

# Incremental Evaluation of a Monotone XPath Fragment

**Japan Advanced Institute for Science and Technology**



**Hidetaka Matsumura, Keishi Tajima**

# Introduction

## Why **incremental evaluation**?

- **incremental maintenance of materialized views**
- **continuous queries, subscription queries**

## **Monotone XPath:**

- **deletion** from DB  **deletion** from query ans
- **insertion** to DB  **insertion** to query ans

# Basic Idea

**We store information on:**

**which elements were contributing  
to which query answers**

**By using this information:**

**upon deletion of elements:**

- **we identify the answers to which the deleted element was contributing**

**upon insertion of elements:**

- **we can skip a part of computation of elements that newly become answers**

# Properties of Contributing Elements

## Those matching steps outside predicates:

- when they are deleted, the corresponding answer elements are also deleted  
→ **no need to monitor them**

## Those matching steps within predicates:

- there may be many elements matching the same step for each answer → **we need counters**
- they **indirectly contribute** to answers **through the joint step** of that predicate

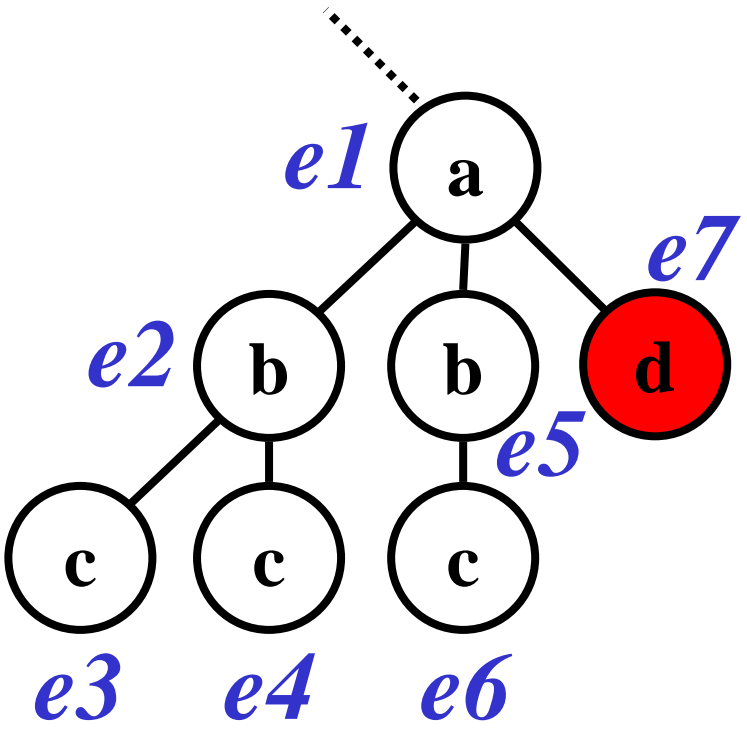
# Matching information and Counter

Q1: /.../a [ b [ c ] ] / d

s1     s2     s3

matching information:

- (e1, s1, e7)
- (e2, s2, e1)     (e5, s2, e1)
- (e3, s3, e2)     (e6, s3, e5)
- (e4, s3, e2)



counter:

- (e7, Q1, [s1=1, ...])
- (e1, s1, [s2=2])
- (e2, s2, [s3=2])
- (e5, s2, [s3=1])

# Processing of Deletions

If  $e3$  has been deleted:

1. search the matching information for  $e3$

2.  $(e3, s3, e2)$  was found

3. decrement the counter  $s3$  for  $e2$ :

$(e2, s2, [s3=1]) \longrightarrow (e2, s2, [s3=0])$

4. If the counter reaches 0:

$e2$  no longer matches  $s2$

search the matching information for  $e2 \dots$

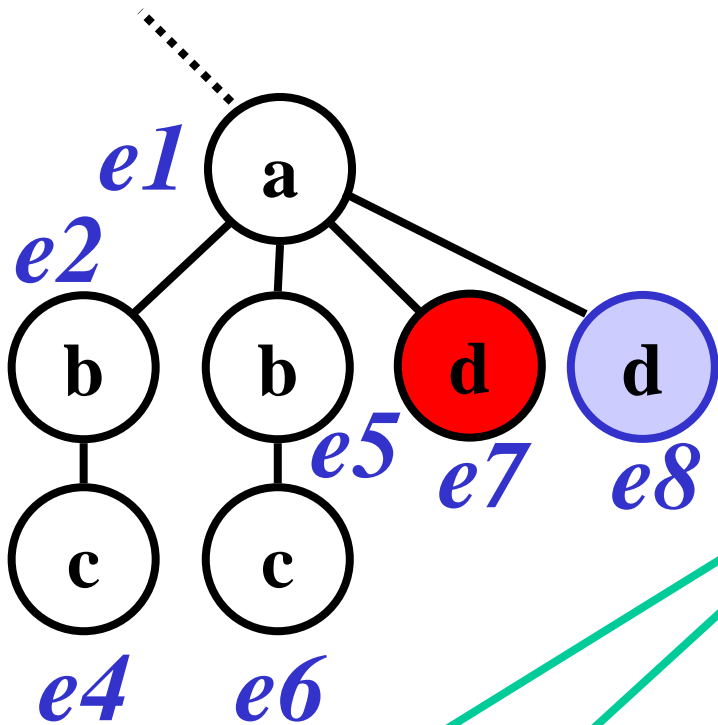
If it does not reach 0

$e2$  still matches  $s2$

stop

# Processing of Insertions

Q1: /.../a [ b [ c ] ] / d



so-called bottom-up  
evaluation strategy

If  $e8$  is inserted :

1. identify which steps in which queries it may match
2. we find it may match the step “d” of Q1
3. we go upward and examine ancestor steps one by one
4. we find ( $e1, s1, e7$ )

→ now we know  $e8$  matches Q1 without evaluating the rest of Q1

# Experiments

- Oracle 9i 64bit / Sun Blade 2000 (900MHz UltraSPARC-CU x 2) / 6GB memory
- XMark data (scaling factor 10, no attributes). 16,703k nodes. 846,755k bytes

Q2: / site [ people / person / name = 'Marek Gill' ] / categories / category / name

Q3: / site / closed\_auctions / closed\_auction [ \* // price = '146.02' ] / buyer

query	# of ans.	# of a / c	size of a / c (byte)
Q2	10,000	20,000	424,017
Q3	5	17	421

Size of matching information and counters

query	naive approach (ms)	our approach (ms)
Q2	3,845	25
Q3	8,816	20

Processing time for element deletion

(ms)	all-at-once	our method
counter at 3rd step	3,395	131
match (no counter)	3,395	3,556
fail at 1st step	20	10
fail at 2nd step	20	71
fail at 3rd step	3,345	3,426
fail at 4th step	11,537	11,557

Processing time for element insertion (Q2)

(ms)	all-at-once	our method
counter at 2nd step	1,862	54
match (no counter)	1,862	2,014
fail at 1st step	10	10
fail at 2nd step	931	34
fail in predicate	1,963	1,976
fail at 4th step	941	1,991

Processing time for element insertion (Q3)