

A Database Take on the Semantic Web

Piazza: Data Management Infrastructure for Semantic Web Applications, Halevy et al., to appear, WWW2003

Zachary G. Ives
University of Pennsylvania

April 16, 2003

Tim Berners-Lee's Vision

A Web where one can pose rich queries and the system “understands semantics” – and can thus provide much more than string matching

➤ The semantic web is a giant “concept mediator”

But how do we implement such a thing?

1. What's the representation of concepts and instances?
2. How do we express relationships between concepts?
3. How do we pose queries?

Last Week's Answer: Knowledge Representation!

What's the representation of concepts and instances?

- Concept definitions in an extended RDF: basically, RDF classes
- Instances are RDF objects

How do we express relationships between concepts?

- RDF triples encode relationships (obj, relationship, obj)
- OWL ontologies represent a taxonomy of concepts
Ontology mapping: Can express that concept in one ontology equals concept in another

How do we pose queries?

We define new concepts in OWL and ask for matching instances

Strengths of KR Techniques

RDF has some nice built-in semantics

- `rdf:id` is guaranteed to be a unique identifier
- All relationships between objects are labeled

KR Concept Definitions can express very powerful relationships

- Students are People and take Classes
- Mothers are those People who are Female and have Children
- In Philadelphia, Buildings prior to 1970 had a Height less than the Height of City Hall

Ontologies can encode relationships between concept definitions

Some Criticisms of KR

How do we scale ontologies to the Web?

- Difficult to come to agreement on how things should be represented
- Difficult to extend and maintain
- *(Same problems as a global mediated schema, only harder!)*

KR answer: no one claims there will be one ontology

- Can specify exact correspondences between ontologies' concepts
- *Does this solve the problem?*

Most of the world's data isn't in RDF!

KR answer: that's OK; we need to redesign from ground up

KR queries require complex reasoning, often EXPTIME

KR answer: people don't usually ask the complicated queries

A Database Perspective

XML + Schema ~ RDF

- Schema allows us to encode special meaning for certain attribs
- CAVEAT: not all relationships between objects are labeled (But does a label = a meaning?)

Views can approximate concept definitions

`Students(ID) :-`

`People(ID,...), Enrolled(ID,cID), Course(cID,...)`

`Mothers(ID) :- People(ID,...,"F"), ParentOf(child,ID)`

- We'll see how to describe the Philly concept later

Can reason about relationships between views:

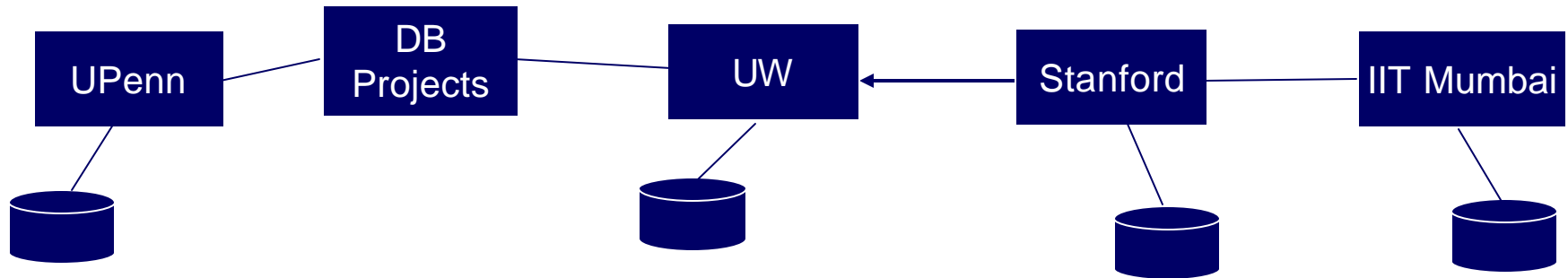
“Query containment”: all answers to one view contained in answers to another view

Roughly analogous to defining a subclass

Why Would We Want to Do This?

- Terabytes of existing data that's in XML (or easily translatable to XML)
 - Hierarchical and relational data
 - Spreadsheets, Java objects, ...
 - XML files, RDF itself!
- We can compose and reason about views
- Avoids need for a single ontology/mediated schema
 - Composition of mappings between schemas
 - Use views to define relationships, i.e., to map between different concepts
- Database techniques generally scale better
 - Datalog execution is at most polynomial in size of data

Piazza and the Semantic Web



Mediates between “structured” data formats, including XML, RDF

- **Itself a semantic web**
 - View-based concept definitions
 - View-based mappings
 - Answers XQueries
- **But can also be used to map XML to RDF (or RDF to RDF)!**
 - KR-based SW is more expressive in some ways
 - Piazza can be used to “import” data from another format

The Issues

- What makes data “structured”?
- What do mappings need to accomplish?
- How do we specify mappings?
- How do we answer queries using mappings?

Structured Data #1/4

We encode data values with:

- Tags describing what the data is
- Hierarchy describing membership

RDF:

- Everything is in triples; hierarchy isn't really used
- Almost exactly corresponds to an ER diagram

XML:

Roughly speaking, a tree representation of an E-R diagram – where some relationships are made implicit

Structured Data #2/4

- RDF explicitly names relationships:

(book, title, "ABC")

(book, writtenBy, author)

(author, name, "John Smith")



- XML does not always:

1.

```
<book>
  <title>ABC</title>
  <writtenBy>
    <author><name>John Smith</name></author>
  </writtenBy>
</book>
```
2.

```
<book>
  <title>ABC</title>
  <author>John Smith</author>
</book>
```

Structured Data #3/4

- RDF is subject-neutral (a graph)
- XML centers around a subject (a tree):
 1. `<book>`
 `<title>ABC</title>`
 `<author>John Smith</author>`
 `</book>`
 2. `<author>`
 `<name>John Smith</name>`
 `<book>ABC</book>`
 `</book>`
- This may result in duplication of contained objects

Structured Data #4/4

- RDF relationship labels help, but don't guarantee the same granularity of concepts:
 1. (Person, eatsForBreakfast, Meal1)
(Person, eatsForLunch, Meal2)
(Person, eatsForDinner, Meal3)
 2. (Person, eatsMeals, MealList)
(MealList, breakfast, Meal1)
(MealList, lunch, Meal2)
(MealList, dinner, Meal3)
 3. (Person, eatsMeals, list of Meal)
(list of Meal := {Meal1, Meal2, Meal3})
- Whether we're in XML- or RDF-land, we still need to map between different levels of abstraction or hierarchy

Some Requirements for Structured Mappings

- Mapping will often need to supply the relationships implicit in a schema
- Equivalence (value and object)
 - $S2:/author/name \Leftrightarrow S1:book/author$
 - $Dollar \Leftrightarrow EuroToDollar(Euro)$
- Fusion:
 - Refs to $S1:book/author$ with same name are the same object
- Label \Leftrightarrow value
 - $(Person,authored,book) \Leftrightarrow (Action,subject,directObject)$
- Splitting: $S3:author/first \Leftrightarrow split(S2:author/name, " ")$
- Sub-concept: $Book \subseteq Publication$
- Super-concept: $House \supseteq HouseInPhilly$
- *What else might be useful?*

Challenges with Mappings

- Information may be lost in one direction of a mapping:
 - $\text{Name} := \text{concat}(\text{FirstName}, \text{LastName})$
 - $\text{Faculty} := \text{Professors} \cup \text{Lecturers}$
- Correspondences may be hard to specify precisely:
 - $\text{Bug} \sim \text{Insect}$
- Data may be dirty or incomplete
- Exact mappings may be computationally expensive

Mappings in Piazza

- Goals:
 - Build on XQuery and XML
 - Remain computationally inexpensive
 - Capture the common mapping types
- Custom mapping language based on templates

```
<output>
  {: $var IN document("doc")/path WHERE condition :}
  <use>$var</use>
</output>
```

 - Designed for translating between parts of data instances (vs. XQuery, which is designed to create a new data instance in a new schema)
 - Restricted to a subset of XQuery that's tractable to reason about
 - Supports special annotations and object fusion

Example Schemas

- Target:

- pubs
 - book*
 - title
 - author*
 - name
 - publisher*
 - name

- Source:

- authors
 - author*
 - full-name
 - publication*
 - title
 - pub-type

Example Piazza Mapping

```
<pubs>
  <book piazza:id={$t}>
    {: $a IN document("...")/authors/author,
      $t IN $a/publication/title,
      $typ IN $a/publication/pub-type
      WHERE $typ = "book"
      PROPERTY $t >= 'A' AND $t <= 'B' :}

    [: <publisher>
      <name> {: PROPERTY $this IN
              {"PrintersInc", "PubsInc"} :}
      </name>
    </publisher> :]
  </book>
</pubs>
```

Semantics of Mappings

- Mappings are, roughly speaking, GAV
- Each template specifies the relationship between a *projection* of a complete instance of the target and an instance of the source
 - Different templates (projections) can be fused using piazza:id
- It's possible to “use the mapping in reverse”
 - We use the notion of “certain answers” as in data integration and our ICDE paper

Query Answering in Piazza

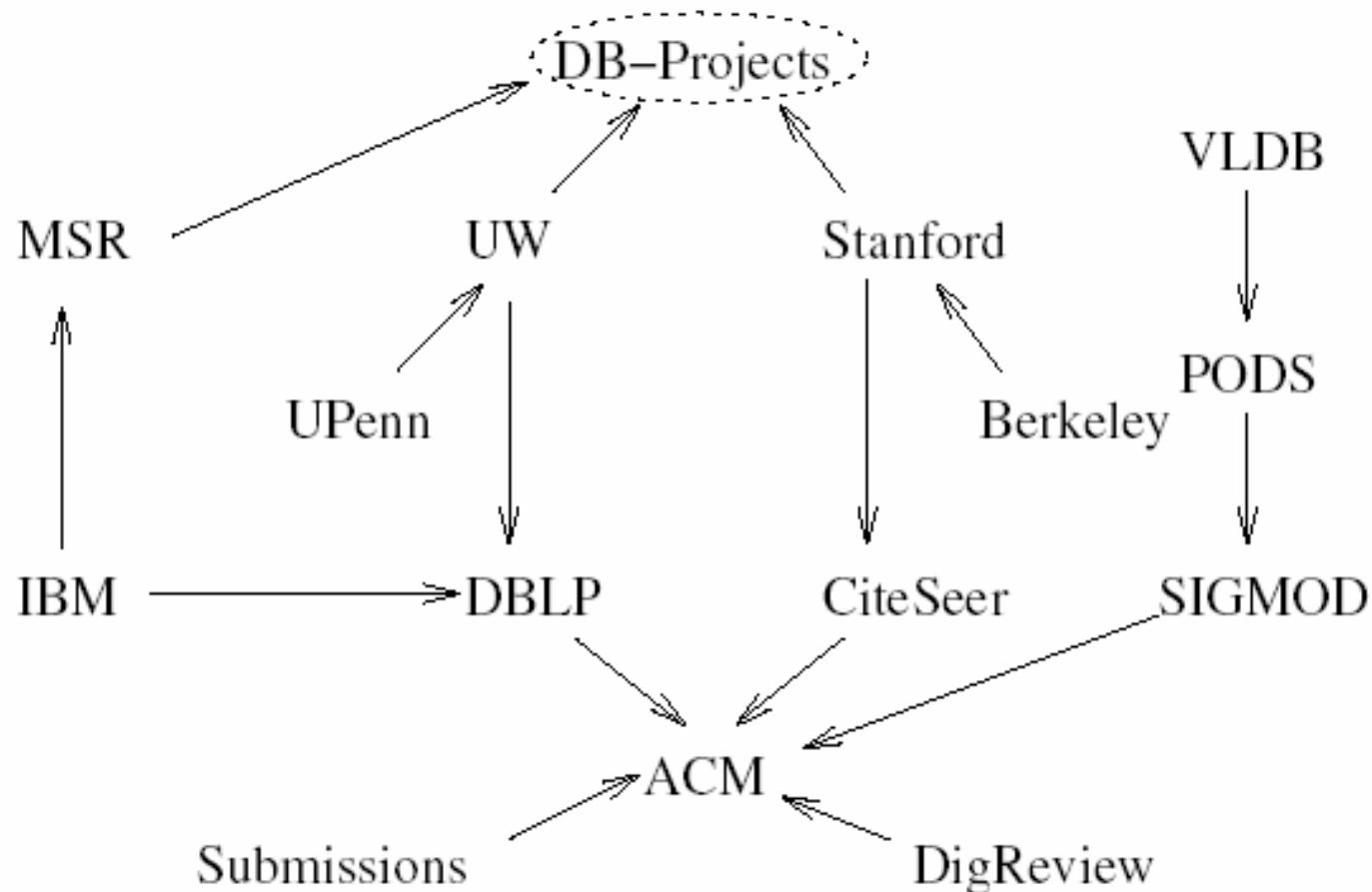
- The “inverted” direction is much more complex than simple inverse rules
 - Need to map between different levels of nesting
 - Typically it’s not possible to invert the mapping without losing some information
- As with the last Piazza paper, we use rule-goal trees to combine “forward” and “inverse” rewritings

OK, That's XML; What about RDF?

- We can map from XML to XML; thus we can go from XML to an XML serialization of RDF
- Caveat: this doesn't give us the full power of the KR-based Semantic Web!
 - We can only create RDF concepts that can be expressed in an XQuery-style view definition
 - To go any further, we may need to supplement these with additional OWL concept definitions
 - But it gets us 80% there and makes the rest much easier – and it supplies mapping capabilities missing from OWL itself

Initial Validation

- A real Semantic Web application using schemas from DB research groups, publication archives, conferences



Experience with Reformulation Cost

(Execution cost will depend on amount of data, low-level issues relating to XML processing)

Query	Description	Reformulation time	# of reformulations
Q1	XML-related projects.	0.5 sec	12
Q2	Co-authors who reviewed each other's work.	0.9 sec	25
Q3	PC members with a paper at the same conference.	0.2 sec	3
Q4	PC chairs of recent conferences + their projects.	0.5 sec	24
Q5	Conflicts-of-interest of PC members.	0.7 sec	36

Conclusion: reformulation cost (the scalability bottleneck) is acceptable, and assuming reasonable XML query performance, we can get reasonably quick answers

A Step Towards the Semantic Web

- Piazza provides several interesting features for the Semantic Web:
 - A decentralized way of mediating between schemas or ontologies
 - A more scalable, database infrastructure for the Semantic Web
 - A way of bridging between different RDF and XML formats

But Many Issues Remain...

- How do we reason about, and control, interactions between different mappings?
- How do we deal with imprecision in mappings?
- What about information loss along a mapping chain?
- Can we handle updates to the data?
- Where do good mappings come from? Can we automate their creation? (More on this next Monday.)