Staircase Join : Teach a Relational DBMS to watch its (Axis) Steps.

Authors:
Torsten Grust
Maurice van Keulen
Jens Teubner

Presented by
Sanjay Kulhari

Agenda

- Background
    - XML and Relational Databases
    - XPath
- XPath Accelerator
    - Pre/Post Plane
    - SQL Based XPath evaluation
- Staircase Join
    - Pruning
    - Partitioning
    - Algorithm

XML and Relational Databases

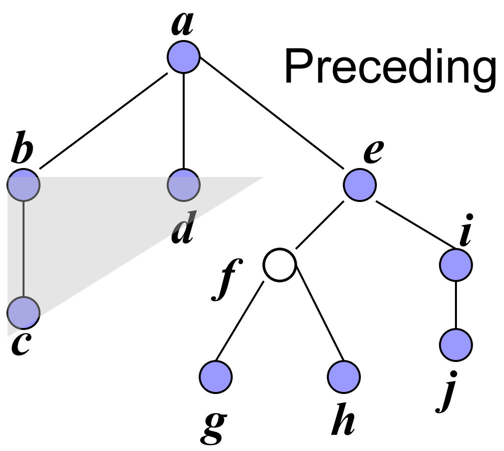
- Specialized data type for XML.
- No. of methods associated with this data type.
- Methods access XML Document Object Model.
- Methods uses XPath expression as argument to search and retrieve nodes.

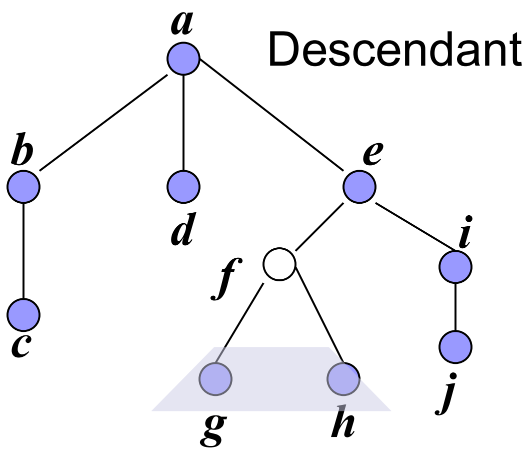
XPath

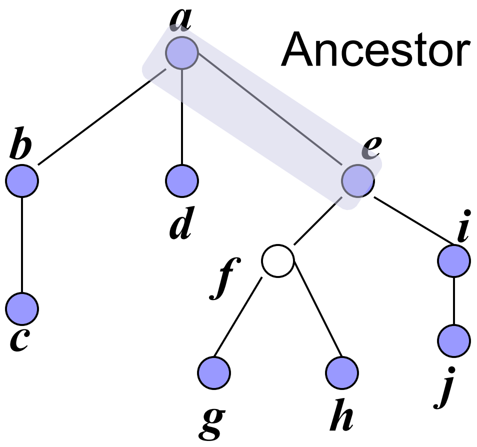
- XPath is a specialized expression language used to parse through XML.
    - State/City[Population > 100000]
- XPath nodes
    - Document, Element, Attribute, Text
- XPath Axes
    - Define and allow access to any node within XML document.
    - Major XPath axes
        - Ancestor
        - Descendent
        - Following
        - Preceding

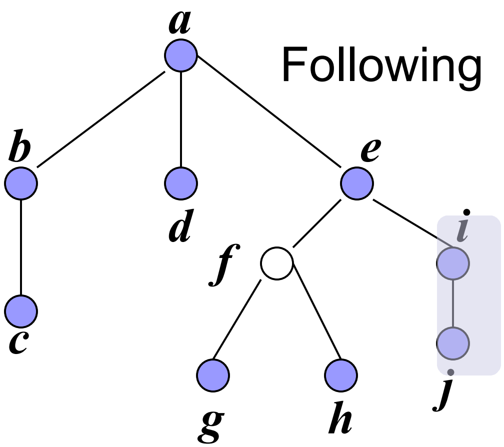
XPath Axes

Context node (f)





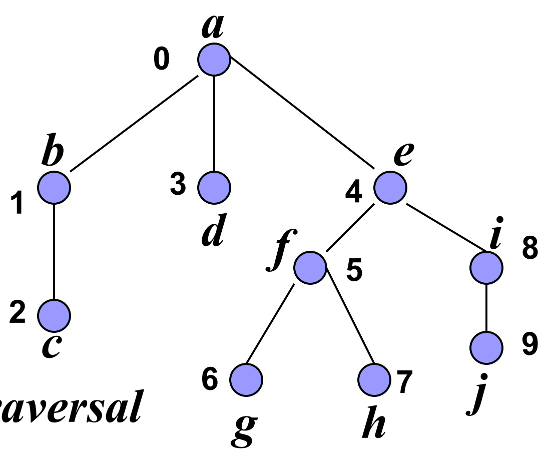




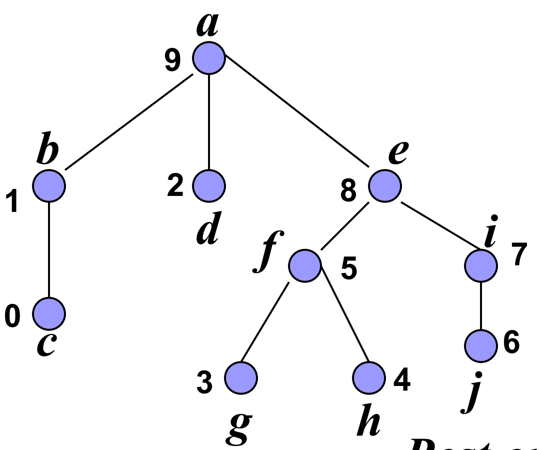
XPath Accelerator

- Relational XML encoding.
  - Document is represented as a relational table.
  - Indexed using indexed structure native to the RDBMS.
  - Queried using relational language.

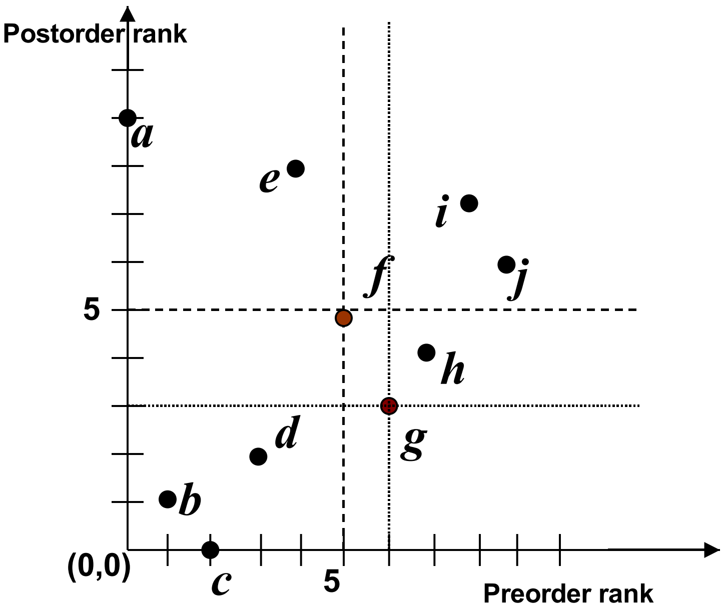
Pre/Post Plane



Pre order traversal



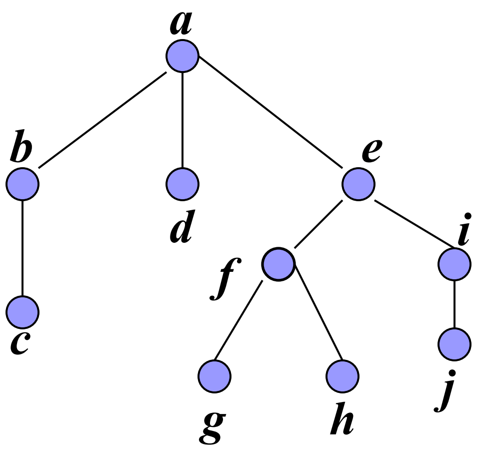
Post order traversal



Ancestor	Following
Preceding	Descendant

	Pre	Post
a	*0*	*9*
b	*1*	*1*
c	*2*	*0*
d	*3*	*2*
e	*4*	*8*
f	*5*	*5*
g	*6*	*3*
h	*7*	*4*
i	*8*	*7*
j	*9*	*6*

SQL-based XPath evaluation



	Pre	Post
a	0	9
b	1	1
c	2	0
d	3	2
e	4	8
f	5	5
g	6	3
h	7	4
i	8	7
j	9	6

V1

	Pre	Post
a	0	9
b	1	1
c	2	0
d	3	2
e	4	8
f	5	5
g	6	3
h	7	4
i	8	7
j	9	6

V2

$(c)/following/descendant = (f, g, h, i, j)$

$$|(v)/descendant| = post\ (v) - \underbrace{pre\ (v) + level\ (v)}_{<=h}$$

$AND\ v2.pre <= v1.post + h\ AND\ v2.post >= v1.pre + h$

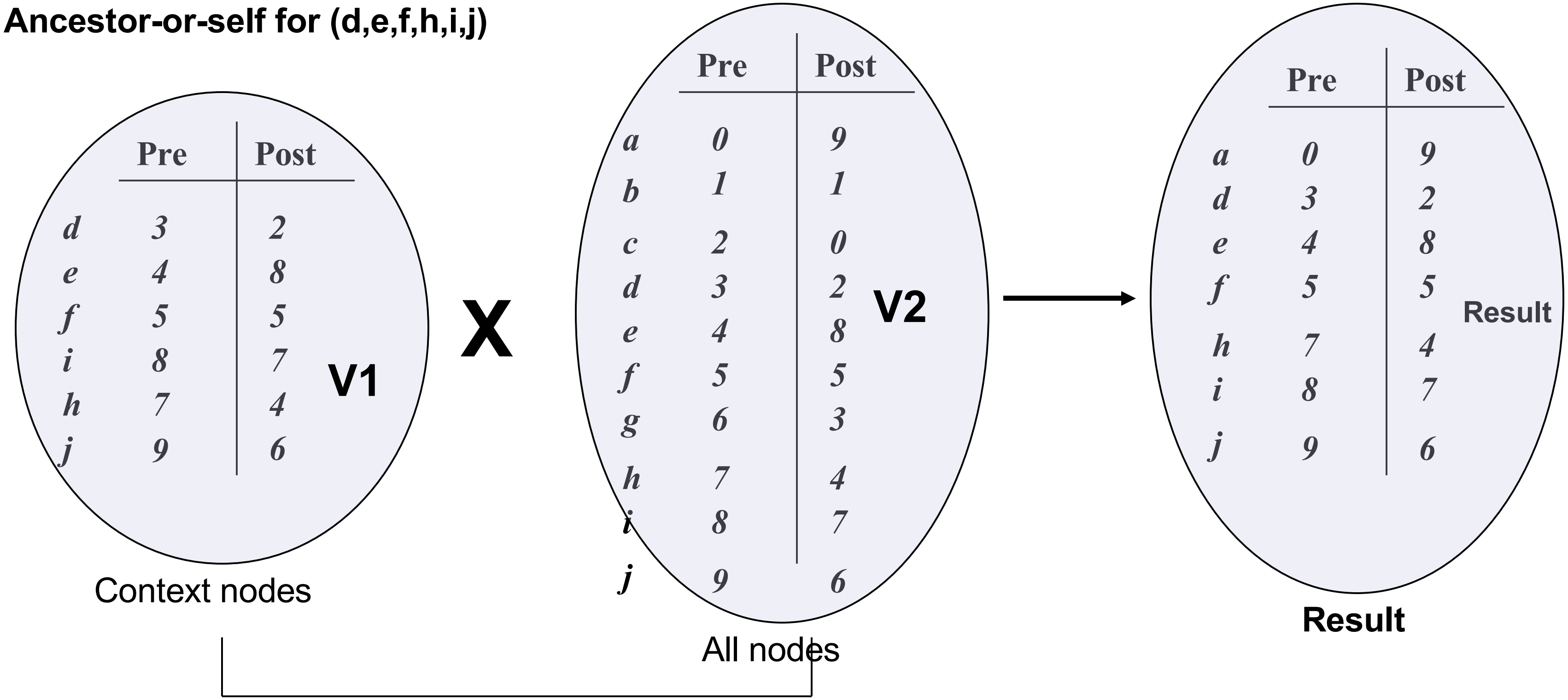
SELECT DISTINCT v2.pre
FROM doc v1,doc v2
WHERE v1.pre > pre(c)
AND v1.pre < v2:pre
AND v1.post > post(c)
AND v1.post > v2.post
ORDER BY v2.pre

Ancestor	Following
Preceding	Descendant

Staircase Join

Basic idea : Join is made between set of context nodes and the pre/post relational table by using knowledge of the pre/post plane.

Ancestor-or-self for (d,e,f,h,i,j)



	Pre	Post
d	3	2
e	4	8
f	5	5
i	8	7
h	7	4
j	9	6

Context nodes

	Pre	Post
a	0	9
b	1	1
c	2	0
d	3	2
e	4	8
f	5	5
g	6	3
h	7	4
i	8	7
j	9	6

All nodes

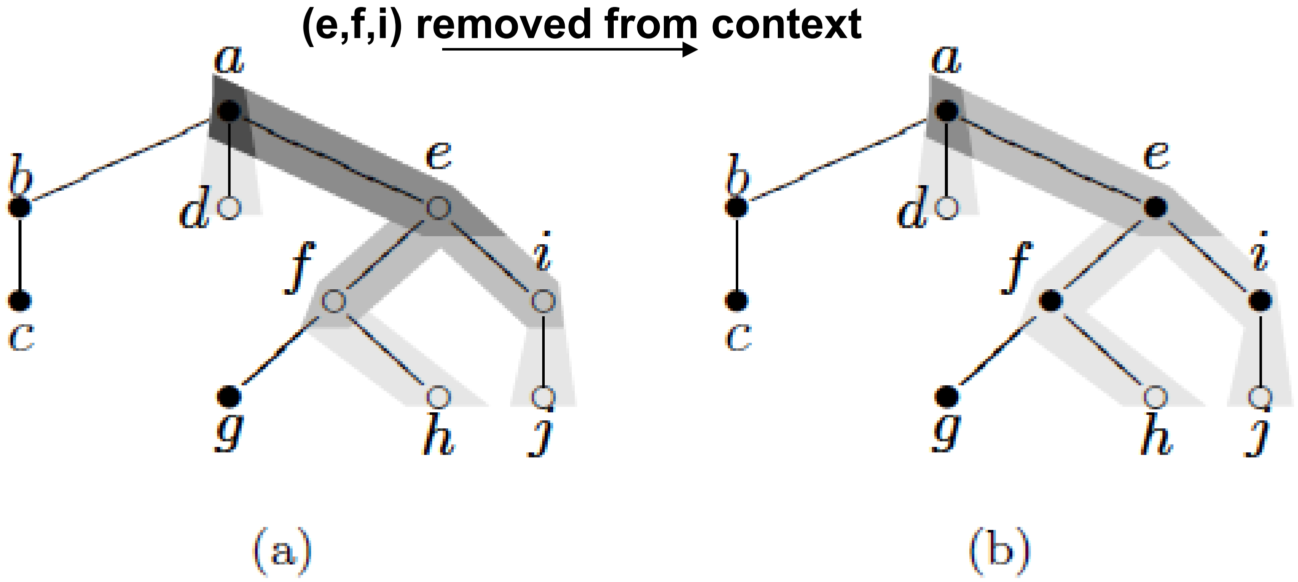
	Pre	Post
a	0	9
d	3	2
e	4	8
f	5	5
h	7	4
i	8	7
j	9	6

Result

Use predicates based on pre/post knowledge

Staircase Join (Cont.)

- Pruning



(e,f,i) removed from context

Ancestor-or-self for (d,e,f,h,i,j)

(d,a), (e,a),(f,e,a), (h,f,e,a),(i,e,a),(j,i,e,a) **11 duplicates**

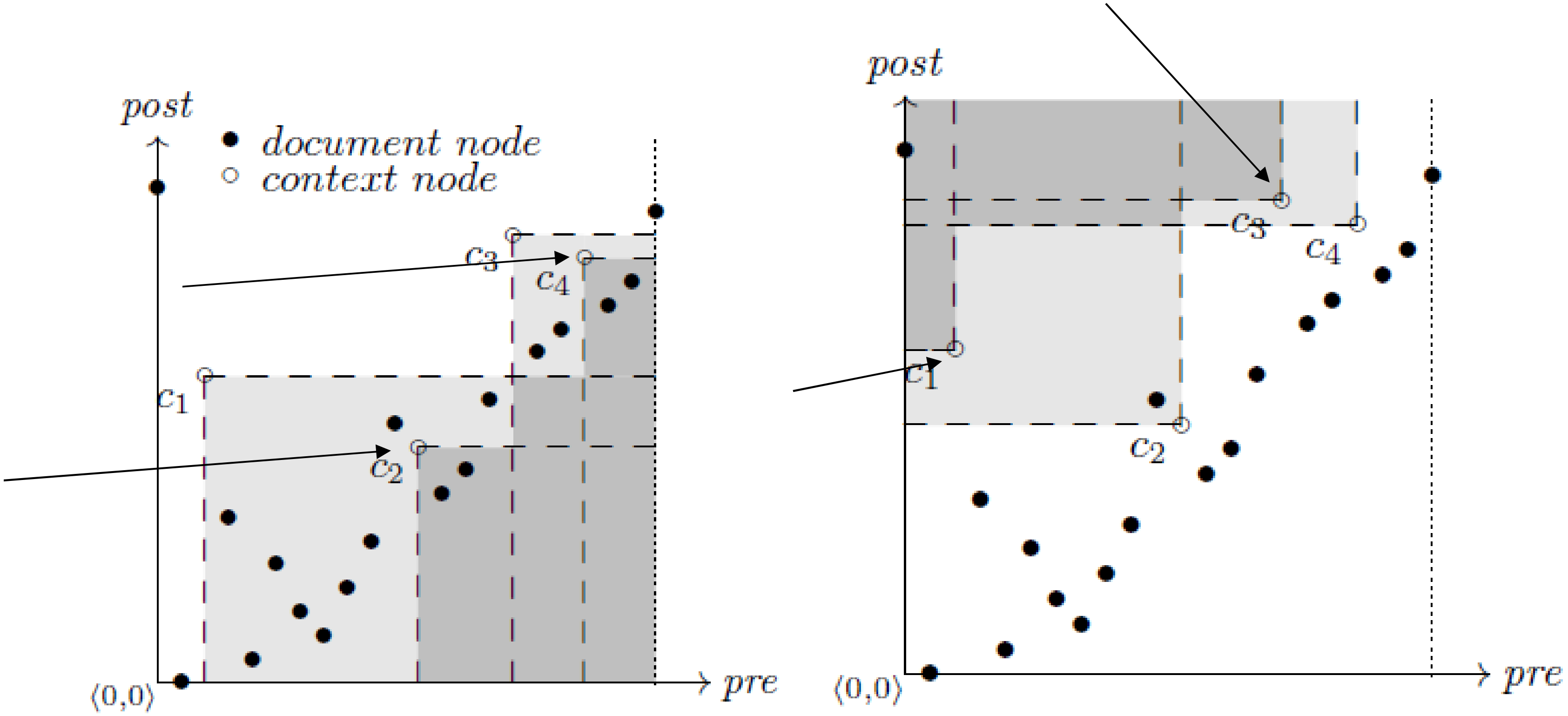
Final result (a,d,e,f,h,i,j)

Ancestor-or-self for (d,h,j)

(d,a), (h,f,e,a),,(j,i,e,a) **3 duplicates**

Final result (a,d,e,f,h,i,j)

Staircase Join (Pruning)

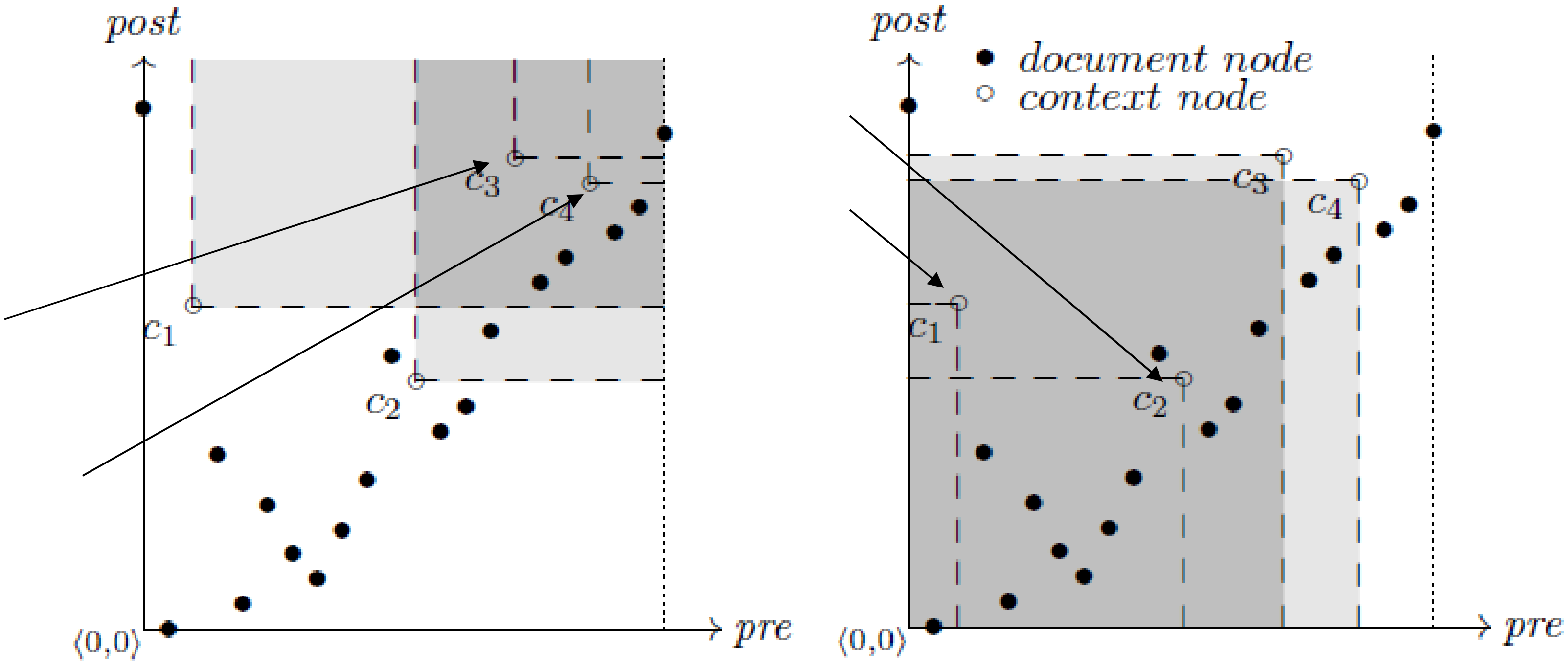


(a) descendant axis

(b) ancestor axis

Overlapping regions

Staircase Join (Pruning)

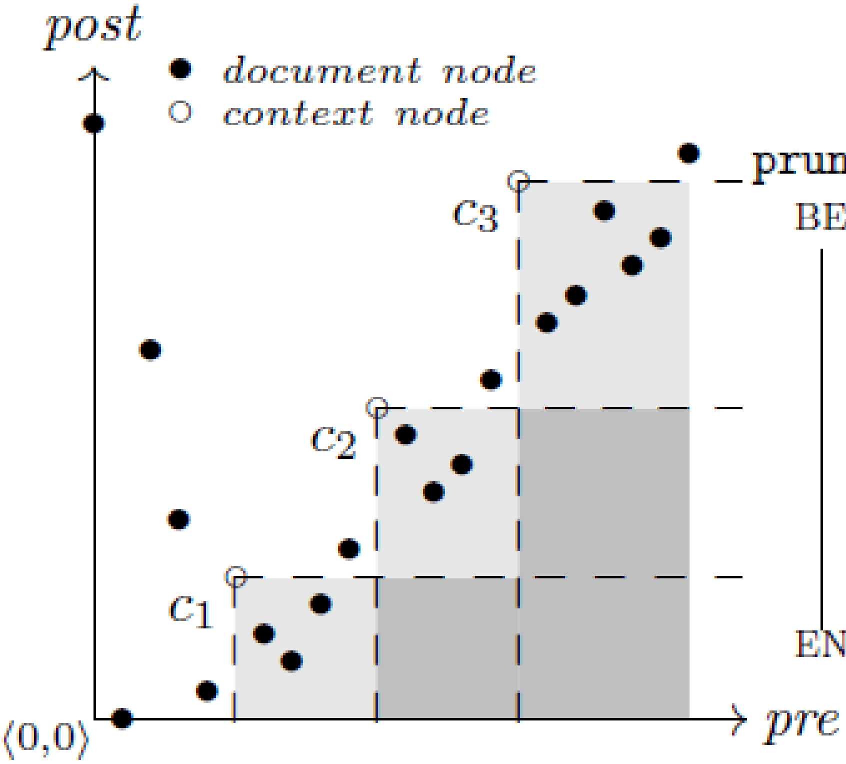


(c) following axis

(d) preceding axis

Overlapping regions

Staircase Join (Pruning)



Pruning procedure for descendent axis

```
prunecontext_desc (context : TABLE (pre,post)) ≡
  BEGIN
    result ← NEW TABLE (pre, post); prev ← 0;
    FOREACH c IN context DO
      IF c.post > prev THEN
        APPEND c TO result;
        prev ← c.post;
    RETURN result;
  END
```

c1,c2 and c3 relate to each other on preceding/following axis

Context establishes a boundary that resembles a staircase.

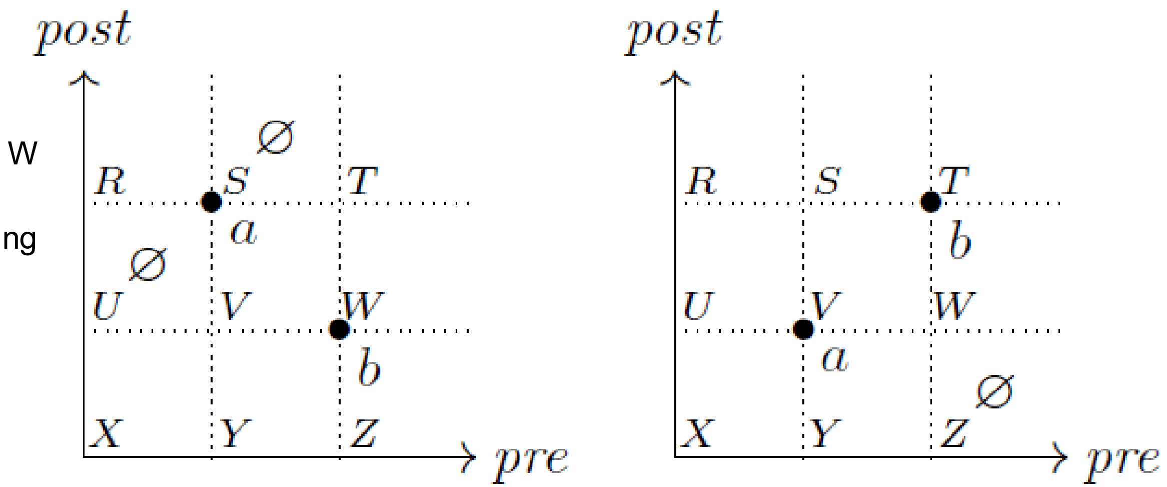
Removal of nodes from overlapping regions

Staircase Join (Pruning)

(a,b)/following = $S \cup T \cup W$

= $T \cup W$

= (b)/following



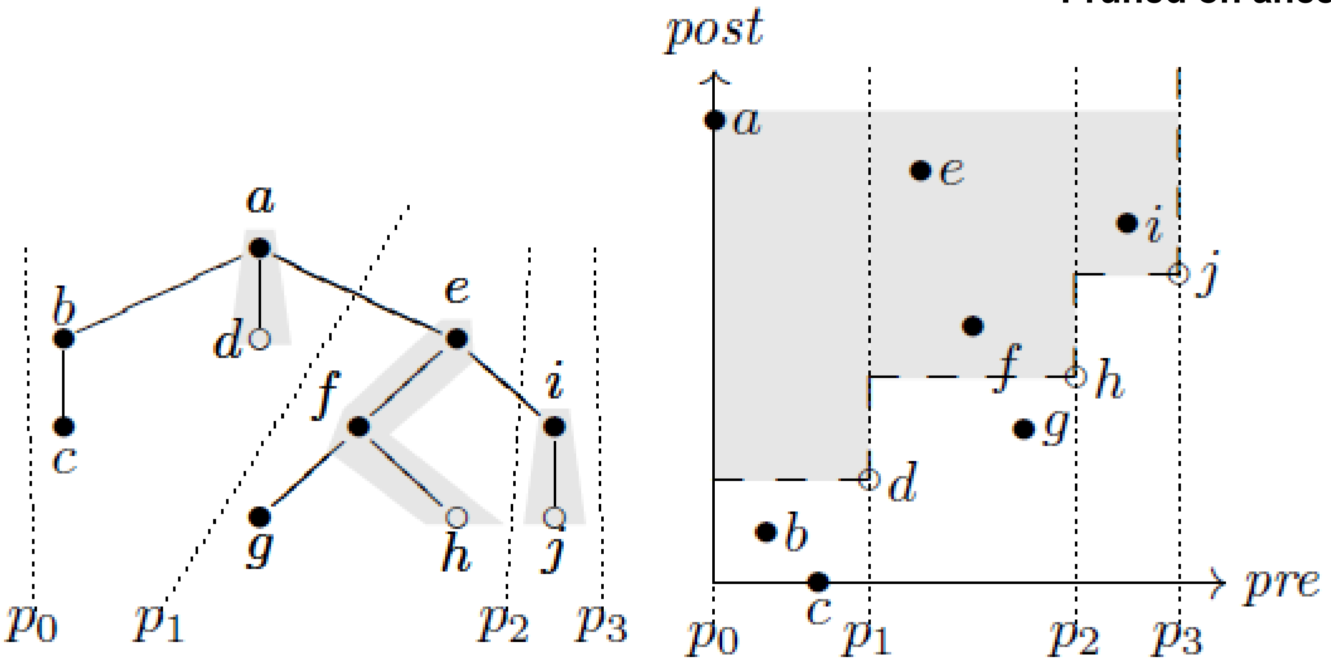
(a) Nodes a and b relate to each other on the `ancestor/descendant` axis.

(b) Nodes a and b relate to each other on the `preceding/following` axis.

Empty regions in pre/post plane

Staircase Join (Partitioning)

Pruned on ancestor axis



(a)

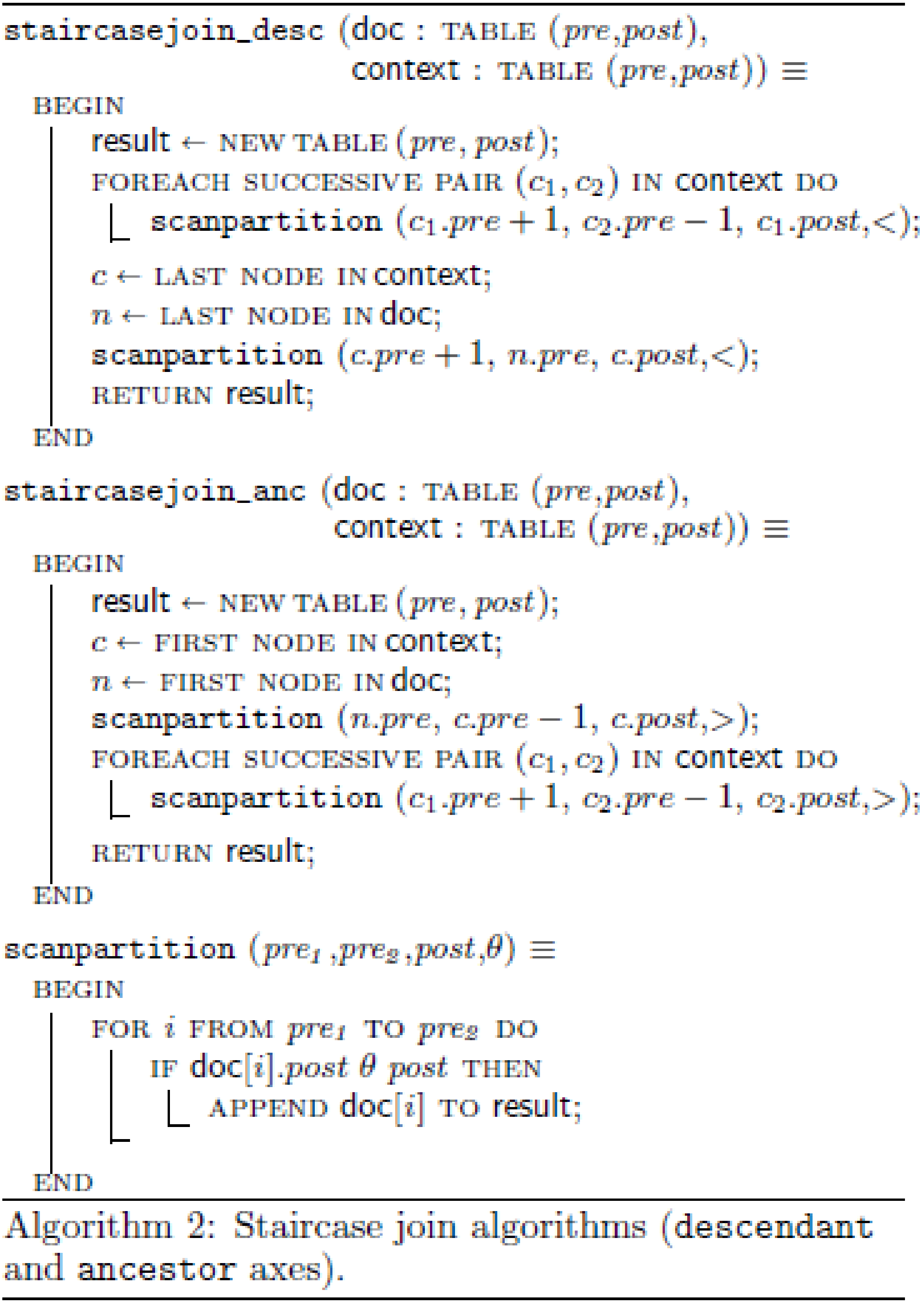
(b)

The partitions [p0; p1), [p1; p2), [p2; p3) of the ancestor staircase separate the ancestor-or-self paths in the document tree

Staircase Join (Algorithm)

Characterstics

2. Scans the doc and context table sequentially
3. Scans both the tables only once for the entire context sequence.
4. Never duplicate nodes.
5. Result nodes are produced in document order.



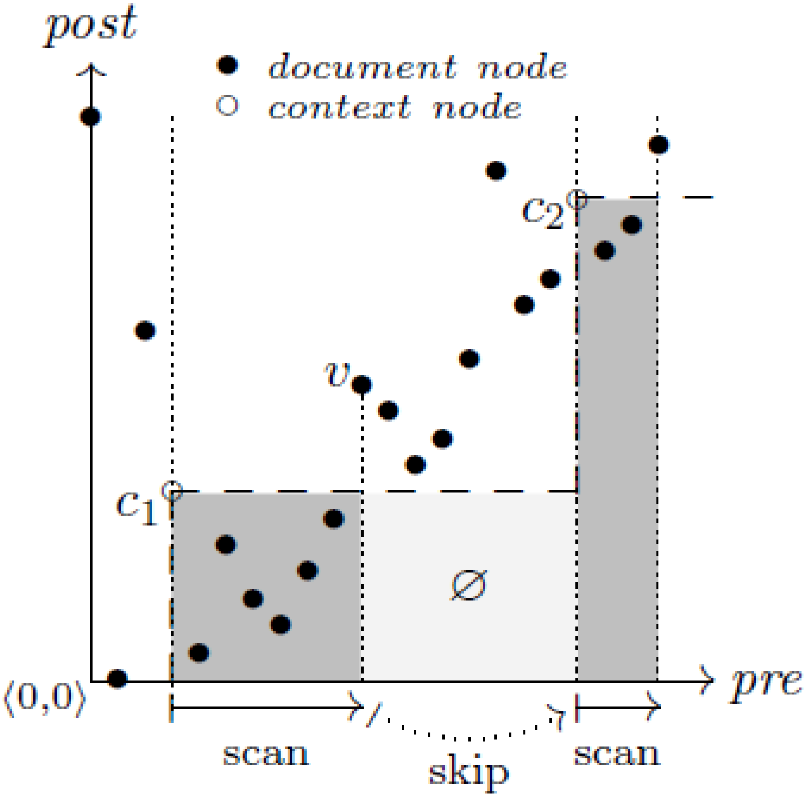
```
staircasejoin_desc (doc : TABLE ($pre$,$post$),
                    context : TABLE ($pre$,$post$)) $\equiv$
  BEGIN
    result $\leftarrow$ NEW TABLE ($pre$, $post$);
    FOREACH SUCCESSIVE PAIR ($c_1, c_2$) IN context DO
      scanpartition ($c_1.pre + 1$, $c_2.pre - 1$, $c_1.post$,$<$);
    $c \leftarrow$ LAST NODE IN context;
    $n \leftarrow$ LAST NODE IN doc;
    scanpartition ($c.pre + 1$, $n.pre$, $c.post$,$<$);
    RETURN result;
  END
staircasejoin_anc (doc : TABLE ($pre$,$post$),
                   context : TABLE ($pre$,$post$)) $\equiv$
  BEGIN
    result $\leftarrow$ NEW TABLE ($pre$, $post$);
    $c \leftarrow$ FIRST NODE IN context;
    $n \leftarrow$ FIRST NODE IN doc;
    scanpartition ($n.pre$, $c.pre - 1$, $c.post$,$>$);
    FOREACH SUCCESSIVE PAIR ($c_1, c_2$) IN context DO
      scanpartition ($c_1.pre + 1$, $c_2.pre - 1$, $c_2.post$,$>$);
    RETURN result;
  END
scanpartition ($pre_1$ ,$pre_2$ ,$post$,$\theta$) $\equiv$
  BEGIN
    FOR $i$ FROM $pre_1$ TO $pre_2$ DO
      IF doc[$i$].$post$ $\theta$ $post$ THEN
        APPEND doc[$i$] TO result;
  END
```

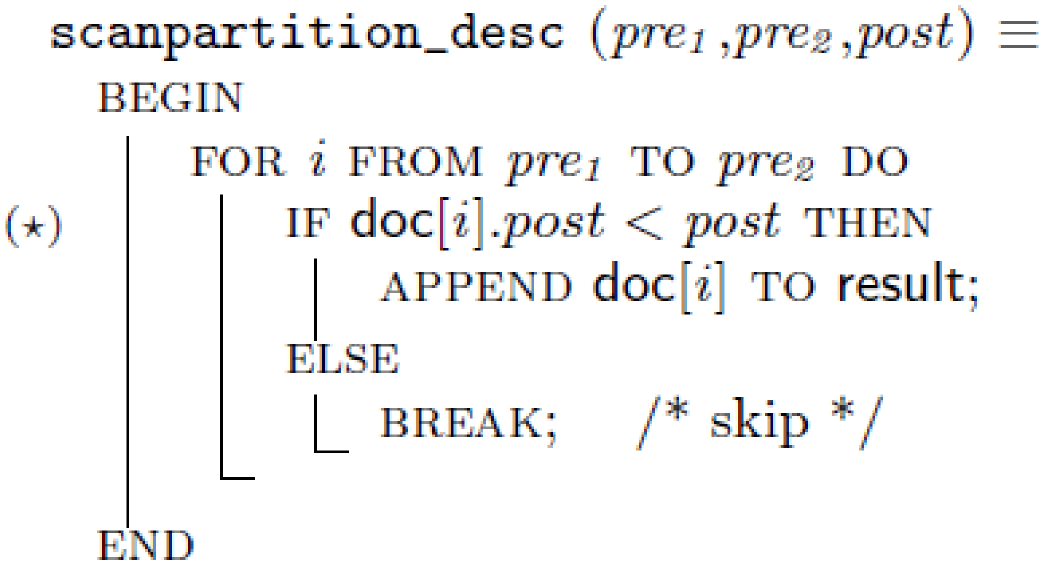
Algorithm 2: Staircase join algorithms (descendant and ancestor axes).

Staircase Join (Skipping)

No node beyond v contributes to the result.

Region between pre(*v*) and pres(*c2*) is skipped

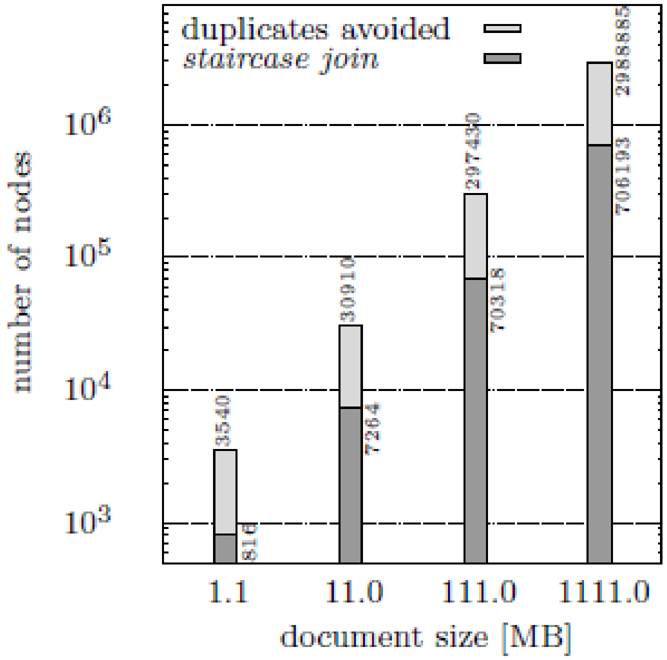




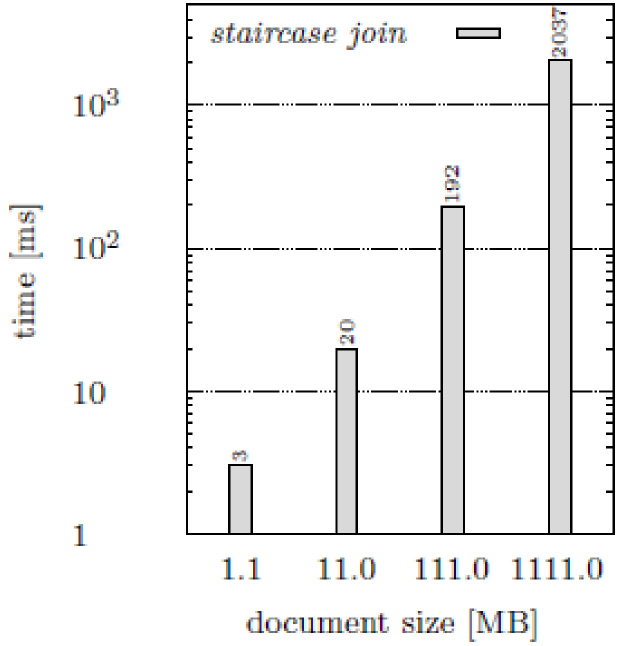
```
scanpartition_desc ($pre_1$, $pre_2$, $post$) ≡
  BEGIN
      FOR $i$ FROM $pre_1$ TO $pre_2$ DO
(⋆)       IF doc[$i$].$post$ < $post$ THEN
              APPEND doc[$i$] TO result;
          ELSE
              BREAK;    /* skip */
  END
```

(c1,c2)/descendant

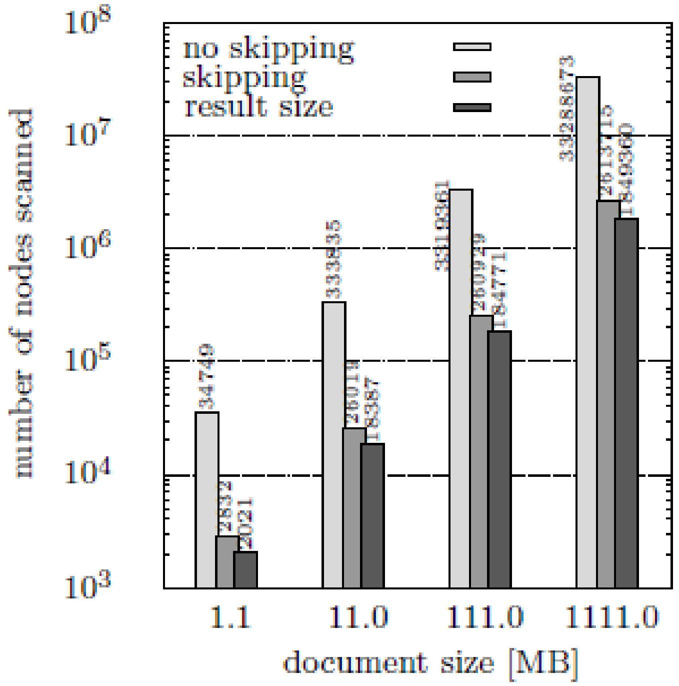
Experimental results



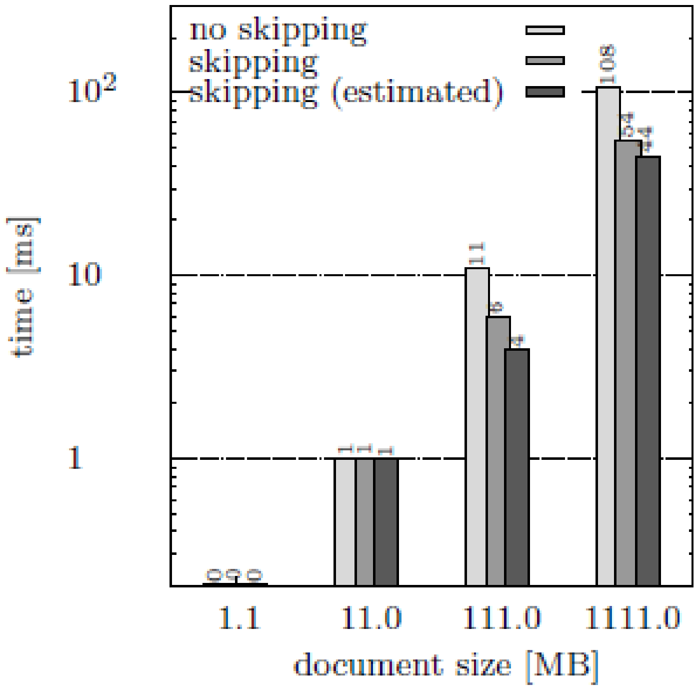
(a) Avoiding duplicates (*Q*2)



(b) Staircase join performance (*Q*2)



(c) Effectiveness of skipping



(d) Effectiveness of skipping

Conclusion

Increased tree awareness can lead to significantly improved XPath performance.

Future research

- To experiment in a commercial disc based RDBMS.
- Use larger documents >> 1GB
- Parallel XPath execution strategy

Thank You