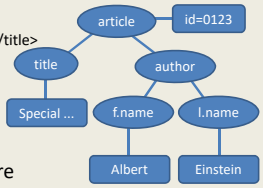
XML

- Extensible Markup Language
  - Origin: SGML (Standard Generalized Markup Language) 1986
  - Similar to HTML (Hyper Text Markup Language) for Web
  - XML 1.0: produced by W3C (World Wide Web Consortium) : 1998
- Trends
  - Recently, more and more data are represented and exchanged by XML on the Internet
    - A format for B2B data exchange
  - Store XML data as Database: XML-Database
    - Examples
      - Bibliography data: DBLP, ACM SIGMOD Record,
      - Bioinformatics data: TrEMBL, Swiss-Prot
      - …

Basic Syntax of XML

- Enclosed with <start tag> and <end tag>
  - <title>Special Theory of Relativity</title>
- Hierarchical Configuration (with attributes)
  - NG: <author>Einstein<title></author>Relativity</title>
  - Well formed :

```
<article id="0123">
  <title>Special Theory of Relativity</title>
  <author>
    <f.name>Albert</f.name>
    <l.name>Einstein</l.name>
  </author>
</article>
```



- Correspond to a tree structure

DTD

- Document Type Definition
  - To create rules for the elements in XML documents
- An example

```
<!DOCTYPE journal-db [
  <!ELEMENT journal (article)+>
  <!ELEMENT article (title, authors)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT authors (author)+>
  <!ELEMENT author (f.name, l.name)>
  <!ELEMENT f.name (#PCDATA)>
  <!ELEMENT l.name (#PCDATA)>
  <!ATTLIST article id IDREFS #REQUIRED>
]>
```

XML-DB vs. RDB

- RDB:
  - Fixed Schema : Table
    - Every tuple has all attributes
  - First Normal Form
    - Each element is atomic
  - Most common query language: SQL
- XML-DB
  - Flexible Schema: semi-structured
  - Hierarchical Configuration
  - Query langate: Xpath, XQuery

XPath

- A language to navigate, and find data, within XML documents.
  - XPath 1.0 became a Recommendation 1999 by W3C
  - XPath 2.0 is the current version
- Select one or more nodes to retrieve the data they contain
  - Expressed by location step
    - axis :: node-test [predicate]
  - axis: child, descendant, parent, following-sibling, …
  - /child::journal/child::article/child::authors/child::author/child::l. name
- Simplified
  - /journal/article/authors/author/l.name
  - /journal//l.name[Einstein]
  - //article[@id="0123"]

XQuery

- A language for querying XML data
  - Recommended by W3C
  - The means to extract and manipulate data from XML documents
    - To XML what SQL is to database tables
  - Use path expressions
  - Provides new XML documents to be constructed
- Syntax
  - FLWOR
    - FOR, LET, WHERE, ORDER BY, RETURN

Example of FLWOR of XQuery

```
for $x in doc("journal-db.xml")//article
let $a := $x/authors/author
where $x/year < 1930
order by $x//l.name descending
return
<articles-before-1930>
  <title>{$x/title}</title>
  <author>{$a/l.name, $a/f.name}</author>
</articles-before-1930>
```


Answer Example for the Query

```
<articles-before-1930>
  <title>Special Theory of Relativity</title>
  <author>Einstein, Albert</author>
</articles-before-1930>
<articles-before-1930>
  <title>Uncertainty Principle</title>
  <author>Heizenberg, Werner</author>
</articles-before-1930>
```


How to store a large amount of XML

- Native XML DB
  - Keep XML Data as it is : Tamigo
  - It cannot use many functions developed for RDB
- Combine with RDB
  - If there are a large number of XML data with other information
    - Store an XML sentence as character strings in RDB
      - It cannot use structure information in XML data
  - If the target XML data is a large set
    - Correspond each schema to an attribute of RDB
      - Less flexibility of the semi-structured configuration of XML
      - It cannot use structure information in XML data
    - Divide each node of a XML tree with label and store as a tuple
      - It can use structure information analyzing the label

Examples of Storing XML

	Journal	
	<article id="0123"> <title>Special Theory of Relativity</title> <author> <f.name>Albert</f.name> <l.name>Einstein</l.name> </author> </article>	
	<article id="56789"> <title>Uncertainty Principle</title> <author> <f.name>Werner</f.name> <l.name>Heizenberg</l.name> </author> </article>	

<m.name>Karl</m.name>

id	title	f.name	l.name	m.name?
01234	Special Theory of Relativity	Albert	Einstein	
56789	Uncertainty Principle	Werner	Heizenberg	

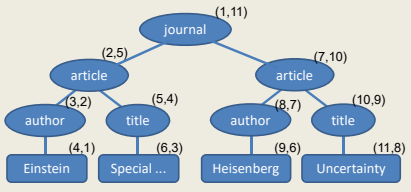
Labeling Methods

- To extract and reconstruct the structural information of XML data
  - Containment relationships, order of sibling, depth of nodes, etc.
  - Important for many applications such as XPath query, keyword search, etc.

Preorder-Postorder Method

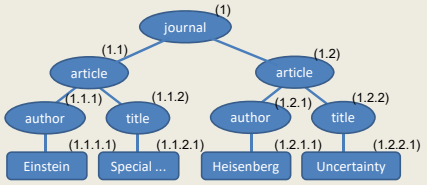
- Assign preorder and postorder numbers to all nodes.
  - Ancestor nodes and descendant nodes satisfy the following condition

ancestor's preorder<descendant's preorder and
ancestor's postorder>descendant's postorder



Dewey Order Method

- The node is expressed by
(parent node's value) . (the order of sibling)
- XML labeling method using Dewey Order has been proposed [Igor et al., 2002]



Storing XML into RDB with Labels

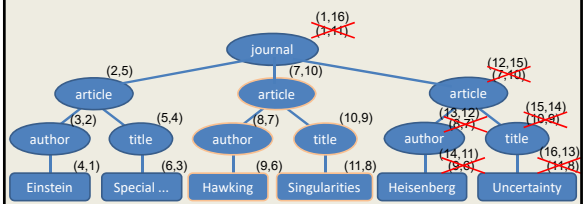
Node	Pre	Post
journal	1	11
article id=01234	2	5
author	3	2
Einstein	4	1
title	5	4
Special ...	6	3
article id=56789	7	10
author	8	7
Heizenberg	9	6
title	10	9
Uncertainty ...	11	8

Node	Dewey
journal	1
article id=01234	1.1
author	1.1.1
Einstein	1.1.1.1
title	1.1.2
Special ...	1.1.2.1
article id=56789	1.2
author	1.2.1
Heizenberg	1.2.1.1
title	1.2.2
Uncertainty ...	1.2.2.1

Both "Einstein" and "Special …" are descendant of "article id=01234"
("2 < 4 and 5>1" and "2<6 and 5>3" / 1.1.1.1 and 1.1.2.1 include 1.1)

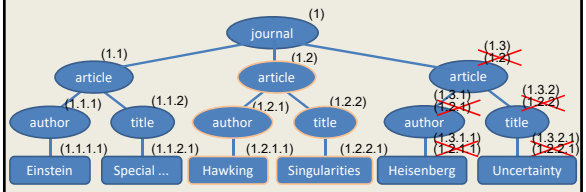
Problem of insertion (PP)

- When an insertion operation occurs, labels of many nodes must be changed
  - Because of using sequential numeric numbers



Problem of insertion (DO)

- When an insertion operation occurs, labels of many nodes must be changed
  - Because of using sequential numeric numbers

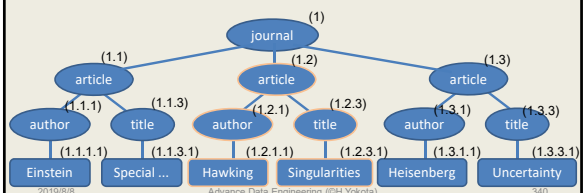


To Solve the Problem

- A number of methods have been proposed
- Prepare intervals between the numbers
  - SPARSE [Eda et al. 2002]
  - QRS (Use float numbers) [Amagasa et al. 2002]
  - Above methods require relabeling of many nodes once the prepared space is used up
- Methods without limitation
  - ORDPATH [O'Neil et al. 2004]
  - DO-VLEI Code [ICDE 2005]

ORDPATH [SIGMOD 2004]

- Use only positive odd numbers for initial labeling
  - Based on Dewey-Order numbers
- Negative integers and even numbers are used for insertion operations
- Used in Microsoft SQL Server 2005

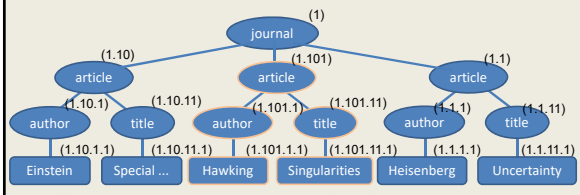


VLEI Code [ICDE2005]

- Variable Length Endless Insertable code
- Definition
  - A bit sequence beginning with 1 $1 \cdot \{0 \mid 1\}^*$
  - Satisfy the following condition $v \cdot 0 \cdot \{0 \mid 1\}^* < v < v \cdot 1 \cdot \{0 \mid 1\}^*$
  - Examples
    - 10<1<11
    - 100<10<101<1<110<11<111
- For any two adjacent VLEI codes v_l and v_r, a new code v between them can be generated by the following equation
  - IF $l(v_l) \leq l(v_r)$ then $v = v_l \cdot 1$ else $v = v_r \cdot 0$
  ($l(v)$ is v's length)
  - Example: *1<11 -> 1<110<11*

DO-VLEI [ICDE2005, IEICE2009]

- Apply VLEI code to Dewey Order Labeling
  - It does not require change of labels for other nodes
- It can convert into bit-strings by a mapping of
  - ".1" → (11), "1" → (10), "0" → (0)
  - Example: "1.10.11" → (10 11 0 11 10)

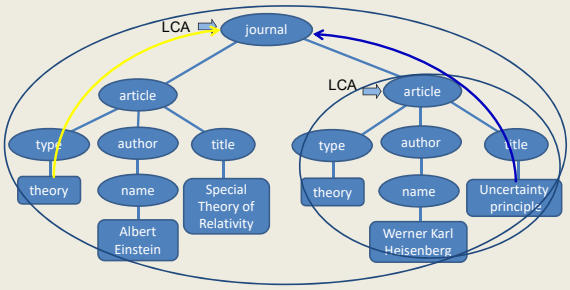


Keyword Search in XML-DB

- Keyword search is important to query XML data
  - XML databases tend to contain text data
  - Users do not know the structure
- Important point of the keyword search in XML database
  - Which sub-tree should be returned as the answer
- Sub-trees containing all keywords are the search target
  - When multiple search keywords are given
- Lowest Common Ancestor (LCA) would be the root of the sub-tree
  - There will be a number of candidate sub-tree for a set of multiple keywords

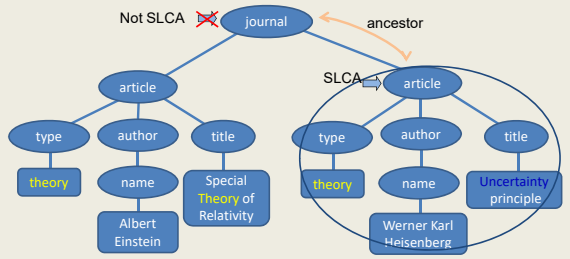
LCAs for a set of Keywords

- Keywords: uncertainty & theory



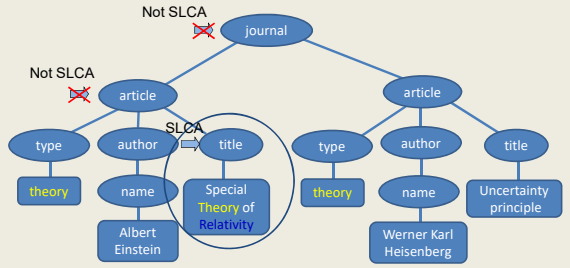
SLCA: Smallest LCA

- The LCA that is not an ancestor of any other LCAs
- Keywords: uncertainty & theory



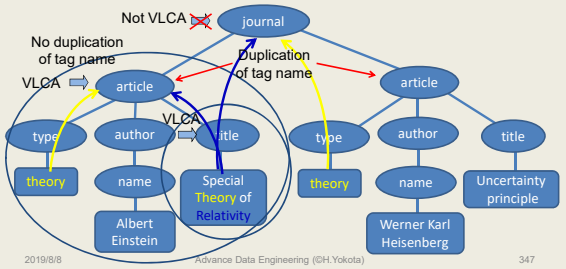
Problem of SLCA

- Sometimes SLCA is not appropriate
- Keywords: relativity & theory



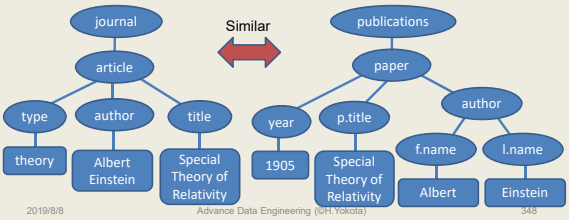
VLCA : Valuable LCA

- Condition: There is no duplication of tag-name between leaves and the VLCA
- Keywords: relativity & theory



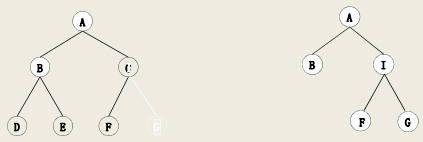
Similarity Search in XML-DB

- Similar contents are represented in different structures in XML
- Similarity Search is useful for
  - Integration of different XML Databases
  - Duplication elimination in an XML Database



Distance Metric for Ordered Labeled Tree

- Tree Edit Distance (TED) [K. Zhang and D. Shasha, 1997]
  - Traditional metric for measuring the edit distance between ordered labeled trees
  - The minimum cost edit operations (insertions, deletions and changes) that transforms one tree to the other



Delete D→ Delete E→ Change C to I→ Insert G→ TED=4

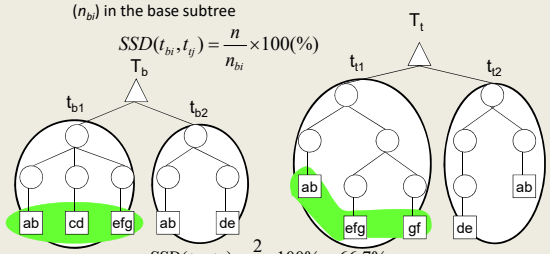
Problem of TED

- Expensive computational cost
  - Worst case: $O(n^4)$ operation
  - Inapplicable for large-scale Data
- For the documents with similar structure
  - It is difficult for TED to distinguish the similarity differences between them
- To tackle the problem, a number of methods have been proposed
  - Tag similarity based method [Butter, 2004]
  - Entropy based method [Helmer, 2007]
  - LAX: Leaf clustering based method [Liang&Yokota 2005]

LAX (1/3)

- Subtree Similarity Degree (SSD)
  - The percentage of matched leaf nodes (n) out of total leaf nodes (n_{bi}) in the base subtree

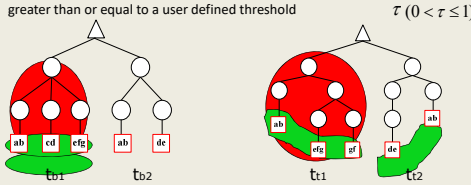
$$SSD(t_{bi}, t_{tj}) = \frac{n}{n_{bi}} \times 100(\%)$$



$$SSD(t_{b1}, t_{t1}) = \frac{2}{3} \times 100\% = 66.7\%$$

LAX (2/3)

- Matched Subtree (T_M) & Hit Subtree (T_H)
  - T_M: The pair of subtrees t_{bi} and t_{tj} that has the maximum subtree similarity degree
  - T_H: The matched subtree T_M is a hit subtee, iff the similarity degree of T_M is greater than or equal to a user defined threshold $\tau\ (0 < \tau \leq 1)$



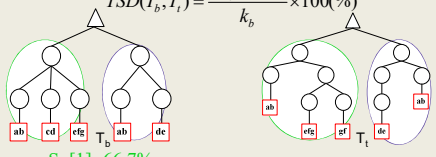
$SSD(t_{b1},t_{t1})=2/3*100\%=66.7\%$ $SSD(t_{b1},t_{t2})=1/3*100\%=33.3\%$

$SSD(t_{b1},t_{t1})=66.7\%>= \tau$?

LAX (3/3)

- Tree Similarity Degree (TSD)
  - Let $S_M[i]$ record the similarity degree of each matched subtree. The tree similarity degree $TSD(T_b,T_t)$ is defined as the mean value of the similarity degree of matched subtrees

$$TSD(T_b,T_t)=\frac{\sum_{i=1}^{k_b} S_M[i]}{k_b}\times 100(\%)$$



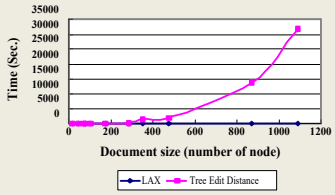
$S_M[1]$=66.7%

$S_M[2]$=100%

$TSD(T_b,T_t)$=(66.7%+100%)/2*100%=83.4%

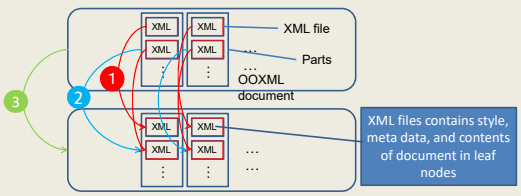
Comparison with TED

- Efficiency
  - Compare execution time using synthetic data
  - *LAX* is much faster than TED



OOXML search system utilizing structure and style

- We propose Structure-based Office document Search (SOS)
  1. Calculate similarity of XML file
  2. Calculate similarity of parts from score of step
  3. Calculate similarity of OOXML files from score of step



Summary of this course

- Data Engineering: A research area managing Data
  - How to store, search for, retrieve and process data efficiently
- Data Warehousing
  - Stock up data of operational transaction processing
  - Apply OLAP (Data Cube) and data mining methods
- Devices to store a large amount data reliably
- Indexing methods to search for target data in storages
- Cost estimation of relational database operations
- Parallel relational database operations / Skew handling
- Distributed databases / B2B connections
- Data engineering for Cloud environment
  - HDFS, KVS, Semantic Web, Security, RDF DB, XML DB