COS 597A:
Principles of
Database and Information Systems

Indexing for XML

Representation Choices

- Convert to relational representation
  - use indexes on relational attributes
  - see text
- Use native representation
  - directly represent tree model
    - node ⇔ element
    - child ⇔ nested element (subelement)
    - leaf ⇔ element value (text()) or attribute value
  - use indexes on values or element ids

Indexes on leaves

- B+ trees on values
  - can limit to certain element type
    - e.g. "date of birth" (v.s. "date of death")
  - B+ tree leaves hold pointers to document nodes (different tree!)

    search key value: pointer to node in doc. tree

    OR

    search key value: pointer to list of nodes in doc. tree

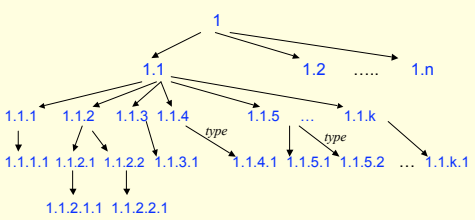
Capturing tree structure (I)

- One method: Summary trees
- For a given XML document
  - tree model (scheme) for node types
  - mapping: node type →
    list of nodes in tree model for document
- Path expression evaluation if *no* path conditions
  - find nodes in summary graph matching path
  - take union of lists of actual nodes
    - gives list of candidates for XQuery WHERE condition

Capturing tree structure II: eXist

- every document node gets a unique ID
- ID give ancestor/decendent "for free"
- uses dynamic level numbering
  - root ID = 1
  - parent ID = $k_0.k_1. \ldots .k_m \Rightarrow$ child ID = $k_0.k_1. \ldots .k_m.i$
    for i^{th} child left to right
  - common prefix gives nearest common ancestor (nca)
  - siblings: nca is parent of both
  - like library catalog numbering
  - variable-length encoding of ID *very* desirable

Example



edges labeled "*type*" point to attribute values with attribute name "*type*"
other leaves hold element values

Index & Calculate

- To calculate A // B
  - Get node IDs with name A: $\{a_i\}$
  - Get node IDs with name B: $\{b_j\}$
  - Do "Structural Join" to find those where a_i is ancestor b_j
    - Join on common tree structure
  - NOT tree traversal
- Need Index on names of elements
  - name element → (doc ID, node ID)
  - name attribute → (doc ID, node ID)

7

Using "structural joins"

- Query evaluation: don't always traverse tree root downward
- given index on values for element name,
  jump to node, then
  work up - must be able to
  OR
  structural join
- not limited to dynamic level numbering rep.
- eXist prefers path expression A // Q[B='C']
  to "FOR … WHERE"
  - get nodes IDs for A, for B='C' and for Q
  - JOIN

8