SemLav: Local-As-View Mediation for SPARQL Queries

Gabriela Montoya, Luis-Daniel Ibanez, Hala Skaf, **Pascal Molli**, Maria-Esther Vidal

Credible Workshop October 2013,

Sophia Antipolis.

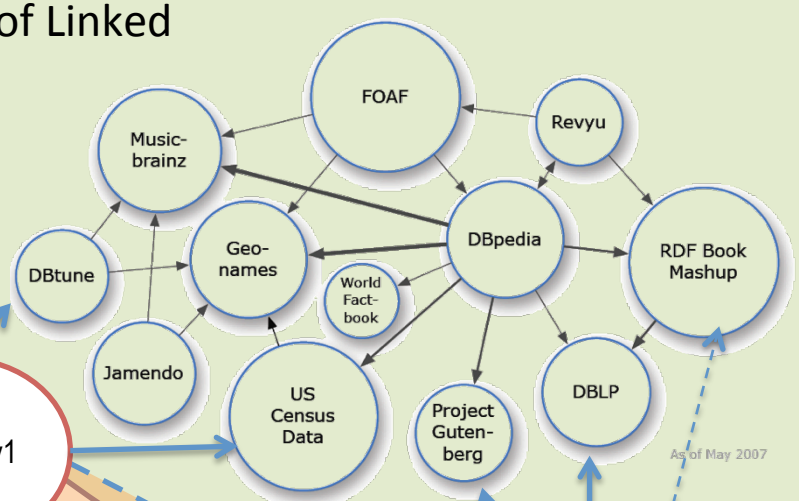
Context

- We want to execute SPARQL queries over a set of distributed, dynamic and semantically heterogeneous data sources
  - Sources with heterogeneous schemas.
  - Served by regular autonomous web services, including FTP, open data API (can include SPARQL endpoint, but it is not the rule).
  - No statistics available.

Why it is important?

- Most open datasets are not yet integrated into linked data
- Be able to extend federated queries to deep web
- Extend the scope of linked data cloud with data that are not 5 stars.

Federation of Linked Open Data



Deep web, web service, Non-Five star open data.

Mediated schema

SemLav1

Wrappers& mappings

Data.gov.ma

Geodata.gov.gr

twitter

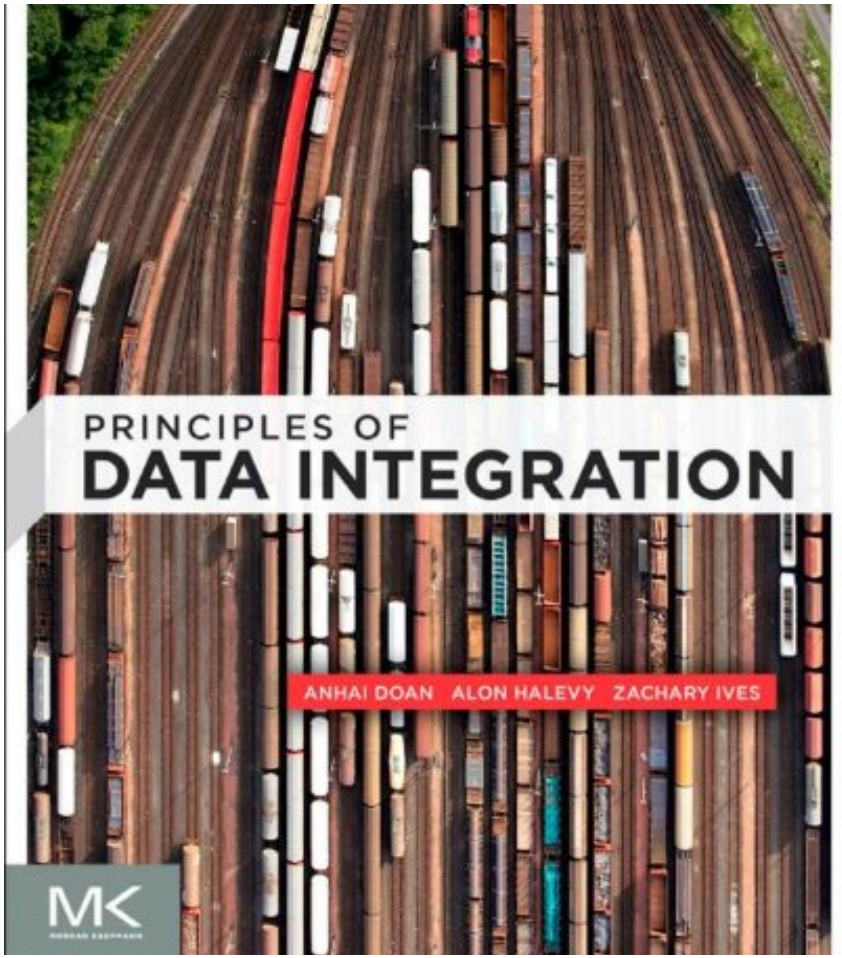
Data.nantes.fr

SemLav2

Wrappers& mappings

Why it is challenging?

- Data integration is generally AI-complete
- Each approach has its own bottleneck,
  - Warehousing -> freshness
  - GAV -> new sources
  - LAV -> rewritings complexity
- **Focus on LAV for freshness and support of dynamicity in the context of SPARQL and semantic web**



LAV Approach: rewrite queries in terms of sources

- Q(P, O, V, C) :- type(P, t1), product(O, P), vendor(O, V), country(V, C).
- 4 Views:
  - S1(P, T, O, V) ⊆
  - S2(V, C) ⊆ cour
  - S3(P, O, S, R) ⊆
  - S4(P, O, S) ⊆ pr

```
SELECT *
WHERE {
  ?X1 rdfs:label  ?X2 .
  ?X1 rdfs:comment ?X3 .
  ?X1 bsbm:productPropertyTextual1 ?X8 .
  ?X1 bsbm:productPropertyTextual2 ?X9 .
  ?X1 bsbm:productPropertyTextual3 ?X10 .
  ?X1 bsbm:productPropertyNumeric1 ?X11 .
  ?X1 bsbm:productPropertyNumeric2 ?X12 .
}
```

- 2 rewritings with 3 WS calls
  - Q(P,O,V,C) :- S1(P,t1,O,V),S2(V,C)
  - Q(P,O,V,C) :- S4(P,O,V),S3(P,O, 0, 1),S2(V,C)

LAV: Query Rewriting Problem

Given a conjunctive query Q and a set of views $V = \{ v_0, \ldots, v_n \}$ over a database D, QRP is to find a set of rewritings R such that:

- *For all instances of views in the bodies of all rewritings in R, the union of the results of executing each query rewriting is contained in the result of evaluating Q in D, i.e.,*
  $\bigcup_{r \in R} r(I(v_{r1}), \ldots, I(v_{ri})) \subseteq Q(D)$
- *R is maximal, i.e., there is no other set R', such that:*

$$\bigcup_{r \in R} r(I(v_{r1}), \ldots, I(v_{ri})) \subset \bigcup_{r' \in R'} r'(I(v_{r'1}), \ldots, I(v_{r'i})) \subseteq Q(D)$$

with $v_{r1}, \ldots, v_{ri}$ the views that appear in rewriting r.

LAV Drawbacks

- The number of candidate rewritings in the worst case is: $\mathbf{(M \times |V|)^N}$.
  - N the number of query subgoals,
  - M the maximal number of views subgoals,
  - and V the set of views,
- Rewriting generation is NP-Complete
- LAV is restricted to conjunctive queries

A. Levy et al, Answering queries using views, Proceedings of the fourteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, ACM, 1995, pp. 95-104.
Y. Arvelo et al, Compilation of Query-Rewriting Problems into Tractable Fragments of Propositional Logic, Proceedings of the 21th national conference on Artificial intelligence - Volume 1, AAAI'06, AAAI Press, 2006, pp. 225-230.
R. Pottinger et al, Minicon: A scalable algorithm for answering queries using views, VLDB, 2000, pp. 484-495.
G. Konstantinidis et al, Scalable query rewriting: a graph-based approach, SIGMOD Conference, ACM, 2011, pp. 97-108.

LAV and SPARQL: Rewritings!

Query	Included Views	# Relevant Views	# Rws covered	# Rewritings
Q1	28	408	1.61E+06	2.04E+10
Q2				1.57E+24
Q4				1.62E+04
Q5				7.48E+07
Q6				3.14E+05
Q8				1.57E+05
Q9				3.40E+01
Q10				4.40E+06
Q11				9.25E+03
Q12				1.50E+09
Q13				6.47E+04
Q14				2.52E+06
Q15				2.04E+10
Q16				3.14E+05
Q17	56	136	1.90E+03	4.62E+03
Q18	23	374	2.80E+05	1.20E+09

476 Views and

Q1=SELECT * WHERE {
?X1rdfs:label ?X2.
?X1 rdfs:comment ?X3 .
?X1 bsbm:productPropertyTextual1 ?X8 . ?X1
bsbm:productPropertyTextual2 ?X9 . ?X1
bsbm:productPropertyTextual3 ?X10 . ?X1
bsbm:productPropertyNumeric1 ?X11 . ?X1
bsbm:productPropertyNumeric2 ?X12 .
}

LAV and SPARQL: NPHard matters!

224 Views and Q1: SELECT * WHERE {
?X1rdfs:label ?X2.
?X1 rdfs:comment ?X3 .
?X1 bsbm:productPropertyTextual1 ?X8 . ?X1
bsbm:productPropertyTextual2 ?X9 . ?X1
bsbm:productPropertyTextual3 ?X10 . ?X1
bsbm:productPropertyNumeric1 ?X11 . ?X1
bsbm:productPropertyNumeric2 ?X12 .
}

Query	Rewriter	5 minutes	10 minutes	20 minutes
Q1	GQR	0	0	0
	MCDSAT	211,125	440,308	898,766
	MiniCon	0	0	0
Q2	GQR	0	0	0
	MCDSAT	67,028	157,909	335,063
	MiniCon	0	0	0

Table 1: Number of rewritings obtained from rewriters GQR, MCDSAT and MiniCon with timeouts of 5, 10 and 20 minutes. Using 224 views and queries Q1 and Q2.

SemLav: Don’t rewrite! Cheat…

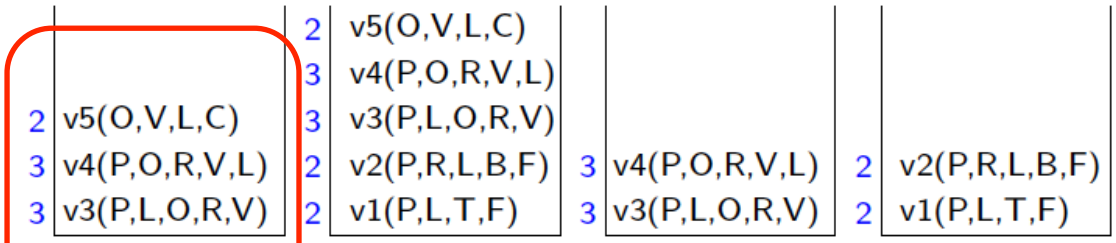
- Let select sources as LAV engine do
  - (and hope to select only a fragment of sources)
- Temporally materialize body of relevant views instead of head, and consider just distinguished variables
  - Will take more space but allow nearly full SPARQL queries to be executed
  - S1(P, T, O, V) ⊆ product(O, P), type(P, T), vendor(O, V)
- Materialize views in smart order
  - Time for First Answers and throughput
  - Determine the order according “coverage” of rewritings…

Covering rewritings…

Query:

Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)

Buckets:



Data sources:

v1(P,L,T,F) :- label(

v2(P,R,L,B,F) :- proc

v3(P,L,O,R,V) :- labe

v4(P,O,R,V,L) :- proc

v5(O,V,L,C) :- vendor(O,V), label(V,L), country(V,C)

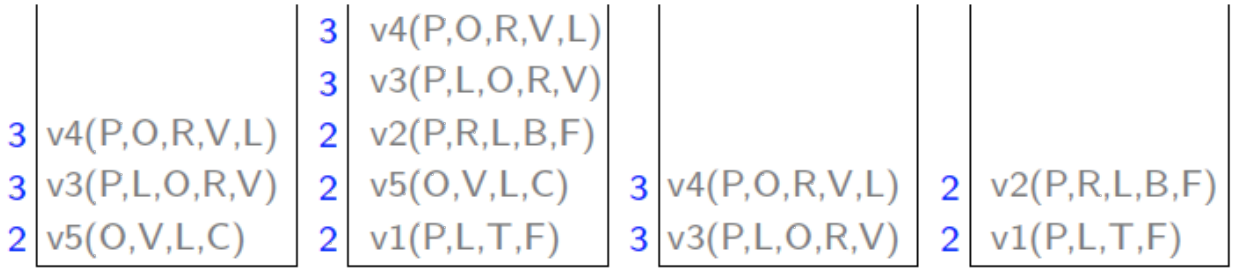
60 rewritings:

- V5,V5,V4,V2
- V5,V4,V4,V2
- ….

Covering rewritings: orders matters

Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)

Buckets:



Vk	#Rw coverage
{}	0x0x0x0=0
{v4}	1x1x1X0=0
{v4,v2}	1x2x1x1=2
{v4,v2,v3}	2x3x2x1=12
{v4,v2,v3,v1}	2x4x2x2=32
{v4,v2,v3,v1,v5}	3x5x2x2=60

Vk	#Rw coverage
{}	0x0x0x0=0
{v5}	1x1x0X0=0
{v5,v1}	1x2x0x1=0
{v5,v1,v3}	2x3x1x1=6
{v5,v1,v3,v2}	2x4x1x1=8
{v5,v1,v3,v2,v4}	3x5x2x2=60

Maximal Coverage Problem (MaxCov)

Given $k > 0$, Q a query defined for dataset D, V a set of views over dataset D, R solution of QRP for Q and V. Let V_k be subset of V of relevant views for Q of size k. Find V_k, such that the set of rewritings covered by V_k, $Coverage(V_k, R)$, is maximal for all subsets of V of size k. $Coverage(V_k, R)$ is defined as:

$$Coverage(V_k, R) = \{r : r \in R \wedge (\forall p : p \in body(r) : p \in V_k)\}$$

SemLav approach

1. Selection and Ranking
   - Quite similar to traditional bucket algorithm
   - Ranking according presence of same atom in different bucket.
2. Compute view loading order according MaxCov
3. Execute query each time a new view loaded
   - good Time for For Answer
   - Allow to execute *nearly* full SPARQL queries

Step1: views selection & ranking

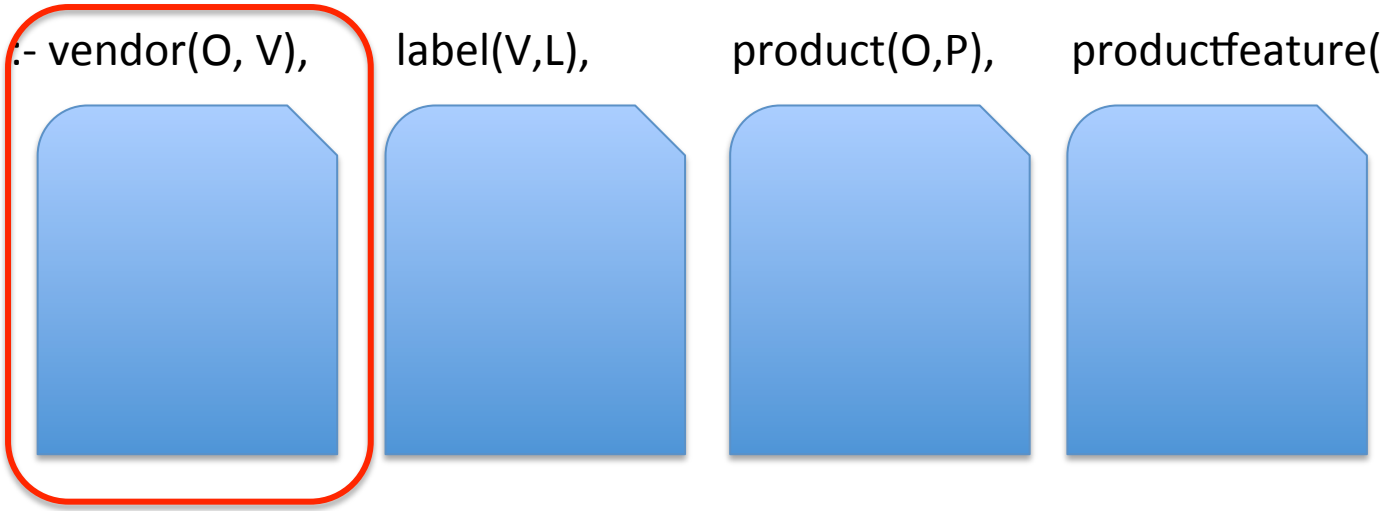
Input: Q : SPARQL Query; V: Set of Views defined as conjunctive queries
Output: *Buckets*: Predicate → List<View>
  for all $q \in body(Q)$ **do**
    $buckets(q) \leftarrow \emptyset$
  end for
  for all $q \in body(Q)$ **do**
    $b \leftarrow buckets(q)$

The complexity of Algorithm 1 is $Max(O(N \times |V| \times M), O(N \times |V| \times log(|V|))$ *where N is the number of query subgoals, V the set of views and M the maximal number of view subgoals.*

      end if
    end for
  end for
  end for
  for all $q \in body(Q)$ **do**
    $b \leftarrow buckets(q)$
    sortBucket(buckets, b, q)
  end for

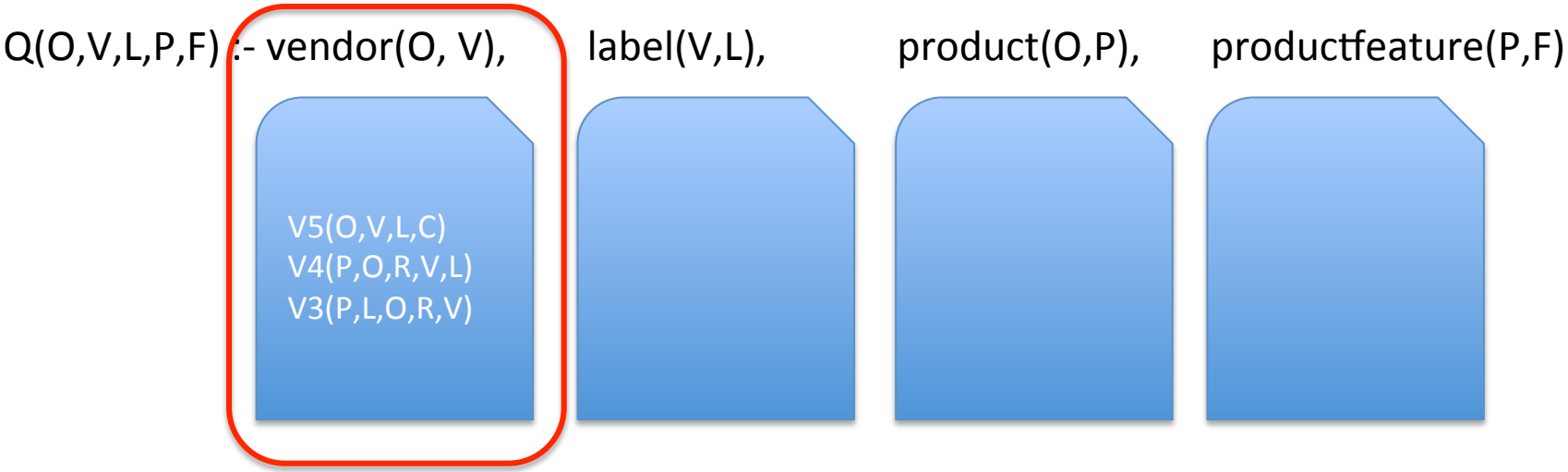
Step1: Selections & Ranking

Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)



v1(P,L,T,F)	⊆	Label(P,L)	Type(P,T)	Productfeature(P,F)	
V2(P,R,L,B,F)	⊆	Producer(P,R)	Label(R,L)	Publisher(P,B)	Productfeature(P,F)
V3(P,L,O,R,V)	⊆	Label(P,L)	Product(O,P)	Price(O,R)	Vendor(O,V)
V4(P,O,R,V,L)	⊆	Product(O,P)	Price(O,R)	Vendor(O,V)	Label(V,L)
V5(O,V,L,C)	⊆	Vendor(O,V)	Label(V,L)	Country(V,C)	

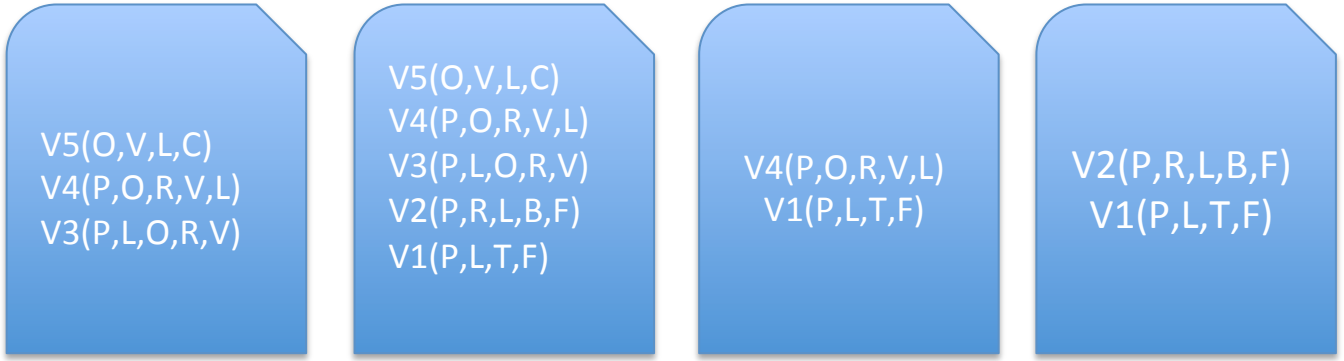
Step1: Selections & Ranking



v1(P,L,T,F)	⊆	Label(P,L)	Type(P,T)	Productfeature(P,F)	
V2(P,R,L,B,F)	⊆	Producer(P,R)	Label(R,L)	Publisher(P,B)	Productfeature(P,F)
V3(P,L,O,R,V)	⊆	Label(P,L)	Product(O,P)	Price(O,R)	Vendor(O,V)
V4(P,O,R,V,L)	⊆	Product(O,P)	Price(O,R)	Vendor(O,V)	Label(V,L)
V5(O,V,L,C)	⊆	Vendor(O,V)	Label(V,L)	Country(V,C)	

Step1: Selections & Ranking

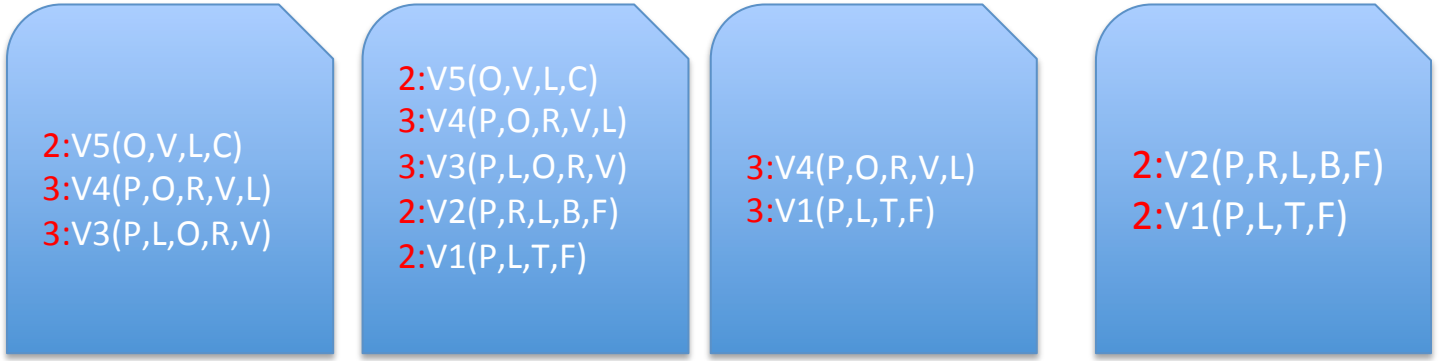
Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)



v1(P,L,T,F)	⊆	Label(P,L)	Type(P,T)	Productfeature(P,F)	
V2(P,R,L,B,F)	⊆	Producer(P,R)	Label(R,L)	Publisher(P,B)	Productfeature(P,F)
V3(P,L,O,R,V)	⊆	Label(P,L)	Product(O,P)	Price(O,R)	Vendor(O,V)
V4(P,O,R,V,L)	⊆	Product(O,P)	Price(O,R)	Vendor(O,V)	Label(V,L)
V5(O,V,L,C)	⊆	Vendor(O,V)	Label(V,L)	Country(V,C)	

Step1: Selections **& Ranking**

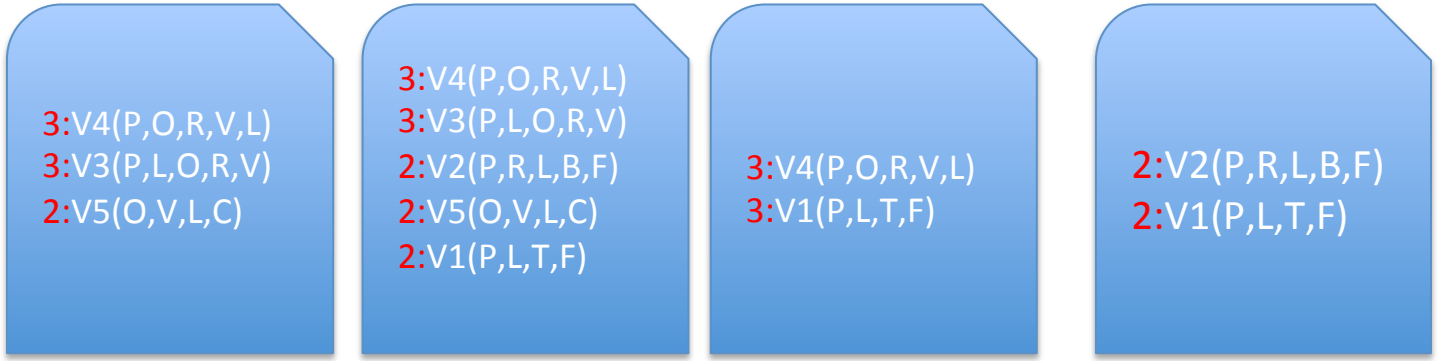
Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)



v1(P,L,T,F)	⊆	Label(P,L)	Type(P,T)	Productfeature(P,F)	
V2(P,R,L,B,F)	⊆	Producer(P,R)	Label(R,L)	Publisher(P,B)	Productfeature(P,F)
V3(P,L,O,R,V)	⊆	Label(P,L)	Product(O,P)	Price(O,R)	Vendor(O,V)
V4(P,O,R,V,L)	⊆	Product(O,P)	Price(O,R)	Vendor(O,V)	Label(V,L)
V5(O,V,L,C)	⊆	Vendor(O,V)	Label(V,L)	Country(V,C)	

Step1: Selections **& Ranking**

Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)



v1(P,L,T,F)	⊆	Label(P,L)	Type(P,T)	Productfeature(P,F)	
V2(P,R,L,B,F)	⊆	Producer(P,R)	Label(R,L)	Publisher(P,B)	Productfeature(P,F)
V3(P,L,O,R,V)	⊆	Label(P,L)	Product(O,P)	Price(O,R)	Vendor(O,V)
V4(P,O,R,V,L)	⊆	Product(O,P)	Price(O,R)	Vendor(O,V)	Label(V,L)
V5(O,V,L,C)	⊆	Vendor(O,V)	Label(V,L)	Country(V,C)	

2: View ordering & Query Evaluation

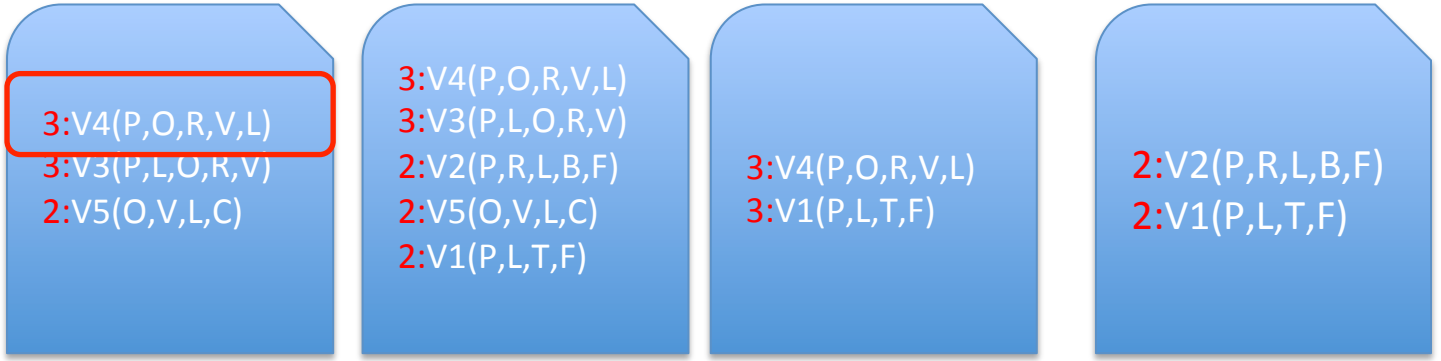
Input: Q : Query; k : Int; *Buckets*: Predicate → List<View> {The buckets are produced by Algorithm 1}
Output: A: Set<Answer>
$Stacks$: Predicate → Stack<View>
V_k : Set<View>
G : RDFGraph

The complexity of Algorithm 2 is $O(k \times VI)$, where k is the number of included views and VI is the size of the largest view instance.

for all $p \in domain(Stacks)$ **do**
$v \leftarrow pop(Stack(p))$
if $v \notin V_k$ **then**
load v into G {only if is not redundant}
$A \leftarrow A \cup exec(Q, G)$ {Option 1: Execute Q after each successful load}
$V_k \leftarrow V_k \cup \{v\}$
end if
end for
end while
$A \leftarrow exec(Q, G)$ {Option 2: execute before exit}

Step1: Selections **& Ranking**

Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)



	Vk	#Rw coverage
0	{}	0x0x0x0=0

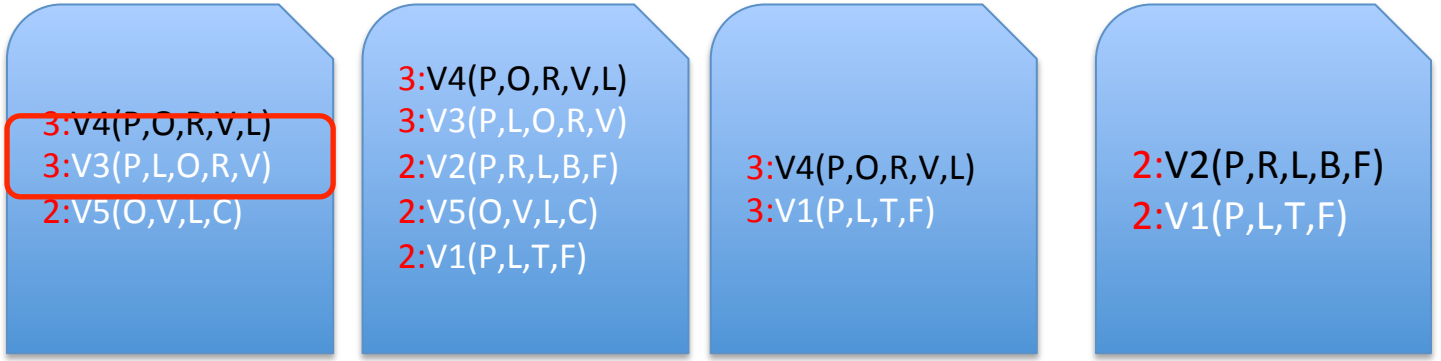
Step1: Selections **& Ranking**

Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)

	Vk	#Rw coverage
0	{}	0x0x0x0=0
1	{v4}	1x1x1X0=0

Step1: Selections **& Ranking**

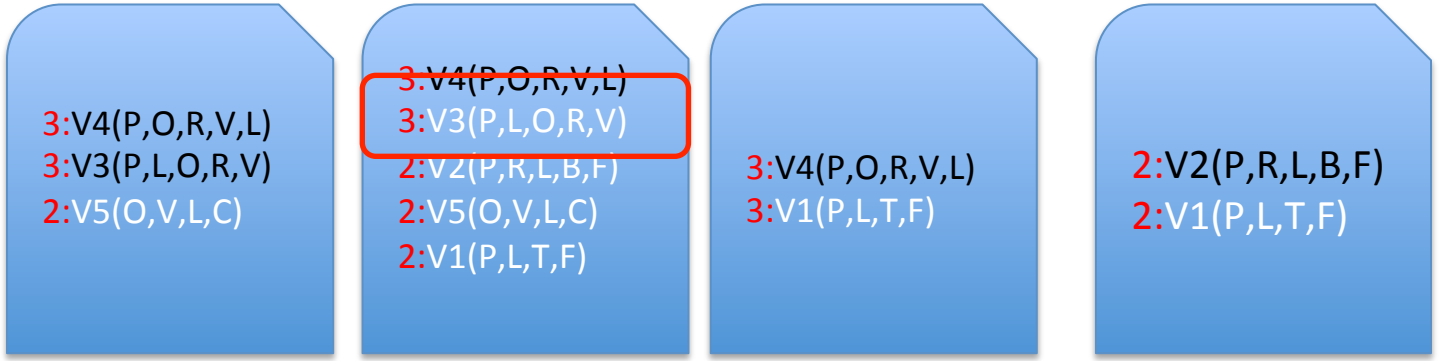
Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)



	Vk	#Rw coverage
0	{}	0x0x0x0=0
1	{v4}	1x1x1X0=0
2	{v4,v2}	1x2x1x1=2

Step1: Selections **& Ranking**

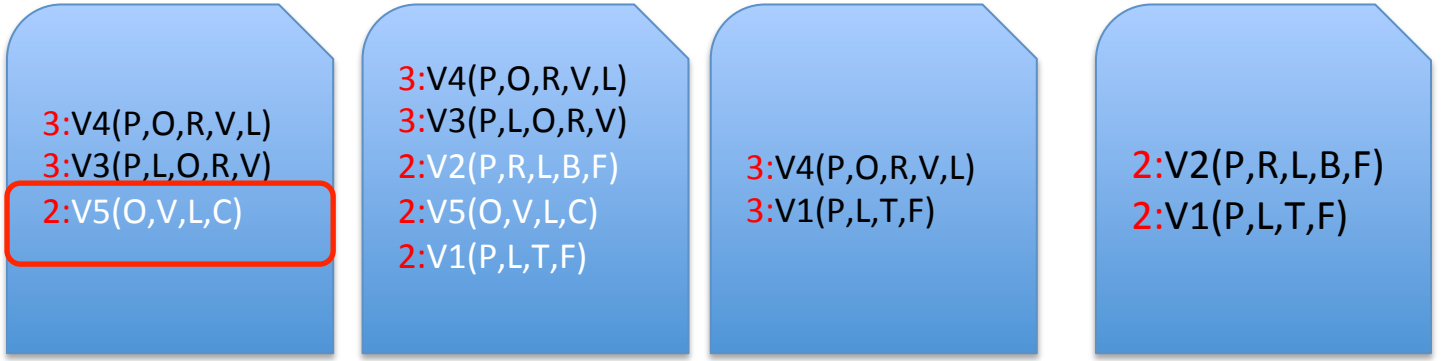
Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)



	Vk	#Rw coverage
0	{}	0x0x0x0=0
1	{v4}	1x1x1X0=0
2	{v4,v2}	1x2x1x1=2
3	{v4,v2,v3}	2x3x2x1=12

Step1: Selections **& Ranking**

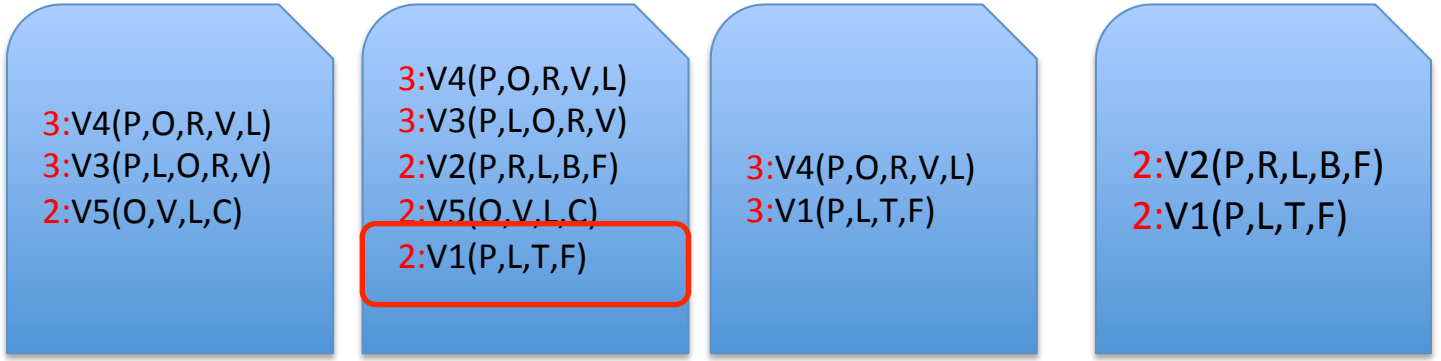
Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)



	Vk	#Rw coverage
0	{}	0x0x0x0=0
1	{v4}	1x1x1X0=0
2	{v4,v2}	1x2x1x1=2
3	{v4,v2,v3}	2x3x2x1=12
4	{v4,v2,v3,v1}	2x4x2x2=32

Step1: Selections **& Ranking**

Q(O,V,L,P,F) :- vendor(O, V), label(V,L), product(O,P), productfeature(P,F)



	Vk	#Rw coverage
0	{}	0x0x0x0=0
1	{v4}	1x1x1X0=0
2	{v4,v2}	1x2x1x1=2
3	{v4,v2,v3}	2x3x2x1=12
4	{v4,v2,v3,v1}	2x4x2x2=32
5	{v4,v2,v3,v1,v5}	3x5x2x2=60

SemLav Properties

- Given a SPARQL query Q, a set of views V on a dataset D, RV the set of relevant views for Q and R a solution to the QRP of Q over V , V_k solution to MaxCov problem found with SemLAV.
  - *Answer Completeness:* If SemLAV executes *Q* over a dataset instance D that includes all data gathered from views in *RV*, then it will produce the complete answer.
- *Effectiveness(V_k)* = |Coverage(V_k,R)|/|R|

SemLav Properties

- *Execution Time* depends on $|RV|$,
- *No memory blocking:*
  - SemLAV guarantees to obtain complete answer if RVs fits.
  - If not, it is necessary to divide this set of relevant views *RV* into several subsets RV_i such that each subset fits into the memory and that for any $r \in R$, all views $v \in body(r)$ are contained in one RV_i.

Expectations and Limitations

- SemLAV effectiveness should be considerably high.
- SemLAV could consume more space than naive rewriting based approach.
- SemLAV obtains more answers in the same amount of time.
- SemLAV obtains answers sooner.

Experimental Setup

- Comparison of SemLav with MCDSAT [6], SSDSAT, Minicon and GQR.
- Dataset of products generated with Berlin Benchmark [7].
  - *Size:* ten millions of triples.
  - 16 conjunctive queries and 14 views from [8], (so we did not choose the queries).
  - The views were horizontally partitioned into 476 views.
  - *Timeout:* 10 minutes.
  - Considering no up to date statistics are available.
- Machine: Intel(R) Xeon(R) CPU X7460@2.66GHz with 128GBof RAM and Debian GNU/Linux 6.0.4 operating system.

[6] Y. Arvelo et al, Compilation of Query-Rewriting Problems into Tractable Fragments of Propositional Logic, Proceedings of the 21th national conference on Artificial intelligence - Volume 1, AAAI'06, AAAI Press, 2006, pp. 225-230.
[7] C. Bizer et al, The berlin sparql benchmark, International Journal on Semantic Web and Information Systems 5 (2009), no. 2, 1-24.
[8] R. Castillo-Espinola, Indexing rdf data using materialized sparql queries, Ph.D. thesis, Humboldt-Universitatzu Berlin, 2012.

Queries

Query	Answer Size	# Subgoals
Q1	6.68E+07	5
Q2	5.99E+05	12
Q4	2.87E+02	2
Q5	5.64E+05	4
Q6	1.97E+05	3
Q8	5.64E+05	3
Q9	2.82E+04	1
Q10	2.99E+06	3
Q11	2.99E+06	2
Q15	2.82E+05	5
Q16	2.82E+05	3
Q17	1.97E+05	2
Q18	5.64E+05	4

(a) Query information

Views	Size
V1-V34	201,250
V35-V68	153,523
V69-V102	53,370
V103-V136	26,572
V137-V170	5,402
V171-V204	66,047
V205-V238	40,146
V239-V272	113,756
V273-V306	24,891
V409-V442	78,594
V443-V476	99,237
V477-V510	1,087,281

(b) Views size

Third party Queries and Views !!
Hard conditions for SemLav…

GQR/MCDsat/Minicon/SSDSAT

Query	Metric	GQR	MCDSAT	MiniCon	SSDSAT	Total Number of Rewritings
Q1	Execution Time (secs)	600.00	600.00	600.00	600.00	
	Number of Rewritings	0	247,304	0	0	2.04E+10
	Throughput (answers/sec)	0.00	412.17	0.00	0.00	
Q2	Execution Time (secs)	600.00	600.00	600.00	600.00	
	Number of Rewritings	0	0	0	0	1.57E+24
	Throughput (answers/sec)	0.00	0.00	0.00	0.00	
Q4	Execution Time (secs)	25.71	84.20	5.47	600.00	

Table 4: Comparison of state-of-the-art engines for queries against LAV views on the dataset of 10,000,736 triples generated by the Berlin SPARQL Benchmark (BSBM) [16] using a scale factor of 28,211 products.

Q6	Execution Time (secs)	600.00	251.51	430.10	600.00	
	Number of Rewritings	0	314,432	314,432	0	3.14E+05
	Throughput (answers/sec)	0.00	1,250.18	731.07	0.00	
Q8	Execution Time (secs)	555.49	191.69	142.63	600.00	
	Number of Rewritings	157,216	157,216	157,216	0	1.57E+05
	Throughput (answers/sec)	283.02	820.16	1,102.30	0.00	
Q9	Execution Time (secs)	0.88	32.24	0.34	49.83	
	Number of Rewritings	34	34	34	34	3.40E+01
	Throughput (answers/sec)	38.51	1.05	101.49	0.68	
Q10	Execution Time (secs)	600.00	600.00	600.00	600.00	
	Number of Rewritings	0	656,140	0	0	4.40E+06
	Throughput (answers/sec)	0.00	1,093.57	0.00	0.00	

GQR/MCDsat/Minicon/SSDSAT

Query	Metric	GQR	MCDSAT	MiniCon	SSDSAT	Total Number of Rewritings
Q11	Execution Time (secs)	12.99	67.03	2.06	600.00	
	Number of Rewritings	9,248	9,248	9,248	0	9.25E+03
	Throughput (answers/sec)	712.15	137.96	4,487.14	0.00	
Q12	Execution Time (secs)	600.00	600.00	600.00	600.00	
	Number of Rewritings	0	440,059	0	0	1.50E+09
	Throughput (answers/sec)	0.00	733.43	0.00	0.00	
Q13	Execution Time (secs)	600.00	98.43	22.40	600.00	
	Number of Rewritings	0	64,736	64,736	0	6.47E+04
	Throughput (answers/sec)	0.00	657.69	2,890.52	0.00	
Q14	Execution Time (secs)	600.00	600.00	600.00	600.00	
	Number of Rewritings	0	913,807	0	0	2.52E+06
	Throughput (answers/sec)	0.00	1,523.01	0.00	0.00	
Q15	Execution Time (secs)	600.00	600.00	600.00	600.00	
	Number of Rewritings	0	308,903	0	0	2.04E+10
	Throughput (answers/sec)	0.00	514.84	0.00	0.00	
Q16	Execution Time (secs)	600.00	233.47	380.81	600.00	
	Number of Rewritings	0	314,432	314,432	0	3.14E+05
	Throughput (answers/sec)	0.00	1,346.81	825.68	0.00	
Q17	Execution Time (secs)	3.97	67.25	1.29	600.00	
	Number of Rewritings	4,624	4,624	4,624	0	4.62E+03
	Throughput (answers/sec)	1,165.62	68.76	3,576.18	0.00	
Q18	Execution Time (secs)	600.00	600.00	600.00	600.00	
	Number of Rewritings	0	463,754	0	0	1.20E+09
	Throughput (answers/sec)	0.00	772.92	0.00	0.00	

GQR/MCDsat/Minicon/SSDSAT

- SSDSAT: poor performances due to constant handling
- GQR: good performances when the number of Rw is low
- Minicon: deliver better thoughput than GQR but no RW for 8/18 queries
- MCDSat: Not always the best, but produce rewritings in most situations 17/18

SemLav Effectiveness: 10mn Execution

Query	Included Views	# Relevant Views	# Rws covered	# Rewritings	Effectiveness
Q1	30	408	2.28E+06	2.04E+10	0.000112
Q2	194	408	2.05E+23	1.57E+24	0.130135
Q4	156	374	8.77E+03	1.62E+04	**0.542017**
Q5	52	374	3.13E+06	7.48E+07	0.041770
Q6	44	136	2.13E+04	3.14E+05	0.067728
Q8	81	136	9.36E+04	1.57E+05	**0.595588**
Q9	34	34	3.40E+01	3.40E+01	**1.000000**
Q10	88	408	3.20E+05	4.40E+06	0.072766
Q11	77	136	5.24E+03	9.25E+03	**0.566176**
Q12	238	408	7.70E+08	1.50E+09	**0.514286**
Q13	245	408	4.26E+04	6.47E+04	**0.657563**
Q14	46	272	1.22E+04	2.52E+06	0.004837
Q15	70	442	5.12E+08	2.04E+10	0.025144
Q16	82	136	1.90E+05	3.14E+05	**0.602941**
Q17	56	136	1.90E+03	4.62E+03	0.411765
Q18	23	374	2.80E+05	1.20E+09	0.000234

Table 5: SemLAV's Effectiveness. For 10 minutes of execution, we report the number of relevant views included in the dataset, the number of covered rewritings and the achieved effectiveness. Also values for total number of views and rewritings are shown.

$$Effectiveness(V_k) = \frac{|Coverage(V_k, R)|}{|R|}$$

Query	Approach	Answer		Time (msecs)					#EQ	MGS	Throughput
		Size	%	WT	GCT	PET	TT	TFA			(answers / msec)
Q1	SemLAV	22,660,216	33	45,434	8,322	547,310	606,697	**6,370**	15	810,638	**37.3501**
	MCDSAT	290	0	13,688	202	299,546	609,381	309,952		810,409	0.0005
	GQR	0	0	0	0	0	600,415	>600,000		0	0.0000
	MiniCon	0	0	0	0	0	600,136	>600,000		0	0.0000
Q2	SemLAV	590,000	98	177,020	30,676	392,439	600,656	**260,333**	66	1,040,373	**0.9823**
	MCDSAT	0	0	15,519	105	7,058	681,246	>600,000		848,276	0.0000
	GQR	0	0	0	0	0	654,483	>600,000		0	0.0000
	MiniCon	0	0	0	0	0	600,054	>600,000		0	0.0000
Q4	SemLAV	287	**100**	555,528	73,771	327	660,938	**104,501**	47	3,659,707	**0.0004**
	MCDSAT	0	0	154,451	371	181,387	601,590	>600,000		279,896	0.0000
	GQR	0	0	557,125	1,181	11,784	600,665	>600,000		84,046	0.0000
	MiniCon	0	0	413,871	650	91,136	601,750	>600,000		177,838	0.0000
Q5	SemLAV	564,220	**100**	523,084	65,333	44,102	632,809	**116,037**	28	3,396,134	**0.8916**

Table 6: Execution of Queries Q1, Q2, Q4-Q6, Q8-Q18 using SemLAV, MCDSAT, GQR and MiniCon, using 20GB of RAM and a timeout of 10 minutes. It is reported the number of answers obtained, wrapper time (WT), graph creation time (GCT), plan execution time (PET), total time (TT) and time of first answer (TFA), number of times original query is executed (#EQ), maximal graph size (MGS) in terms of number of triples and throughput (number of answers obtained per millisecond).

Q9	SemLAV	28,211	**100**	2,938	697	1,338	5,107	**1,235**	18	169,839	**5.5240**
	MCDSAT	28,211	100	5,609	445	1,643	41,505	34,392		5,417	0.6797
	GQR	28,211	100	3,310	132	1,281	5,709	1,435		5,417	4.9415
	MiniCon	28,211	100	3,086	129	1,362	5,004	862		5,417	5.6377

Query	Approach	Answer Size	%	Time (msecs) WT	GCT	PET	TT	TFA	#EQ	MGS	Throughput (answers / msec)
Q10	SemLAV	2,993,175	**100**	161,047	25,659	417,234	607,841	**9,810**	44	869,340	**4.9243**
	MCDSAT	332,488	11	19,801	67	383,421	600,000	207,191		603,769	0.5541
	GQR	0	0	0	0	0	600,639	>600,000		0	0.0000
	MiniCon	0	0	0	0	0	600,138	>600,000		0	0.0000
Q11	SemLAV	2,993,175	**100**	195,950	27,442	377,255	601,042	**8,352**	43	816,308	**4.9800**
	MCDSAT	1,943,141	64	141,876	389	391,852	600,000	72,939		402,528	3.2386
	GQR	1,442,134	48	248,275	689	340,937	600,000	14,435		307,089	2.4036
	MiniCon	1,956,539	65	217,321	415	385,019	605,021	6,832		402,539	3.2338
Q12	SemLAV	598,635	**100**	258,097	41,062	303,023	609,509	**5,784**	121	1,041,369	**0.9822**
	MCDSAT	0	0	424,369	498	15,271	607,408	>600,000		509,271	0.0000
	GQR	0	0	0	0	0	600,418	>600,000		0	0.0000
	MiniCon	0	0	0	0	0	600,189	>600,000		0	0.0000
Q13	SemLAV	598,635	**100**	452,288	65,043	126,345	671,893	**183,844**	124	3,509,975	**0.8910**
	MCDSAT	0	0	250,542	312	141,728	610,452	>600,000		402,531	0.0000
	GQR	0	0	36,563	344	19,757	600,376	>600,000		31,948	0.0000
	MiniCon	0	0	143,879	625	219,882	605,727	>600,000		206,689	0.0000
Q14	SemLAV	344,885	61	544,919	58,563	32,752	636,387	**29,201**	24	2,921,646	**0.5419**
	MCDSAT	10,308	1	382,674	587	63,689	614,123	133,200		1,206,075	0.0168
	GQR	0	0	0	0	0	600,714	>600,000		0	0.0000
	MiniCon	0	0	0	0	0	600,319	>600,000		0	0.0000
Q15	SemLAV	282,110	**100**	471,609	63,548	109,762	645,172	**2,911**	37	3,255,223	**0.4373**
	MCDSAT	8,298	2	90,061	271	168,041	622,474	217,445		361,882	0.0133
	GQR	0	0	0	0	0	819,679	>600,000		0	0.0000
	MiniCon	0	0	0	0	0	600,171	>600,000		0	0.0000
Q16	SemLAV	282,110	**100**	407,107	53,611	187,986	648,826	**2,531**	46	3,356,755	**0.4348**
	MCDSAT	8,298	2	437,590	852	32,015	601,584	103,641		74,682	0.0138
	GQR	1	0	26,460	79	94	619,761	619,702		1,136,305	0.0000
	MiniCon	252	0	110,366	181	122,022	603,821	400,416		1,151,769	0.0004
Q17	SemLAV	197,112	**100**	547,255	67,857	28,783	644,090	**1,504**	32	3,002,144	**0.3060**
	MCDSAT	156,533	79	412,525	1,727	60,858	600,067	70,476		23,192	0.2609
	GQR	45,037	22	245,953	177	350,406	600,000	27,178		1,098,117	0.0751
	MiniCon	5,779	2	262,608	361	334,810	600,001	26,952		1,099,508	0.0096
Q18	SemLAV	0	0	582,334	65,083	3,543	651,094	>600,000	12	2,806,533	0.0000
	MCDSAT	0	0	256,304	257	100,820	607,091	>600,000		411,901	0.0000
	GQR	0	0	0	0	0	600,791	>600,000		0	0.0000
	MiniCon	0	0	0	0	0	600,186	>600,000		0	0.0000

Experimentations conclusions

- SemLAV effectiveness should be considerably high.
  - Yes
- SemLAV could consume more space than naive rewriting based approach.
  - Yes, overlapping of datasources can reduce maxsize. Maxsize of one rewriting can also be high
- SemLAV obtains more answers in the same amount of time.
  - Yes.
- SemLAV obtains answers sooner.
  - Yes, in most cases. But also, real distribution of answer can give advantage to rewriters in some cases…

query1 - Hotel

query2 - Hotel

query3 - Activity

query4 - Hotel, Activity

query5 - Hotel

query6 - Hotel, DBPedia

```
PREFIX ex:<http://example.org/hotel/>
SELECT * WHERE {
    ?Hotel ex:hasName ?Name .
    ?Hotel ex:hasPostalCode ?PC .
    ?Hotel ex:hasAddress ?Address .
    ?Hotel ex:hasMail ?Mail .
    ?Hotel ex:hasWebSite ?WebSite .
}
```

Evaluate query!

Query1

Number of answer:	127
Total execution time:	8922 ms
Graph size:	719
Wrapper Time:	8543 ms
Graph Creation Time:	25 ms
Execution Time:	208 ms

Relevant Views

view2_0
view3_0
view1_0

Query1

[http://example.org/hotel/hotelLocation83, CHATEAU (DU), 44000, 5, Place de la Duchesse Anne, hotelduchateau44@orange.fr, www.hotelduchateau-nantes.fr]
[http://example.org/hotel/hotelLocation27, LUTETIA, 44500, 13, avenue Olivier Guichard, contact@lutetia-labaule.com, www.lutetia-labaule.com]
[http://example.org/hotel/hotelLocation118, HOTEL ANAIADE, 44600, 45 route de Fondeline, ha4411@inter-hotel.com, www.inter-hotel.fr]
[http://example.org/hotel/hotelLocation73, CHOLET (LE), 44000, 10, Rue Gresset, hotel.cholet@wanadoo.fr, www.hotelcholet-nantes.com]
[http://example.org/hotel/hotelLocation97, NUIT DE RETZ, 44710, Zone de Loisirs - rue du Grand Pré, nuitderetz@orange.fr, www.hotel-nuitderetz.com]
[http://example.org/hotel/hotelLocation1, LE BRETAGNE, 44410, 33 rue du Mes, contact@lebretagne.net, www.lebretagne.net]
[http://example.org/hotel/hotelLocation42, LES TENNIS MEN, 44500, 1, avenue de Lyon, hotel.tennis@wanadoo.fr, www.hotel-lestennis.com]
[http://example.org/hotel/hotelLocation57, HOTEL F1 - NANTES EST PORTE DE SAINTE LUCE, 44303, 345, Route de Sainte Luce, H2260@accor.com, www.hotelf1.com]
[http://example.org/hotel/hotelLocation13, ETAP HOTEL NANTES OUEST, 44220, Route de Vannes, H2527@accor.com,

Conclusions

- SemLav mediators can extend the scope federation of linked data to deep web.
  - Not for intensive querying, but usefull for unfrequent ad-hoc queries
  - Looks like best effort wharehousing on the fly…
- SemLAV can consume more space than traditional approaches, but even considering a small subset of views, it covers an exponential number of rewritings and allow nearly full SPARQL queries
- MaxCov problem formalizes the intuition of maximizing a view subset coverage of rewritings can improve performances when no statistics available.

Future Work

- Room for optimisation:
  - Change the RDF store used for one that allows to load graphs in parallel.
- Solve the MaxCov problem without the restriction of only distinguished variables in views.
- Solve the MaxCovUMR problem, where the memory size available is limited, and we should choose the set of views that fitting in memory covers more rewritings (sort of bagpack).

References

- Montoya, Gabriela, Luis Daniel Ibáñez, Hala Skaf-Molli, Pascal Molli, and Maria-Esther Vidal. "SemLAV: Local-as-View Mediation for SPARQL queries." (2013) http://hal.univ-nantes.fr/hal-00841985/
- (submitted)

Mappings and Mediators

- The resulting GLAV rules can now be used to generate the appropriate RDF for a source in terms of the domain ontology, as in data exchange [3]. Alternatively, the mappings can be interpreted dynamically by a mediator, as in data integration [15]. The mediator would provide a SPARQL endpoint exposing the ontology and executing queries directly over the original sources.

Knoblock, Craig A., et al. "Semi-automatically mapping structured sources into the semantic web." The Semantic Web: Research and Applications. Springer Berlin Heidelberg, 2012. 375-390.