University of Washington
Department of Computer Science & Engineering

Managing XML and Semistructured Data

Lecture 16: Indexes

Prof. Dan Suciu

Spring 2001

In this lecture

- Indexes
  - XSet
  - Region algebras
  - Dataguides
  - T-indexes

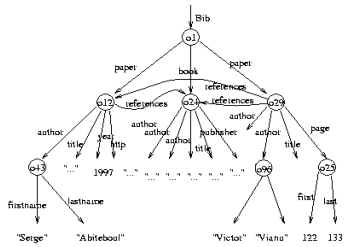
Resources

- Index Structures for Path Expressions by Milo and Suciu, in ICDT'99
- XSet description: http://www.openhealth.org/XSet/
- **Data on the Web** Abiteboul, Buneman, Suciu : *section 8.2*

The problem

- Input: large, irregular data graph
- Output: index structure for evaluating regular path expressions

The Data



Semistructured data instance = a large graph

The queries

- Regular expressions (using Lorel-like syntax)

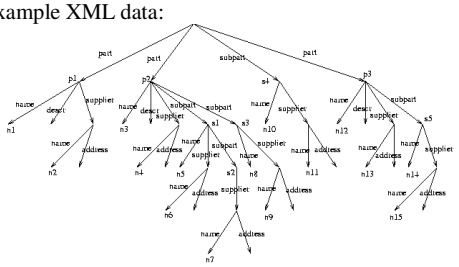
```
SELECT X
FROM
(Bib.*.author).(lastname|firstname).Abiteboul X
```

Analyzing the problem

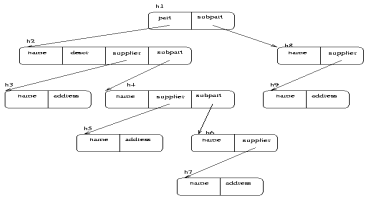
- what kind of data
  - tree data (XML)
  - graph data
- what kind of queries
  - restricted regular expressions (e.g. XPath)
  - arbitrary regular expressions

XSet: a simple index for XML

- Part of the Ninja project at Berkeley
- Example XML data:



XSet: a simple index for XML



Each node = a hashtable

Each entry = list of pointers to data nodes (not shown)

XSet: Efficient query evaluation

- `SELECT X FROM part.name X` -yes
- `SELECT X FROM part.supplier.name X` -yes
- `SELECT X FROM part.*.subpart.name X` -maybe
- `SELECT X FROM *.supplier.name X` -maybe

Will gain when index fits in memory

Region Algebras

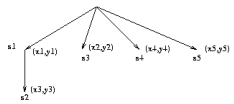
- structured text = text with tags (like XML)
- powerful indexing techniques
  [Baeza-Yates, Gonnet, Navarro, Salminen, Tompa, etc.]
- New Oxford English Dictionary
- critical limitation:ordered data only (like text)
- less critical limitation: restricted regular expressions

Region Algebras

- data = sequence of characters $[c_1 c_2 c_3 \ldots]$
- region = interval in the text
  - representation (x,y) = $[c_x, c_{x+1}, \ldots c_y]$
  - example: **`<section> … </section>`**
- region set = a set of regions
  - example all **`<section>`** regions (may be nested)
- region algebra = operators on region set,
  `s1 op s2`

Representation of a region set

- Example: the **`<subpart>`** region set:



Region algebra: some operators

- s1 intersect s2 = {r | r∈ s1, r ∈s2}
- s1 included s2 = {r | r∈s1, ∃r' ∈ s2, r ⊆ r'}
- s1 including s2 = {r | r∈ s1, ∃r' ∈ s2, r ⊇ r'}
- s1 parent s2 = {r | r∈ s1, ∃r'∈ s2, r is a parent of r'}
- s1 child s2 = {r | r∈ s1, ∃r' ∈ s2, r is child of r'}

Examples:

<subpart> included <part> = { s1, s2, s3, s5}

<part> including <subpart> = {p2, p3}

Efficient computation of Region Algebra Operators

Example: s1 included s2

s1 = {(x1,x1'), (x2,x2'), …}

s2 = {(y1,y1'), (y2,y2'), …}

(i.e. assume each consists of disjoint regions)

Algorithm:

if xi < yj then i := i + 1

if xi' > yj' then j := j + 1

otherwise: print (xi,xi'), do i := i + 1

Can do in sub-linear time when one region is very small

From path expressions to region expressions

part.name	name child (part child root)
part.supplier.name	name child (supplier child (part child root))
*.supplier.name	name child supplier
part.*.subpart.name	name child (subpart included (part child root))

Region expressions correspond to simple XPath expressions

DataGuides

- Goldman & Widom [VLDB 97]
  - graph data
  - arbitrary regular expressions

DataGuides

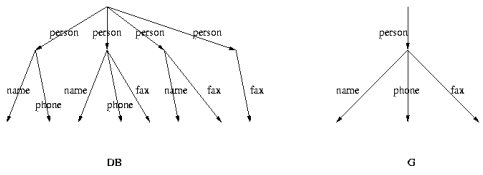
Definition

given a semistructured data instance DB, a *DataGuide* for DB is a graph G s.t.:

- every path in DB also occurs in G
- every path in G occurs in DB
- every path in G is unique

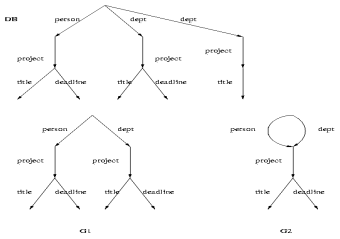
Dataguides

Example:



DataGuides

- Multiple DataGuides for the same data:



DataGuides

Definition

Let w, w' be two words (I.e word queries) and G a graph

$$w \equiv_G w' \text{ if } w(G) = w'(G)$$

Definition

G is a *strong* dataguide for a database DB if $\equiv_G$ is the same as $\equiv_{DB}$

DataGuides

Example:

- G1 is a strong dataguide
- G2 is not strong

$$\text{person.project} \; !\equiv_{DB} \text{dept.project}$$
$$\text{person.project} \; !\equiv_{G2} \text{dept.project}$$

DataGuides

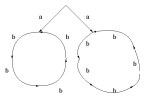
- Constructing the strong DataGuide G:

```
Nodes(G)={{root}}
Edges(G)=∅
while changes do
   choose s in Nodes(G), a in Labels
   add s'={y|x in s, (x -a->y) in Edges(DB)} to Nodes(G)
   add (x -a->y) to Edges(G)
```

- Use hash table for Nodes(G)
- This is precisely the powerset automaton construction.

DataGuides

- How large are the dataguides ?
  - if DB is a tree, then size(G) <= size(DB)
    - why? answer: every node is in exactly one extent of G
    - here: dataguide = XSet
  - How many nodes does the strong dataguide have for this DB ?



20 nodes (least common multiple of 4 and 5)

Dataguides usually fail on data with cyclic schemas, like:

T-Indexes

- Milo & Suciu [ICDT 99]
- 1-index:
  - data graph
  - arbitrary regular expressions
- 2-index, T-index: for more complex queries, consisting of more regular expressions.

1-Indexes

- A first attempt:
- Database: DB = (V,E,Roots)
- Queries: regular path expressions q(DB)

$\forall u \in V.\quad Lu \Leftrightarrow \{a_1 \ldots a_n \mid v_0 \xrightarrow{a_1} \ldots \xrightarrow{a_n} v_n \in DB,\ v_0 \in Root,\ v_n = u\}$

$\forall u,v \in V.\ u \equiv v \ \Leftrightarrow L_u = L_v$

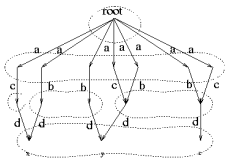
$\forall u \in V.\quad [u] = \{v \mid u \equiv v\}$

1-Indexes

I = { Nodes(I) = { [u] | u in nodes(DB) }
Edges(I) = { s → s' | ∃u ∈ s, ∃u' ∈ s', (u →$_a$ u') ∈ Edges(DB)}

q(DB) = { u | ∃ s ∈ q(I), u ∈ s }

Example:



Inefficient: construction cost (PSPACE)

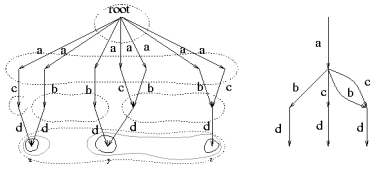
1-indexes

- IDEA: Use Simulation or Bisimulation instead of ≡
  Fact: $u \approx_b v \Rightarrow u \approx_s v \Rightarrow u \equiv v$
  Use the same construction, but [u] now refers to $\approx_b$ instead of ≡.

  Works because $L_u = L_{[u]}$
  Efficient PTIME algorithms exist for computing $\approx_b$ and $\approx_s$ [Paige&Tarjan, Henzinger&Henzinger&Kopke]

1-Indexes

- Example



DB I

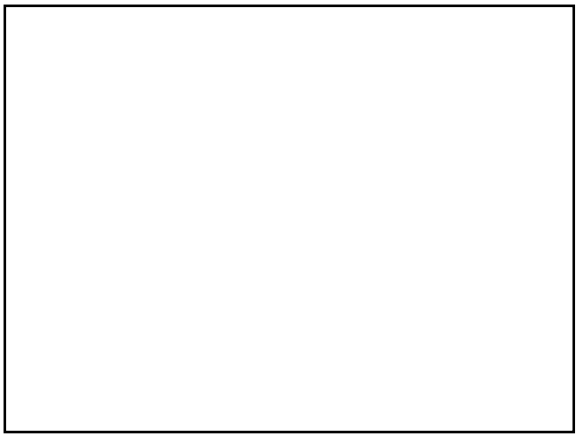
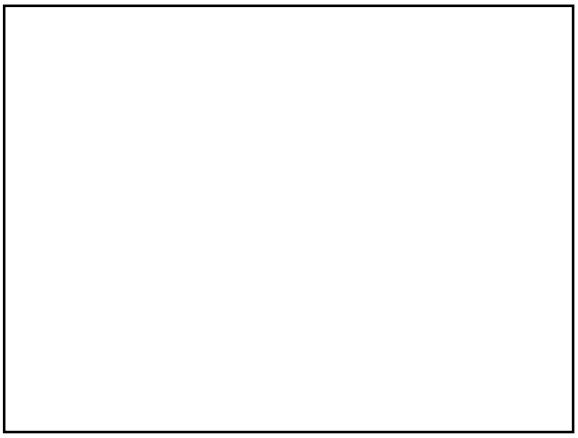
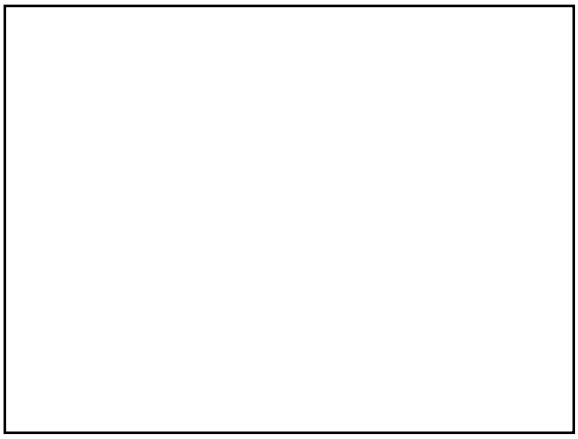
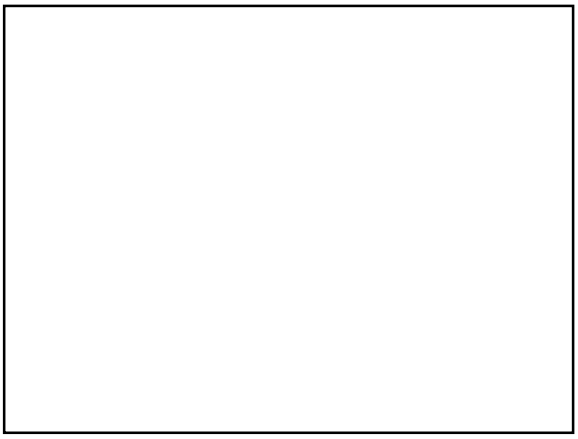
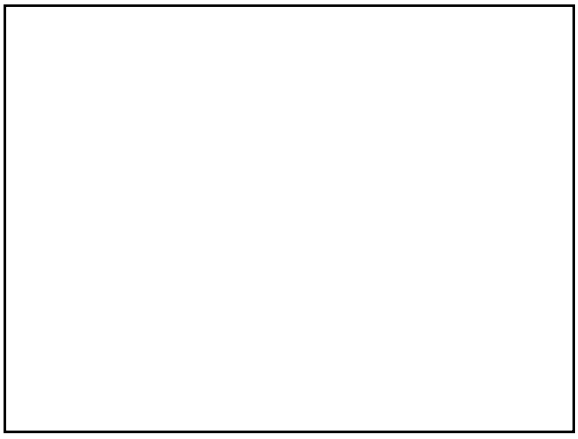
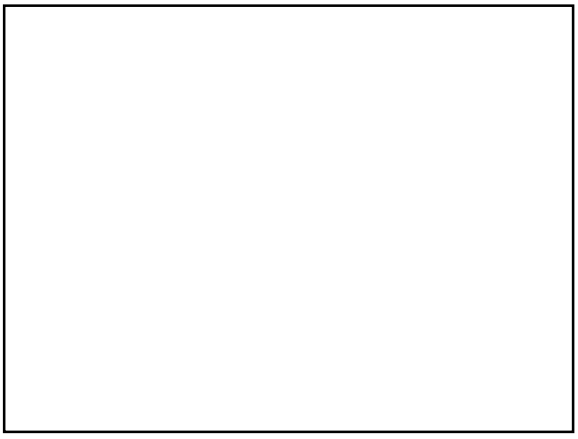
1-Indexes

- Analyzing the 1-index
- always: size(I) <= size(DB) (unlike Dataguide)
- always: can compute in O(nlogn) time n=size(DB)
- When DB is a tree: $\approx_b, \approx_s, \equiv$ coincide
  - no penalty for $\approx_b, \approx_s$
  - 1-index = Dataguide = XSet

1-Indexes

- Analyzing the 1-index:
- Do we have size(I) << size(DB) ? No. Two worst cases:

- Facts:
  - in theory: except for these two DB's, size(I) << size(DB)
  - in practice: it's a different story. Experiments: size(I) ≈ 1/3 size(DB)

Conclusions

- work on structured text: relevant but restrictive
- trees are simple: XSet = Dataguides = 1-index (conceptually)
- 1-index: scales to cyclic data too
- more complex queries: 2-index, T-index
- T-index: space/generality tradeoff
- Problem: how to use a specific T-index to answer a given query. Query rewriting (see [ICDT'99]).
- Need external-memory algorithm for bisimulation/simulation.