



# Applying GRIPP to XML Documents containing XInclude and XLink Elements

Silke Trißl, Florian Zipser & Ulf Leser

Humboldt-Universität zu Berlin

Knowledge Management in Bioinformatics

# Outline

---

- ❑ Problem definition
  - ❑ XML Documents are trees! – Really?
  
- ❑ Indexing XML Documents - GRIPP
  - ❑ Index Construction
  - ❑ Querying GRIPP
  
- ❑ Experimental Evaluation
  
- ❑ Conclusion

Problem  
Really trees?  
GRIPP  
Indexing  
Querying  
Experiments  
Conclusion

# XML Documents

❑ XML documents are trees! - **Really?**

❑ Consider Publications

```
<DBLP>
  ...
  <PAPER>
    <TITLE>Accelerating XPath location steps</TITLE>
    <AUTHORS>
      <AUTHOR>Torsten Grust</AUTHOR>
    </AUTHORS>
    <INPROCEEDINGS>SIGMOD 2002</INPROCEEDINGS>
    <REFERENCES> to other publications
  </PAPER>
  ...
</DBLP>
```

How to express this  
in XML?

Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

Conclusion

# XML Additions

---

- ❑ W3C specifies **XLink** und **XInclude** elements
  - ❑ XInclude
    - Element to include other parts of the document or other documents
  - ❑ XLink
    - Only a reference to other elements
    - Document can form a graph
- ❑ XPointer to address target element

Problem

Really trees?

GRIPP

Indexing

Querying

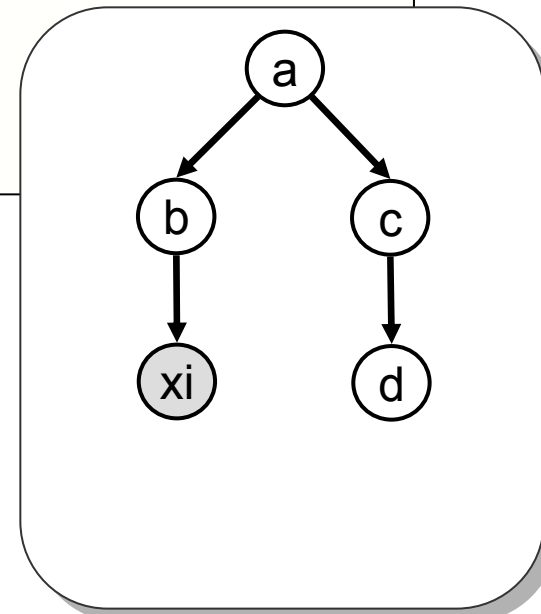
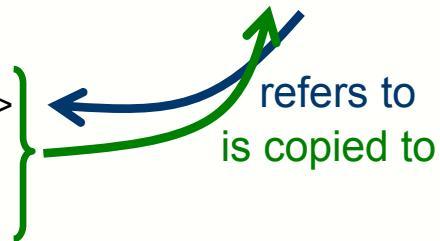
Experiments

Conclusion

# XInclude Elements

*a.xml*

```
<a xmlns:xi="http://www.w3.org/2001/XInclude/">
  <b xml:id="b">b
    <xi:include href="a.xml#c">
  </b>
  <c xml:id="c">
    <d>d</d>
  </c>
</a>
```



Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

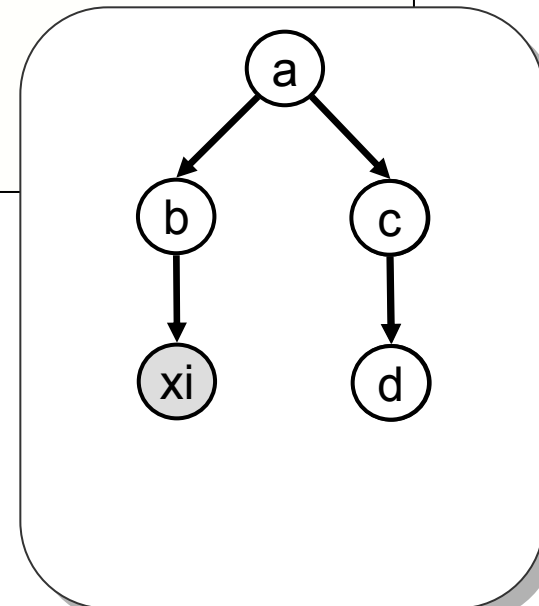
Conclusion

# XInclude Elements

*a.xml*

```
<a xmlns:xi="http://www.w3.org/2001/XInclude/">
  <b xml:id="b">b
    <c xml:id="c">
      <d>d</d>
    </c>
  </b>
  <c xml:id="c">
    <d>d</d>
  </c>
</a>
```

is copied to



- ❑ XInclude element is replaced during preprocessing
- ❑ Document forms a directed, acyclic graph (DAG)

Problem  
Really trees?  
GRIPP  
Indexing  
Querying  
Experiments  
Conclusion

# XML Additions

---

- ❑ W3C specifies [XLink](#) und [XInclude](#) elements
  - ❑ XInclude
    - Element to include other parts of the document or other documents
  - ❑ XLink
    - Only a reference to other elements
    - Document can form a graph
- ❑ XPointer to address target element

Problem

Really trees?

GRIPP

Indexing

Querying

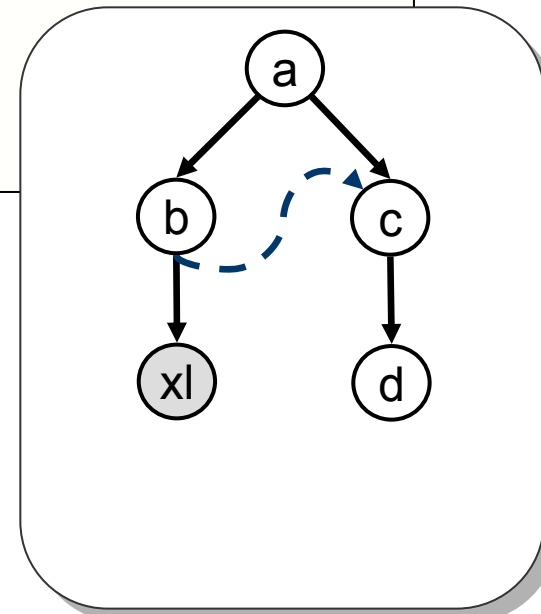
Experiments

Conclusion

# XLink Elements

*a.xml*

```
<a xmlns:xlink="http://www.w3.org/2001/XLink/">
  <b xml:id="b">b
    <xlink:simple xlink:href="a.xml#c" />
  </b>
  <c xml:id="c"> ← refers to
    <d>d</d>
  </c>
</a>
```



- ❑ Only a reference to target element
- ❑ Document can form a graph containing cycles

Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

Conclusion

# Outline

---

- Problem definition
  - XML Documents are trees! – Really?
  
- Indexing XML Documents - GRIPP
  - Index Construction
  - Querying GRIPP
  
- Experimental Evaluation
  
- Conclusion

Problem  
Really trees?  
GRIPP  
Indexing  
Querying  
Experiments  
Conclusion



# Indexing XML Documents - GRIPP

- ❑ XML documents with XLinks are **graphs**
- ❑ Index structure for graphs: **GRIPP**
  - ❑ Graph Indexing based on Pre- and Postorder Numbering [Trißl & Leser 2007]
  - ❑ Applicable to very large graphs
    - 5 million nodes, 10 million edges
  - ❑ Completely RDBMS-based

Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

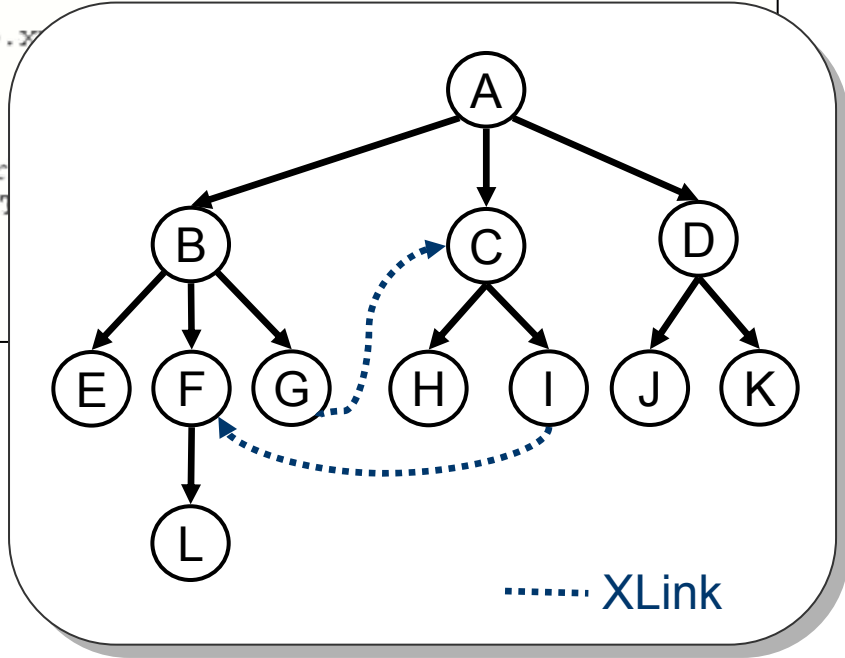
Conclusion

# Sample Document

```

<DBLP xml:id="A" xmlns:xlink="http://www.w3.org/1999/xlink/">
  <PAPER xml:id="B">
    <TITLE xml:id="E">Accelerating XPath Evaluation in any RDBMS</TITLE>
    <AUTHOR xml:id="F">Torsten Grust
      <ORGANIZATION xml:id="L">University of Konstanz</ORGANIZATION>
    </AUTHOR>
    <CROSSREF xml:id="G">
      <xlink:simple xlink:href="dblp.xml#C />
    </CROSSREF>
  </PAPER>
  <PAPER xml:id="C">
    <TITLE xml:id="H">Accelerating XPath location Steps</TITLE>
    <AUTHOR xml:id="I">
      <xlink:simple xlink:href="dblp.xml#B />
    </AUTHOR>
  </PAPER>
  <TECHNICAL_REPORT xml:id="D">
    <TITLE xml:id="J">Extensible Markup Language</TITLE>
    <AUTHOR xml:id="K">Tim Bray</AUTHOR>
  </TECHNICAL_REPORT>
  ...
</DBLP>

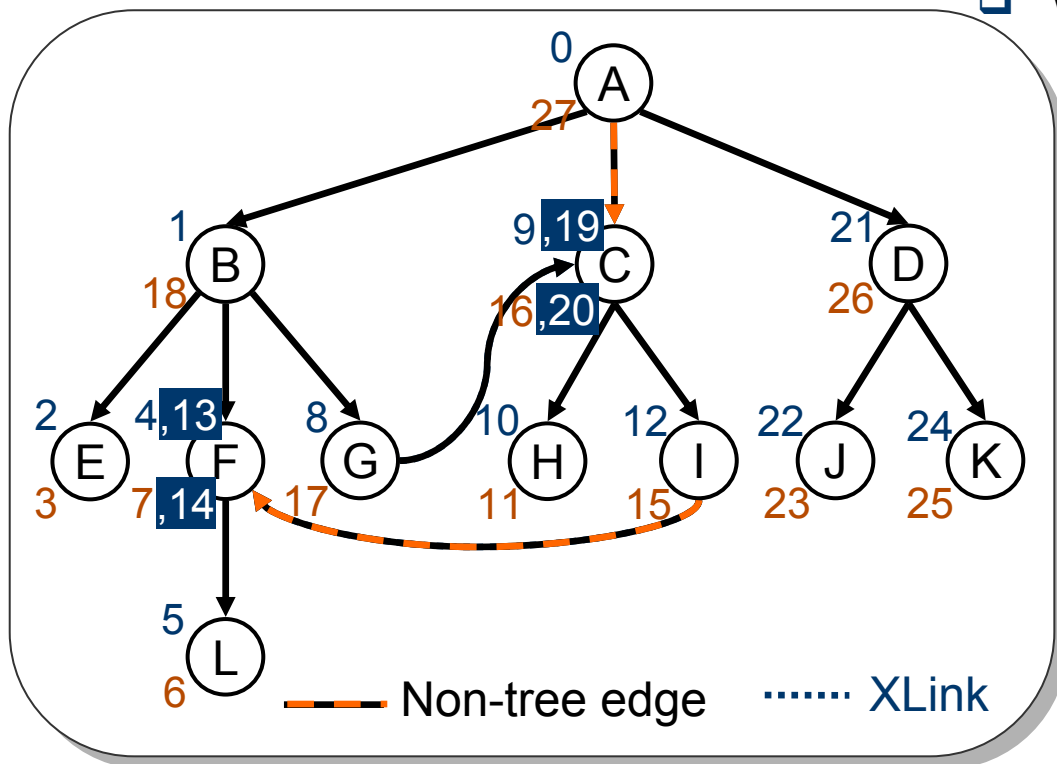
```



- Problem
- Really trees?
- GRIPP
- Indexing
- Querying
- Experiments
- Conclusion

# GRIPP – Index creation

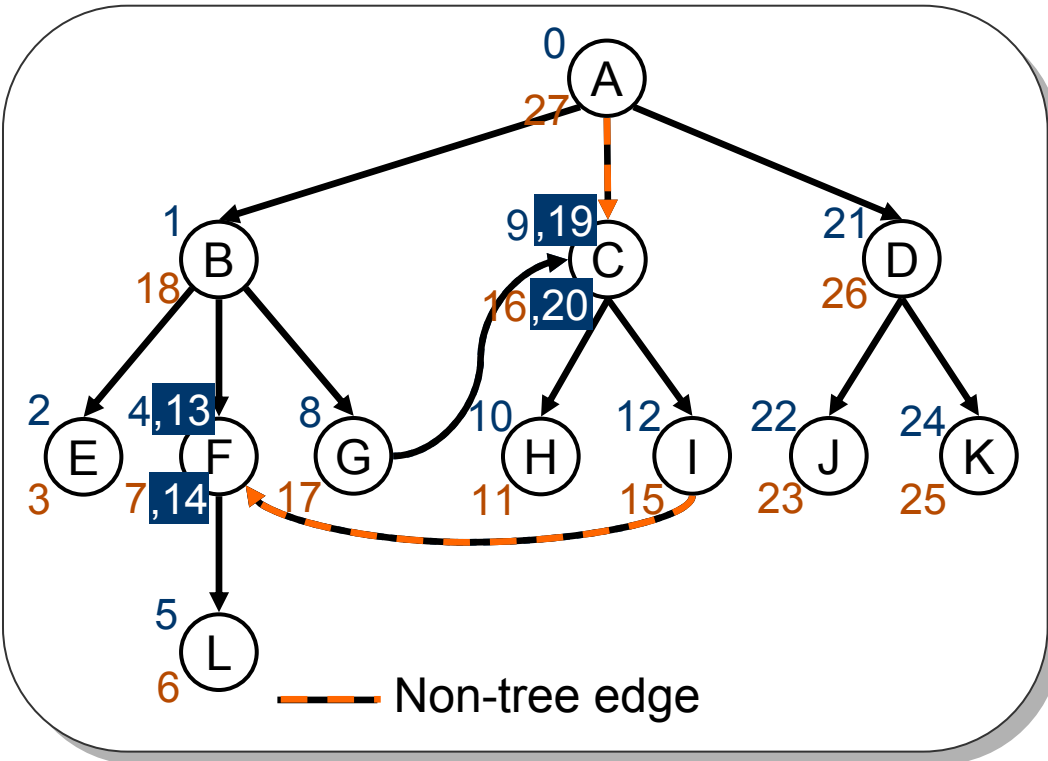
- ❑ Index creation during depth-first traversal
  - ❑ Start at the root node



- ❑ We reach a **node  $v$** 
  - ❑ for the first time
    - add *tree instance* of  $v$  to  $IND(G)$
    - proceed traversal
  - ❑ **again**
    - add *non-tree instance* of  $v$  to  $IND(G)$
    - do not traverse child nodes of  $v$

# Index Table, $IND(G)$

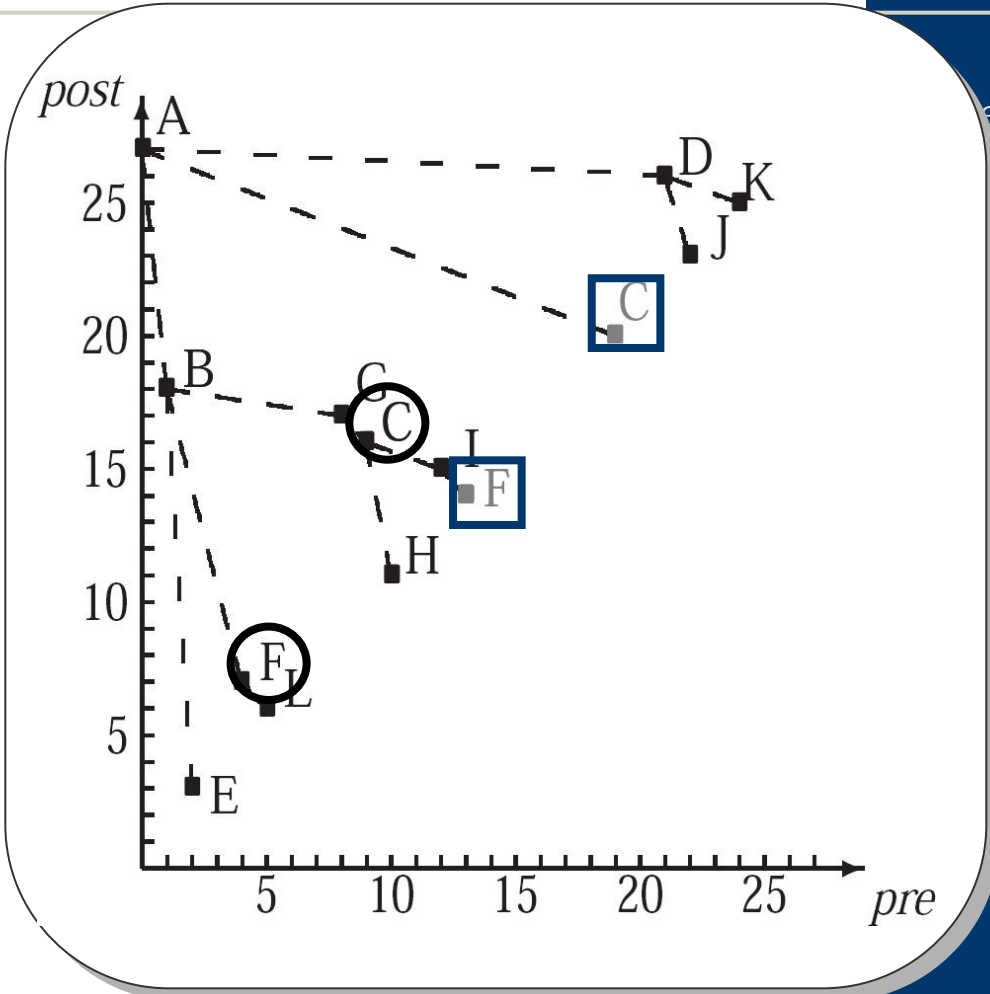
## Index table, $IND(G)$



ID	pre	post	instance
A	0	27	tree
B	1	18	tree
E	2	3	tree
F	4	7	tree
L	5	6	tree
G	8	17	tree
C	9	16	tree
H	10	11	tree
I	12	15	tree
F	13	14	non-tree
C	19	20	non-tree
D	21	26	tree
J	22	23	tree
K	24	25	tree

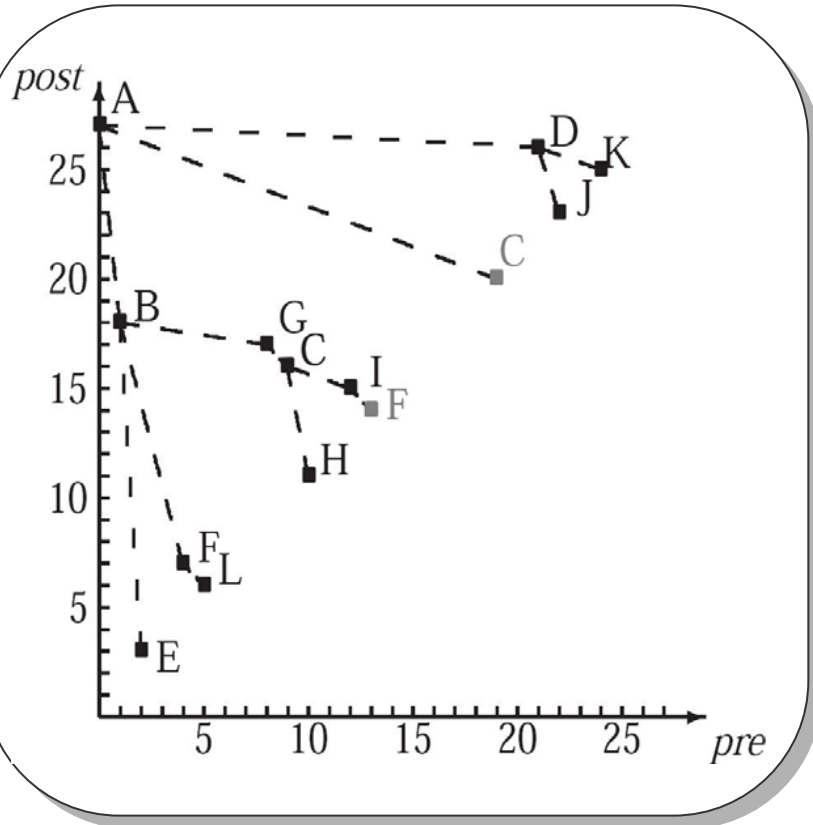
# The Order Tree, $O(G)$

ID	pre	post	instance
A	0	27	tree
B	1	18	tree
E	2	3	tree
F	4	7	tree
L	5	6	tree
G	8	17	tree
C	9	16	tree
H	10	11	tree
I	12	15	tree
F	13	14	non-tree
C	19	20	non-tree
D	21	26	tree
J	22	23	tree
K	24	25	tree

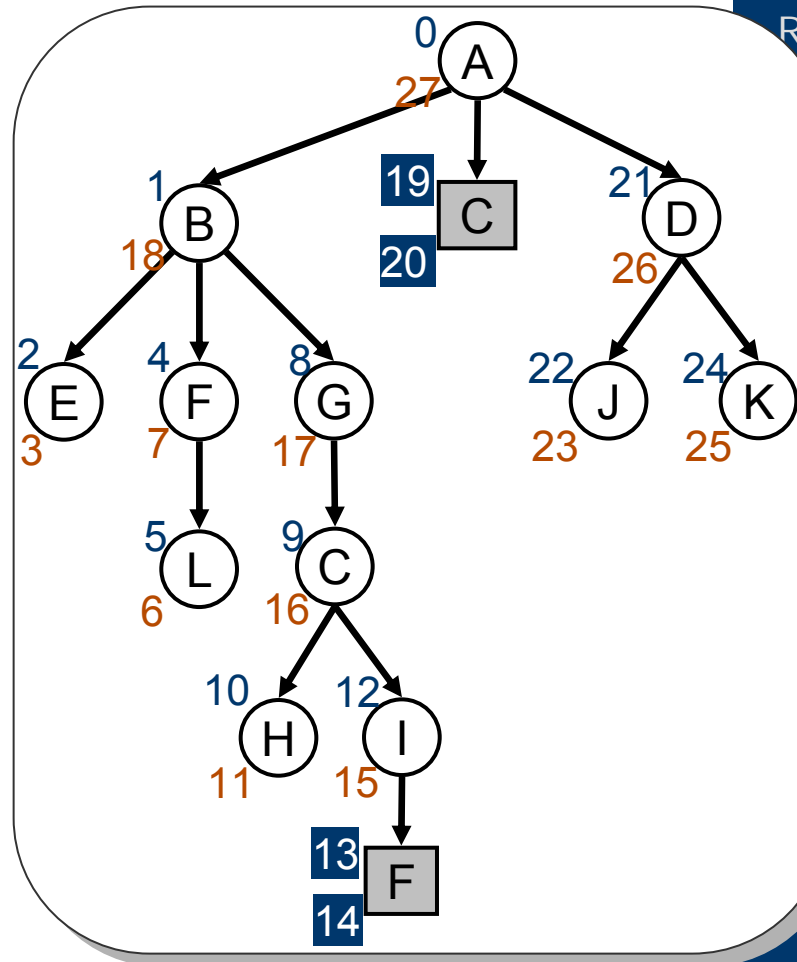


- Tree instance: Inner or leaf node
- Non-tree instance: Always leaf node

# The Order Tree – Restructured



Order tree  $O(G)$



Problem

Really trees?

pp

Indexing

Querying

Experiments

Conclusion

# Answering Queries

- XPath expression  $//v//w$ 
  - Identify bindings of  $v$  and  $w$ 
    - Given by ID of the element
  - Answer  $//v//w$ 
    - Is  $w \in \text{descendant}/v$  ?
    - Condition:

$w \in \text{descendant}/v$  iff  
 $\text{pre}_v < \text{pre}_w < \text{post}_v$

Problem

Really trees?

GRIPP

Indexing

Querying

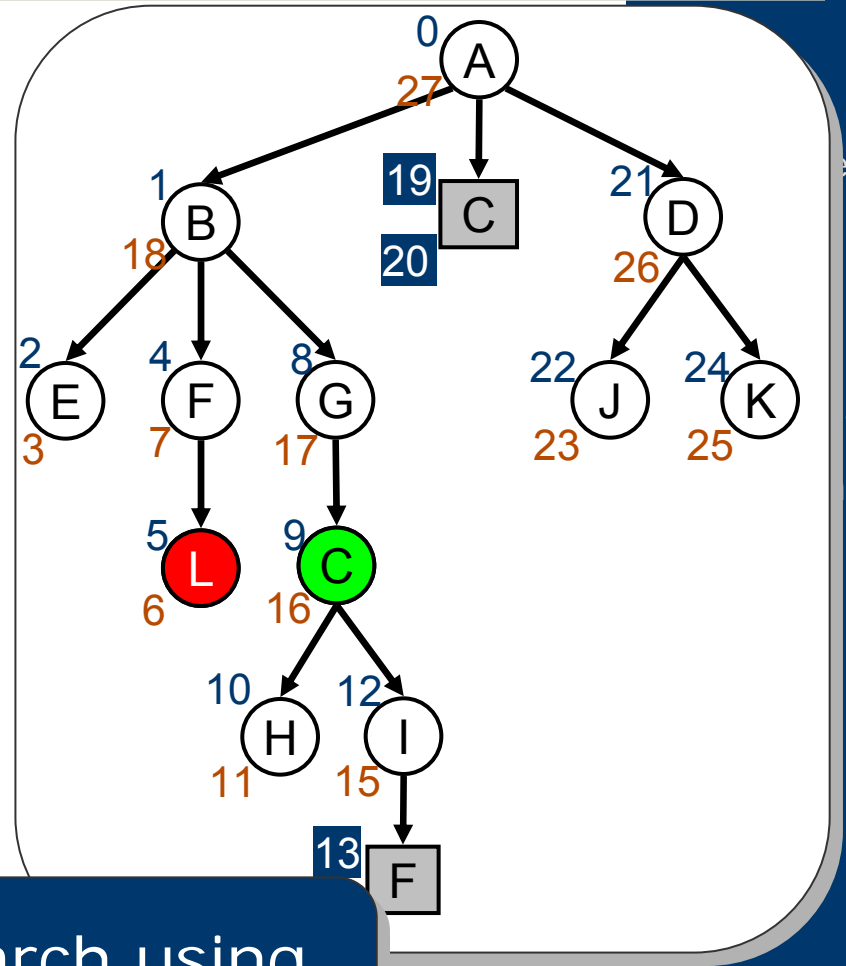
Experiments

Conclusion

# Answering //v//w with GRIPP

## Example //c//l

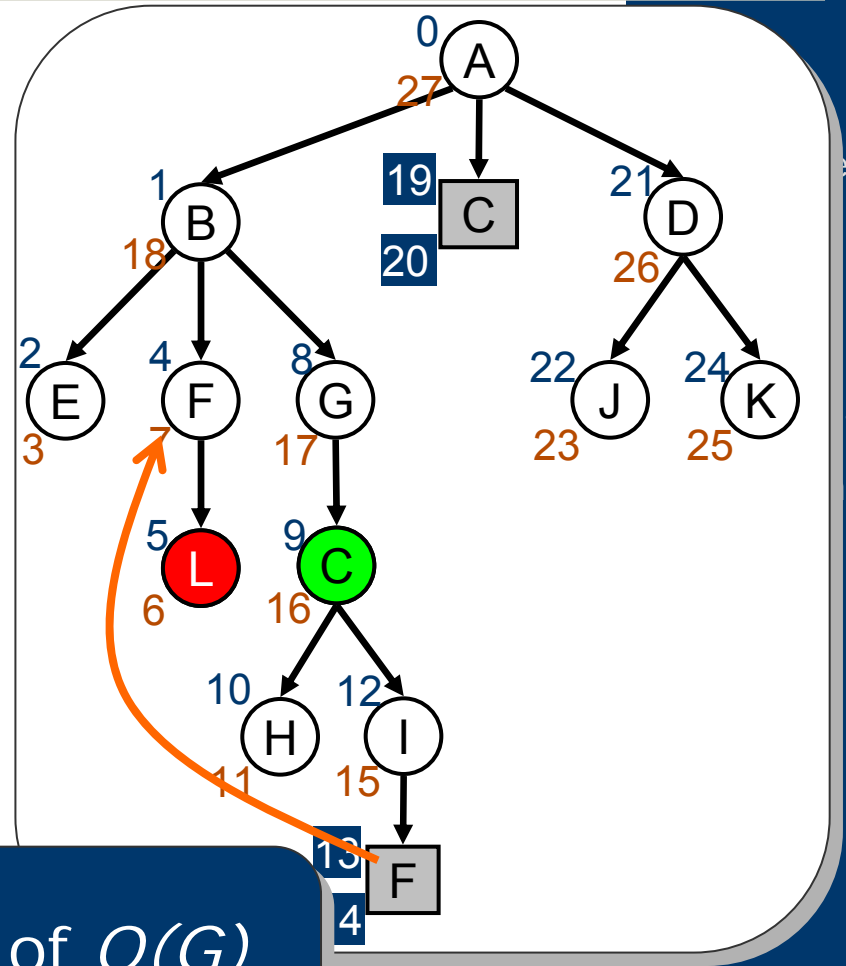
1. Find tree instance of c
2. Check  $pre_c < pre_L < post_c$ 
  - $9 < 5 < 16 \Rightarrow \text{False}$
3. Find non-tree instances in descendant/c
  - F is non-tree instance
    - Has no descendants in  $O(G)$
    - But...



**Extend the search using non-tree instances**

# Answering //v//w with GRIPP

- Example //c//L
- 1. Find tree instance of c
- 2. Check  $pre_c < pre_L < post_c$ 
  - $9 < 5 < 16 \Rightarrow \text{False}$
- 3. Find non-tree instances in descendant/c
  - For each non-tree instance go to Step 1



□ //c Depth-first search of  $O(G)$  using non-tree instances

# Outline

---

- Problem definition
  - XML Documents are trees! – Really?
  
- Indexing XML Documents - GRIPP
  - Index Construction
  - Querying GRIPP
  
- Experimental Evaluation
  
- Conclusion

Problem  
Really trees?  
GRIPP  
Indexing  
Querying  
Experiments  
Conclusion

# Experimental Setup

- ❑ eXist – native XML database
  - ❑ Load documents into database
  - ❑ Query for  $//v//w$  using XQuery
  
- ❑ GRIPPread - RDBMS
  - ❑ Read, immediately index and load index
  - ❑ Query  $IND(G)$  using PL/SQL
  
- ❑ GRIPPdb - RDBMS
  - ❑ Read document, generate graph and load graph
  - ❑ Create  $IND(G)$  inside a RDBMS

Problem

Really trees?

GRIPP

Indexing

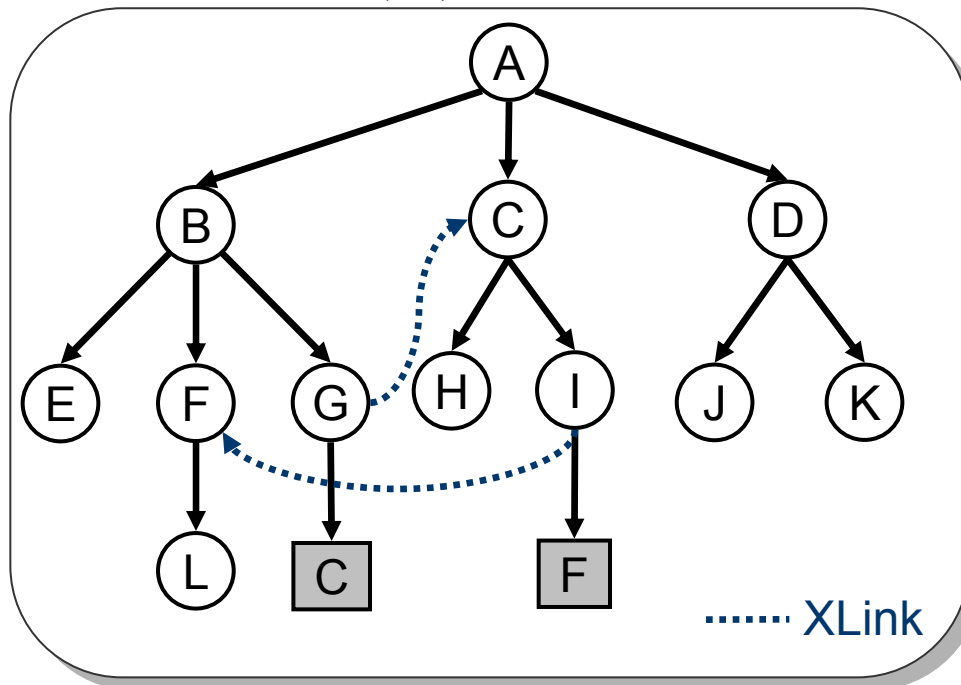
Querying

Experiments

Conclusion

# GRIPPread

- Read and immediately index XML document using pre- and postorder values
  - Target of XLink element becomes non-tree instance in  $IND(G)$



Order tree  $O(G)$

Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

Conclusion

# Experimental Setup

- ❑ eXist – native XML database
  - ❑ Load documents into database
  - ❑ Query for  $//v//w$  using XQuery
- ❑ GRIPPread - RDBMS
  - ❑ Read, immediately index and load index
  - ❑ Query  $IND(G)$  using PL/SQL
- ❑ GRIPPdb - RDBMS
  - ❑ Read document, generate graph and load graph
  - ❑ Create  $IND(G)$  inside a RDBMS

Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

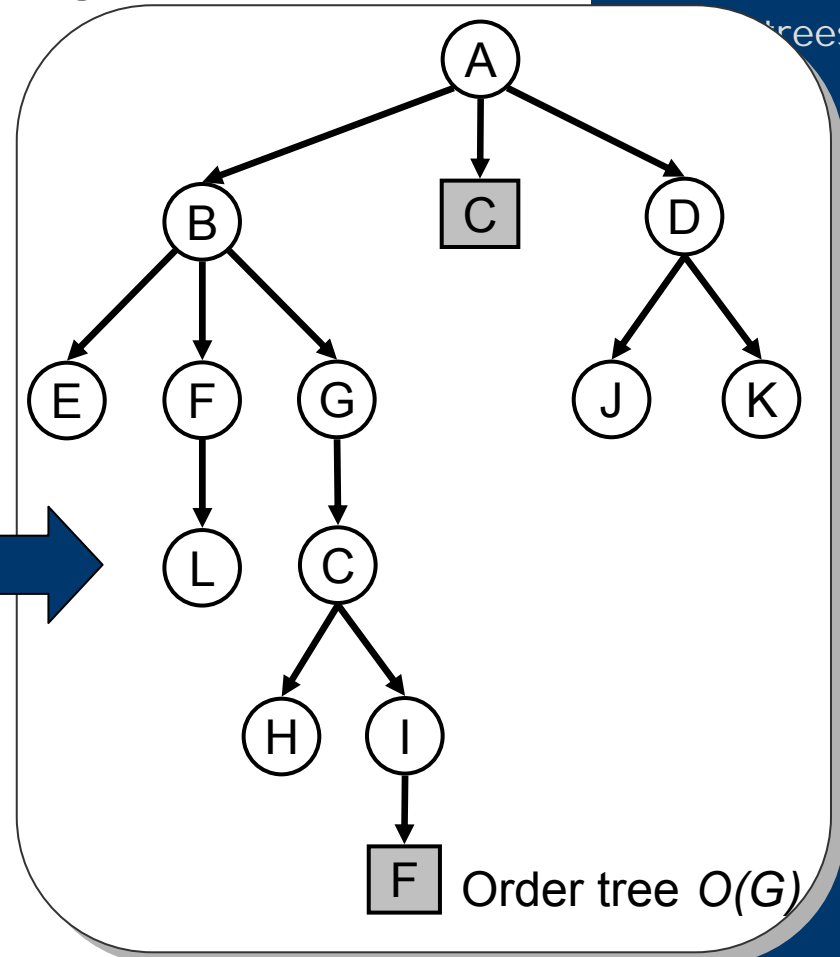
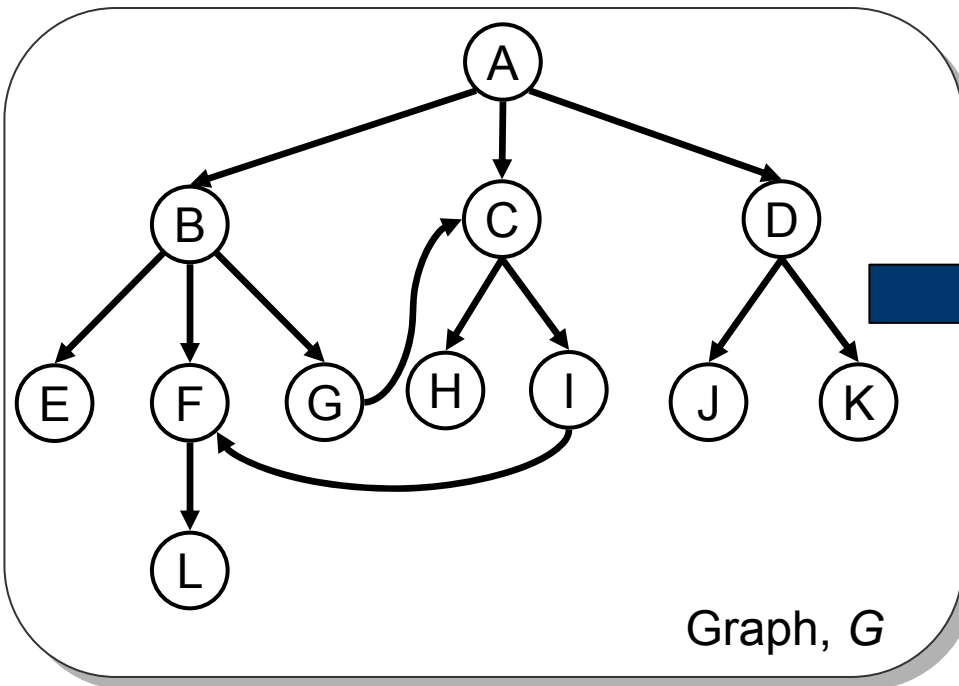
Conclusion

# GRIPPdb

1. Read document and create graph
2. Load graph into RDBMS
3. Create the GRIPP index

Problem

trees?



# XML Documents

## ❑ Generated documents

- ❑ Size 1.2 – 152 MB
- ❑ % XLink elements 5 – 80%

## ❑ Real-world documents

- ❑ DBLP
  - 345 MB (8 Million elements)
  - 5% XLink elements
  - Max. tree depth 6

Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

Conclusion

# Index Creation Time

- Time to read, load, and index the XML document

Linear increase

Size	# Nodes	# Edges	GRIPP <i>read</i>	GRIPP <i>db</i>	eXist
1.2 MB	13,210	13,869	5	59	4
7.1 MB	64,693	67,926	17	675	70
13.2 MB	144,775	152,013	33	2,760	97
61.1 MB	644,240	676,451	157	55,490	-
152.4 MB	1,600,683	1,680,716	560	-	-

Time in seconds for XML documents containing 5% XLink elements

Problem

Really trees?

GRIPP

Indexing

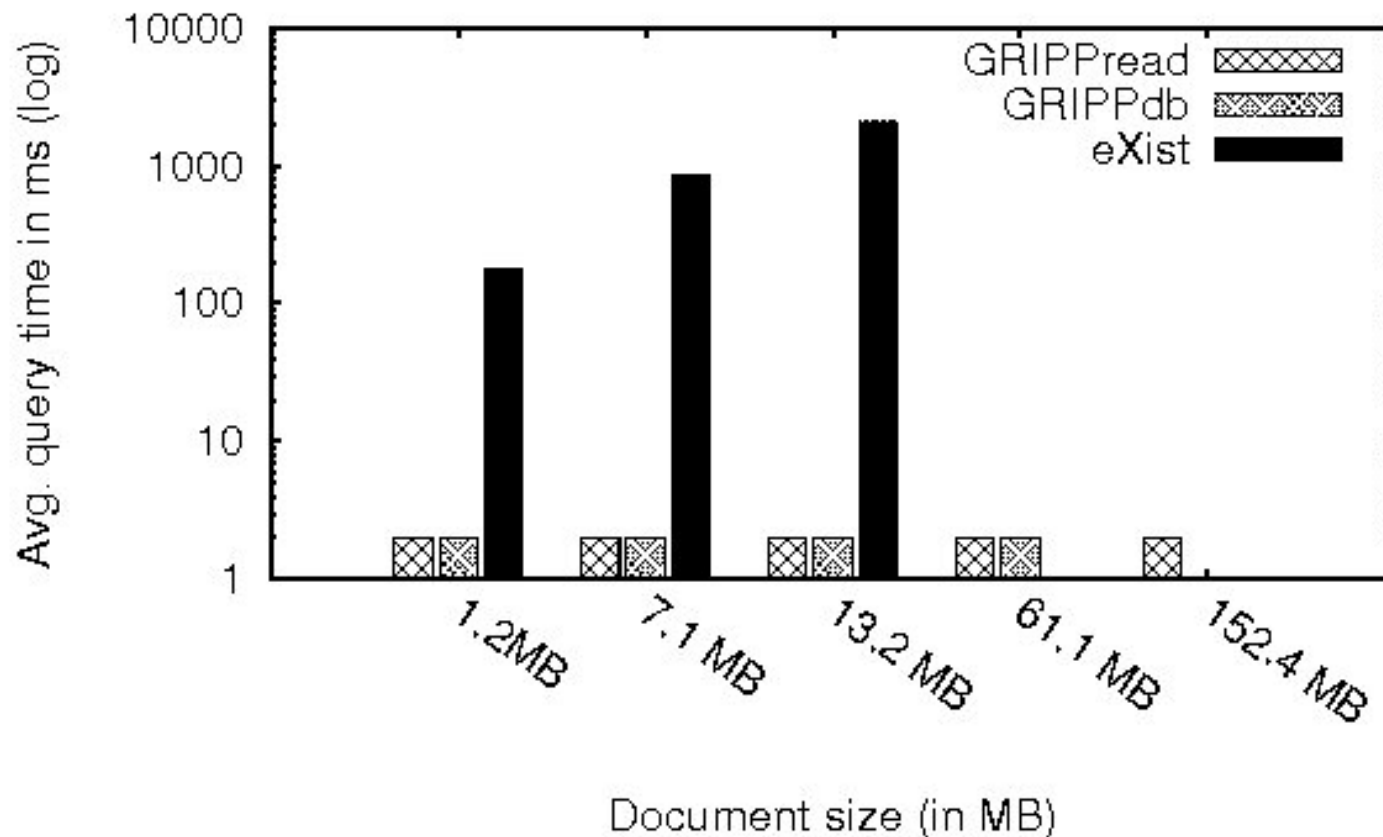
Querying

Experiments

Conclusion

# Query Times

## Increasing document size



Query time in ms for XML documents containing 5% XLink elements

# eXist so much slower?

## ❑ GRIPRead and GRIPdb

- ❑ Answer `//v//w = TRUE/FALSE` using PL/SQL

## ❑ eXist

- ❑ Answer `//v//w` using XQuery
- ❑ XQuery has a different semantic
  - Returns all nodes  $w \in \text{descendant}/v$
  - Does not terminate after  $w$  is found
  - Does not support XLink elements
    - Work-around: Use functions

Problem

Really trees?

GRIPP

Indexing

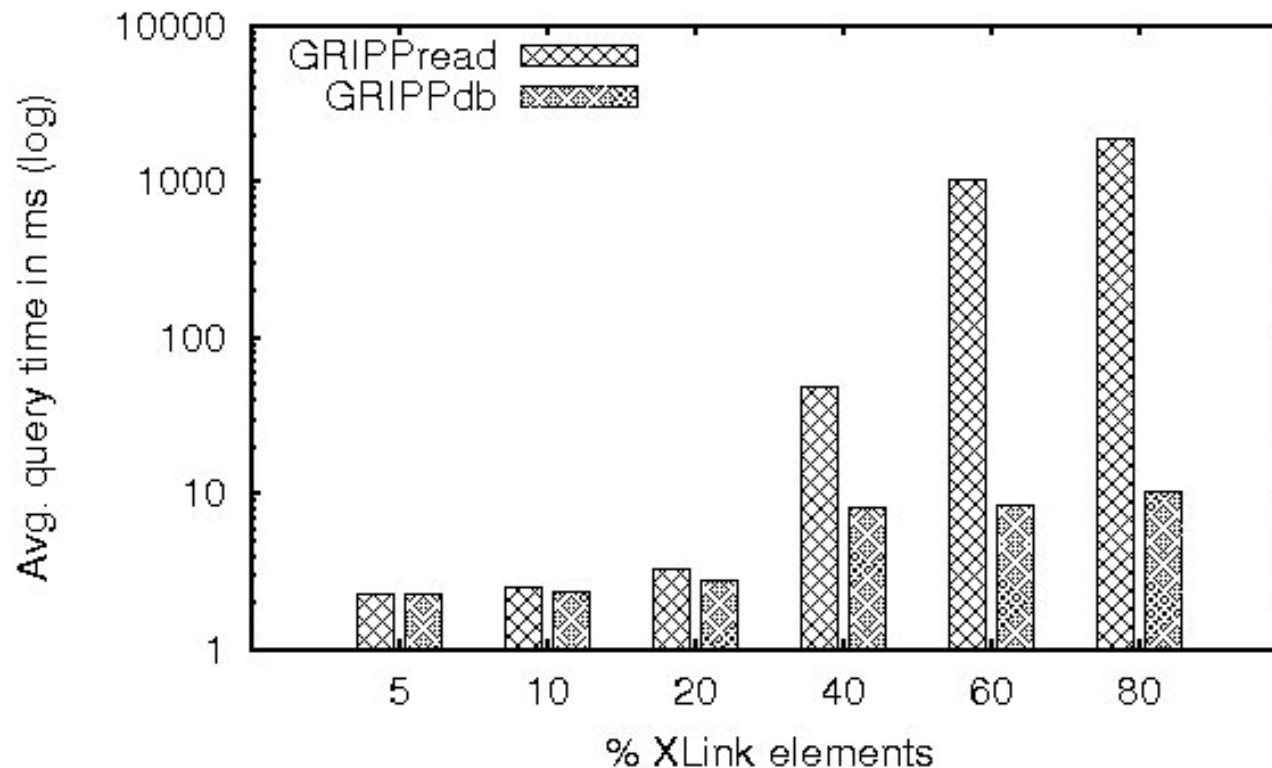
Querying

Experiments

Conclusion

# Query Times – Why GRIPPdb?

- Increasing number of XLink elements



Query time in ms for XML documents of size 1 – 2 MB

Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

Conclusion

# Real-world Documents

## □ DBLP

- 345 MB (8 Million elements)
- 5% XLink elements

## □ GRIPP*read*

- Indexing time **21 minutes**
- Query time **3.4 ms (on average)**

## □ Not possible with eXist

- Main-memory exception!

Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

Conclusion

# Conclusion

- ❑ *GRIPPread vs. GRIPpdb*
  - ❑ Indexing
    - *GRIPPread*  $\ll$  *GRIPpdb*
    - In *GRIPPread* document order is preserved
  - ❑ Querying
    - Few XLink elements ( $\leq 20\%$ )
      - *GRIPpdb*  $\sim$  *GRIPPread*
    - Many XLink elements ( $> 20\%$ )
      - *GRIPpdb*  $\ll$  *GRIPPread*
- ❑ **BUT:** XML documents might not contain more than 20 % XLink elements

Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

Conclusion

# Conclusion

## ❑ GRIPP vs. eXist

### ❑ Reading and indexing

- GRIPP $read$  < eXist << GRIPP $db$
- GRIPP can handle large XML documents

### ❑ For querying for `//v//w = TRUE / FALSE`

- GRIPP $read$  ~ GRIPP $db$  << eXist

GRIPP is applicable to XML documents containing XInclude or XLink elements!

Problem

Really trees?

GRIPP

Indexing

Querying

Experiments

Conclusion



# Thanks for your attention!

Silke Trißl, Florian Zipser & Ulf Leser

Humboldt-Universität zu Berlin

Knowledge Management in Bioinformatics