Formalization and Experiences of R2RML-based SPARQL to SQL query translation using Morph

Freddy Priyatna, Oscar Corcho

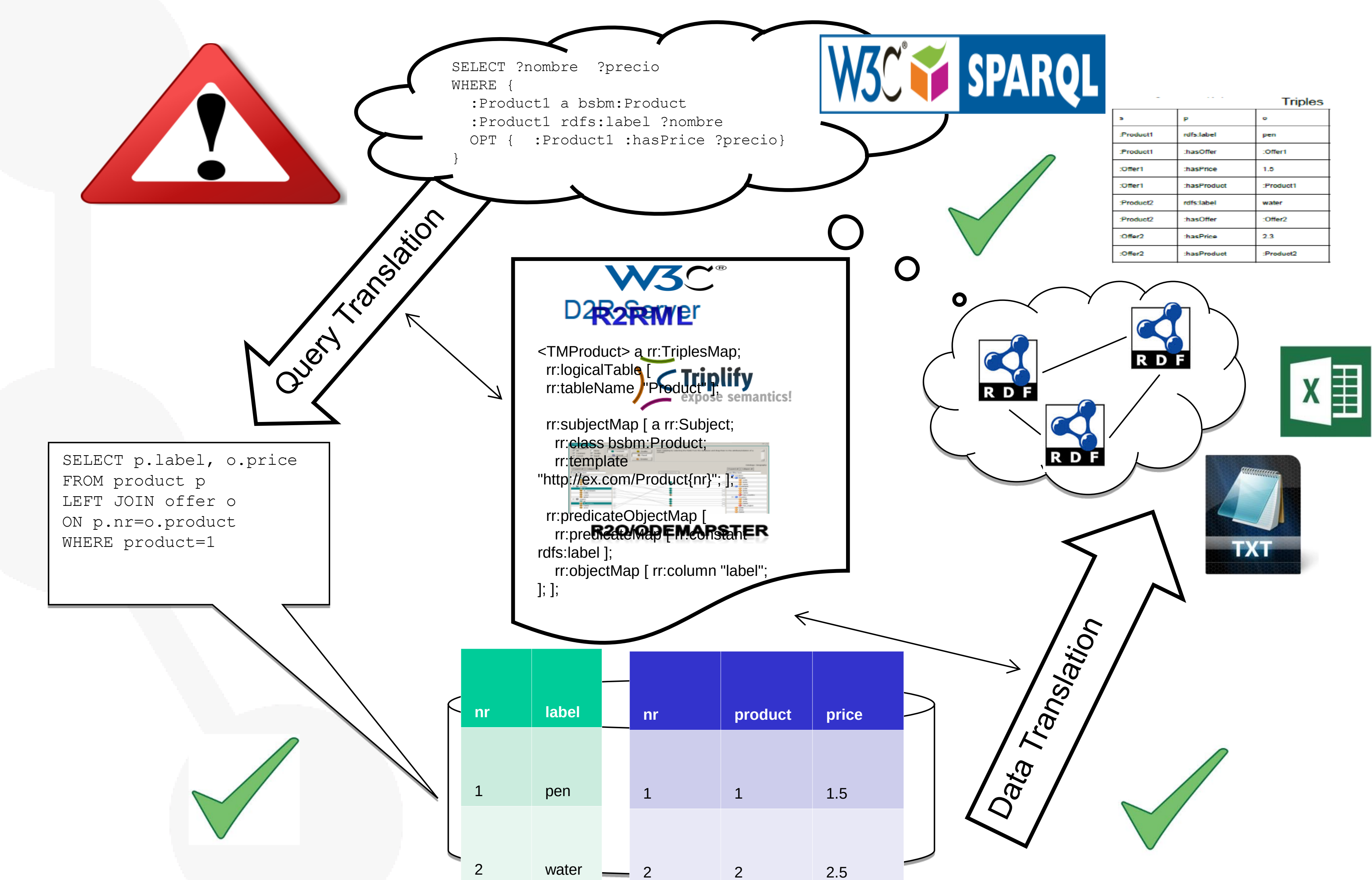
Ontology Engineering Group
Universidad Politécnica de Madrid
Madrid, Spain
[fpriyatna,ocorcho]@fi.upm.es

Juan Sequeda

Dept. of Computer Science
University of Texas at Austin
Austin, USA
jsequeda@cs.utexas.edu

OEG, March 27th, 2014

Motivation



SELECT ?nombre ?precio
WHERE {
:Product1 a bsbm:Product
:Product1 rdfs:label ?nombre
OPT { :Product1 :hasPrice ?precio}
}
W3C SPARQL
Triples
s | p | o
:Product1 | rdfs:label | pen
:Product1 | :hasOffer | :Offer1
:Offer1 | :hasPrice | 1.5
:Offer1 | :hasProduct | :Product1
:Product2 | rdfs:label | water
:Product2 | :hasOffer | :Offer2
:Offer2 | :hasPrice | 2.3
:Offer2 | :hasProduct | :Product2
Query Translation
W3C
D2R Server
R2RML
<TMProduct> a rr:TriplesMap;
rr:logicalTable [
rr:tableName "Product"];
Triplify
expose semantics!
rr:subjectMap [a rr:Subject;
rr:class bsbm:Product;
rr:template
"http://ex.com/Product{nr}";];
rr:predicateObjectMap [
rr:predicateMap [rr:constant
rdfs:label];
rr:objectMap [rr:column "label";
];];
R2O/ODEMAPSTER
RDF
RDF
RDF
X
TXT
SELECT p.label, o.price
FROM product p
LEFT JOIN offer o
ON p.nr=o.product
WHERE product=1
nr | label
1 | pen
2 | water
nr | product | price
1 | 1 | 1.5
2 | 2 | 2.5
Data Translation

SPARQL to SQL: whose problem?

Integrative Cancer Research INTEGRATE Through Innovative Biomedical Infrastructures

```
SELECT DISTINCT `T21_v_livingsubject`.`id`,
`T18_participation`.`roleId`, `T8_v_observation`.`methodCodeTitle`,
`T14_v_observation`.`referenceRangeMax`,
`T7_v_observation`.`methodCode`, `T10_v_observation`.`valueST`,
`T18_participation`.`actId`, `T20_role`.`id`,
`T12_v_observation`.`units`, `T1_v_observation`.`id`,
`T3_v_observation`.`effectiveTime`, `T18_participation`.`entityId`,
`T5_v_observation`.`targetSiteCodeTitle`,
`T17_v_observation`.`codeVocId`,
`T13_v_observation`.`referenceRangeMin`,
`T6_v_observation`.`targetSiteCodeVocId`,
`T19_participation`.`entityId`,
`T16_v_observation`.`actionNegationInd`,
`T15_v_observation`.`title`, `T9_v_observation`.`methodCodeVocId`,
`T20_role`.`entityId`, `T2_v_observation`.`id`, `T23_entity`.`id`,
`T18_participation`.`typeCode`, `T22_v_livingsubject`.`birthTime`,
`T1_v_observation`.`code`, `T24_entity`.`code`,
`T11_v_observation`.`valuePQ`, `T4_v_observation`.`targetSiteCode`
FROM `v_observation` AS `T15_v_observation`, `v_observation` AS
`T10_v_observation`, `v_observation` AS `T5_v_observation`,
`v_observation` AS `T13_v_observation`, `entity` AS `T24_entity`,
`role` AS `T20_role`, `participation` AS `T20_participation`,
`v_observation` AS `T7_v_observation`, `v_observation` AS
`T9_v_observation`, `v_observation` AS `T4_v_observation`,
`participation` AS `T19_participation`, `v_livingsubject` AS
`T22_v_livingsubject`, `v_observation` AS `T11_v_observation`,
`v_observation` AS `T17_v_observation`, `v_observation` AS
`T3_v_observation`, `v_observation` AS `T16_v_observation`,
`v_livingsubject` AS `T21_v_livingsubject`, `participation` AS
`T18_participation`, `v_observation` AS `T2_v_observation`,
`v_observation` AS `T14_v_observation`, `v_observation` AS
`T6_v_observation`, `v_observation` AS `T12_v_observation`,
`v_observation` AS `T18_v_observation`, `role` AS `T23_role`,
`entity` AS `T23_entity`, `v_observation` AS `T1_v_observation`,
`v_observation` AS `T8_v_observation`
WHERE (`T10_v_observation`.`id` = `T6_v_observation`.`id` AND
`T10_v_observation`.`valueST` IS NOT NULL AND
`T11_v_observation`.`id` = `T6_v_observation`.`id` AND
`T11_v_observation`.`valuePQ` IS NOT NULL AND
`T12_v_observation`.`id` = `T6_v_observation`.`id` AND
`T12_v_observation`.`units` IS NOT NULL AND `T13_v_observation`.`id`
= `T6_v_observation`.`id` AND
`T13_v_observation`.`referenceRangeMin` IS NOT NULL AND
```

BIZKAISENSE

```
SELECT DISTINCT `T2_Observations`.`date`, `T1_Stations`.`code`,
`T5_Properties`.`name`, `T5_Observations`.`date`,
`T3_Stations`.`code`, `T6_Stations`.`code`, `T1_Properties`.`name`,
`T1_Observations`.`date`, `T6_Observations`.`date`,
`T5_Stations`.`code`, `T7_Observations`.`value`,
`T6_Properties`.`name`
FROM `Properties` AS `T3_Properties`, `Observations` AS
`T3_Observations`, `Stations` AS `T7_Stations`, `Properties` AS
`T6_Properties`, `Observations` AS `T2_Observations`, `Properties`
AS `T1_Properties`, `Observations` AS `T6_Observations`,
`Properties` AS `T5_Properties`, `Stations` AS `T3_Stations`,
`Properties` AS `T7_Properties`, `Stations` AS `T1_Stations`,
`Stations` AS `T2_Stations`, `Stations` AS `T6_Stations`,
`Properties` AS `T2_Properties`, `Observations` AS
`T5_Observations`, `Stations` AS `T4_Stations`, `Observations` AS
`T1_Observations`, `Stations` AS `T5_Stations`, `Observations` AS
`T7_Observations`, `Observations` AS `T4_Observations`,
`Properties` AS `T4_Properties`
WHERE (`T1_Observations`.`date` = `T3_Observations`.`date` AND
`T1_Observations`.`date` IS NOT NULL AND `T1_Observations`.`prop_id`
= `T1_Properties`.`id` AND `T1_Observations`.`station_id` =
`T1_Stations`.`id` AND `T1_Properties`.`name` =
`T3_Properties`.`name` AND `T1_Properties`.`name` IS NOT NULL AND
`T1_Stations`.`code` = `T3_Stations`.`code` AND
`T1_Stations`.`code` IS NOT NULL AND `T2_Observations`.`date` <
'2011-01-02T00:00:00' AND `T2_Observations`.`date` =
`T3_Observations`.`date` AND `T2_Observations`.`date` >=
'2011-01-01T00:00:00' AND `T2_Observations`.`date` IS NOT NULL AND
```

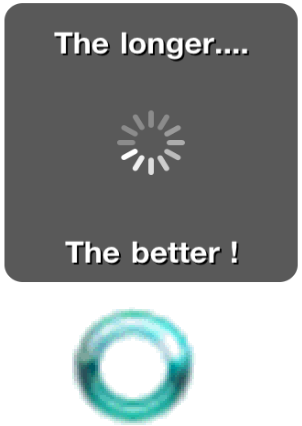
RÉPENER

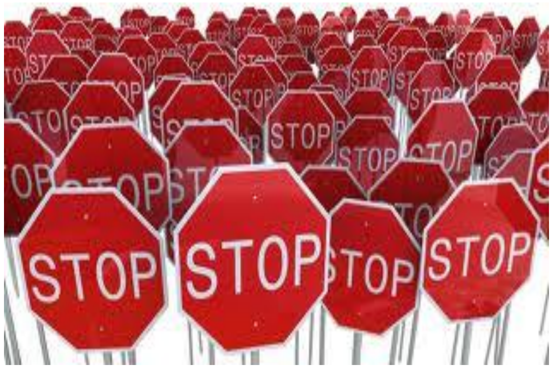
```
SELECT DISTINCT `T2_icaen`.`Expr1`, `T5_icaen`.`Expr1`,
`T6_icaen`.`Expr1`, `T8_icaen`.`CARACT_GEN_SUP`,
`T4_icaen`.`building_life_cycle_phase`,
`T3_icaen`.`building_life_cycle_phase`, `T1_icaen`.`Expr1`
FROM `icaen` AS `T2_icaen`, `icaen` AS `T8_icaen`, `icaen` AS
`T7_icaen`, `icaen` AS `T1_icaen`, `icaen` AS `T6_icaen`, `icaen` AS
`T5_icaen`, `icaen` AS `T4_icaen`, `icaen` AS `T3_icaen`
WHERE (`T1_icaen`.`Expr1` = `T2_icaen`.`Expr1` AND `T1_icaen`.`Expr1`
IS NOT NULL AND `T2_icaen`.`Expr1` = `T3_icaen`.`Expr1` AND
`T2_icaen`.`Expr1` = `T5_icaen`.`Expr1` AND `T2_icaen`.`Expr1` IS NOT
NULL AND `T3_icaen`.`Expr1` IS NOT NULL AND
`T3_icaen`.`building_life_cycle_phase` =
`T4_icaen`.`building_life_cycle_phase` AND
`T3_icaen`.`building_life_cycle_phase` IS NOT NULL AND
`T4_icaen`.`building_life_cycle_phase` IS NOT NULL AND
`T5_icaen`.`Expr1` = `T6_icaen`.`Expr1` AND `T5_icaen`.`Expr1` IS NOT
NULL AND `T6_icaen`.`Expr1` = `T7_icaen`.`Expr1` AND
`T6_icaen`.`Expr1` IS NOT NULL AND `T7_icaen`.`CARACT_GEN_SUP` =
`T8_icaen`.`CARACT_GEN_SUP` AND `T7_icaen`.`CARACT_GEN_SUP` IS NOT
NULL AND `T7_icaen`.`Expr1` IS NOT NULL AND
`T8_icaen`.`CARACT_GEN_SUP` IS NOT NULL)
```

... to use rdb2rdf tools to expose large science archives for data integration.

SPARQL for expressing scientific queries, and the performance of several triple stores and RDB2RDF tools for executing queries over a moderately sized sample of a large astronomical data set. We found that more research and improvements are required into SPARQL and RDB2RDF tools to efficiently expose existing science archives for data integration.

Challenges





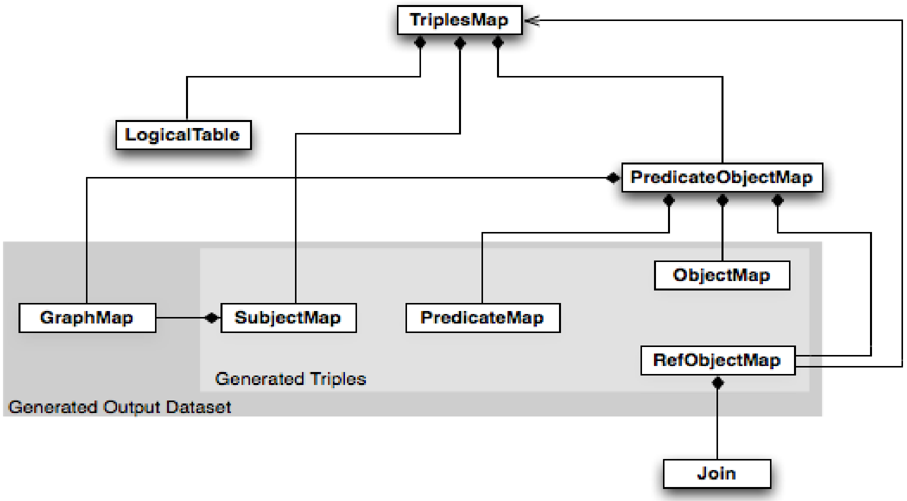
- Query answering takes a lot of time
  - Inefficient queries
  - Some calculation is performed on memory

- Can't be evaluated
  - Too many tables

- Lack of formalization

R2RML

- TriplesMap -> RDF Triples
- LogicalTable: Base Table or SQL Query
- SubjectMap -> subject
- PredicateObjectMap
  - PredicateMap -> predicate
  - ObjectMap -> object
  - RefObjectMap: join



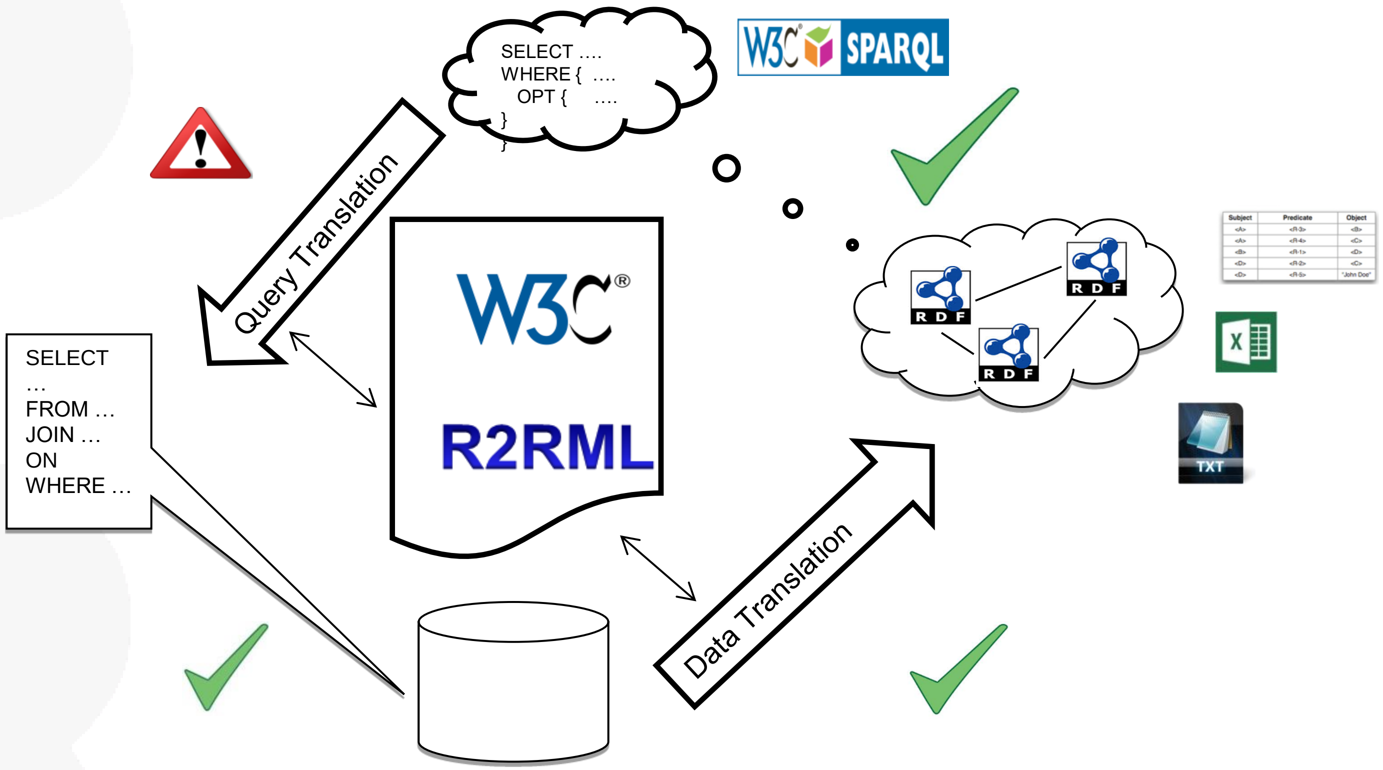
Query Translation in Triples Stores

- Chebotko
- Elliot

Query Translation in RDB2RDF

- R2O
- D2R
- Ontop
- Unbehauen
- Garrote

R2RML-based Query Translation



• **Query translation approaches applied for the implementation of query evaluation over RDB-backed triple stores can be extended to work with R2RML mappings**
• **R2RML-based query translation can be optimized, while preserving semantics, so that they can be applied in real-world cases**

Extension of trans function

ARTICLE IN PRESS

$trans(tp,\alpha,\beta) =$

Select Distinct $genPR\text{-}SQL(tp,\beta,name)$ From $\alpha(tp)$ Where $genCond\text{-}SQL(tp,\beta)$; (13)

$trans(gp_1\ AND\ gp_2,\alpha,\beta) =$

Select Distinct $name(a),_{[a|a\in(terms(gp_1)-terms(gp_2))]}$ $name(b),_{[b|b\in(terms(gp_2)-terms(gp_1))]}$
Coalesce($r_1.name(c)$,$r_2.name(c)$) As $name(c),_{[c|c\in(terms(gp_1)\ \cap\ terms(gp_2))]}$
From ($trans(gp_1,\alpha,\beta)$) r_1 Inner Join ($trans(gp_2,\alpha,\beta)$) r_2 (14)
On (True And$_{[c|c\in(terms(gp_1)\ \cap\ terms(gp_2))]}$
($r_1.name(c)$=$r_2.name(c)$ Or $r_1.name(c)$ Is Null Or $r_2.name(c)$ Is Null));
where $r_1 = alias()$ and $r_2 = alias()$.

$trans(gp_1\ OPT\ gp_2,\alpha,\beta) =$

Select Distinct $name(a),_{[a|a\in(terms(gp_1)-terms(gp_2))]}$ $name(b),_{[b|b\in(terms(gp_2)-terms(gp_1))]}$
Coalesce($r_1.name(c)$,$r_2.name(c)$) As $name(c),_{[c|c\in(terms(gp_1)\ \cap\ terms(gp_2))]}$
From ($trans(gp_1,\alpha,\beta)$) r_1 Left Outer Join ($trans(gp_2,\alpha,\beta)$) r_2 (15)
On (True And$_{[c|c\in(terms(gp_1)\ \cap\ terms(gp_2))]}$
($r_1.name(c)$=$r_2.name(c)$ Or $r_1.name(c)$ Is Null Or $r_2.name(c)$ Is Null));
where $r_1 = alias()$ and $r_2 = alias()$.

$trans(gp_1\ UNION\ gp_2,\alpha,\beta) =$

Select $name(a)_{[a|a\in A]}$, $name(b)_{[b|b\in B]}$, $r_1.name(c)_{[c|c\in C]}$ As $name(c)$
From ($trans(gp_1,\alpha,\beta)$) r_1 Left Outer Join ($trans(gp_2,\alpha,\beta)$) r_2 On (False) (16)
Union
Select $name(a)_{[a|a\in A]}$, $name(b)_{[b|b\in B]}$, $r_3.name(c)_{[c|c\in C]}$ As $name(c)$
From ($trans(gp_2,\alpha,\beta)$) r_3 Left Outer Join ($trans(gp_1,\alpha,\beta)$) r_4 On (False);
where r_1, r_2, r_3, and $r_4 = alias()$; A, B, and C are ordered sets $(terms(gp_1) - terms(gp_2))$, $(terms(gp_2) - terms(gp_1))$, and $(terms(gp_1)\ \cap\ terms(gp_2))$, respectively.

$trans(gp\ FILTER\ expr,\alpha,\beta) =$

Select * From ($trans(gp,\alpha,\beta)$) $alias()$ Where $transexpr(expr)$; (17)

$trans(SELECT\ (v_1,v_2,\ldots,v_n)\ WHERE(gp),\alpha,\beta) =$

Select Distinct $name(v_1)$,$name(v_2)$,...,$name(v_n)$ From ($trans(gp,\alpha,\beta)$) $alias()$; (18)

trans(tp)

- Step1: Multiple mapped TriplesMaps
  - Ex: :fred hasID ?fredID
    - TriplesMapStudent
    - TriplesMapResearcher
  - UNION of trans(tp, tm)
  - m is TriplesMap corresponding to the triple pattern tp
- Step2: Unbounded predicate
  - Ex: :fred ?p ?o
    - :fred hasID ?o
    - :fred hasName ?o
  - UNION of trans(tp, tm, predicateURI)
  - predicateURI is the mapped predicate defined in tm

trans(tp, tm, predicateURI) =
SELECT *genPRSQL(tp, beta, name)*
FROM *alpha*
WHERE *genCondSQL(tp)*

Previously on Morph ...

Query Translation - $\alpha^M(tp)$ - Definition

Definition (Mapping α under M)

G
a
R
α
a

Query Translation - $\beta^M(tp, pos)$ - Algorithm

$\beta^M(tp, pos, M)$:
TriplesMap = *getTriplesMap*(*tp*, *M*)

Query Translation - genCondSQL - Algorithm (2/2)

name, and a set
erates an SQL
relational
.predicate,

27 / 51

Query Translation - One of The Results

Freddy Priyatna – Query Translation for R2RML 43 / 51

R2RML-based Query Translation: Examples

```
<TMProduct> a rr:TriplesMap;
  rr:logicalTable [ rr:tableName  "Product" ];

  rr:subjectMap [ a rr:Subject; rr:class bsbm:Product;
    rr:template "http://ex.com/Product/{nr}"; ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant :hasProductName];
    rr:objectMap [ rr:column "label"; ]; ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant :hasProductOffer ];
    rr:objectMap [ rr:parentTriplesMap <TriplesMapOffer>;
      rr:joinCondition [ rr:child "nr" ; rr:parent "product" ; ] ];
].

<TMOffer> a rr:TriplesMap;
  rr:logicalTable [ rr:tableName  "Offer" ];

  rr:subjectMap [ a rr:Subject; rr:class bsbm:Offer;
    rr:template "http://ex.com/Offer/{nr}"; ];

  rr:predicateObjectMap [
    rr:predicateMap [ rr:constant bsbm:price ];
    rr:objectMap [ rr:column "price"; ]; ].
```

- Trans(tp, tm, predicateURI) =

  SELECT *genPRSQL*(tp, *beta*, *name*)

  FROM *alpha*(tp)

  WHERE *genCondSQL*(tp)

tp	Alpha(tp)	Beta(tp, pos)	genCondSQL(tp)	genPRSQL(tp, Beta, name)
?product :hasProductName ?prodName	{ Product }	Sub:Product.nr Pre:' hasProductName ' Obj: Product.label	Product.label IS NOT NULL	Product.nr AS var_product 'hasProductName' AS iri_hasProductName Product.label AS var_prodName
:Product/1 :hasProductOffer :Offer/3847	{ Product, Offer }	Sub: Product.nr Pre: 'hasProductOffer ' Obj: Offer.nr	Product.nr=1 AND Product.nr=Offer.product AND Offer.nr=3847	Product.nr AS iri_Product1 'hasProductOffer' AS iri_hasProductOffer Offer.nr AS iri_Offer3847

Query Optimizations

SPARQL

```
SELECT DISTINCT ?pr ?productLabel
?productComment ?productProducer
WHERE {
   ?pr hasLabel ?label .
   ?pr hasComment ?comment .
   ?pr hasProducer ?producer .
}
```

Naïve Translation

```
SELECT DISTINCT pr, productLabel, productComment, productProducer
FROM (
  SELECT p_1.pr, p_1.productLabel,
    p_2_3.productComment, p_2_3.productProducer
  FROM
   (SELECT nr AS pr, label AS productLabel FROM product
     WHERE nr IS NOT NULL AND label IS NOT NULL) p_1,
   (SELECT p_2.pr, p_2.productComment, p_3.productProducer FROM
     (SELECT nr AS pr, comment AS productComment FROM product
        WHERE nr IS NOT NULL AND comment IS NOT NULL) p_2,
   (SELECT nr AS pr, producer AS productProducer FROM product
     WHERE nr IS NOT NULL AND producer IS NOT NULL) p_3
  WHERE p_2.pr= p_3.pr) p_2_3
WHERE p_1.pr = p_2_3.pr
) gp
```

Self-join elimination

```
SELECT DISTINCT pr, productLabel, productComment, productProducer
FROM
   (SELECT p.nr AS pr, p.label AS productLabel
     , p.comment as productComment, p.producer AS productProducer
   FROM product p
   WHERE p.nr IS NOT NULL AND p.label IS NOT NULL
   AND p.comment IS NOT NULL AND p.producer IS NOT NULL
   ) gp
```

Subquery Elimination

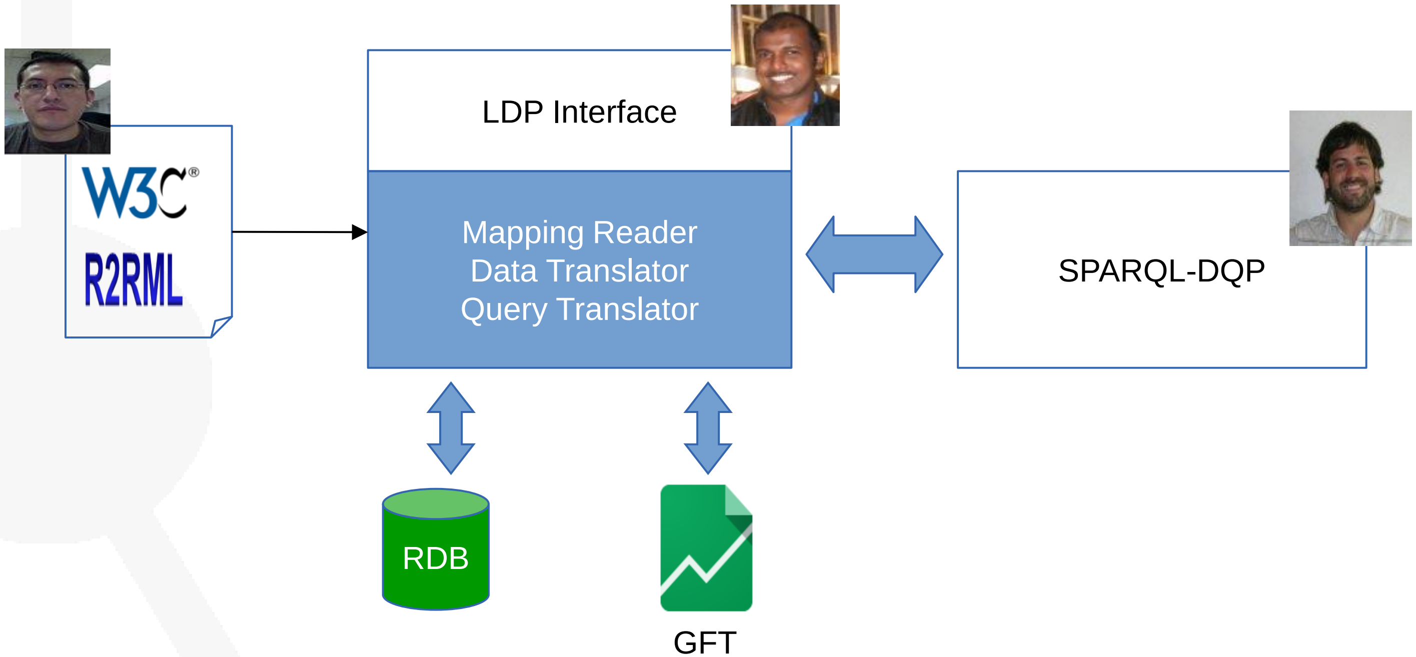
```
SELECT DISTINCT p1.nr AS pr, p1.label AS productLabel,
  p2.comment AS productComment, p3.producer AS productProducer
FROM product p1, product p2, product p3
WHERE p1.nr = p2.nr AND p2.nr = p3.nr
  AND p1.nr IS NOT NULL AND p2.nr IS NOT NULL
  AND p3.nr IS NOT NULL AND p1.label IS NOT NULL
  AND p2.comment IS NOT NULL AND p3.producer IS NOT NULL
```

Both

```
SELECT DISTINCT p.nr AS pr, p.label AS productLabel,
  p.comment AS productComment, p.producer AS productProducer
FROM product p
WHERE p.nr IS NOT NULL AND p.label IS NOT NULL
  AND p.comment IS NOT NULL AND p.producer IS NOT NULL
```

Implementation and Evaluation

- Morph: R2RML Engine
  - https://github.com/fpriyatna/morph

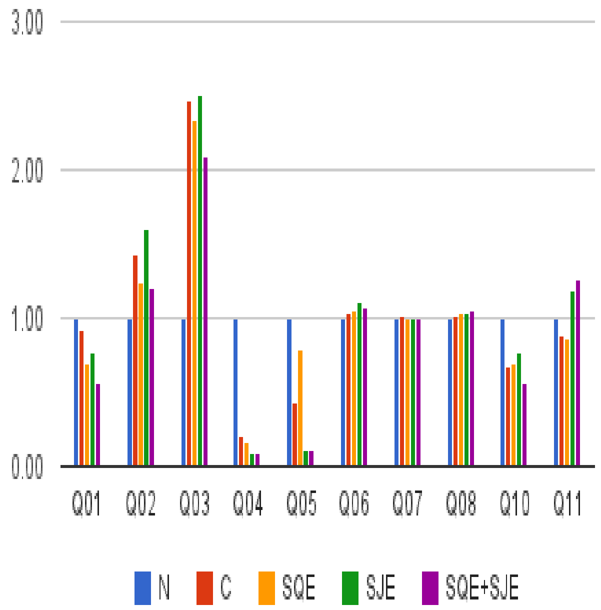


- Synthetic benchmark
  - BSBM (Ecommerce)

- Three Spanish/EU projects
  - BizkaiSense (Enviromental)
  - Repener (Energy)
  - Integrate (Clinical)

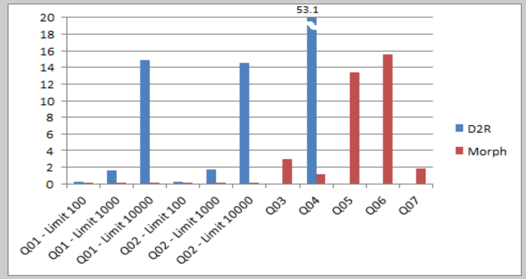
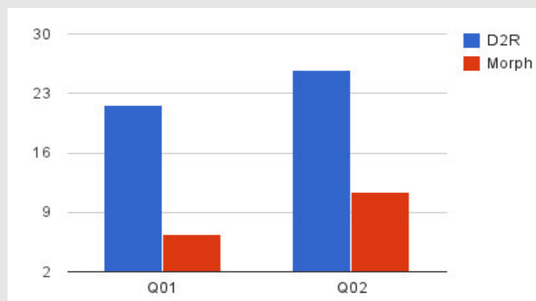
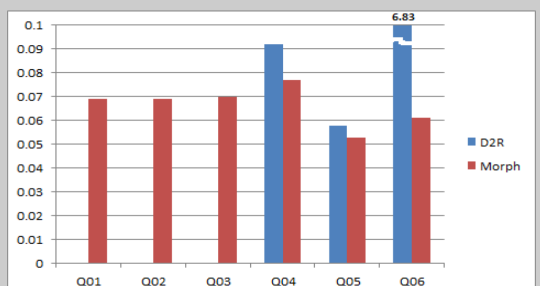
Evaluation: Synthetic Benchmark

- Diverse type of operators/patterns
- Diverse type of solution modifiers
- Combination between low/high selectivity
- 100 millions triples
- 10 SELECT queries
  - Native (N)
  - Naive (C)
  - Subquery Elimination (SQE)
  - Selfjoin Elimination (SJE)
  - SQE+SJE
  - 20 run for each query



- Similar performance for queries types (N, C, SQE, SJE, SQE+SJE) for most queries
  - Non-corelated subquery elimination
  - Joins over indexes columns
- Q04
  - N: corelated subquery
  - Others: joins
- Q05
  - Join order
- Native queries should have been better optimized

Evaluation: Real Cases

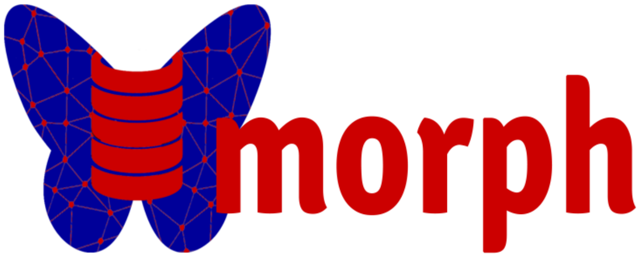
Project Name	Tipical Queries	Result	Observation
BizkaiSense	Q01: observations coming from a particular weather or air quality station Q03: the average measures by property for a given week Q05: the maximum measure in all the stations for each property in a given day		Morph queries takes less time Q3, Q5, Q7 by D2R queries are not able to be evaluated
RÉPENER	Q01: buildings and their climatezone and building life cycle phase Q02: all buildings and their climatezone, building life cycle phase, and conditioned floor area.		Morph queries are 2-3x faster
Integrate	Q01/Q02/Q03 are similarly structured, with a code that specifies whether to obtain tumor size, tumor stage, or pregnant women. Q04 obtains multiple participants who have been treated with Antracyclines Q05 obtains demographic information (people that are older than 30 years old) Q06 obtains images andinformation that a diagnosis is based on		Too many joined-tables error message for D2R Q01/Q02/Q30 Similar performance of Morph and D2R for Q04 and Q05 Significantly faster for Morph Q06 vs D2R Q06

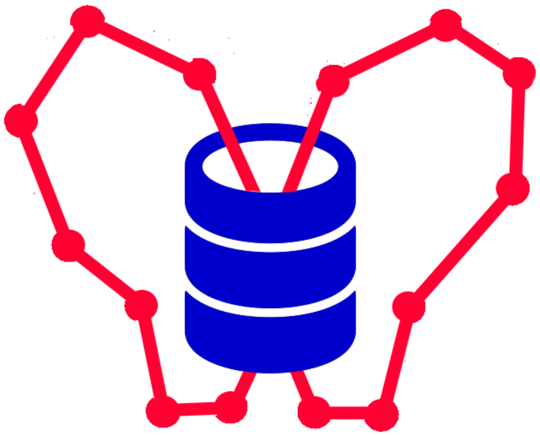
Conclusion & Future Work

Employing Semantics Query Optimizations for exploiting more indexes

Making our implementation via Web Service

Some Pictures





Formalization and Experiences of R2RML-based SPARQL to SQL query translation using Morph

Freddy Priyatna, Oscar Corcho

Ontology Engineering Group
Universidad Politécnica de Madrid
Madrid, Spain
[fpriyatna,ocorcho]@fi.upm.es

Juan Sequeda

Dept. of Computer Science
University of Texas at Austin
Austin, USA
jsequeda@cs.utexas.edu

OEG, March 27th, 2014