

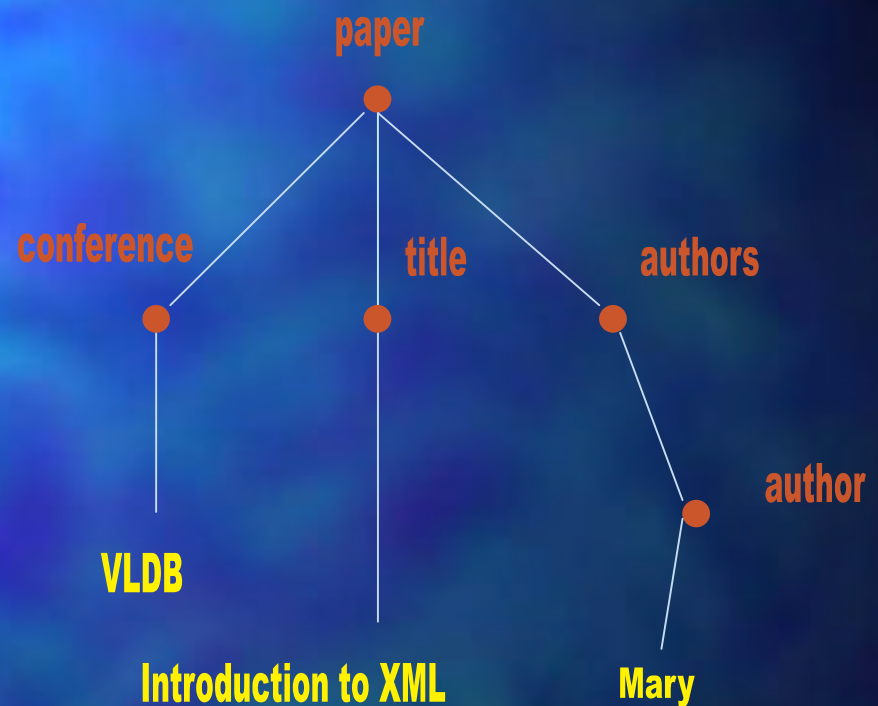
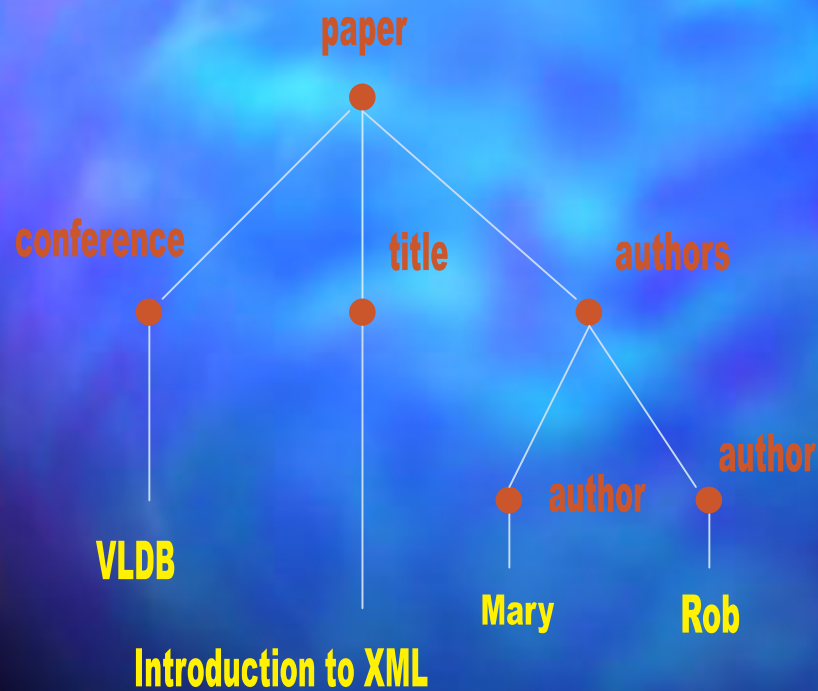
PART III
Correlating XML Data Sources

Nick Koudas
AT&T Research

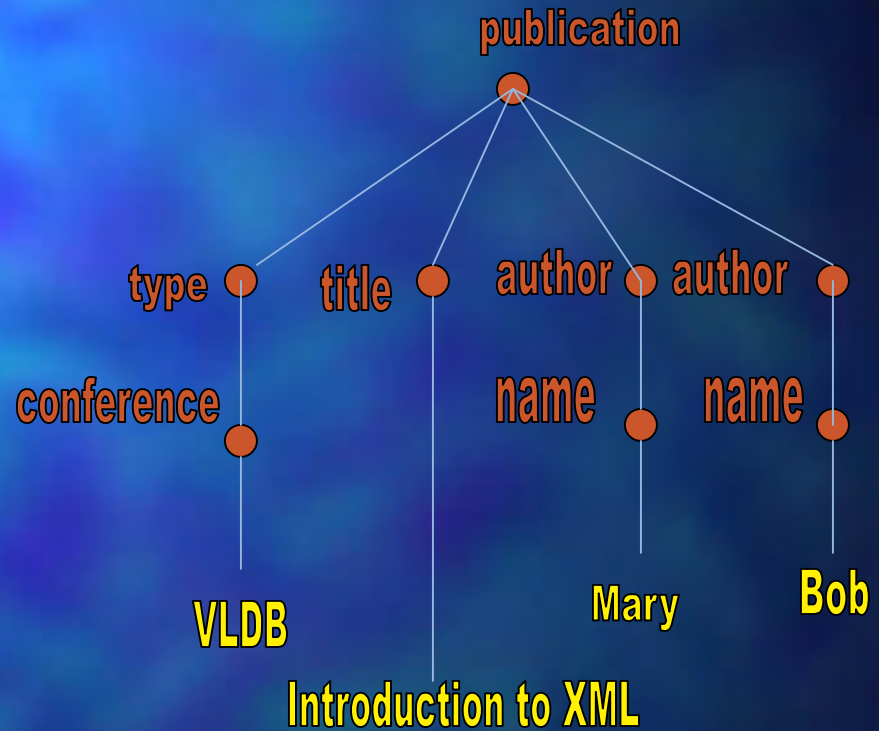
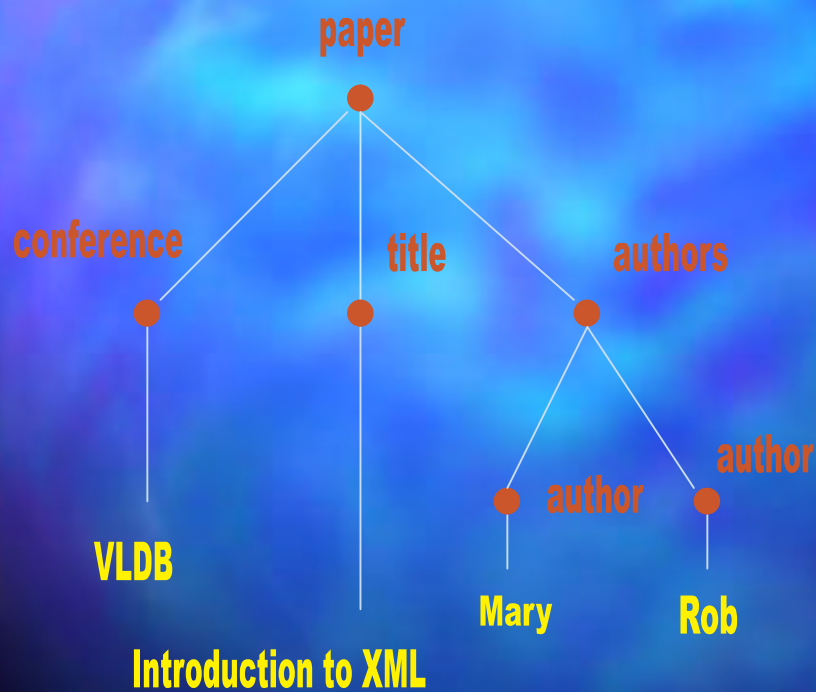
XML Data Correlation Steps

- Matching a pair of documents approximately
 - techniques for approximate match of a document pair
 - derive a 'cost' for matching the pair
- Matching all pairs of documents from two XML data sources
 - techniques for efficiently reporting all pairs with 'matching cost' less than a specified threshold

..and the problem begins..



..and the problem begins..



Approximately Matching Documents

- Efficient techniques to correlate XML data sources.
- Correlation predicate should account for mismatches in
 - Content
 - Structure

Approximate Matching Content

- Element context of string type is prevalent in XML
 - e.g., `<name> John Doe </name>`
- Requirement for techniques to match strings approximately
- Adapt well known metrics for quantifying string distance

Approximately Matching Content

- String edit distance between two strings s_1, s_2
 - minimum number of operations to transform string s_1 to s_2
- Allowable operations
 - character insert, delete, modify
 - Operations assume unit cost
 - `String`, `String` at distance 1

Approximately Matching Content

- Well known dynamic programming algorithm to compute edit distance, given two strings
- Worst case an $O(n^2)$ operation for strings for length n
- Many variants
 - Operations
 - Block moves ('John Doe', 'Doe John' unit cost)
 - Weighted operations
 - assigned relative weights to operations

Approximately Matching Content

- Vector string representation
- Map string to a vector based on token (character or q-gram) frequency
- Assess strings similarity using the inner product of vectors in that space
- Cosine similarity measure, between 0 and 1 for normalized vectors

Approximate Match in Structure

- What is the structural distance between a pair of XML documents?
- XML documents are ordered labeled trees
- Distance between two ordered labeled trees
 - minimum number of operations to transform one into the other
- Operations
 - tree node: insert, delete, re-label

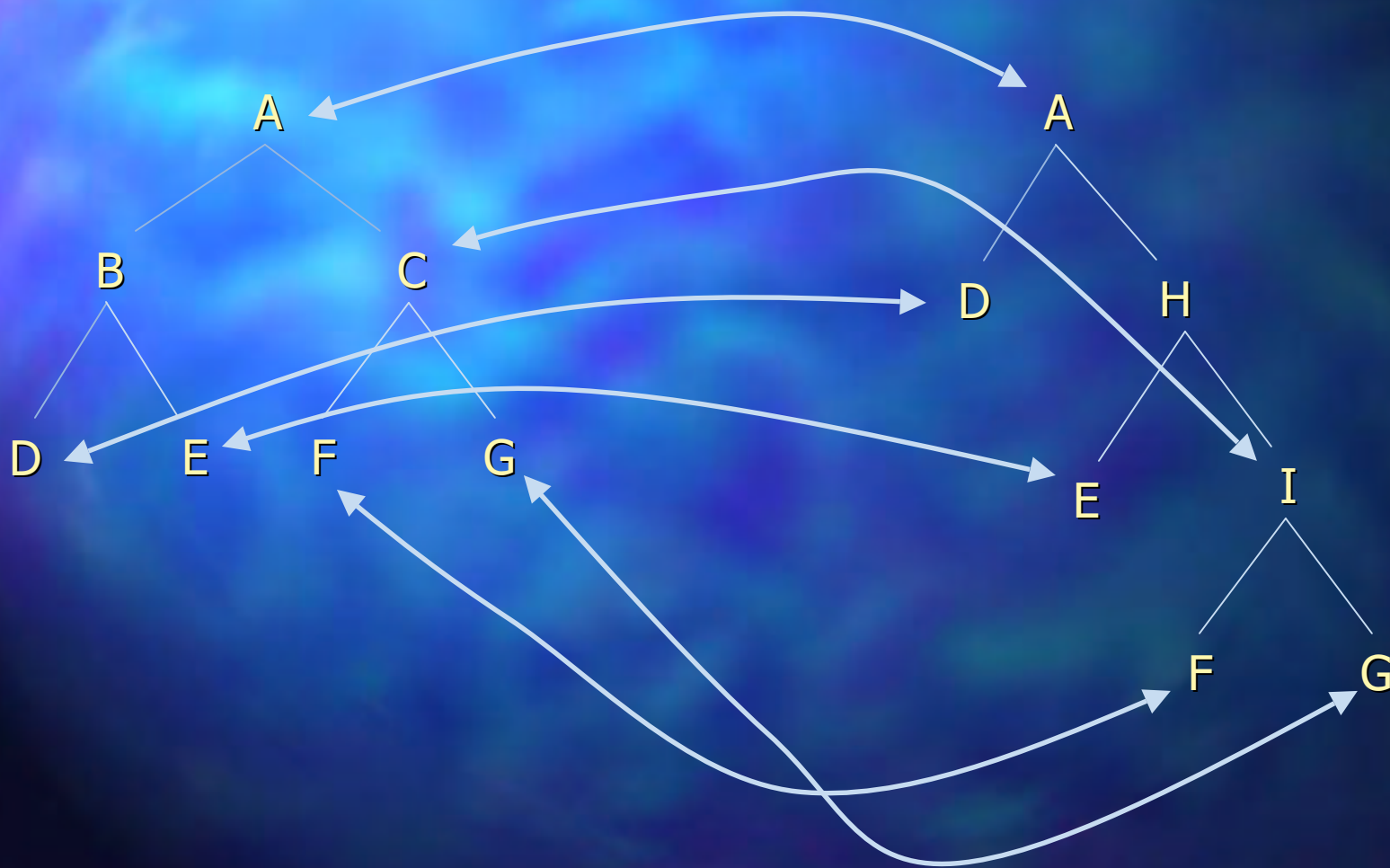
Approximate Match in Structure

- Various proposals available in the literature
- Varying degrees of complexity depending on assumption about XML trees and operations allowed
- Operations can have unit or other cost
- Two step approach
 - Derive a matching M between nodes of the two trees
 - Working on M derive an 'edit script', i.e the minimal sequence of operations transforming one tree to the other

Tree edit distance Tdist

- Operations allowed
 - node insert, delete, re-label
- Derive matching M
 - examine all pairs of nodes establishing an association between nodes
 - for tree of n nodes, $O(n^2)$ time
- Apply allowable operations in sequence to derive the minimum cost edit script

Tree edit distance ($Tdist(T_1, T_2)$)



Tree edit distance



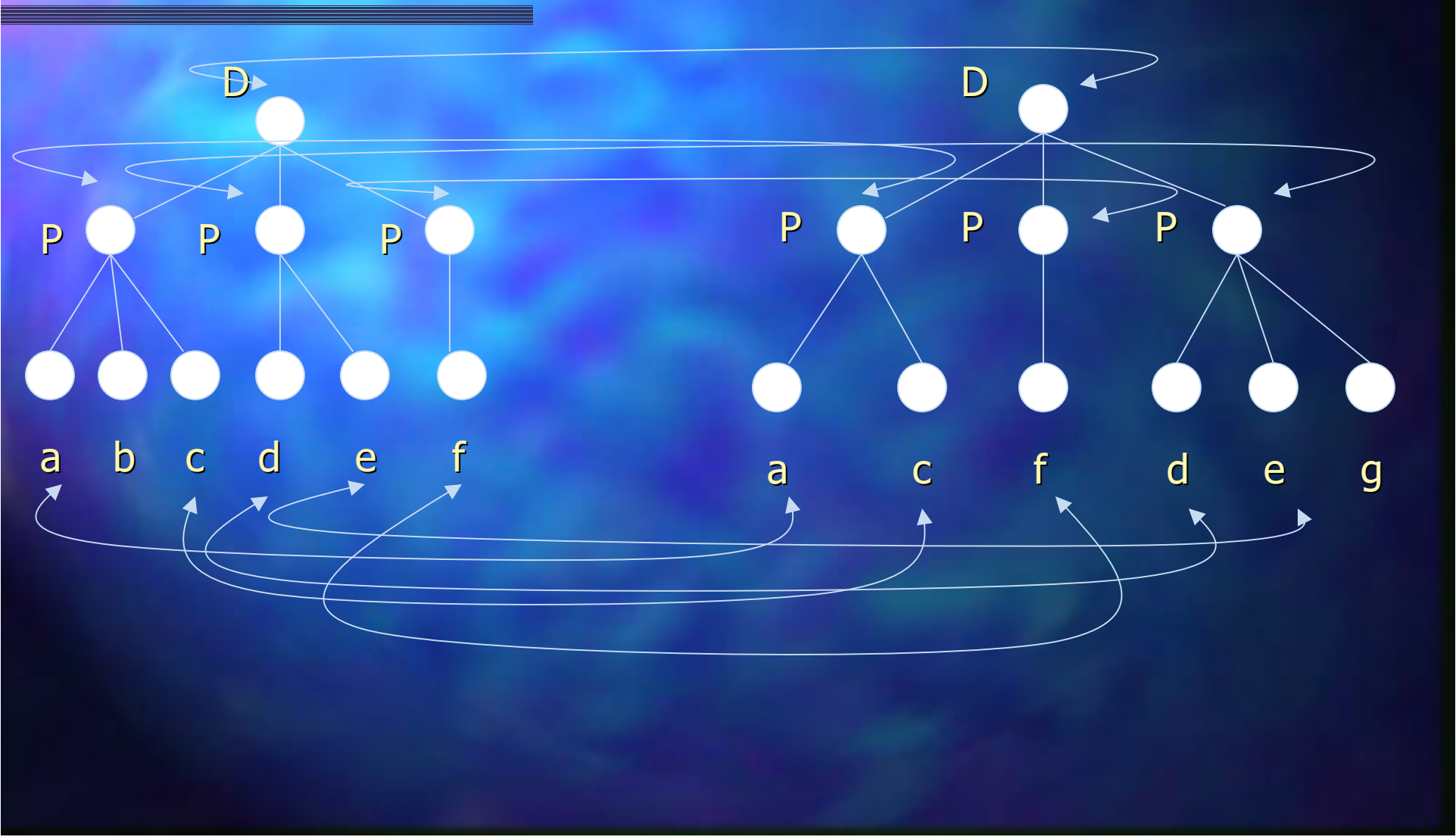
- Tree edit distance is a metric
- Expensive, $O(n^4)$ for trees with n nodes, $O(n^2 \log^2 n)$ for balanced trees

Algorithm FastMatch (Chawathe et.al, SIGMOD 1996)

- Proposed for change detection in hierarchical structured data.
- Operations allowed
 - tree node: insert, delete, re-label
 - tree node move
- Assumption: number of tree duplicate nodes is limited
- Two step approach:
 - Compute node Matching then derive edit script

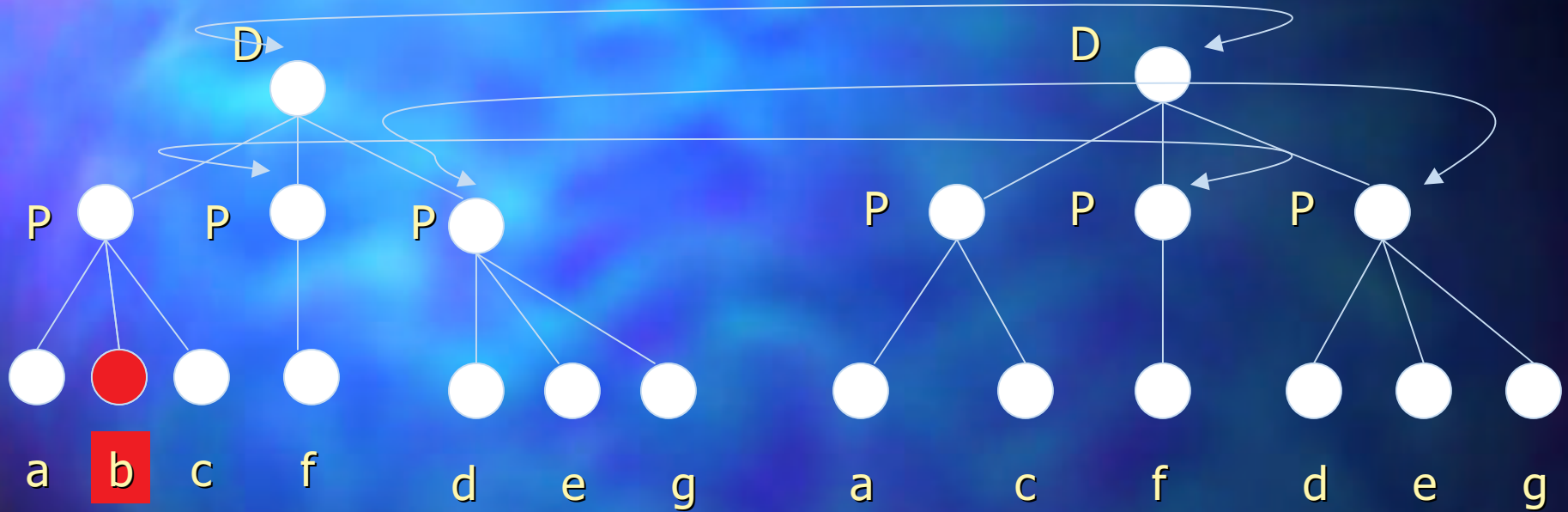
Algorithm FastMatch (Chawathe et.al, SIGMOD 1996)

- Improve complexity of deriving the matching by heuristically limiting the number of pairs considered during the match.
- Intuitively the heuristic is aiming to identify how different are two sub-trees structurally and consider matching nodes with not too dissimilar sub-trees



Algorithm FastMatch (Chawathe et.al, SIGMOD 1996)

- Once matching is derived the algorithm derives the edit script given the matching by
 - updating labels
 - aligning leaf nodes by assessing the longest common sub-sequence between sets of ordered leafs
 - inserting, moving and deleting nodes
- Complexity can be linear on the number of nodes.

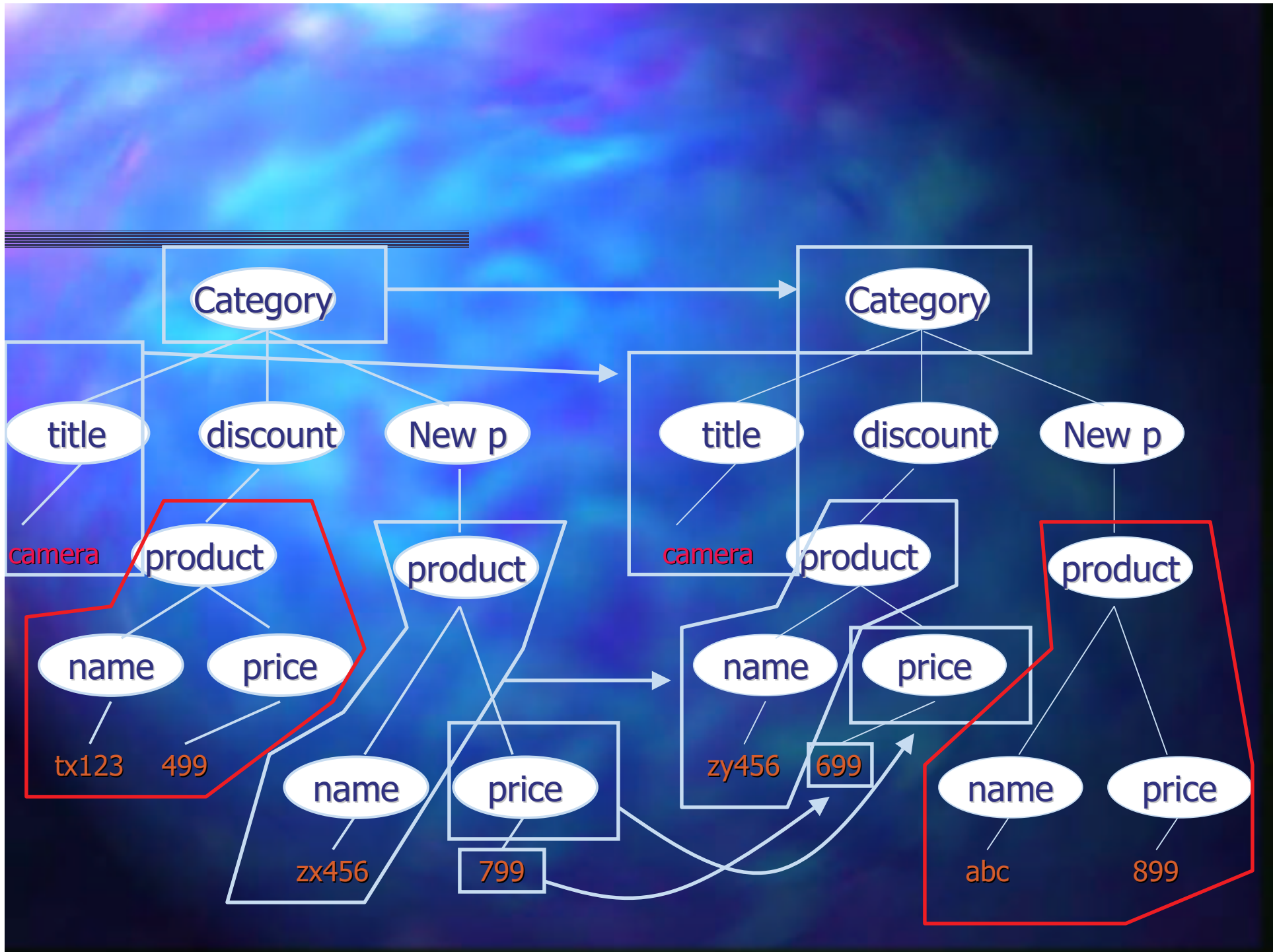


Algorithm MH-DIFF (Chawathe et.al, SIGMOD 1997)

- Raises assumptions of previous algorithm
 - ordered nodes, few duplicate node labels
- Reduces the problem to a minimum cost edge cover in a bipartite graph
- Complexity of the heuristic solution $O(n^3)$
- $O(n^2)$ in many practical scenarios

Algorithm BULD-diff (Cobena et. al., VLDB 2000)

- Proposed in the context of version management of XML documents
- Operations
 - sub-tree delete, insert
 - update of the value of a node or text attribute
 - move of a node or part of a sub-tree
- Similar execution style
 - derive a matching and then an edit script



XML Data Correlation Steps

- Matching a pair of documents approximately
 - techniques for approximate match of a document pair
 - derive a 'cost' for matching the pair
- Matching all pairs of documents from two XML data sources
 - techniques for efficiently reporting all pairs with 'matching cost' less than a specified threshold

Approximate XML Join

[SIGMOD '02]

- Let S_1 and S_2 be two XML data sources with the same or different DTD's
- Let $\text{dist}(d_1, d_2)$, $d_1 \in S_1, d_2 \in S_2$, a function assessing the distance (scoring, ranking) between entire XML documents or parts of documents (trees or sub-trees)
- We seek efficient ways to report all pairs (d_1, d_2) with $\text{dist}(d_1, d_2) \leq T$, for user specified T

Approximate XML Join

[SIGMOD '02]

- Ranking function should be a metric
- Bounds for tree edit distance
- Notion of reference sets
 - XML documents from which we encode relative distances
- Techniques for selecting reference sets based on sampling
- Efficient join and index based join [ICDE 2003] algorithms