Supporting Efficient Streaming and Insertion of XML Data in RDBMS

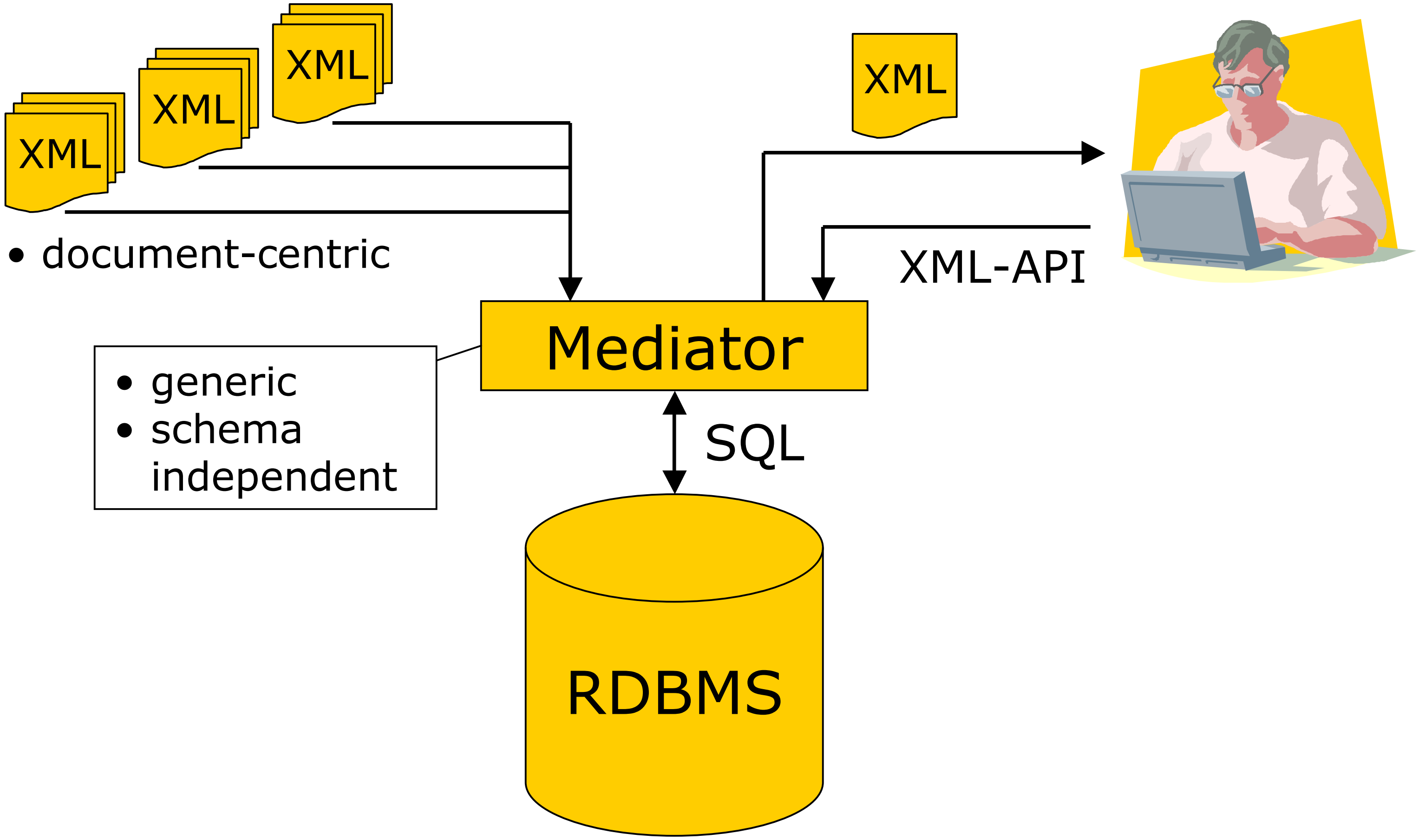
- Timo Böhme
  boehme@informatik.uni-leipzig.de
- Erhard Rahm
  rahm@informatik.uni-leipzig.de

http://dbs.uni-leipzig.de

Overview

- Motivation
- Numbering Schemes
- Dynamic Level Numbering (DLN)
- Evaluation

Motivation



XML XML XML

- document-centric

XML

XML-API

Mediator

- generic
- schema independent

SQL

RDBMS

Motivation

- demand to manage XML data with RDBMS
  - mature, scalable
  - query across XML and SQL data
  - single (existing) DBMS
- mapping XML ↔ relational
  - document-centered
  - data/schema-centered
  - structure-centered

Motivation
Mappings

- document-centered
  - save document as a whole (CLOB)
    - + fast storage/retrieval
    - + maintain original structure
    - – updates costly
    - – query evaluation may need parsing of XML data
- data/schema-centered
  - shredding XML into schema specific tables
    - + enable data-centered SQL operations
    - – problems with structure-oriented queries
    - – element sibling order, schema needed

Motivation

Mappings (2)

- structure-centered
  - mapping graph structure into predefined relations
    - + no schema needed
    - + structure-oriented/navigational queries
    - + efficient transformation of XML interfaces
    - – no direct SQL operations by user
  - mapping to parent-child relationships insufficient
    - determine ancestors/descendants
    - reconstruction of document fragments
  - solution: numbering schemes (NS)

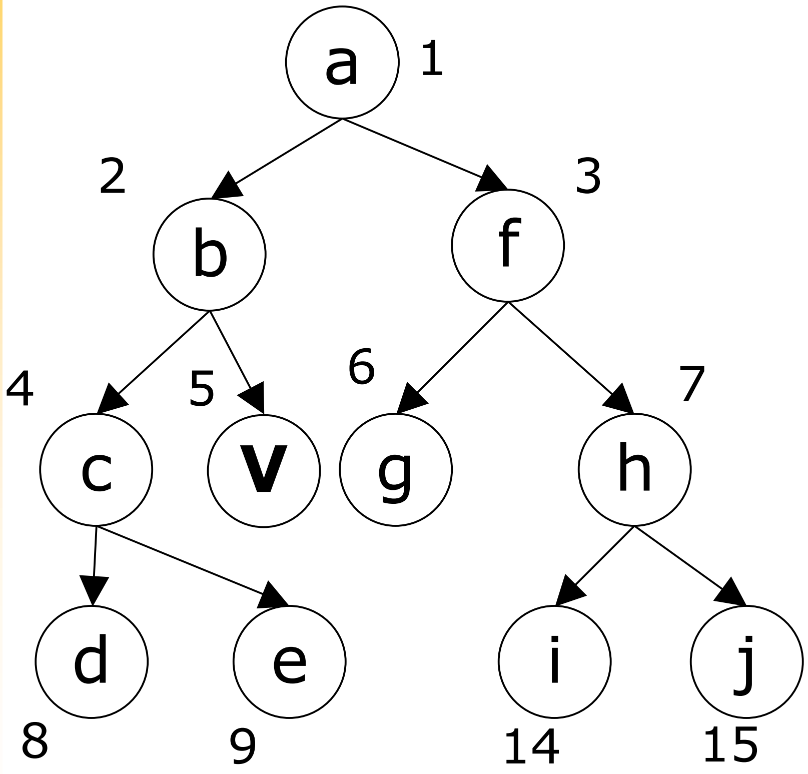
Motivation

Goal

- efficiently manage document-centric XML data in RDBMS
  - support sequential processing (streaming)
  - efficient updates (insertion of complex subtrees)
  - fast reconstruction
  - no manual interaction
- best suited mapping strategy: structure-centered with semantically rich NS
  - existing NS fail on streaming or updating

Related Work
Numbering Schemes

k-ary tree



- supports
  - calculation of parent ID

$$parent(i) = \left\lfloor \frac{(i-2)}{k} + 1 \right\rfloor$$

  - calculation of jth child ID

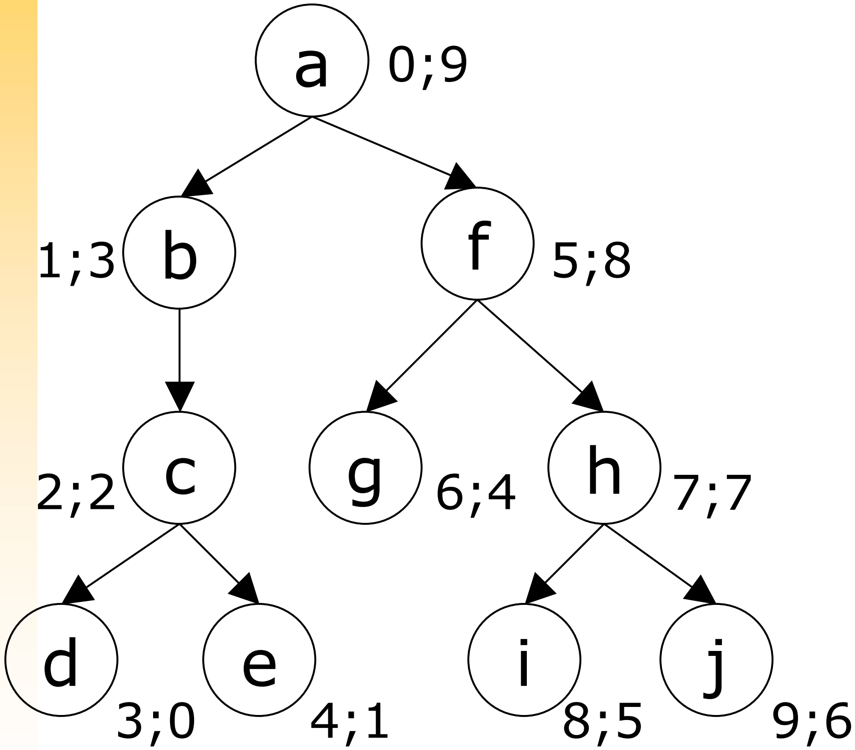
$$child(i,j)=k(i-1)+j+1$$

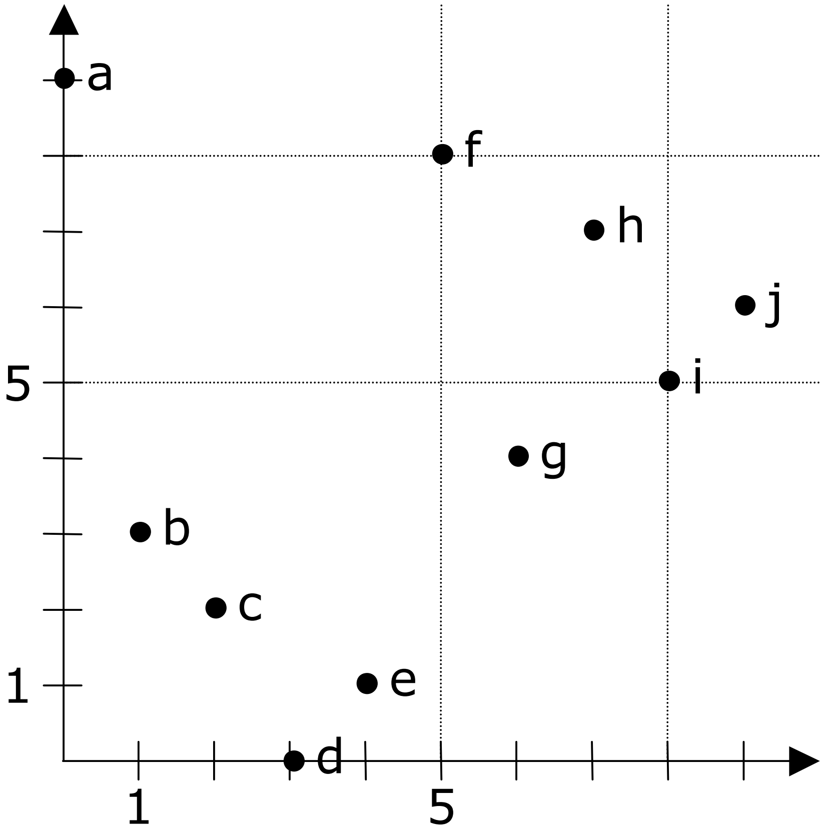
- negative
  - fixed, known fan-out
  - insertion costs
  - missing support for other XPath axis

Related Work

Numbering Schemes (2)

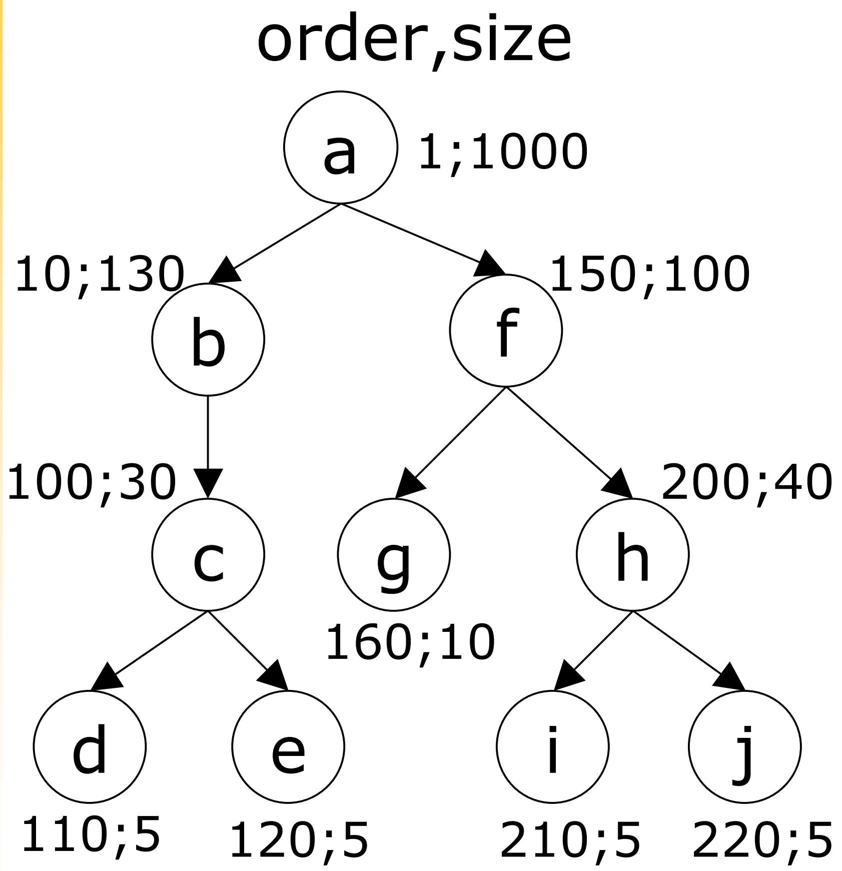
preorder, postorder





Related Work

Numbering Schemes (3)



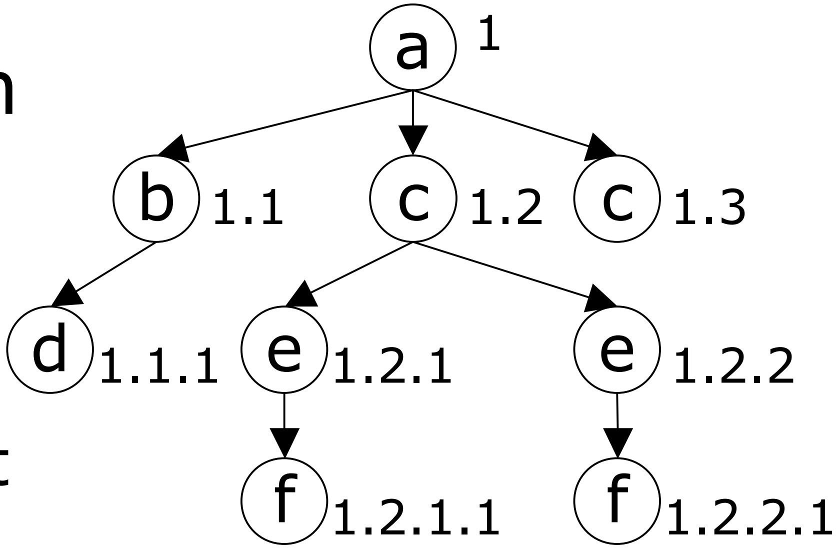
- supports
  - containment (descendant)
  - ancestor
  - restricted insert operations
- negative
  - parent, child not supported
  - assigning size values has to be delayed

Objectives for a new NS

- support wide range of documents
  - irregular documents (low and high fan-out)
  - streaming of large documents
- support efficient operations
  - explicitly express total node order
    - maintain global node order
    - fast reconstruction of document fragments
  - minimize necessity for renumbering after update operations
  - efficient processing of XPath expressions
  - exploitable by query optimizer of RDBMS

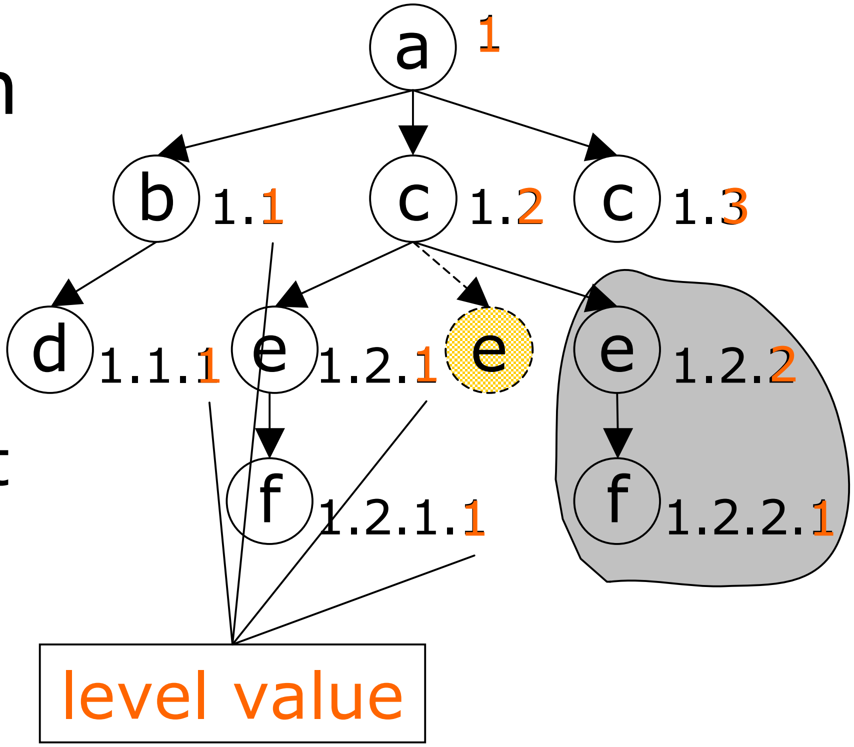
Dynamic Level Numbering

- based on Dewey Decimal Classification
- pro
  - ancestor/descendant deducible
  - sequential assignment of IDs



Dynamic Level Numbering

- based on Dewey Decimal Classification
- pro
  - ancestor/descendant deducible
  - sequential assignment of IDs
  - restricted update domain

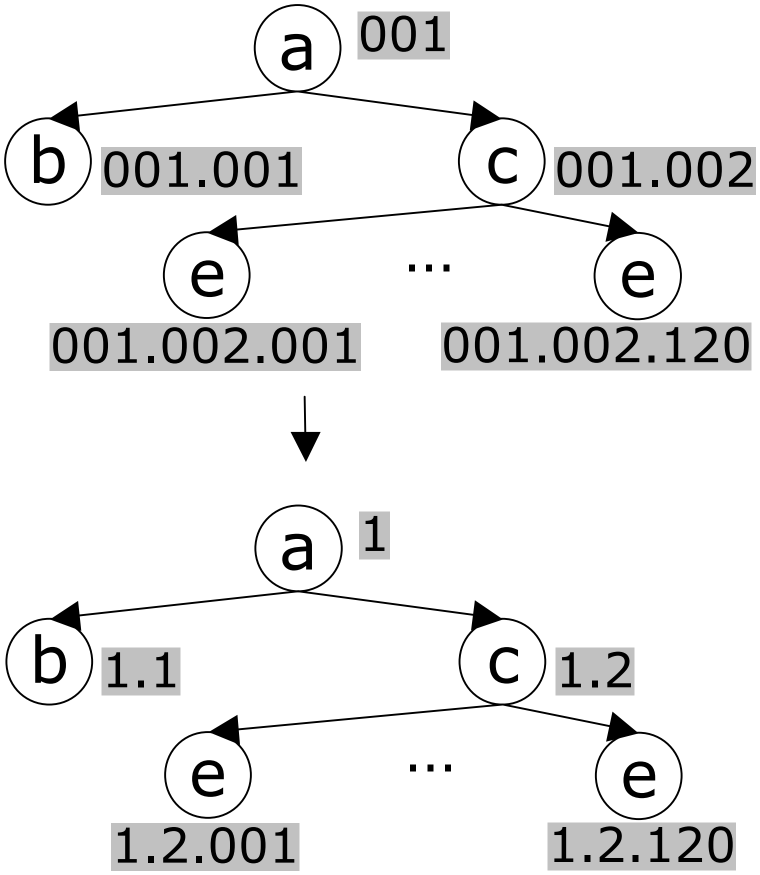


Decimal Classification

- cons of decimal classification
  - ID length depends on path length
  - updates can be costly
  - string comparison may return wrong result (e.g. 1.9 and 1.10)
    alternative:
    - level value with fix number of characters (001.009, 001.010)
    - level value coded in UTF-8 (0-127 1 Byte, 128-2047 2 Byte, 2.048-65.553 3 Byte etc., comparable at binary level)

DLN Enhancements

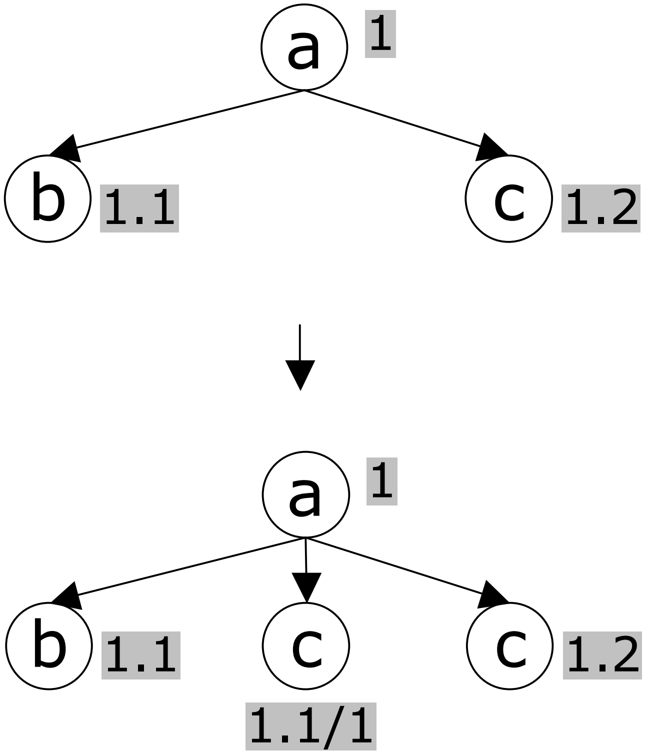
- optimization of needed characters
  - level values have variable number of digits
  - but: all sibling nodes have same number of digits



DLN

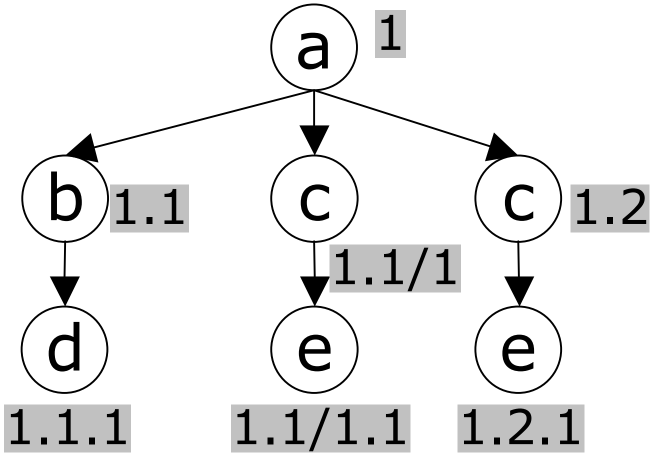
Enhancements (2)

- prevention of renumbering
  - supplement level values with subvalues (recursive)
  - separate subvalues and level values by character
    - greater than level separator
  - last subvalue >0



DLN
Enhancements (2)

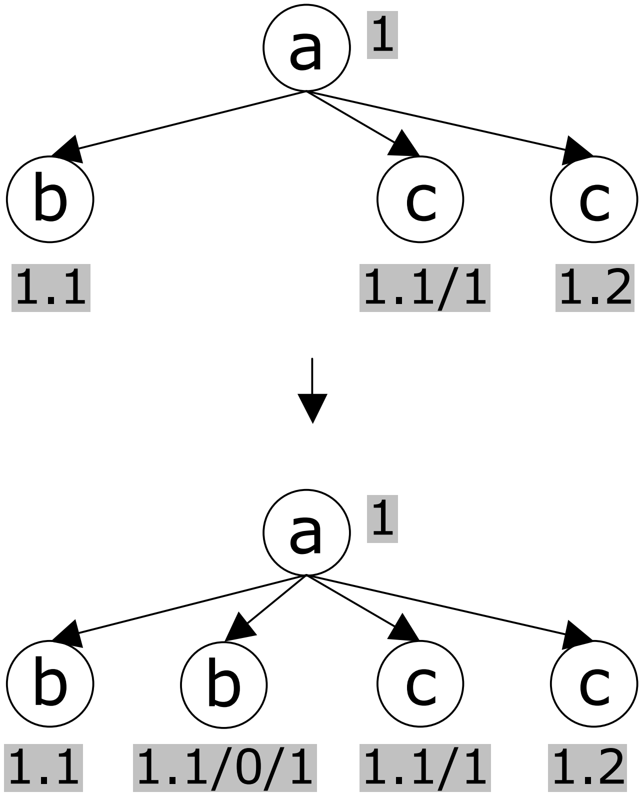
- prevention of renumbering
  - supplement level values with subvalues (recursive)
  - separate subvalues and level values by character
    - greater than level separator
  - last subvalue >0



1.1.1 < 1.1/1 < 1.1/1.1 < 1.2

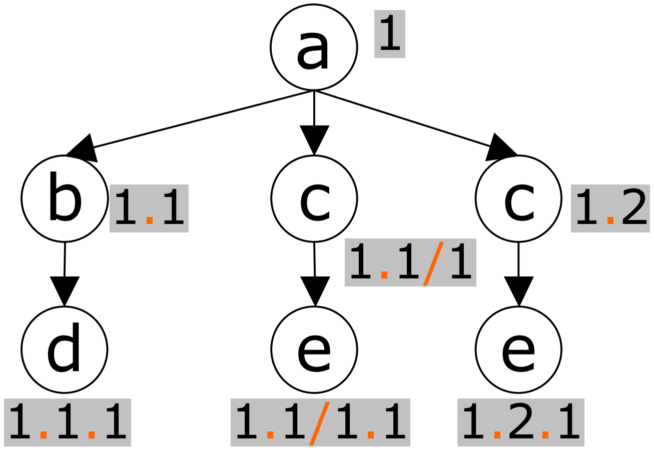
DLN
Enhancements (2)

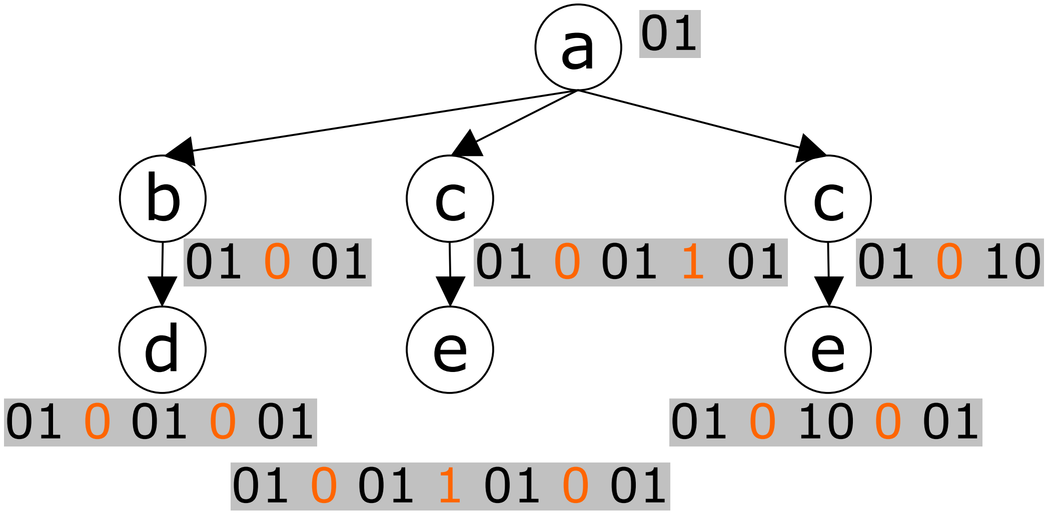
- prevention of renumbering
  - supplement level values with subvalues (recursive)
  - separate subvalues and level values by character
    - greater than level separator
  - last subvalue >0



DLN
Enhancements (3)

- binary coding to reduce ID length





1.1 <
1.1.1 <
1.1/1 <
1.1/1.1 <
1.2

01 0 01 <
01 0 01 0 01 <
01 0 01 1 01 <
01 0 01 1 01 0 01 <
01 0 10

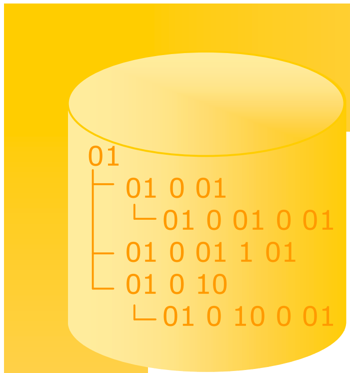
DLN
Properties

- preorder numbering
- update domain
  - worst case: identical to decimal classification
  - need for renumbering highly reduced by the use of subvalues
- supports parent, ancestor/descendant, preceding, following relationships
- ID length reduced due to variable digit number and binary coding

DLN
Streamed Data

- variable digit number not applicable
- solution (algorithm)
  - start with small bit number *n*
  - after *n-1* bits are exhausted add subvalue and use combined bit range

bit pattern	# of ids	enhanced # of ids
0XXX	1..7	1..7
10XX 1 XXXX	8..71	8..86
110X 1 XXXX 1 XXXX	72..583	87..613
1110 1 XXXX 1 XXXX 1 XXXX	584..4679	614..4724



DLN Variants

• streaming DLN – same number of bits for all values – different but fixed number of bits for level values and subvalues	• no metadata • suitable for all kinds of data
• simple DOM DLN – variable number of bits per value	• big metadata • minimal bit usage
• fixed DOM DLN – fixed number of bits per value	• no metadata • efficient use of combined bit range
• restricted DOM DLN – variable number of bits per value but restricted to a fixed upper bound	• small metadata • minimal overall bit usage

DLN – Evaluation

Maximum ID length

	Streaming DLN		DOM DLN			ORDPATH	
	4 bit	3/4 bit	simple	fixed	restr.	A	B
Nasa	64	2	-8	0	-9	0	-1
Cities	39	1	-1	-5	-5	5	7
Dictionary	69	-1	-13	-5	-12	9	-6
Novel	24	6	6	0	-2	3	-1
Pop. Places	44	2	-5	-10	-9	7	-1
Religion	54	-1	-8	-10	-10	-7	0
Shakespeare	44	5	-2	-5	-4	1	5
Sigmod	44	9	0	0	-2	2	7
Treebank	199	-21	-64	-5	-71	-3	-57
WFB	49	0	-3	0	-4	3	5
Courses	44	0	-3	-5	-7	-2	0

DLN – Evaluation
XMLRDB Prototype

NODE	
document	: int
dlnId	: long
parent	: long
rightSibling	: long
name	: int
value	: string
valType	: char
nodeType	: char

ATTR	
document	: int
dlnId	: long
name	: int
value	: string

TEXT	
document	: int
dlnId	: long
value	: string

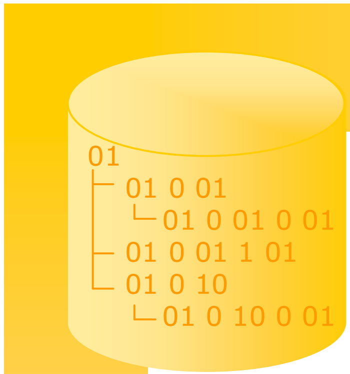
NAMEMAP	
nameId	: int
name	: string

DLN – Evaluation
Performance

doc	#nodes	n/kB	Insertion (n/sec)			Reconstruction (n/sec)
			w/o idx	w/o txtidx	with txtidx	
XMach-1	$2,2*10^6$	11	2.500	-	-	-
Cities	$3,5*10^4$	44	2.380	1.729	1.699	10.310
Sigmod	$3,8*10^4$	38	2.180	1.579	1.522	9.860
Bible	$2,5*10^4$	8	1.899	1.405	1.152	8.525

DLN – Evaluation
Performance

- queries with descendant axis (a//b) optimized by NS
  - range query with node id of a and maximum child id (calculated from a)
  - example
    - *//issue[.//author='Michael Stonebraker']]/volume* – 40 ms
    - */SigmodRecord/issue[articles/article/authors [author='Michael Stonebraker']]/volume* – 451 ms
      (both queries needed 190 ms to build XML result)



Summary

- new numbering scheme (DLN) for generic structure-centered storage of XML data in RDBMS
- insert operations without renumbering of existing nodes
- usable with streamed data
- benefits from structure information
- inserting, retrieving and querying efficiently supported