CS561 WEB DATA MANAGEMENT

PRESENTATION:

I ON BLANK NODES
II EFFICIENT QUERY ANSWERING AGAINST DYNAMIC RDF DATABASES

Lantzaki Christina - May 2013

On Blank Nodes

Alejandro Mallea, Marcelo Arenas, Aidan Hogan and Axel Polleres

ISWC 2011

In one slide

- Study blank nodes
  - theoretical perspective
  - inside the Semantic Web standards
- Lack of consistency regarding the treatment of blank nodes inside the W3C stack
  - The standard query language SPARQL can return different results for two graphs considered equivalent by the RDF semantics
- Empirical analysis of blank nodes in published RDF data
- Discuss proposals for handling blank nodes

- Theoretical Perspective
- The blank nodes in the standards
- Published blank nodes
- Solutions
- Conclusion

Theoretical perspective

- Anonymous existential variables
  - Do not have names

```
:Chris :hasAddress [:street “Knossou”], [:number “42”].
```

  - Labels are used in some serializations

```
:Chris :hasAddress _:b1
 _:b1 :street “Knossou”
 _:b1 :number “42”
```

- Anonymous existential variables
  - Indicate the existence of a thing

```
:Chris :hasAddress _:b1
:Chris :hasAddress _:b2
```

“Chris has an address”

Theoretical perspective (cont)

- Anonymous existential variables
  - A blank node is valid in a limited context
  - Labels of blank nodes only have local identity

G1

```
:Chis :hasAddress _:b1

:Chis :hasAddress _:x
```

G2

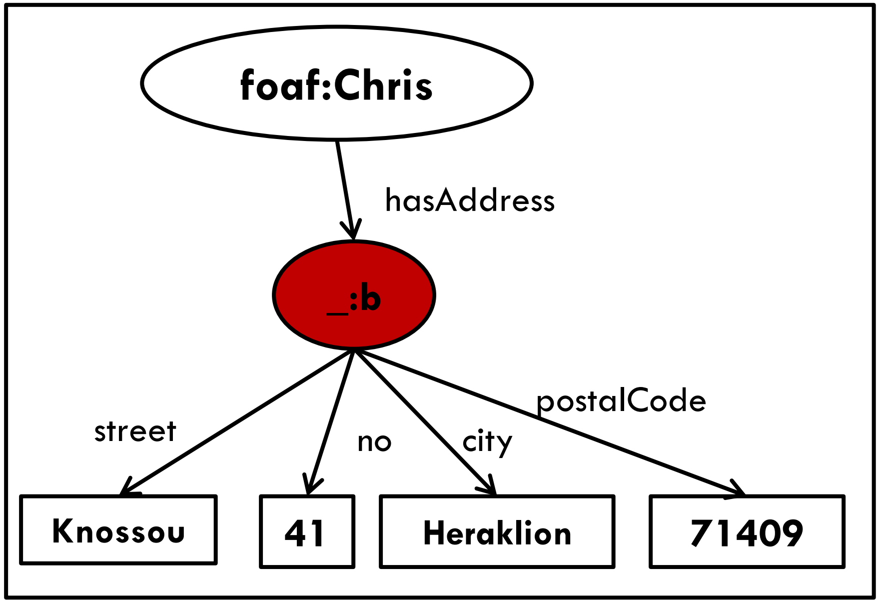
```
:Maria :hasFriend _:b1

:Maria :hasFriend _:y
```

- Theoretical Perspective
- The blank nodes in the standards
- Published blank nodes
- Solutions
- Conclusion

Why are blank nodes useful?

- Represent resources whose identity is unknown, but their attributes are known



- Express the multi-relationship model

- Describe RDF containers

- Hide sensitive information

Labeling

- All RFD syntaxes allow blank nodes to be explicitly labeled
  - Allows blank nodes to be referenced outside of nested elements
  - Creation of cyclic structures becomes possible
  - Labeling may vary across time and across parsers

Blank nodes in RDFS and OWL

- Blank nodes are used as existential variables
  - as surrogates for literals in RDFS

```
:Federer apt:name “Roger Federer”
apt:name rdfs:range apt:PlayerName
```

```
“Roger Federer” rdf:type apt:PlayerName
```

NOT VALID

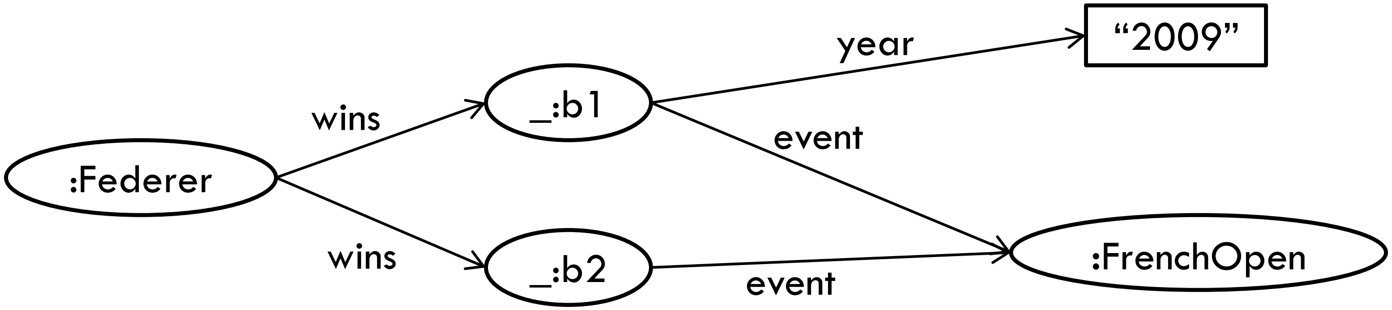
```
_:bFederer rdf:type apt:PlayerName
```

Isomorphism check and RDFS entailment become NP-Complete

Blank nodes in SPARQL

- Standard query language for RDF

- Considers blank nodes as constants scoped to the graph they appear in



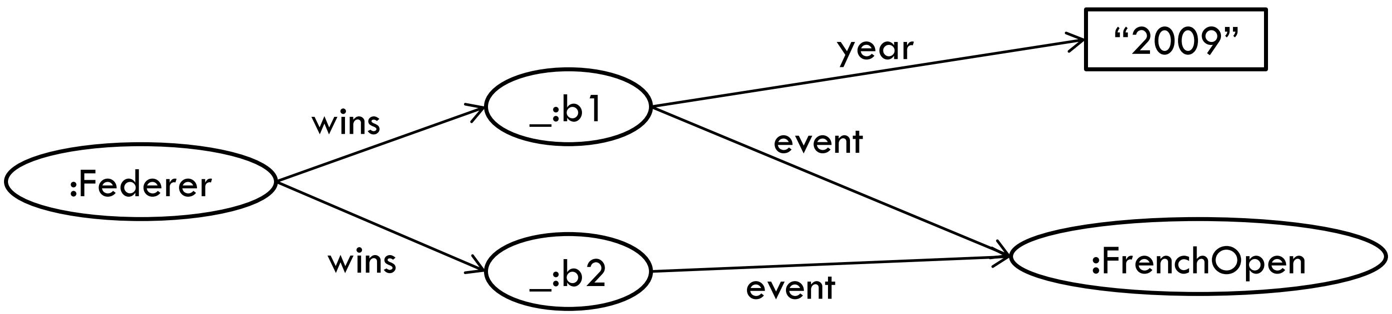

```
SELECT ?X WHERE {
       :Federer :wins ?X .
       ?X :event :FrenchOpen .}
```

?X
_:b1
_:b2

Blank nodes in SPARQL

- Standard query language for RDF

- Considers blank nodes as constants scoped to the graph they appear in



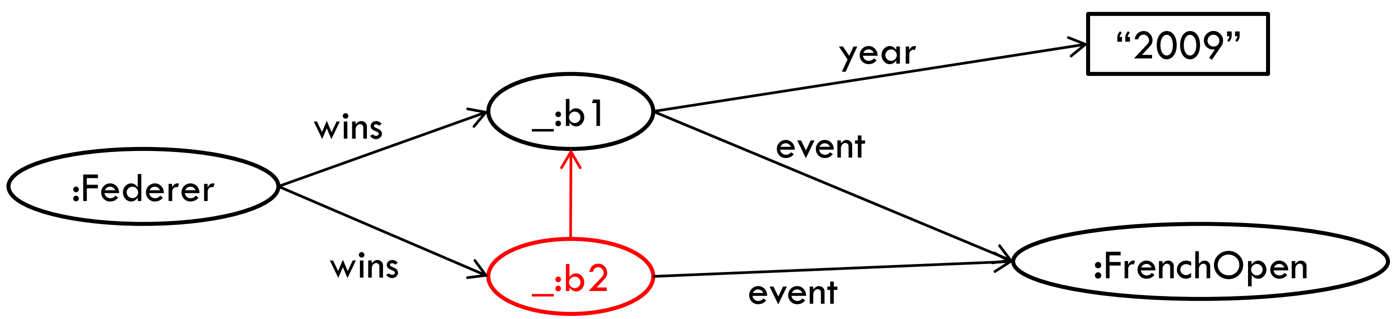

```
SELECT DISTINCT ?X WHERE {
                 :Federer :wins ?X .
                 ?X :event :FrenchOpen .}
```

?X
_:b1
_:b2

Blank nodes in SPARQL

- Standard query language for RDF

- Considers blank nodes as constants scoped to the graph they appear in

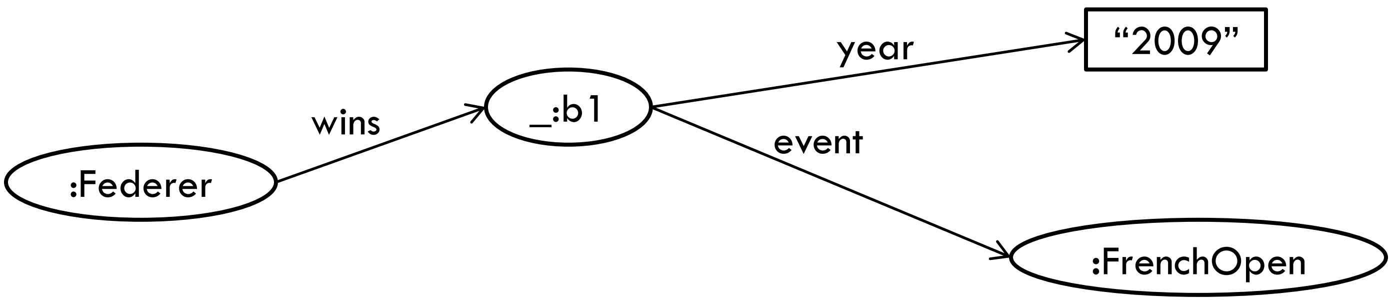



```
SELECT DISTINCT ?X WHERE {
                 :Federer :wins ?X .
                 ?X :event :FrenchOpen .}
```

_:b2 is a subset of _:b1 under the RDF semantics

Blank nodes in SPARQL

- Standard query language for RDF
- Considers blank nodes as constants scoped to the graph they appear in




```
SELECT DISTINCT ?X WHERE {
                 :Federer :wins ?X .
                 ?X :event :FrenchOpen .}
```

?X
_:b1

Merging in RDF

G1

```
:Chris :hasAddress _:b1
```

G2

```
:Chris :hasAddress _:b1
```

Merge(G1, G2)

```
:Chris :hasAddress _:x
:Chris :hasAddress _:y
```

- Theoretical Perspective
- The blank nodes in the standards
- Published blank nodes
- Solutions
- Conclusion

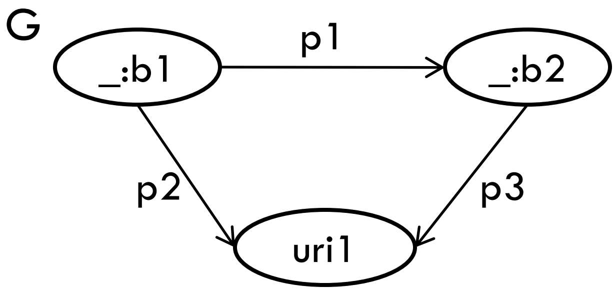
Published blank nodes

Over a corpus of 965 MB triples, 4 MB RDF/XML documents, 783 domains

- average 7.5 % blank nodes per domain
- 44% of the domains did not publish any blank nodes
  - hi5.com foaf domain 85% bnodes (148MB)
  - rdfabout.com 42% bnodes (460 KB)
  - freebase.com 15% bnodes (1 MB)
  - bbc.co.uk 7% bnodes (100 KB)

Blank node Structures

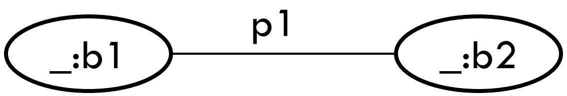
Blank(G): the largest subgraph of G consisting of blank nodes, seen as an undirected graph



Blank node Structures (cont)

Blank(G): the largest subgraph of G consisting of blank nodes, seen as an undirected graph

Blank(G)



•The treewidth is a measure of cyclicity of the graph

•The higher the treewidth of a blank(G), the harder the entailment becomes [Pihcler et al. 2008]

For bounded treewidth (tree bnode strctures) problems like Entailment, Merging become easy to solve, but in the general case it is NP-Complete.

Blank node Structures (cont)

Over their corpus

treewidth	# graphs
1	518,831
2	8,134
3	208
4	99
5	23
6	-
7	1

Blank node Structures (cont)

Over their corpus

treewidth	# graphs
1	518,831
2	8,134
3	208
4	99
5	23
6	-
7	1

98.4 % of the graphs are acyclic

One graph with 451 blank nodes and 887 edges

Survey of publishing

- Made a simple poll asking publishers about their intention when they publish blank nodes
  - A portion of publishers avoid publishing blank nodes
  - Some use blank nodes as constants
  - Some use them according to RDF semantics

Divisive issue

- Theoretical Perspective
- The blank nodes in the standards
- Published blank nodes
- Solutions
- Conclusion

Solutions

- Not allowing blank nodes to be explicitly labeled
  - Bnode structures only form trees

- Skolemization
  - A way of removing existential variables from a formula in normal form
  - Existentially quantified variables are replaced by “fresh” constants that are not used in the original formula
  - process for skolemizing bnodes into URIs -- i.e., converting bnodes into URIs -- that RDF users could use to eliminate bnodes from RDF graphs
  - The merge problem is solved
  - Lose locallity

Skolemization Schemes

- Set of guidelines for Skolemization
- Does not require changes in the semantics of RDF
- Defines a standard syntax for Skolem constants

Skolemization Schemes (cont)

- Centralized
  - central service that gives out fresh URIs upon request ensuring uniqueness in a global scale
  - Costs of bandwidth and maintenance may be high
  - Against the spirit of the SW community

- Decentralized
  - Naming conflicts are typically avoided by pay level domains
  - The RDF working group discussion to add the syntax

`http://example.com/.well-known/bnode/ffewrwer0Q`

- Theoretical Perspective
- The blank nodes in the standards
- Published blank nodes
- Solutions
- Conclusion

Conclusion

- Confusion between the semantics of blank nodes across standards
  - Compatibility with the RDF standards is violated
- The needs of publishers are not always aligned with the semantics of RDF
- Entailment, Isomorphism check complexity problems
- Skolemizations is useful
- Need for a standard skolemization schema

Efficient Query Answering against Dynamic RDF Databases

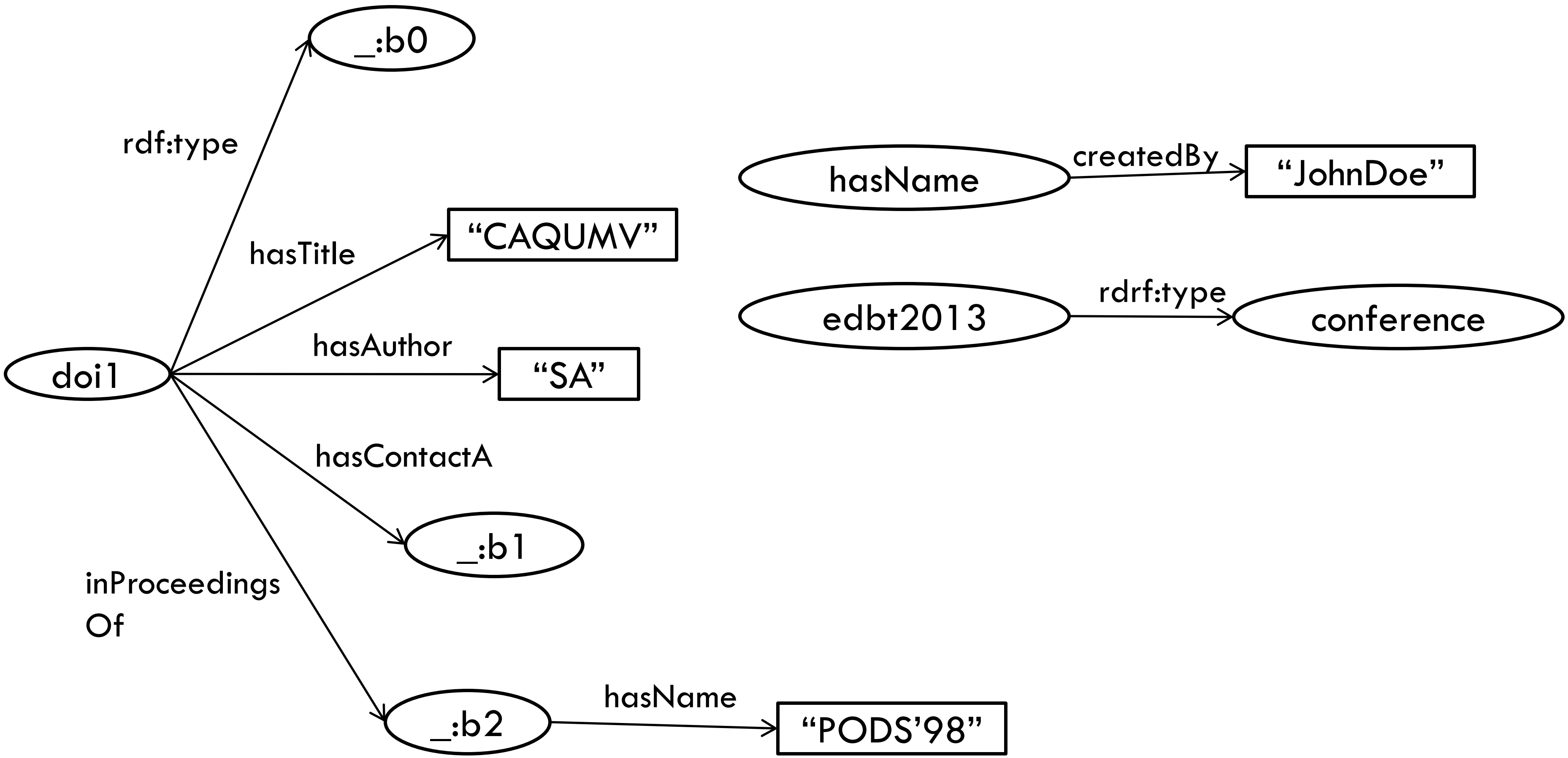
François Goasdoué, Ioana Manolescu, and Alexandra Roati ¸s

EDBT 2013

In one slide

- Efficiently querying RDF data by translating SPARQL queries into efficient RDBMS-style operations
- Extend the **database fragment of RDF** by **the support of blank nodes**
- Propose **BGP answering techniques** for the database fragment working on top of every standard conjunctive query processor
  - A novel incremental RDF saturation maintenance algorithm
  - A novel reformulation-based query answering algorithm
- Implement the query answering techniques and evaluate them

Running Example



- Background
- RDF meets RDBMS
- DB fragment for RDF
- Query Answering techniques
  - Saturation-based query answering
  - Reformulation-based query answering
- Evaluation
- Conclusion

OWA under the RDFS Entailment

- OWA: Open World Assumption
  - **Implicit triples** are considered to be part of the RDF graph, even though they are not explicitly present in it

```
:hasFriend rdfs:domain :Person
:Anne :hasFriend :Marie
:Anne rdf:type :Person
```

  - The implicit triples are given through the **entailment rules**
    - rdfs:subclass, rdfs:subproperty, rdfs:domain, rdfs:range

Immediate entailment


```
:doi1 :hasContactA _:b1
:hasContactA rdfs:subproperty :hasAuthor
:hasAuthor rdfs:domain :paper
```

- 1st application (According to rule rdfs3)

```
:doi1 :hasAuthor _:b1
```

- 2nd application (According to rule rdfs2)

```
:doi1 rdf:type :paper
```

Graph Saturation

- The new graph that is built after adding to the graph G all its implicit triples is called **finite saturation of G**, denoted by G∞

$$\bullet\ G^0 = G$$

$$\bullet\ G^\alpha = G^{\alpha-1} \cup \{s\ p\ o \mid G^{\alpha-1} \vdash^i_{\mathrm{RDF}} s\ p\ o\}$$

- Also known as **closure of graph**, denoted by C(G)

A triple t is entailed through G, if and only if t is part of G∞

Basic Graph Pattern Queries

- BGP Queries
  - Boolean `ASK WHERE {t1, t2, … , ta}`
  - Non-boolean `SELECT x WHERE {t1, t2, …, ta}`
  - **`q(x) = t1, t2, …, ta`**
- Query Evaluation
  - Results only contains explicit triples
  - Treat blank nodes as non-distinquished variables
- Query Answering
  - Results are obtained by the evaluation of the query against the finite saturate graph

The evaluation of a query may lead to an incomplete answer set

Basic Graph Pattern Queries (cont)

- `q(x)  = y1 hasAuthor x`
  `y1 inProccedingsOf y2`
  `y2 y3 "PODS '98"`

Query Evaluation

μ = {y1 → doi1, x → "SA" , y2 → _:b2, y3 → hasName}

`q(G') = {"SA"}`

Query Answering

μ = {y1 → doi1, x → _:b1 , y2 → _:b2, y3 → hasName}

`q(G'∞) = {"SA", _:b1}`

- Background
- RDF meets RDBMS
- DB fragment for RDF
- Query Answering techniques
  - Saturation-based query answering
  - Reformulation-based query answering
- Evaluation
- Conclusion

RDF meets RDBMS

- RDF graphs are seen as a special case of incomplete relational databases based on V-tables
- V-tables allow using variables in their tuples

RDF	RDBMS
Graph	V-table
triple	tuple
Blank nodes	variables

- V-table querying computes the exact answer set of any conjunctive query

RDF meets RDBMS (cont)

The SPARQL evaluation q(G) is obtained by the relational evaluation of the conjunctive query after blank nodes are replaced by fresh variables

BGP query answering boils down to conjunctive query answering on a saturated database

- Background
- RDF meets RDBMS
- DB fragment for RDF
- Query Answering techniques
  - Saturation-based query answering
  - Reformulation-based query answering
- Evaluation
- Conclusion

The database fragment of RDF

- A restriction of RDF
  - a fragment that can be efficiently implemented on top of any conjunctive query engine
  - Restricting RDF Entailment to RDFS schema rules
  - any triple allowed in the RDF is also allowed in the DB fragment

- A graph that belongs to the DB fragment is called database
- Database db = <S, D>
- S : schema-level of db - triples can only be RDFS statements
- D: instance-level of db

- Background
- RDF meets RDBMS
- DB fragment for RDF
- Query Answering techniques
  - Saturation-based query answering
  - Reformulation-based query answering
- Evaluation
- Conclusion

Query Answering techniques

- Saturation-based: the saturation of the database is computed using the allowed entailment rules

- Reformulation-based : reformulates the query such that the evaluation of the new query yields exactly the answer-set of the original query against the database

Saturation-based query answering

- Saturate algorithm
  - Input: db
  - Computes the saturation of the db using the allowed entailment rules
  - Output: Saturate∞(db)
  - Upper bound for time O(#db^3)
- Requires time to be computed
- Space to be stored
- For each update of the graph the saturation must be somehow recomputed

Saturation maintenance

- Do not re-compute the saturation, just modify it to reflect the update
- Update = triples deletion/addition

- Keep track of the multiple ways in which a triple was entailed

- Naïve Algorithm
  - Record the inference paths of each implicit triple
  - For each update decide whether this adds or removes a reason why a triple belongs to the saturation
  - When the count of reasons is terminated, then the triple must be removed
  - Cannot scale

Saturation maintenance (cont)

- Their approach
  - Keep track of the number of reasons why a triple has been inferred
  - A triple appears in the saturation as many times as it can be entailed
  - $\text{Saturate}^{\infty}_{+}(\text{db})$
- Optimization: every triple is tagged with:
  - A boolean to indicate whether it is explicit or entailed
  - An integer indicating how many times it appears

Reformulation-based query answering

- Reformulate algorithm
  - exhaustively applies the set of reformulation rules
  - Each rule is a transformation of the form Input/Output
  - Input: <logical condition on db, logical condition on q>
  - Output: query q'
  - Each rule produces a new query when the rule's input conditions are satisfied
  - Applying all the rules gives the result of the reformulation

Reformulation-based query answering (cont)

- Handling the results of reformulating a query directly to a conjunctive query processor for evaluation, may introduce erroneous answers

WHY?

- The semantics of blank nodes in the BGP queries treat them as variables
- The reformulate algorithm treat the blank nodes as constants

Reformulation-based query answering (cont)

- Query q(x,y) = x rdf:type y

Reformulate(q,db) contains

q(x,confP) = x rdf:type _:b0

During the evaluation of the query

μ = {x → edbt2013 _:b0 → conference}

Answer tuple : < edbt2013 , confP > erroneous

Non-standard Evaluation

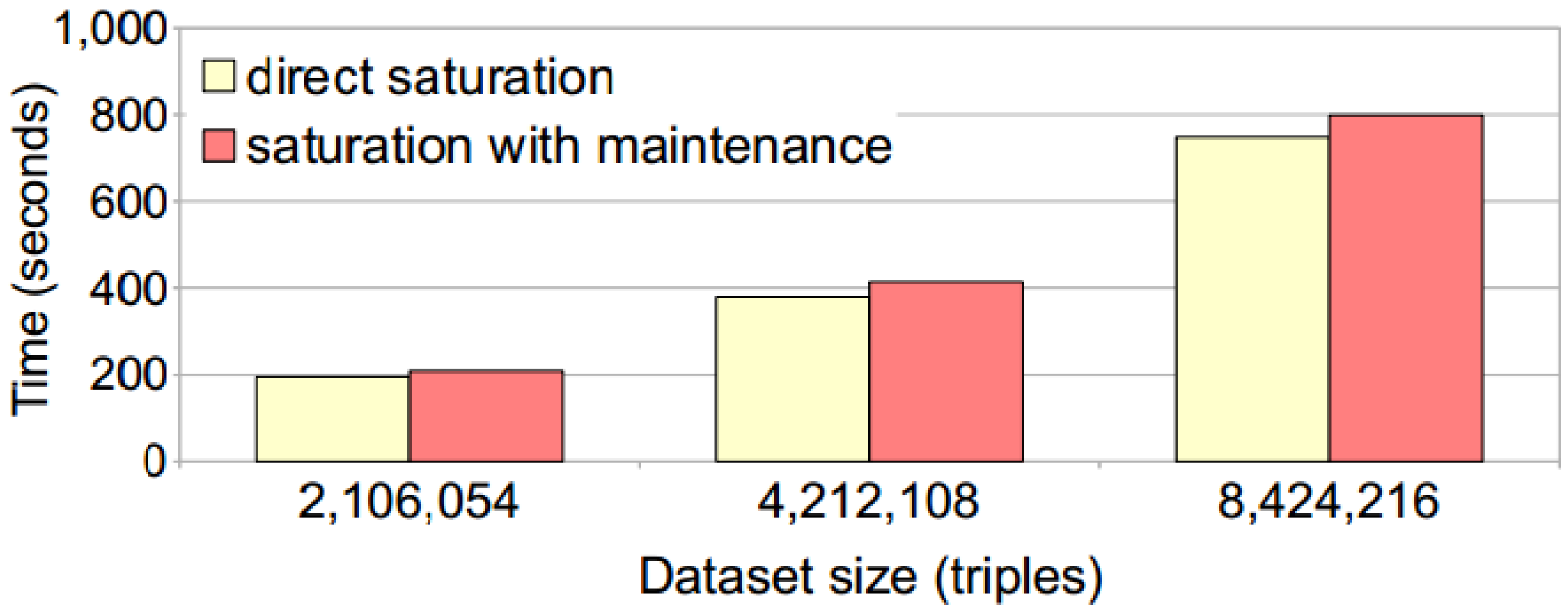
- Translate q into SQL taking care to:
  - **Enclose blank nodes within quotes, so that the RDBMS treats each as a constant, to be matched only by the exact same value in the database**

- Background
- RDF meets RDBMS
- DB fragment for RDF
- Query Answering techniques
  - Saturation-based query answering
  - Reformulation-based query answering
- Evaluation
- Conclusion

Settings

- Algorithms
  - Saturate Algorithm (Saturate, Saturate+)
  - Reformulate Algorithm
- Data Structures
  - Sat(s, p, o)
  - SatM(s, p, o, isExplicit, count)
- RDF Datasets
  - Barton
  - DBPedia
  - DBLP

Scalability of Saturate and Saturate+

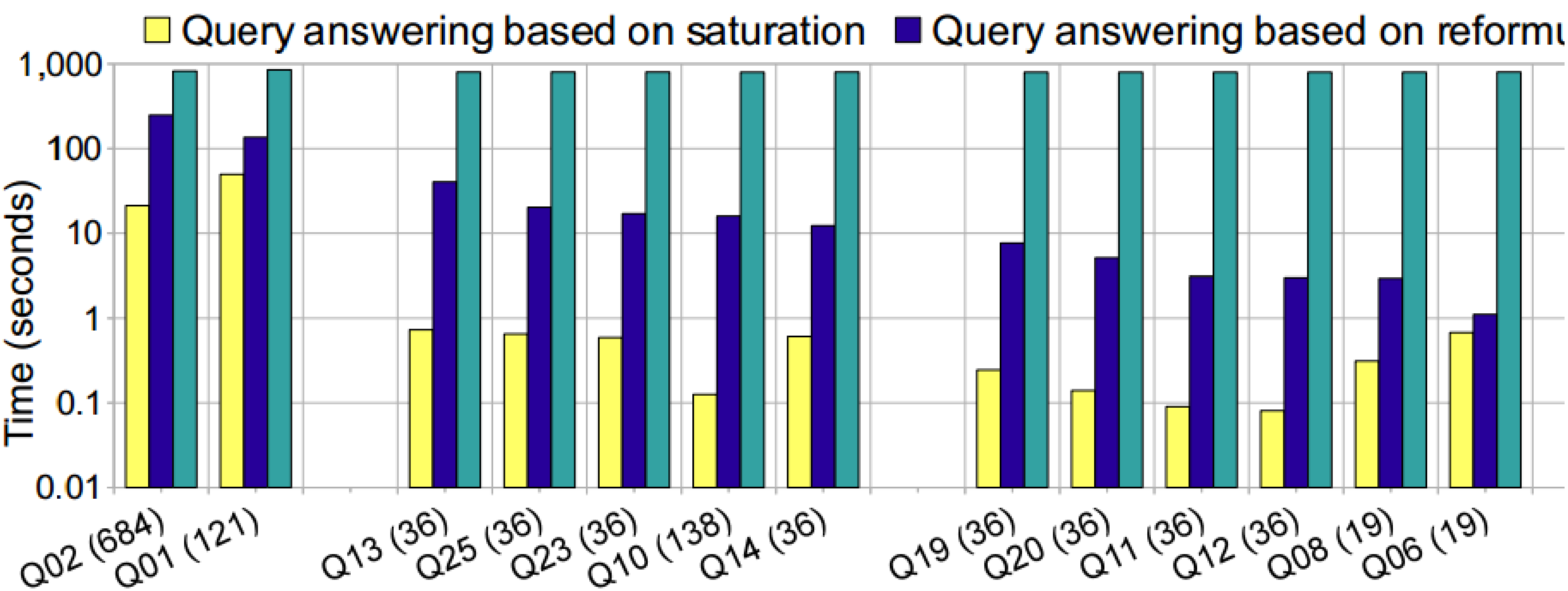


Time grows almost linearly, far less than O(#db^3)

Other maintenance systems scale poorly, perform more costly maintenance operations

Query Answering Times

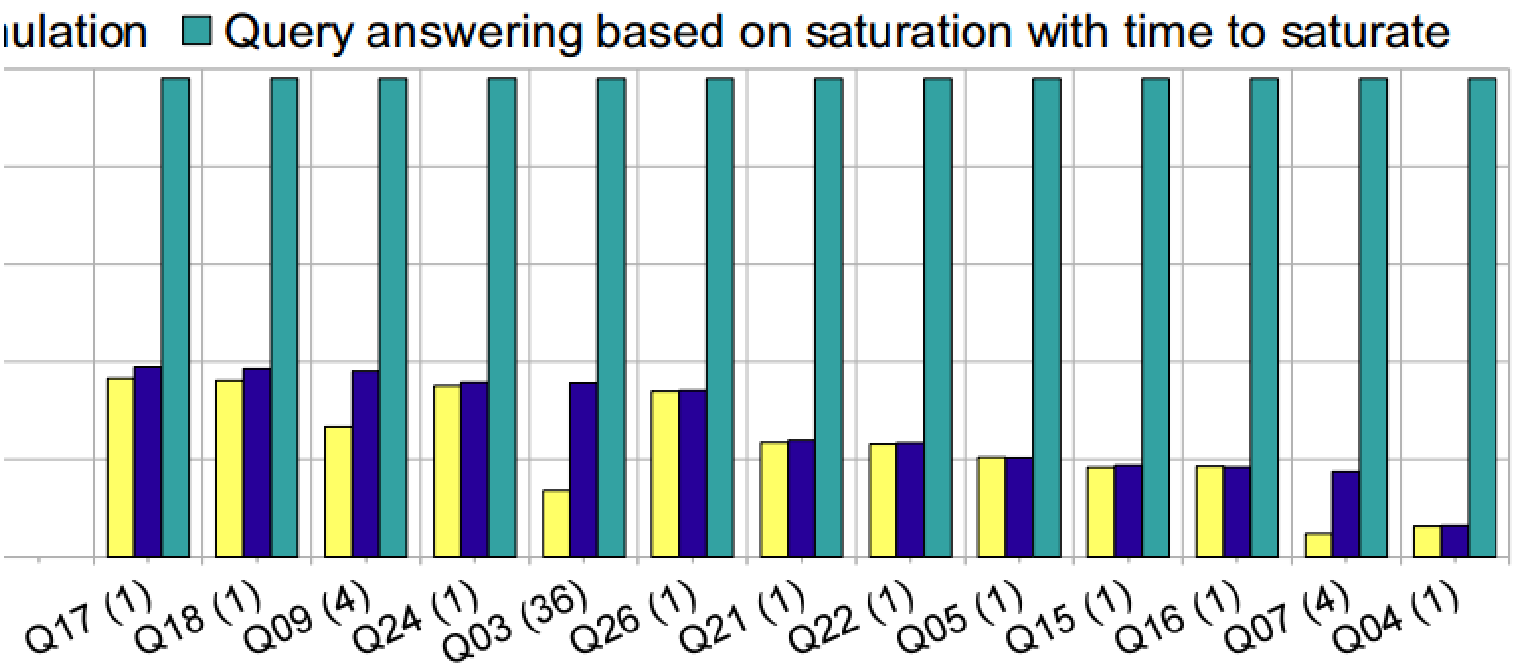
Set of 26 queries from the DBLP graph



Saturation-based time is much smaller than reformulation-based time

Query Answering Times (cont)

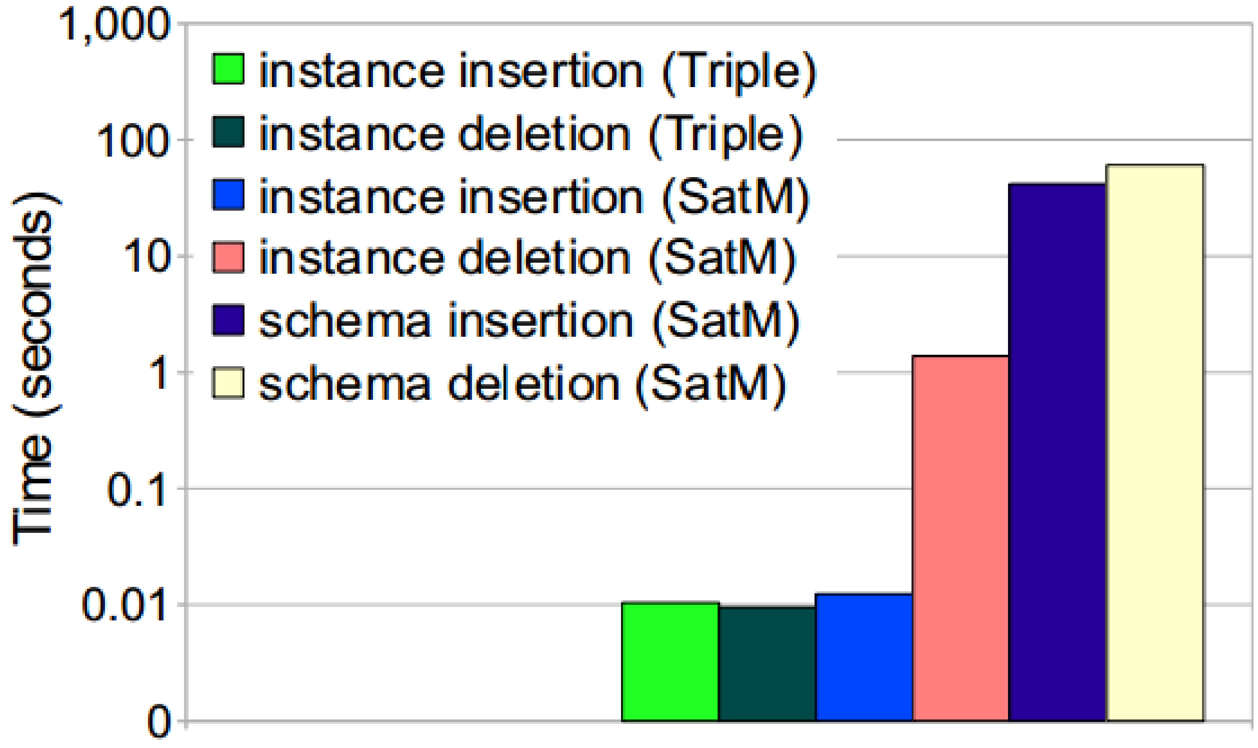
Set of 26 queries from the DBLP graph



If we factor saturation+ saturation-based becomes more expensive. But saturation cost is only paid once!

In the presence of updates

- Updates have no impact on reformulation
- Saturation needs to maintain Sat_M structure



Experimental Results

- Saturation is best for large reformulation queries
- Reformulation is efficient for small-to-moderate reformulation queries

- Saturation can be maintained at a reasonable cost for instance level updates
  - For schema level updates is much more expensive
- Updates have a small impact on reformulation

- Background
- RDF meets RDBMS
- DB fragment for RDF
- Query Answering techniques
  - Saturation-based query answering
  - Reformulation-based query answering
- Evaluation
- Conclusion

Conclusion

- Extend the state of the art in BGP query answering techniques
- Saturation-based query answering techniques
- Reformulation-based query answering techniques

Future plans

- An automated strategy to choose between the two techniques