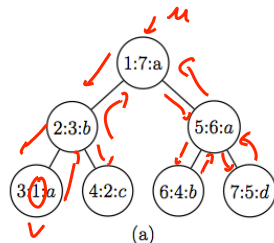


## XASR: eXtented Access Support Relations



R	pre	post	parent_pre	lab
1	7	1		a
2	3	1		b
3	1	2	a	a
4	2	2	c	c
5	6	1	a	a
6	4	5	b	b
7	5	5	d	d

1 <a>  
 2 <b>  
 3 <a><1/a>  
 4 <c><1/c>  
 5 </b>  
 6 <a>  
 7 <b><1/b>  
 <a><1/d>  
 </a>  
 </a>

Figure 2: Tree (a) and XASR (b).

**Example 2.1** The tree shown in Figure 2 (a) can be represented by the XASR of Figure 2 (b). We can define the descendant axis as an SQL view

```

CREATE VIEW descendant AS
SELECT r1.pre, r2.pre FROM R r1, R r2
WHERE r1.pre < r2.pre AND r2.post < r1.post;

```

and the child axis as

```

CREATE VIEW child AS
SELECT parent_pre, pre FROM R
WHERE parent_pre is not NULL;

```

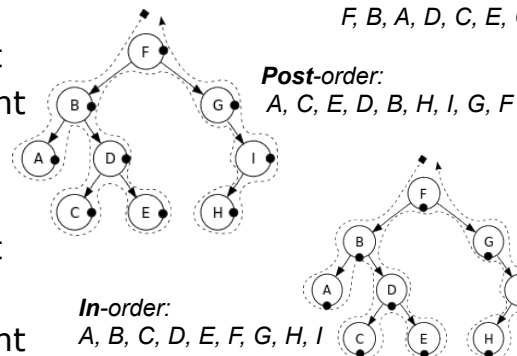
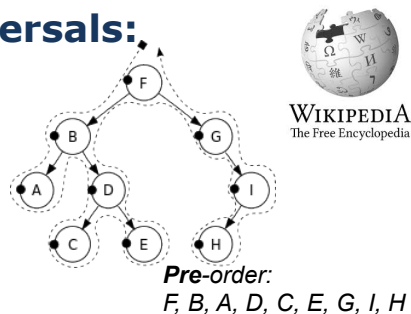
[3] Processing queries on tree-structured data efficiently, Koch, C., PODS 2006.

ECS-165B

1

## Depth-First Tree Traversals:

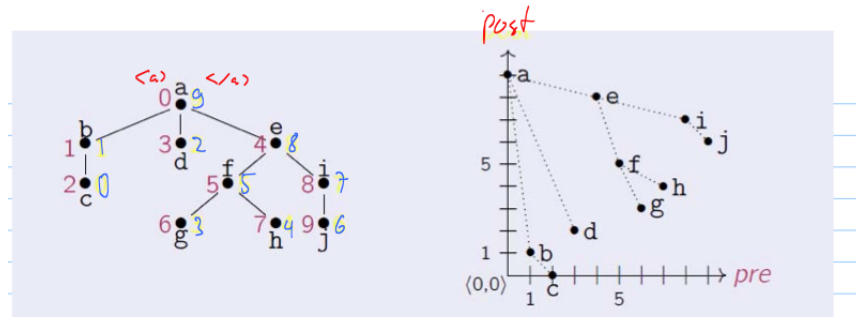
- **Pre-order:**
  - Visit the root
  - Traverse left child
  - Traverse right child
- **Post-order**
  - Traverse left
  - Traverse right
  - Visit root
- **In-order**
  - Traverse left
  - Visit root
  - Traverse right



ECS-165B

2

## Pre- and Post-Order Traversal Numbers



$\langle a \rangle$   
 $\langle b \rangle$   
 $\langle c \rangle$   
 $\langle d \rangle$   
 $\langle f \rangle$   
 $\langle g \rangle$   
 $\langle e \rangle$   
 $\langle h \rangle$   
 $\langle i \rangle$   
 $\langle j \rangle$

for each node  $v$   
 store  $(pre, post)$  number  
 (here:  $(x, y)$  coordinates)

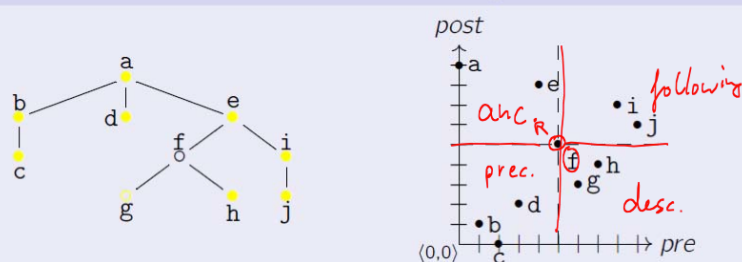
[5] Database-Supported XML Processors, Lectures by Torsten Grust. U Tübingen, 2013

ECS-165B

3

Know your descendants, ancestors, following and preceding nodes via their pre- and post-order coordinates ...

Plane partitions  $\equiv$  XPath axes,  $o$  is arbitrary!



Pre/post plane regions  $\equiv$  major XPath axes

The **major XPath axes** descendant, ancestor, following, preceding correspond to rectangular **pre/post plane windows**.

[5] Database-Supported XML Processors, Lectures by Torsten Grust. U Tübingen, 2013

ECS-165B

4

## Example

$(e, f) / \underline{\text{descendant}}::\text{node}()$

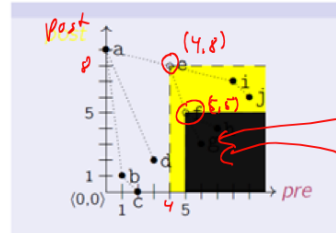
[5] Database-Supported XML Processors,  
Lectures by Torsten Grust, U Tübingen, 2013

### Context & frag. encodings

context	pre	post	...
f →	5	5	
c →	4	8	

accel	pre	post	...
	0	9	
	1	1	
	2	0	
	3	2	
	4	8	
	5	5	
	6	3	
	7	4	
	8	7	
	9	6	



### SQL query with expanded `window()` predicate

```

SELECT  DISTINCT v1.*
FROM    context v, accel v1
WHERE   (v1.pre > v.pre) AND (v1.post < v.post)
ORDER BY v1.pre

```

Handwritten red annotations: "to the right" under `v1.pre > v.pre` and "below" under `v1.post < v.post`.

ECS-165B

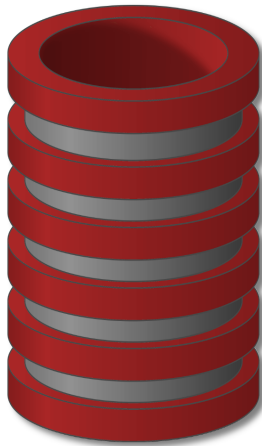
5

## References

- Lots of basic XPath material out there ...
  - Google, e.g., "XPath tutorial"
- Some (quite) advanced resources:
  - [1] Monadic queries over tree-structured data, Gottlob, G. and Koch, C., Logic in Computer Science (LICS), 2002.
  - [2] Efficient Algorithms for Processing XPath Queries, Georg Gottlob, Christoph Koch, and Reinhard Pichler, ACM TODS, 2005
  - [3] Processing queries on tree-structured data efficiently, Koch, C., PODS 2006.
  - [4] An XML Toolkit for Light-weight XML Stream Processing, Yours Truly (TJ Green) et al., 2003,  
<http://homes.cs.washington.edu/~suciu/XMLTK/>
  - [5] Database-Supported XML Processors, Torsten Grust, U Tübingen, Winter 2012/13**  
<http://db.inf.uni-tuebingen.de/teaching/ws1213/dbxml>

ECS-165B

6



# Querying XML

## XQuery

Db-class.org, Jennifer Widom, Stanford

### Querying XML

### XQuery

Not nearly as mature as Querying Relational

- Newer
- No underlying algebra

Sequence of development

1. XPath ✓
2. ~~XSLT~~
3. XQuery ✓

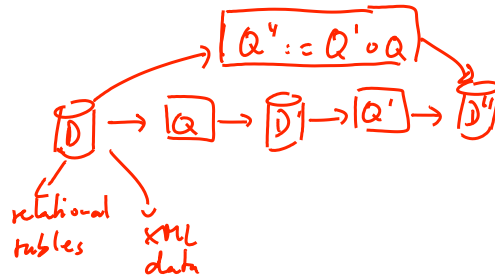
//a/b/c ..  
 "for \$a in ..  
   for \$b in ..  
     for \$c in ..  
       ...

Db-class.org, Jennifer Widom, Stanford

## XQuery

## XQuery

- Expression language (compositional) "closed"
- Each expression operates on & returns *sequence of elements*
- XPath is one type of expression



Db-class.org, Jennifer Widom, Stanford

## XQuery

## XQuery: FLWOR expression

```

For $var in expr
Let $var := expr
where condition
Order By expr
Return expr

```

- All except **Return** are optional
- **For** and **Let** can be repeated and interleaved

Db-class.org, Jennifer Widom, Stanford

## XQuery

## Mixing queries and XML

```
<Result> { ...query goes here... } </Result>
```

**Demo: XQuery examples  
over bookstore data**

Db-class.org, Jennifer Widom, Stanford

[https://class.stanford.edu/courses/Engineering/db/2014\\_1/courseware/ch-querying\\_xml/seq-vid-xquery\\_demo/](https://class.stanford.edu/courses/Engineering/db/2014_1/courseware/ch-querying_xml/seq-vid-xquery_demo/)

That's the basic expression of the XQuery language.

Here we only have the F, W and R part, "for" "where" and "return".

This query is going to return the titles of books that cost less than ninety dollars where Ullman is an author.

**Let's go ahead and take a look at the constructs of the query.**

The four construct, as we described, has an expression in this case it's an XPath expression that returns the books in our document. It binds the variable dollar B to each book one at a time and run the rest of the query for that finding the next thing it does is check whether the

(Titles of books costing less than \$90 where "Ullman" is an author)