XSPARQL

An XML-RDF transformation and query language combining XQuery and SPARQL

Axel Polleres[1] Thomas Krennwallner[1,2] Nuno Lopes[1]
Jacek Kopecký[3] Waseem Akhtar[1]

[1]DERI, National University of Ireland, Galway

[2]Knowledge-Based Systems Group, Institute for Information Systems, TU Wien

[3]STI Innsbruck, University of Innsbruck, Austria

Lightning Talk presented by Alexandre Passant (DERI)

Motivation

relations.xml

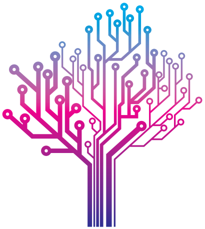
```
    <relations>
      <person name="Alice">
        <knows>Bob</knows>
        <knows>Charles</knows>
      </person>
      <person name="Bob">
        <knows>Charles</knows>
      </person>
      <person name="Charles"/>
    </relations>
```

?

↔

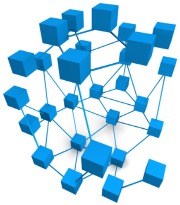
relations.rdf

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:b1 a foaf:Person;
     foaf:name "Alice";
     foaf:knows _:b2;
     foaf:knows _:b3.
_:b2 a foaf:Person; foaf:name "Bob";
     foaf:knows _:b3.
_:b3 a foaf:Person; foaf:name "Charles".
```



$\leftarrow$ SPARQL + XSLT, XQuery

XSLT, Xquery $\rightarrow$



both not an ideal fit...

Can we do better? YES!

Mapping RDF to RDF

Generate fullname from first and last name:

```
construct { _:b foaf:name {fn:concat("""",$N," ",$F,"""")} }
from <vcard.rdf>
where {
  $P vc:Given $N .
  $P vc:Family $F .
}
```

```
_:b1 foaf:name "Waseem Akhtar"
_:b2 foaf:name "Jacek Kopecky"
_:b3 foaf:name "Axel Polleres"
.
.
.
```

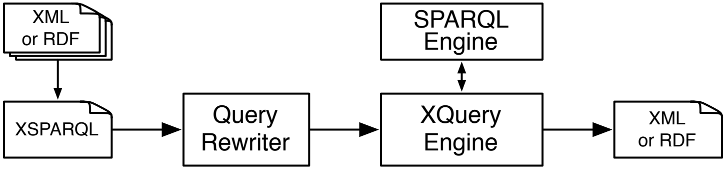
Mapping RDF to XML

```
<relations>{
for $Person $Name
from <relations.rdf>
where { $Person foaf:name $Name }
order by $Name
return <person name="{$Name}">{
  for $FName
  from <relations.rdf>
  where {
    $Person foaf:knows $Friend .
    $Person foaf:name $Name .
    $Friend foaf:name $FName
  }
  return <knows>{$FName}</knows>
 }</person>
}</relations>
```

```
<relations>
 <person name="Alice">
  <knows>Bob</knows>
  <knows>Charles</knows>
 </person>
 <person name="Bob">
  <knows>Charles</knows>
 </person>
 <person name="Charles"/>
</relations>
```

XSPARQL Semantics + Implementation

- Formal semantics of XSPARQL: extension of the XQuery semantics by plugging in SPARQL semantics in a modular way



- Rewriting algorithm is defined for embedding XSPARQL into native XQuery plus interleaved calls to a SPARQL endpoint

- Benefits: rely on off-the-shelf components

http://xsparql.deri.org/