > 100]
```

- Gets the ships which have a name of a British monarch, more than 100 seats and a skipper with bad ratings.

Further XQuery functionality

- Declare namespace prefixes in prolog
 - declare namespace foo =
"http://bar.com/"
- Specify documents to read from
 - doc("ships.xml")
 - collection("customers")
- Declare user defined functions
 - declare function math:pi() {
3.14159265
}

Creating elements and attributes

- using XML syntax
 - `<white_ships id='01' count='4'>`
 {
 //ship[@color = 'white']
 }
`</white_ships>`
- using computed constructors
 - `element white_ships`
 {
 attribute id {'01'},
 attribute count {
 fn:count(//ship[@color = 'white'])
 },
 //ship[@color = 'white']
 }

Joins

- FLWOR expression: For Let where OrderBy Return
- Equivalent to SELECT FROM WHERE
 - `for $skipper in doc("a.xml")//skipper,`
 `$ship in doc("b.xml")/ships/ship`
`let $ship_id := $ship/@id`
`where $skipper/@ship = $ship_id`
`return`
 `<reservation>`
 {
 \$skipper/name/text(),
 " sails ",
 \$ship/name/text()
 }
 `</reservation>`

Ordering results

```
- for $skipper in doc("a.xml")//skipper,  
    $ship in doc("b.xml")/ships/ship  
let $ship_id := $ship/@id  
where $skipper/@ship = $ship_id  
order by $skipper/rating  
return  
    <reservation>  
    {  
        $skipper/name/text(),  
        " sails ",  
        $ship/name/text()  
    }  
    </reservation>
```

XQuery function library

- Collection of powerful functions
- All in namespace
fn = <http://www.w3.org/2003/05/xpath-functions>
- Operate on
 - Numbers
 - Strings
 - Booleans
 - Durations, Dates, Times
 - QNames
 - Nodes and Sequences

What is Typing

- Queries can be evaluated before accessing the data storage by reading the Data Dictionary
- RDBMS offer strong typing through DDL
 - Every column has a defined type: INT, VARCHAR, BLOB, ...
- XML offers typing through XML Schema
 - Allows validation of documents
 - Used in XQuery

XML Schema and XQuery

- Allows static and dynamic type checking
- definition of functions with types
 - `declare foo:bar($a as xs:string) as xs:int`
- casting
 - `"2003" cast as xs:double`
- typeswitch
 - ```
typeswitch($customer/billing-address)
 case $a as element(*, USAddress)
 return $a/state
 case $a as element(*, CanadaAddress)
 return $a/province
 case $a as element(*, JapanAddress)
 return $a/prefecture
 default return "unknown"
```

# XUpdate and Database Connectivity

## An Update language for XMLDBMS

- Simple Update language specified by XML:DB group
- No official standards organization like W3C or OASIS, but XUpdate has wide adoption
- Other solutions: Proprietary extensions of XQuery

## What is XUpdate

- XML application, updates are specified in XML and XPath 1.0
- Possible operations
  - xupdate:insert-before
  - xupdate:insert-after
  - xupdate:append
  - xupdate:update
  - xupdate:remove
  - xupdate:rename
  - xupdate:variable
  - xupdate:value-of

## Database Connectivity

- RDBMS use JDBC or ODBC etc.
- There is no official standard for XMLDBMS, but
  - XML:DB API
    - By same standards body as XUpdate
    - For Java, .NET, Perl, Python, SOAP, XML-RPC
  - OJXQI by Oracle
  - XQJ by Sun and others (JSR 225)

# Thanks for your attention!

- Any Questions?

Lars@trieloff.net

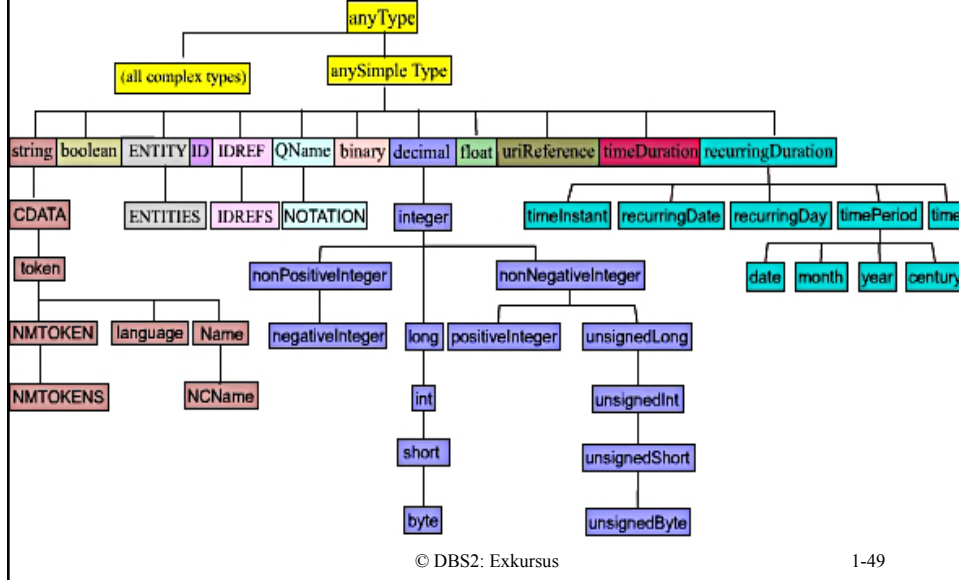
Mail@jan-sievers.de

Jens.Huendling@hpi.uni-potsdam.de

## XML Schema and XQuery

- An advanced schema language
- Supports
  - Namespaces
  - User defined types
  - Inheritance
- Massive Industry Adoption
  - Used in WSDL, XSLT 2.0, etc.
  - And **XQuery**

## XML Schema basic types



## Defining data types in XML Schema and XQuery

- ```

<xs:complexType
  name="User" />
<xs:sequence>
  <xs:element name="name">
    <xs:complexType>
      <xs:element name=
        "first" type="xs:string"
        minOccurs="0" />
      <xs:element
        name="last"
        type="xs:string" />
    </xs:complexType>
  </xs:element>
  <xs:element ref="rating"
    minOccurs="0" />
</xs:sequence>
<xs:attribute name="id"
  type="xs:ID"
  use="required">
</xs:complexType>

```
- ```

define type User {
 attribute id of type
 xs:ID,
 element name of type
 AnonymousType1,
 element rating ?
}
Define type
AnonymousType1 {
 element first of type
 xs:string ?,
 element last of type
 xs:string
}

```

## Data types in XQuery

- Allows static and dynamic type checking
- definition of functions with types
  - declare foo:bar(\$a as xs:string) as xs:int
- casting
  - "2003" cast as xs:double
- typeswitch
  - typeswitch(\$customer/billing-address)
    - case \$a as element(\*, USAddress)
      - return \$a/state
    - case \$a as element(\*, CanadaAddress)
      - return \$a/province
    - case \$a as element(\*, JapanAddress)
      - return \$a/prefecture
    - default return "unknown"