Dynamic schema discovery

András Z. Salamon

10 September 2014
stitching tables back together from RDF
Dematerialisation

given table,
generate RDF tuples representing it
RDF: subject (IRI or blank), predicate (IRI),
object (IRI or blank or literal)

(IRI = Internationalized Resource Identifier)
Dematerialisation

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>McLeod</td>
<td>NULL</td>
<td>98</td>
</tr>
<tr>
<td>Pitt</td>
<td>82</td>
<td>39</td>
</tr>
<tr>
<td>BNI</td>
<td>name</td>
<td>age</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>-----</td>
</tr>
<tr>
<td>McLeod</td>
<td>NULL</td>
<td>98</td>
</tr>
<tr>
<td>Pitt</td>
<td>82</td>
<td>39</td>
</tr>
</tbody>
</table>
Dematerialisation

<table>
<thead>
<tr>
<th>BNI</th>
<th>name</th>
<th>age</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>McLeod</td>
<td>NULL</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>Pitt</td>
<td>82</td>
<td>39</td>
<td></td>
</tr>
</tbody>
</table>

for each row
assign blank node identifier (BNI)
for each attribute
print (BNI, attribute, entry) if entry != NULL
Dematerialisation

<table>
<thead>
<tr>
<th>BNI</th>
<th>name</th>
<th>age</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>_:...a2</td>
<td>McLeod</td>
<td>NULL</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>Pitt</td>
<td>82</td>
<td>39</td>
</tr>
</tbody>
</table>

for each row assign blank node identifier (BNI)
for each attribute
print (BNI, attribute, entry) if entry != NULL

(_:...a2,<datatype>#name,"McLeod")
(_:...a2,<datatype>#integer,98)
(_:...a2,<datatype>#dataset,tableURI)
Dematerialisation

<table>
<thead>
<tr>
<th>BNI</th>
<th>name</th>
<th>age</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>_:...a2</td>
<td>McLeod</td>
<td>NULL</td>
<td>98</td>
</tr>
<tr>
<td>_:...d0</td>
<td>Pitt</td>
<td>82</td>
<td>39</td>
</tr>
</tbody>
</table>

For each row:
- Assign blank node identifier (BNI)
- For each attribute
  - Print (BNI, attribute, entry) if entry != NULL

(_::...d0,<datatype>#name,"Pitt")
(_::...d0,<datatype>#age,82)
(_::...d0,<datatype>#integer,39)
(_::...d0,<ontology>#dataset,tableURI)
Rematerialisation

given a collection of RDF tuples, reconstruct table they came from
Data

Problem Schemas & simulation Making it practical
Structure

Problem Schemas & simulation Making it practical
Database schema

keep headings of table, throw away data (easy)
dependencies (hard)

what if table not given?
Database

Problem

Schemas & simulation

Making it practical
Database schema (alternative view)

“most general” entry in each relation

```
_::...a2

#name
McLeod

#score
98

#dataset
tableURI

_::...d0

#name
Pitt

#age
82

#score
39

#dataset
tableURI
```
Database schema (alternative view)

“most general” entry in each relation

Table URI

Row

McLeod 98

Pitt 82 39
Database schema (alternative view)

“most general” entry in each relation

<table>
<thead>
<tr>
<th>tableURI</th>
</tr>
</thead>
<tbody>
<tr>
<td>#row</td>
</tr>
<tr>
<td>http:// ... d0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#name</th>
<th>#age</th>
<th>#score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pitt</td>
<td>82</td>
<td>39</td>
</tr>
</tbody>
</table>
Up-ended tree
Database schema (yet another view)

A simulates B if A is at least as general as B
keep only maximal elements in this order
quotient of RDF graph: keep only most general
linked data is forest-like: tangled core, tree-like periphery
Personal view

linked data is forest-like: tangled core, tree-like periphery

most real RDF graphs are trees or are close to trees (Pichler, Polleres, Wei, Woltran 2008)
98.4% of connected components were trees (Mallea, Arenas, Hogan, Polleres 2011)
power-law distributions (Luo, Fletcher, Hidders, De Bra, Wu 2013)
Computer science forest
Real forest

Problem

Schemas & simulation

Making it practical
Tree-like periphery
Forest-like tree
Computing largest simulation

$n$ nodes in RDF graph ($\sim 500 \times 10^6$) takes $O(n^3)$ time, $O(n^2)$ space (oops...)
bit messy...
less messy
$k$-bisimulation

(Luo, Fletcher, De Bra, Wu, van Heeswijk @TU Eindhoven; Hidders @TU Delft; Vansummeren @ULB 2012–4; Buneman, S., Viglas, Klein, Waites @Ed 2011–2) $k$-step simulation

one layer at a time (leaves and parent)
do $k$ layers ($k$ small)
output hashes of local schemas as stream
can do in parallel (Hadoop)
(also DAG representation of XML: Maneth/S.)
Not perfect
typos in predicates or dirty data: heuristics, ML relies on sorting (but incremental updates OK) literals not always equivalent: type inference needs row with all attributes: concept analysis can’t touch tangled core
Conclusion

dynamic schema discovery works well
“$k$-bisimulation” ($k$-step simulation)
ongoing work