Numeric Reasoning in the Semantic Web

Chimène Fankam, Stéphane Jean, Guy Pierra

Presented by:
Giovanni Thenstead

Advanced Semantic Web CSCI-6965 (Spring 2009)
Rensselaer Polytechnic Institute

Feb-03-2009
Aim of Presentation

• Introduce new ontology database for reasoning and query processing.

• Show alternative reasoning engine that utilizes various labeling techniques.

• Explore a new database architecture that supports mixed ontology and labeling schemes (i.e., a meta-scheme).

• Introduce new ontology-based query language.
Background Knowledge

• Binary Relation Properties:

  ![Transitivity Diagram]

  ![Reflexivity Diagram]

  ![Symmetry Diagram]

• Eager and Lazy Reasoning:
  - Eager reasoning is used as an optimization technique for preprocessing deduced facts in order to speed-up future query requests.
  - Lazy reasoning is the opposite of Eager reasoning whereby deduction is performed on-the-fly for each query request.

• Labeling schemes:
  - Interval labeling
  - Geometric labeling
Background Knowledge

• Ontology-Based Database (OBDB) Type-II
  – Class of databases that partitions (keeps separate) the ontology data tables from the instance data tables.

• Partially Ordered Sets/Tree Order for Subdivisioning

• Table Schemes:
  – Binary Table
  – Table Per Class
What is Numeric Reasoning?

Reasoning that uses number- (or string-oriented-) labeling for qualified data instances in an Type-III OBDB.

– Problems with this approach:
  • Current Type-II OBDBs do not support numeric reasoning
    – However, new proposals are being made for Type-III OBDBs. These Type-III OBDBs may support:
      » Multiple ontology models (OWL, DAML+OIL, PBLIB, etc.).
      » New SQL-like query language.
      » New database management system (DBMS) that can automatically transform qualified (are they transitive, antisymmetric, and reflexive? Spatial or temporal?) data instances, via labeling, into an appropriate extended form.
      » Will add new meta-schemes (stored as new DBMS tables) for ontologies and labels.
Issues with the Semantic Web

• Scaling on large-size data
  - Applications need to manage an amount of ontology data that doesn’t fit in memory.

• Reasoning over large-size data
  - Often times, OBDBs that are very scalable to real world ontology and instance data are poor at reasoning, i.e., they have subpar response time when reasoning is being performed.
Why Numeric Reasoning?

• Scalability over large-size data.
  – Labeling decreases instance data representation, making it more efficient to store in memory.

• OntoDB.
  – New Type-III OBDB that extends current Type-II OBDBs with new meta-scheming for handling multiple ontologies and labels.

• Labeling.
  – Eliminates the need for reasoning over deductive facts.

• Can exploit Eager Reasoning for optimization.
  – Because labeling is stable and requires less storage overheads, eager reasoning (preprocessing of instance data) can be used as an effective strategy for improving query processing.
Databases are efficient at handling large size data and also at performing numeric and string queries.

Eager reasoning can be expensive. It’s better to represent data as numbers or strings instead of deduced facts which can get unstable (i.e. data size varies).
Deductive Capabilities of OBDBs

Popular approaches used for reasoning by current OBDBs:

– Eager Reasoning.
  • Deduced facts such as transitive closures for all transitive relationships can be derived and then materialized (stored as, for example, database views – these are essentially virtual database tables created on-the-fly).
  • Drawback:
    – requires extra storage and update overhead.

– Lazy Reasoning.
  • Deduction is done on-the-fly using virtual deduced facts.
  • Done on various database mechanisms such as views, labeling schemes, or sub-table relationships found on object-relational databases.
  • Drawback:
    – requires extra processing cost but doesn’t impose storage and update overheads.

– Subsumption Reasoning.
  • Data instances that establish a subclass-like relationship with respect to another (i.e., classOf, instanceOf relationships.) – this can be useful for carrying-out transitive closures.
OBDB Approaches: Type-I OBDB

Fig. 1. Type 1 OBDBs approach
OBDB Approaches: Type-II OBDB

**Ontology**

<table>
<thead>
<tr>
<th>Class ID</th>
<th>Name</th>
<th>SubClassOf ID</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Person</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Student</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Worker</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>Address</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property ID</th>
<th>Name</th>
<th>Domain prop</th>
<th>Domain class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>name</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>age</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>grade</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Range prop</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>xsd:string</td>
</tr>
<tr>
<td>2</td>
<td>xsd:integer</td>
</tr>
<tr>
<td>3</td>
<td>xsd:string</td>
</tr>
</tbody>
</table>

**Instances**

<table>
<thead>
<tr>
<th>Person ID</th>
<th>Student ID</th>
<th>Address ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student#1</td>
<td></td>
<td>Address#1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student#1</td>
<td>Peter</td>
</tr>
<tr>
<td>Worker#1</td>
<td>John</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grade ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student#1</td>
<td>PhD</td>
</tr>
<tr>
<td>Worker#1</td>
<td>1500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address#1</td>
<td>France</td>
</tr>
</tbody>
</table>

Fig. 2. Type 2 OBDBs approach
By the properties of **Transitivity and Inclusion:**
- **Subdivisions** can be established based on the topographic hierarchy.
  - As such, a partial order on these subdivisions can be used to exploit their positional-relationship to create some range (i.e. unique interval under which a subdivision's domain falls in respect to its scope in an hierarchy).
Numeric Reasoning over Partially Ordered Sets

Fig. 5. translation of a tree structure into two numeric values
Numeric Reasoning over Spatial and Temporal Domains

Spatial Domain

Example of Geometric labeling using circles

Temporal Domain

Time Line (in Years)
### Implementation: Labeling Scheme

#### LABELING SCHEME

<table>
<thead>
<tr>
<th>schemeld</th>
<th>numberOfColumns</th>
<th>listColumnsNames</th>
<th>listColumnsTypes</th>
<th>label</th>
<th>less or eq</th>
<th>defaultScheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>{bound1, bound2}</td>
<td>{int, int}</td>
<td>interval</td>
<td>include</td>
<td>TRUE</td>
</tr>
<tr>
<td><em>geo_rectangle</em></td>
<td>4</td>
<td>{xmin, xmax, ymin, ymax}</td>
<td>{float, float, float}</td>
<td>NULL</td>
<td>MBRContains</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

#### PROPERTY SCHEMES

<table>
<thead>
<tr>
<th>propId</th>
<th>schemeld</th>
<th>listProperties</th>
<th>activeScheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>{7,8}</td>
<td>TRUE</td>
</tr>
<tr>
<td>4</td>
<td><em>geo_rectangle</em></td>
<td>{9,10,11,12}</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
Query processing on a Table Per Class backend can be 10 times as fast compared to its Binary Represented counterpart – and provided the queries are being carried out with a class name specified.
Implementation: Query Rewriting

Queries that are not apart of OntoQL (proposed Type-III OBDB query language) are automatically converted to their OntoQL counterparts. This is important so that the correct ontologies in the OntoDB DBMS are queried, given their namespace.

Fig. 11. COG - OntoDB: example query rewriting
Implementation: Type-III OBDB

Fig. 7. COG - OntoDB : ontology part

Fig. 9. COG - OntoDB : the data part
Questions

• Does the labeling scheme support more complex geographic features (i.e., with holes, or irregular geometries) than what is provided?

  *Gregory Todd Williams*
  
  – While it may be possible to extend the current labeling scheme definitions (via the label_schemes table) to include other shapes, what you are asking does not appear to be a possibility that can be achieved given the unique dynamics properties that would have to be considered.

• How is sub-classing supported?

  *Jesse Weaver*
  
  – As mentioned earlier, this can be achieved through Subsumption Reasoning which utilizes transitivity closures on derived data instances with antisymmetric, transitive, and reflexive properties.

• How reliable is the automatic transformation process for non-OntoDB ontology and instance data?

  *Joshua Shinavier*
  
  – There is a default mechanism, so if no implementation entry (ontology scheme) exists for an unknown instance data, then whatever tricks available to the DBMS for default events may be applied.
Application to Research

• This may apply to my research interest, “Distributed Reasoning,” in the following ways:
  
  – In the case of interval labels:
    • finding possible vertical partitioning points may be used to help distribute workload across multiple reasoners.
  
  – Maybe interesting to distribute the underlying OntoDB DBMS so that instead of storing the ontology and their respective instance data on one engine, they may be spread across multiple reasoning engines.
References

Questions?