Pay-as-you-go Matching of Relational Schemata to OWL Ontologies With IncMap

Christoph Pinkel\(^1\), Carsten Binnig\(^2\), Evgeny Khramtsov\(^3\) and Peter Haase\(^4\)

\(^1\)Fluid Operations AG — \(^2\)University of Mannheim — \(^3\)University of Oxford

Motivation

![Diagram showing the relationship between ontology, relational schema, and query translation in the context of OBDA (Open Database Access).

Mapping Construction in OBDA — High Human Effort

- OBDA helps at working with complex data where traditional query formulation requires massive human effort.
- You can formulate queries against an ontology that represents the users’ view of the domain.
- Query translation, however, requires mappings.
- Typically, mappings need to be constructed/maintained manually.
- Mapping construction becomes the new weak point in terms of human effort involved.
- IncMap reduces effort with semi-automatic, pay-as-you-go approach to match ontologies and relational schemata.

IncGraph: Structurally Unified Data Structure for Matching OWL Ontologies With Relational Schemata

- Build a simple, directed labeled graph from the ontology
- Cover only structural properties important for matching:
  - Object properties linked by "ref" edges
  - Datatype properties linked by "val" edges
- Build a similar graph from the relational schema:
  - Tables and attributes as nodes
  - FK references linked by "ref" edges
  - Attributes linked by "val" edges
- Now both are structurally similar and intuitive to align
  - Increase structural similarity further by adding inverse edges for "ref" edges
  - Optionally add further edges (heuristics) to overcome differences in typical design patterns: edges from query workload structure, shortcuts...

IncMap

- Input: IncGraphs (ontology/schema). set of confirmed matches from previous iterations (as can be produced in step 2)
- Match nodes (cross product) Initialize with lexical scores
- Improve scores by considering structure
  - Currently: using Similarity Flooding (Melnik et al.)
  - Duplicates initial scores in fixpoint computation using structural commonalities between IncGraphs
- Output: ranked match candidates

Step 1: Automatic Basic Match Construction

- Input: IncGraphs (ontology/schema), set of confirmed matches from previous iterations (as can be produced in step 2)
- Match nodes (cross product) Initialize with lexical scores
- Improve scores by considering structure
  - Currently: using Similarity Flooding (Melnik et al.)
  - Duplicates initial scores in fixpoint computation using structural commonalities between IncGraphs
- Output: ranked match candidates

Step 2: Manual Confirmation of Relevant Matches

- Input: ranked match candidates required for current query
- Ask user to confirm/reject suggestions
- Output: confirmed matches (used in subsequent iterations):
  - A) "Initialize": replace initial (lexical) scores
  - B) "Self-confidence": force new scores permanently
  - C) "Influence Nodes": construct additional nodes to influence fixpoint computation

Experiments

- Results for Basic Matches (non-incremental)
- IMDB: Incremental Runs
- Music Ontology: Incremental Runs