Optimizing Analytical Queries over Semantic Web Sources

by Dilshod Ibragimov
under the supervision of Esteban Zimányi (ULB),
Torben Bach Pedersen (AAU), Katja Hose (AAU).

Outline

- Introduction and Objectives
- A framework for *Exploratory* OLAP
- Performance optimization in a federation of SPARQL endpoints
- Performance optimization on a single endpoint
- Conclusion and Future Works

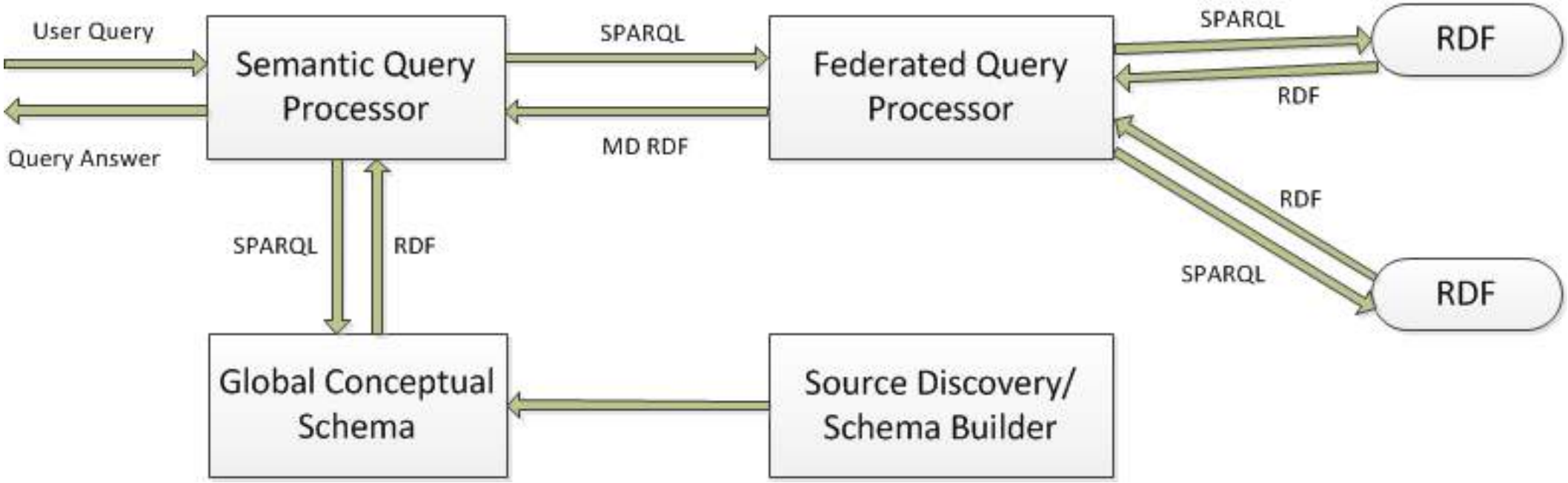
Semantic Web

- W3C promote standards for common data formats and exchange protocols to enable machine-to-machine communication
- Information is given a well-defined meaning
- Meaning is expressed by Resource Description Framework (RDF)
  - subject-predicate-object: <P1> <is called> < Jimmy Wales >
- To enable exploration, data are linked on the Web using RDF
- SPARQL is used for RDF graph pattern matching to extract information or to construct new graphs
  - SELECT, CONSTRUCT, ASK, DESCRIBE
  - SPARQL 1.1 (aggregate functions, subqueries, federated queries…)

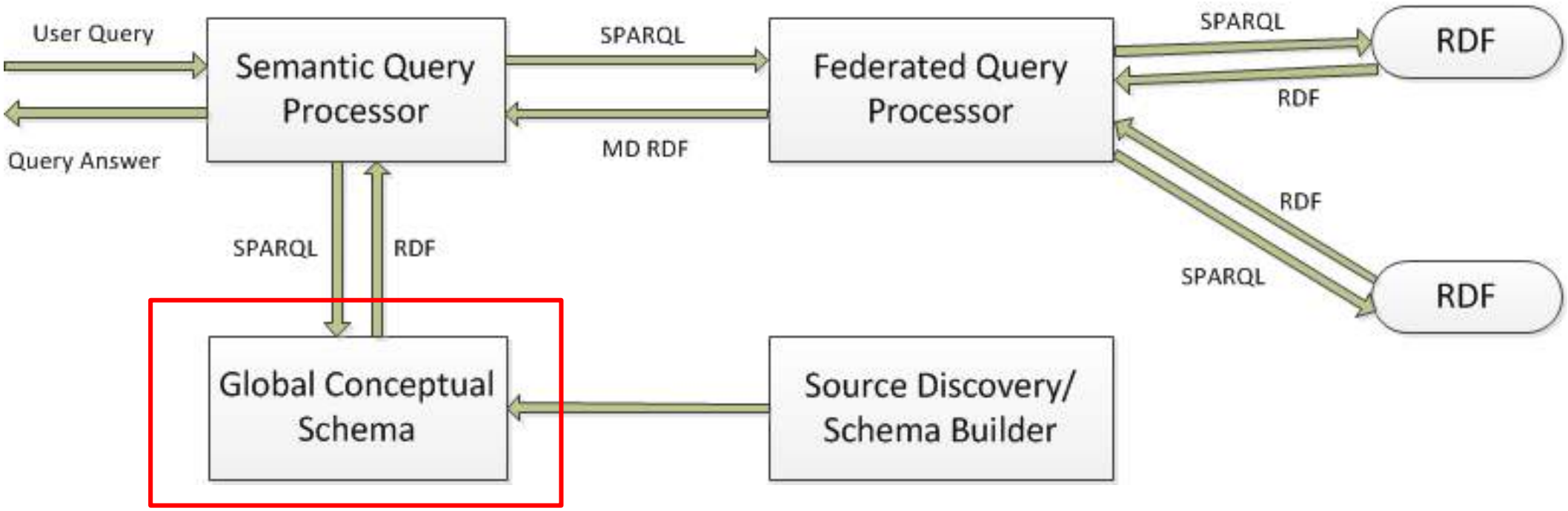
Business Intelligence and Semantic Web

- Benefit of integrating data published on the Web into decision making process
- Traditional (Relational Data Warehouse) approaches fail because:
  - External data source changes impact entire ETL
  - Need to handle graph-like data
  - Entailment (deriving new information using RDF semantics), blank nodes and other RDF specifics are better handled by native RDF systems
- Aim: To investigate RDF data analytics over federated data sources in a context of BI applications considering the RDF features, namely heterogeneity, rich semantics, entailment, ease of publication, etc.

A framework for Exploratory OLAP

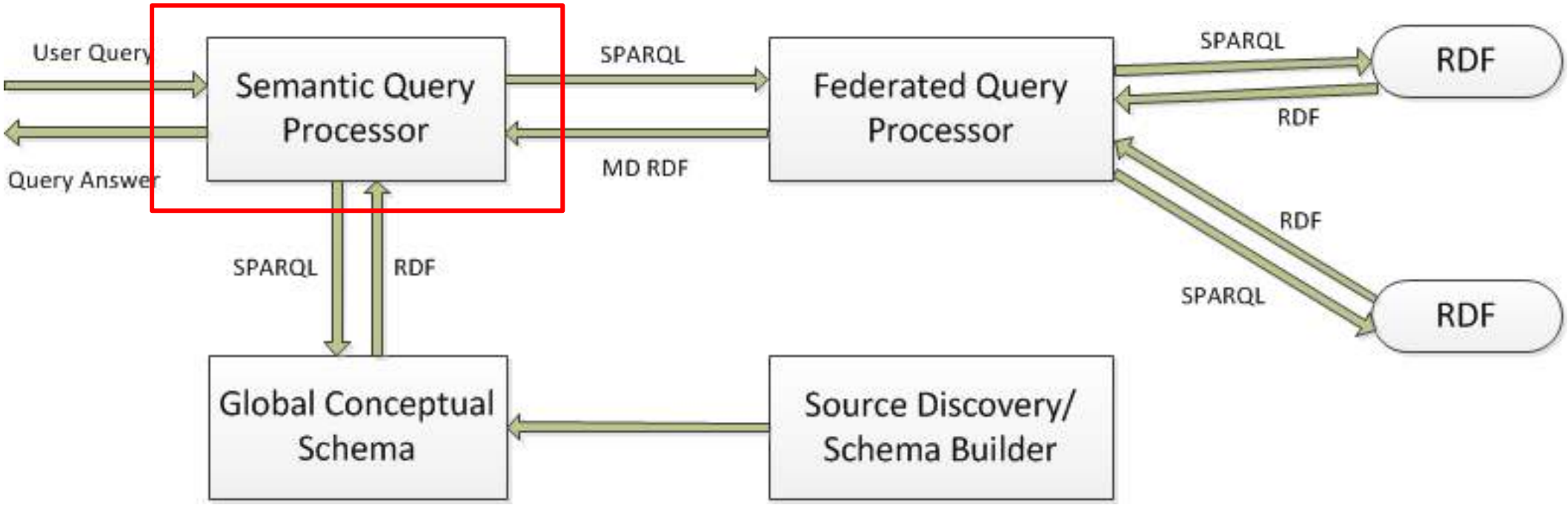


Proposed System



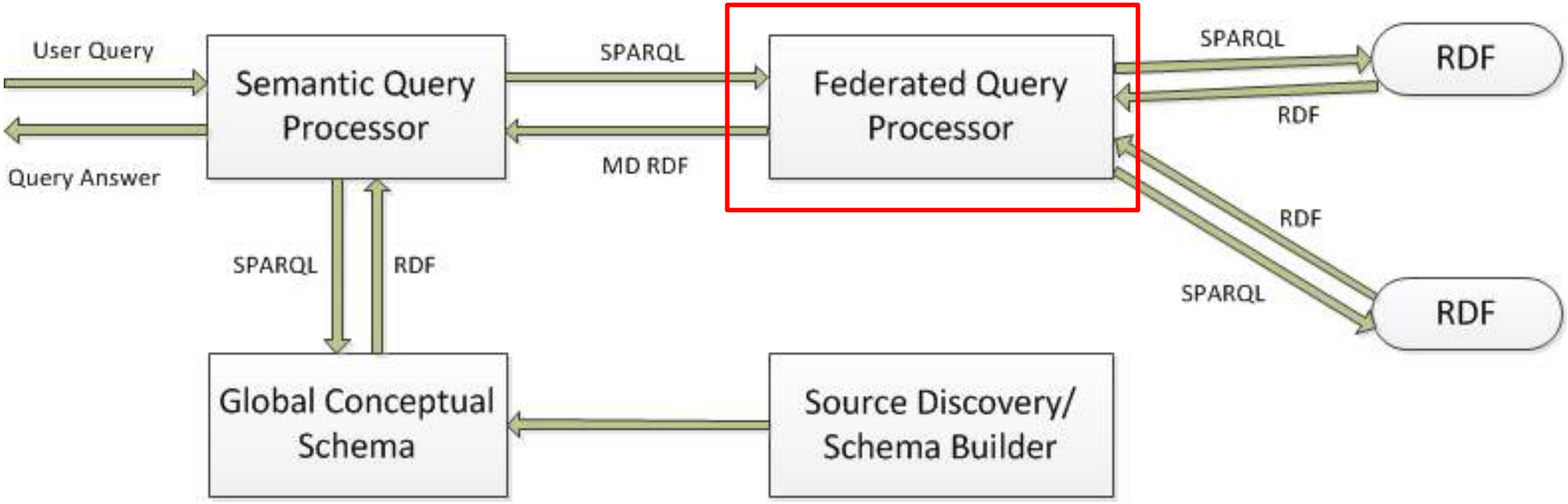
Global Conceptual Schema defines the high-level view of the system - expressed in QB4OLAP, VoID

Proposed System



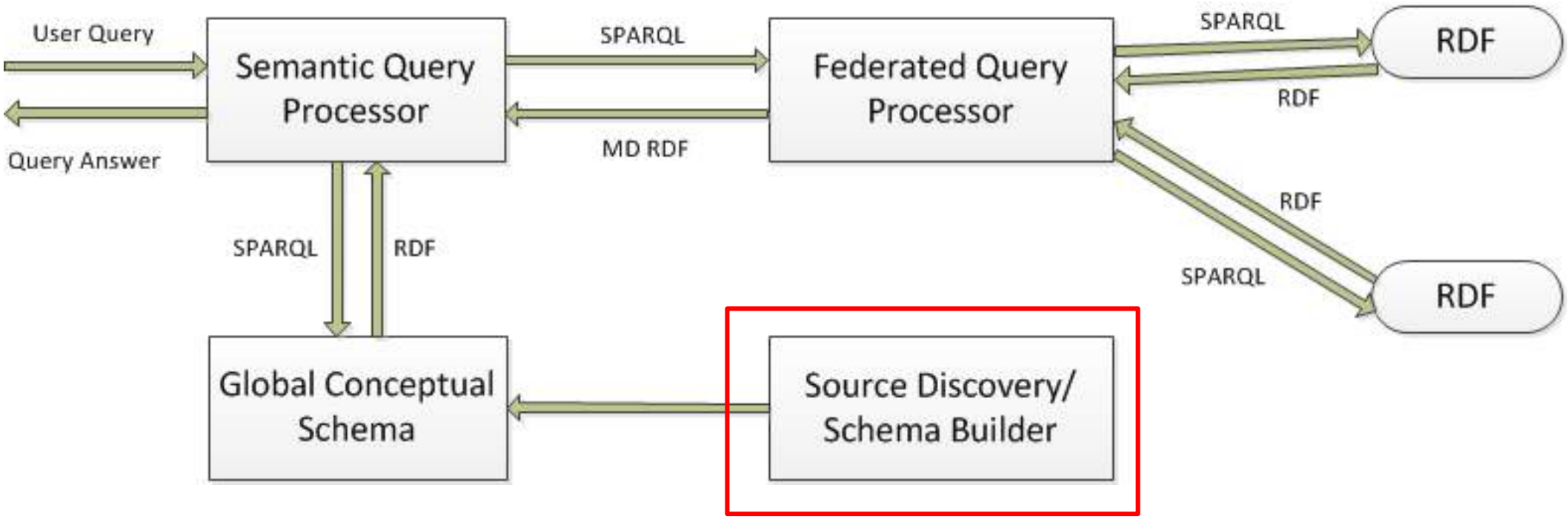
Semantic Query Processor (using the Global Conceptual Schema) converts a user query to a new format and passes it to the Federated Query Processor

Proposed System



Federated Query Processor retrieves data from several federated data sources

Proposed System



Source Discovery/Schema Builder is responsible for the discovery of data sources and construction of the Global Conceptual Schema

Optimizing performance in a federation of SPARQL endpoints

Motivating Example

- Earthquake in the Pacific in March 2011 ⟹ a nuclear accident
- Hourly observation (one year) of radioactivity statistics at 47 prefectures (places represented by URI from GeoNames)
- Interesting analyses - AVG MIN and MAX radioactivity separately for each prefecture
- Virtuoso, Sesame, and Jena timed out for federated queries
- Network traffic analyzer showed that:
  - Virtuoso and Fuseki query GeoNames for each radioactivity observation (more than 400,000 requests)
  - Sesame is trying to download all triples that match the SERVICE query pattern (more than 7.8 million triples)

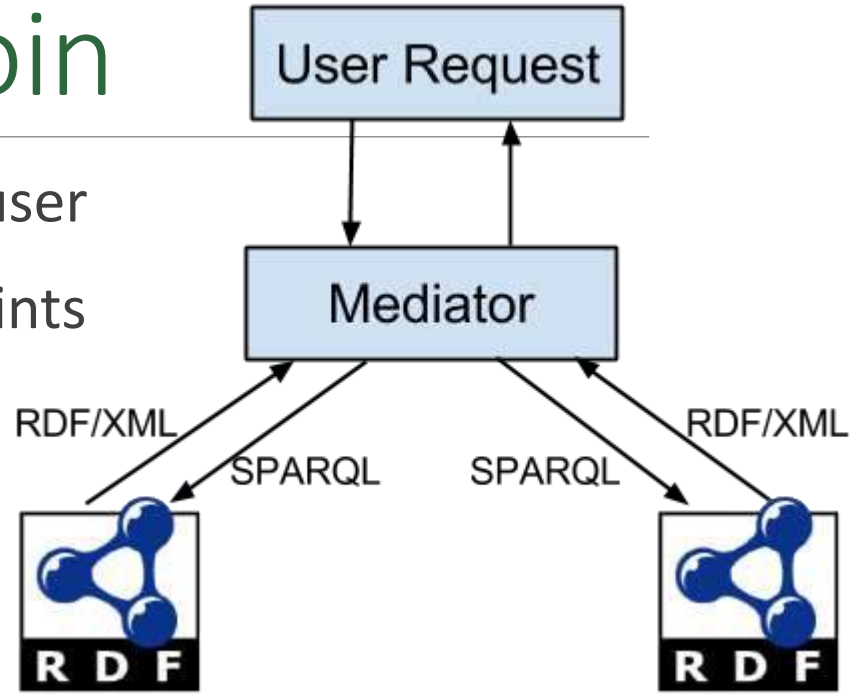
Basic Strategies - Mediator Join

- The mediator/federator receives the query from the user
- The query optimizer sends separate queries to endpoints

and merges the results

- Strong point – parallelization
- Weak point – expensive for large intermediate

results/datasets



Basic Strategies - Semi-Join

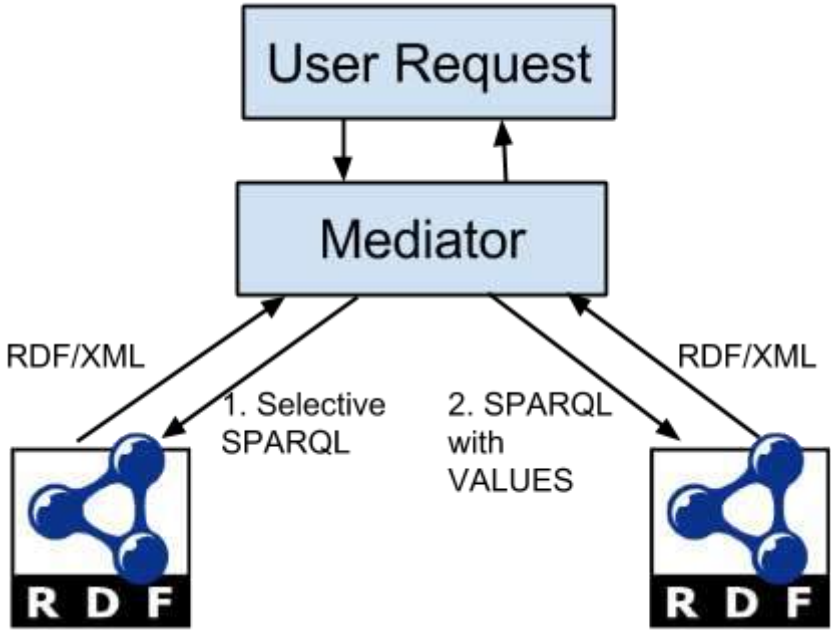
- Main principle is to execute the subquery with the smallest result first and use the retrieved results as bindings for the join variable in other subqueries (SPARQL structure)
- Efficient for highly selective subqueries (with FILTER statement)

```
SELECT ?regName ( AVG (?radioValue ) AS ?average ) WHERE {
    ?s ev:place ?placeID .      ?s ev:time ?time .      ?s rdf:value ?radioValue .
    SERVICE <http://lod2.openlinksw.com/sparql> {
        ?placeID gn: parentFeature ?regionID .      ?regionID gn:name ?regName .
    }
    FILTER(?radioValue < 0.08) .
} GROUP BY ?regName
```

Basic Strategies - Semi-Join (Cont)

```
SELECT ?placeID ?radioVal
WHERE {
  ?s rdf:value ?radioVal ;
  ev:place ?placeID; ev:time ?time.
  FILTER (?radioValue < 0.08) .
}
```

```
SELECT ?placeID ?regName
WHERE { ?placeID gn:parentFeature ?rgID.
  ?rgID gn:name ?regName.
  VALUES (?placeID) {
  <http://sws.geonames.org/1852083/>...}
}
```



- Weak point - VALUES is not yet widely adopted in existing endpoints. SPARQL 1.0 compliant alternatives of UNION (or FILTER) must often be used

Basic Strategies - Partial Aggregation

- If results are grouped by SERVICE query variables, further optimization is possible (motivating query example)

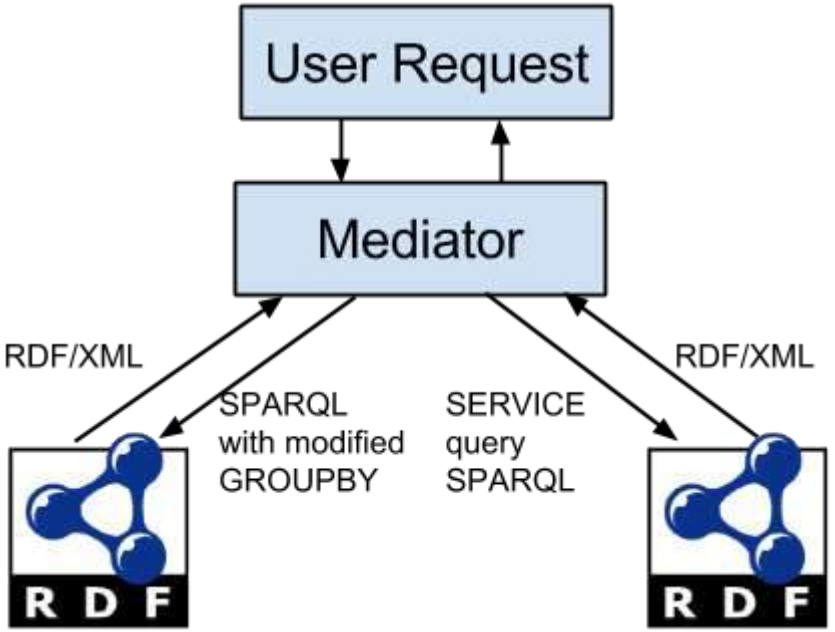
1) First group by the observation place (?placeID)

```
SELECT ?placeID (SUM (?floatRV ) AS ?avgSUM) (COUNT (?floatRV ) AS ?avgCNT )
WHERE {
    ?s ev:place ?placeID .      ?s ev:time ? time .      ?s rdf:value ? radioValue .
    BIND (xsd:float (?radioValue ) as ?floatRV ) .
}
GROUP BY ?placeID
```

Basic Strategies - Partial Aggregation (Cont)

- *Then execute SERVICE query*

```
SELECT ?placeID ?regName WHERE {
    ?placeID gn:parentFeature ?regionID .
    ?regionID gn:name ?regName .
    VALUES (?placeID) {
        (<http://sws.geonames.org/1852083/>) ….
    }
}
```



- *Final step – join the intermediate results and compute the final result* (distributed/algebraic functions)

CODA – Cost-based Optimizer for Distributed Aggregate Queries

- Decomposes the original query into multiple subqueries (query Q_M and SERVICE queries Q_{e1} … Q_{eN})
- Estimates query execution costs for different query execution plans
- Chooses the one with minimum costs

CODA - Costs

- Overall costs C_Q

$$C_Q = C_P + C_C$$

- Communication costs C_C for subquery S_i:

$C_C(S_i) = C_O + c_{S_i} * C_{map}$; C_O - communication establishing overhead , c_{S_i} - result size, and C_{map} - single result transfer cost

- Processing costs

$C_P = c_{agg_i} * C_{AGG}$; c_{agg_i} - number of aggregated observations, C_{AGG} - cost for processing a single observation

CODA - Estimating Constants and Result Sizes

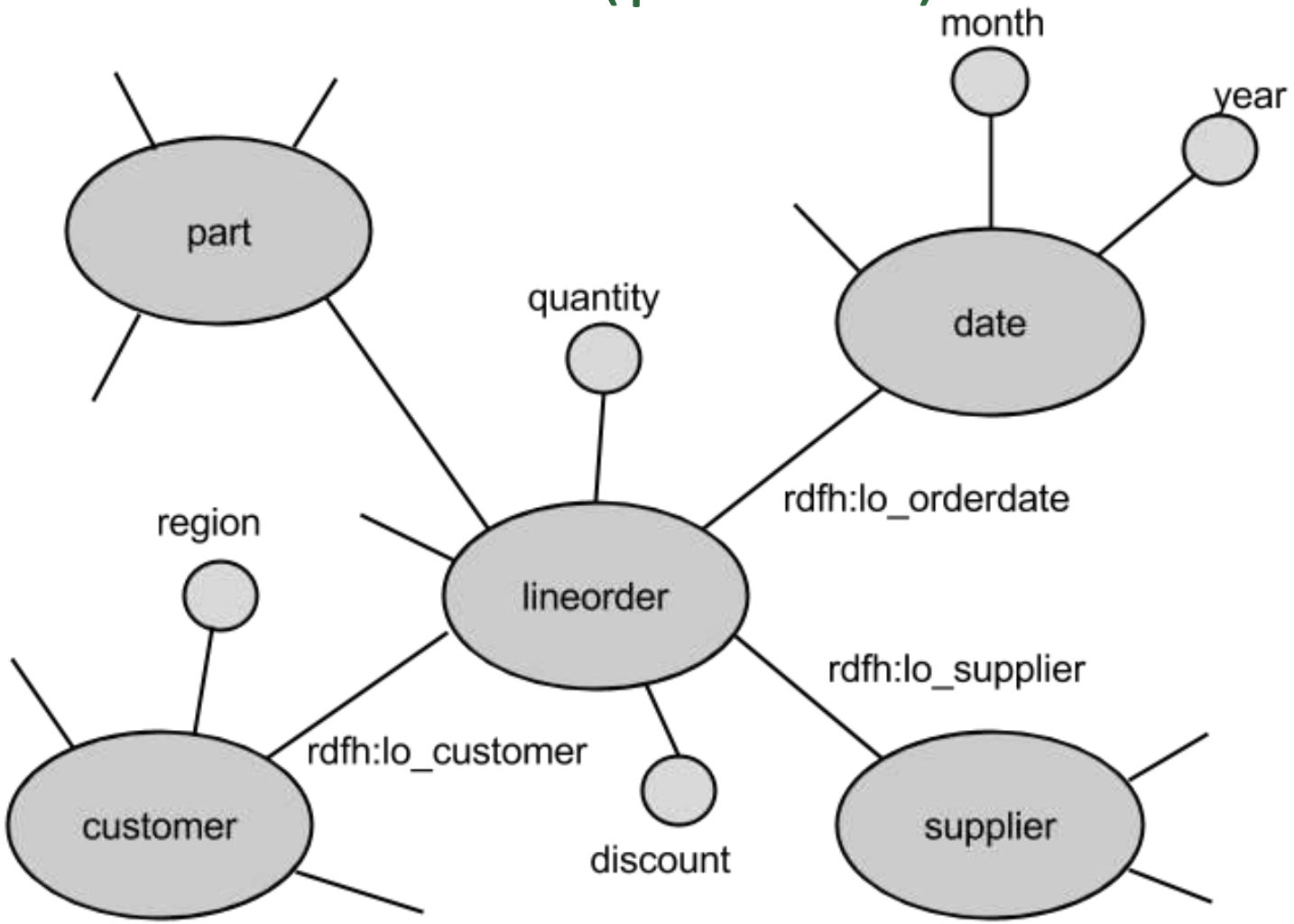
- C_{map}, C_O, C_{AGG} - estimated using probe queries ("SELECT * WHERE { ?s #p ?o . FILTER(?o = #o) } LIMIT #L"; "ASK {}" or "SELECT (1 AS ?v) {}"; "SELECT COUNT(?s) WHERE {?s ?p ?o } GROUP BY ?o"
- Result size estimation - VoID statistics (dataset, property partition, class partition)
- Single patterns - C_{res} for (?s ?p ?o) is given by c_t, (s ?p ?o) estimated as c_t/c_s, (?s ?p o) as c_t/c_o, and (s ?p o) as $c_t/(c_s * c_o)$; FILTER influence estimates
- Joins - estimates depend on shape (star vs path).

Not perfectly accurate but the aim is to find out which execution plan is more efficient (not to predict the execution costs)

Test Case – SSB as RDF

- Star Schema Benchmark converted to RDF (strongly resembling SSB tabular structure)
- We generated data for different scale factors (1 to 5 - 6M to 30M observations, 110,5M to 547,5M triples)
- Different configurations
  - two endpoints (one endpoint containing main observation data and one SERVICE endpoint containing supporting data)
  - three endpoints (two SERVICE endpoints containing supporting data)
  - four endpoints (three SERVICE endpoints containing supporting data)

SSB RDF schema (partial)

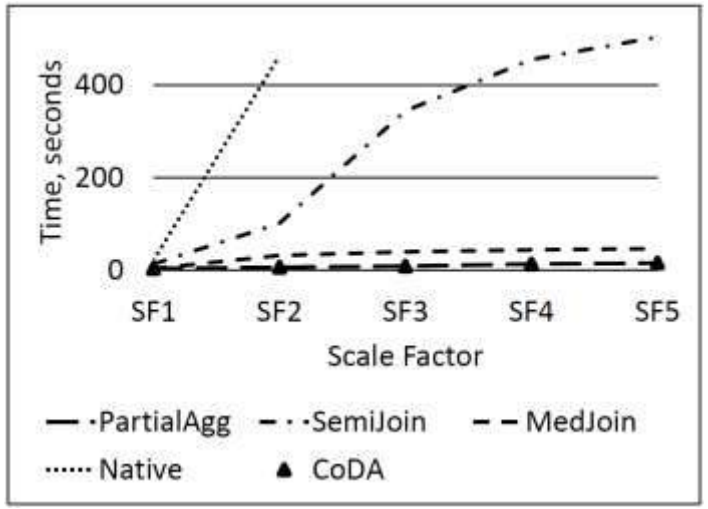


Test Case – SSB Queries

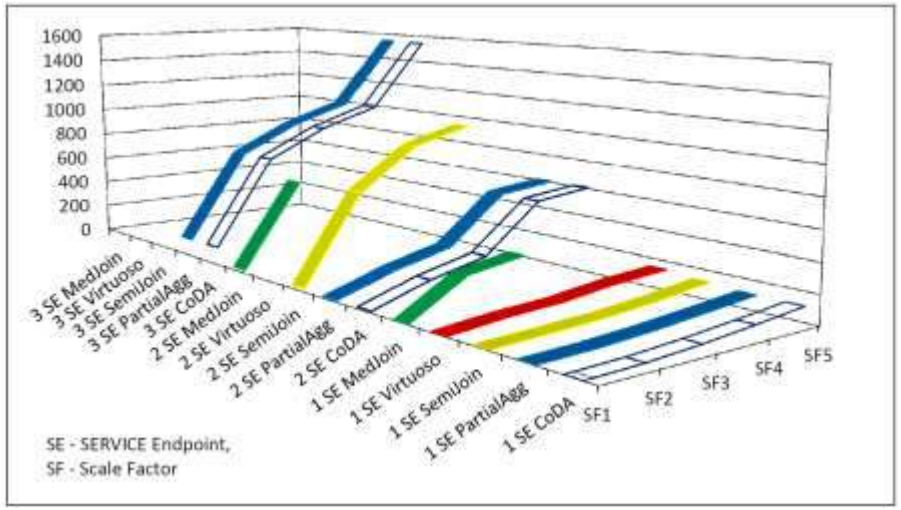
Query Prototypes	Query No	Query Parameters for Various Selectivities
Prototype 1. Amount of revenue increase that would have resulted from eliminating certain company-wide discounts.	Q1.1	Discounts 1, 2, and 3 for quantities less than 25 shipped in 1993.
	Q1.2	Discounts 1, 2, and 3 for quantities less than 25 shipped in 01/1993.
	Q1.3	Discounts 5, 6, and 7 for quantities less than 35 shipped in week 6 of 1993.
Prototype 2. Revenue for some product classes, for suppliers in a certain region, grouped by more restrictive product classes and all years.	Q2.1	Revenue for 'MFGR#12' category, for suppliers in America
	Q2.2	Revenue for brands 'MFGR#2221' to 'MFGR#2228', for suppliers in Asia
	Q2.3	Revenue for brand 'MFGR#2239' for suppliers in Europe
Prototype 3. Revenue for some product classes, for suppliers in a certain region, grouped by more restrictive product classes and all years.	Q3.1	For Asian suppliers and customers in 1992-1997
	Q3.2	For US suppliers and customers in 1992-1997
	Q3.3	For specific UK cities suppliers and customers in 1992-1997
	Q3.4	For specific UK cities suppliers and customers in 12/1997
Prototype 4. Aggregate profit, measured by subtracting revenue from supply cost.	Q4.1	For American suppliers and customers for manufacturers 'MFGR#1' or 'MFGR#2' in 1992
	Q4.2	For American suppliers and customers for manufacturers 'MFGR#1' or 'MFGR#2' in 1997-1998
	Q4.3	For American customers and US suppliers for category 'MFGR#14' in 1997-1998

Test Case – Results

	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3
Scale Factor 1													
Virtuoso	T/O	T/O	760	500,3	107,8	21,3	215,8	21,2	1,4	1,4	863	969	7,3
SemiJoin	**1,3**	**0,2**	**0,1**	12,7	13,5	12,6	**14,1**	11,0	6,5	**0,2**	4,8	**8,1**	4,5
PartialAgg	1,6	1,4	0,8	**9,4**	**4,5**	3	17,5	**2,8**	**0,5**	0,3	**4**	18,5	**1,0**
MedJoin	249,5	213,4	82,9	11	5,2	**2,9**	98,9	3,4	0,8	0,3	26,4	32	1,1
CoDA	**1,3**	**0,2**	**0,1**	**9,4**	**4,5**	**2,9**	**14,1**	**2,8**	**0,5**	**0,2**	**4**	**8,1**	**1,0**



One SERVICE Endpoint, Q2.3



One to Three SERVICE Endpoints, Q4.3

Different Federation Setup

- CoDA did not consider Horizontal Federations
- Related data are often published at multiple endpoints
  - Statistics
  - Weather data
  - Census data
- Challenges for processing such data
  - Analyzing data *across* endpoints
  - Data source heterogeneity
  - Deriving new information from data using RDF semantics

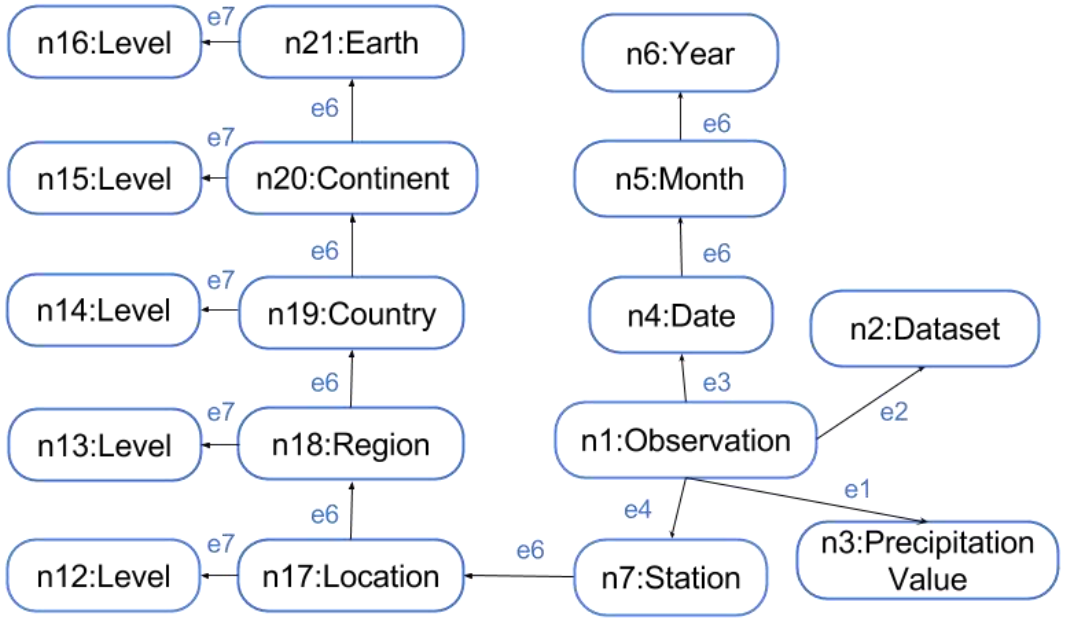
LITE – OLAP-style AnalytIcs in a FederaTion of SPARQL Endpoints

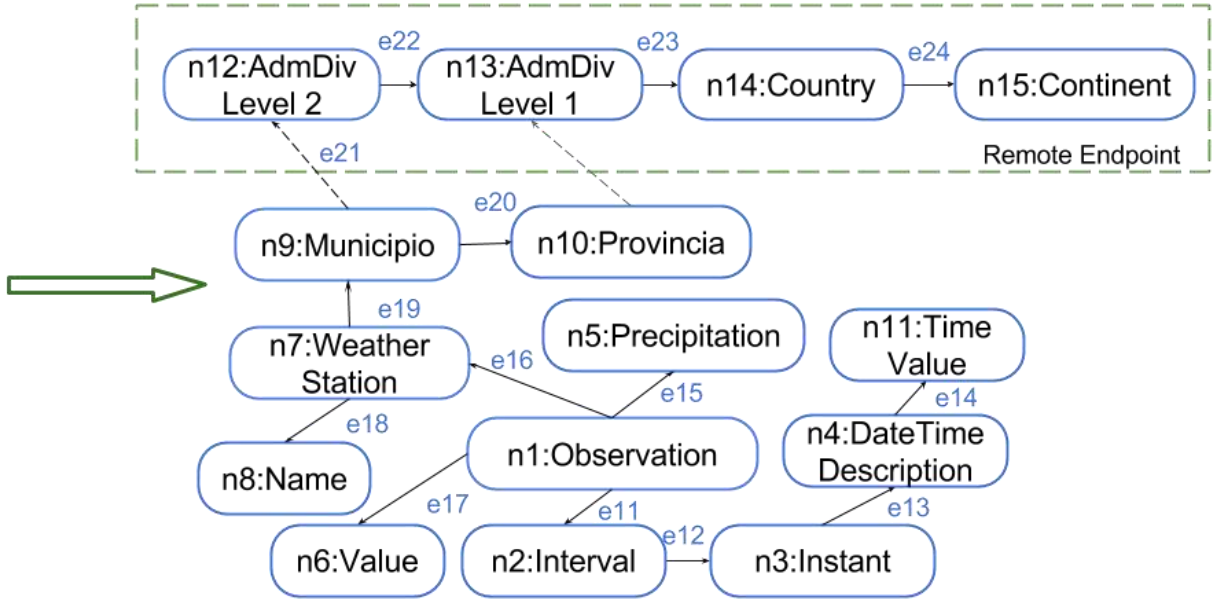
- A native RDF/SPARQL-based approach for efficient support of analytical queries
- An extended vocabulary for specifying the mapping between the multidimensional mediated schema (global) and the local schemas of the sources
- An algorithm for rewriting SPARQL queries in a federation of endpoints that takes into account hierarchical information encoded in RDFS

Key Concepts

- Representing schemas as graphs:
  - Local schemas are represented as graphs that ***highlights*** the structure of the data
  - Global schema can be built using multidimensional vocabularies or more general approaches
- Extending SPIN vocabulary for mapping
  - Adapt SPIN to represent RDF schema graphs
  - New terms: *pattern*, *schemaMatch*, *sameConcept*
- Mapping schema fragments (subgraphs that reflect core concepts in schemas)

Global and Local Schemas Mapping





Global Schema built based on Local Schemas

Precipitation Data from Spain (Example)

Query Rewriting Algorithm

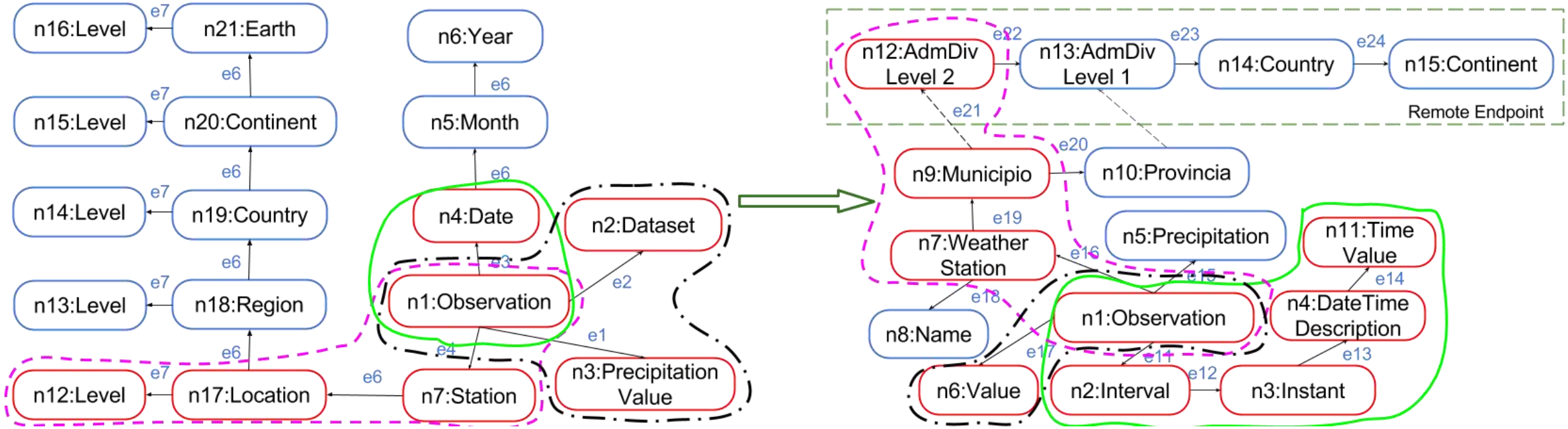
1. Identify Global Schema subgraph that corresponds to user query (match graph pattern of user query to schema graph)
2. Identify subset of fragments to construct subgraph from Step 1.
3. Using mapping - find counterparts in Local schemas
4. Build a schema of local data source that corresponds to the user query
5. For result variables in fragments – replace nodes in global query with same concept node in local query
6. Remaining nodes are replaced with newly generated variables
7. Rewrite aggregate function (if needed)

Query Rewriting Algorithm

```
SELECT ?loc (SUM(?prec) as ?totalPrec)
WHERE {
  ?obs rdf:type qb:Observation .
  ?obs qb:dataSet ex:Precipitation .
  ?obs ex:precipitationLevel ?prec .
  ?obs ex:station ?station .
  ?station qb4o:inLevel ex:Station .
  ?station skos:broader ?loc .
  ?loc qb4o:inLevel ex:Location .
  ?obs sen:samplingTime ?date .
  FILTER(?date=`2014-02-07'^^xsd:dateTime)
} GROUP BY ?loc
```

```
SELECT ?loc (SUM(?prec) as ?totalPrec)
WHERE{ ?node1 a ae:Observation .
  ?node1 ae:valueOfObservedData ?prec .
  ?node1 ssn:observedBy/ssn:locatedIn ?node9 .
  ?node9 owl:sameAs ?node12 .
  SERVICE <http://lod2.openlinksw.com/sparql> {
    ?node12 gn:name ?loc . }
  ?node1 tm:inInterval/tm:hasBeginning ?node3 .
  ?node3 tm:inDateTime/tm:inXSDDateTime ?date .
  FILTER(?date=`2014-02-07'^^xsd:dateTime)
} GROUP BY ?loc
```

Query Rewriting Algorithm



Global and Local Optimization

- Rule-based global optimization
  - Rule 1 – use internal hierarchies to rollup to a level that corresponds to some level of the external hierarchy
  - Rule 2 – restriction on the values for hierarchy levels (not to retrieve irrelevant data)
  - Rule 3 – reduce the amount of transferred data (local endpoint to mediator) by rolling-up on local endpoints
- Cost-based local optimization (CoDA)

Accounting for Implicit Hierarchies

- RDFS defines semantic constraints between classes and properties
- Standard properties
  - rdfs:subClassOf,
  - rdfs:subPropertyOf,
  - rdfs:domain,
  - rdfs:range
- Accounting for hierarchical information present in the data using rdfs:subClassOf or rdfs:subPropertyOf
  - Add new triple pattern (UNION) for hierarchical relations between predicates in RDFS Schema (rdfs:subPropertyOf)
  - Add new triple pattern (UNION) with newly generated variable replacing the predicate for hierarchical relations between classes (rdfs:subClassOf)

Evaluation

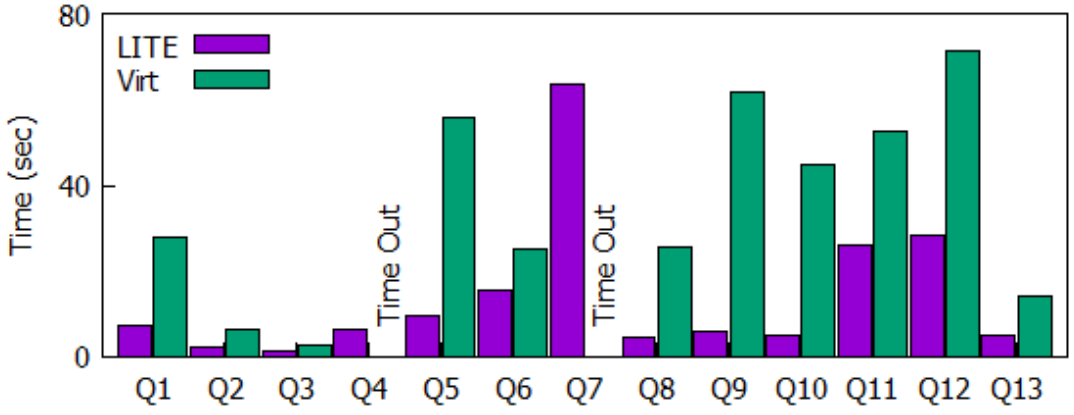
- Benchmark data generators – no option to generate required data
- Extended SSB (linking to existing cities). Generated data for three scale factors (mln triples):

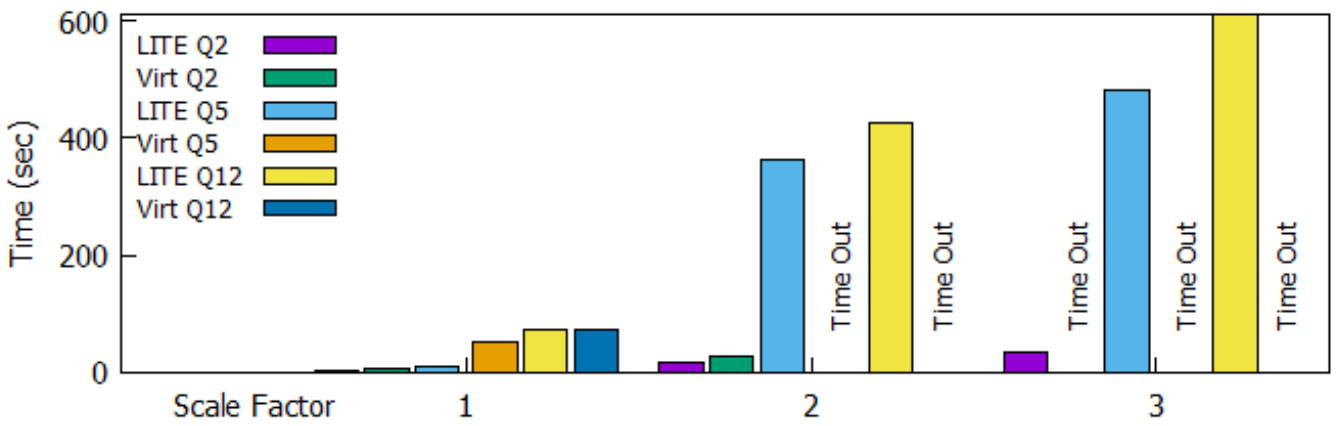
	SF1	SF2	SF3
eSSB	120M	240M	360M

- Linked each city/country to counterparts from GeoNames using owl:sameAs predicate
- 64-bit Ubuntu 14.04 LTS with CPU Intel(R) Core(TM)i7-950, 24GB RAM, 600GB HDD; Virtuoso v07.10.3207; dotNetRDF
- Executed 5 times after a single warm-up run; average runtime reported

Performance over Different Scale Factors

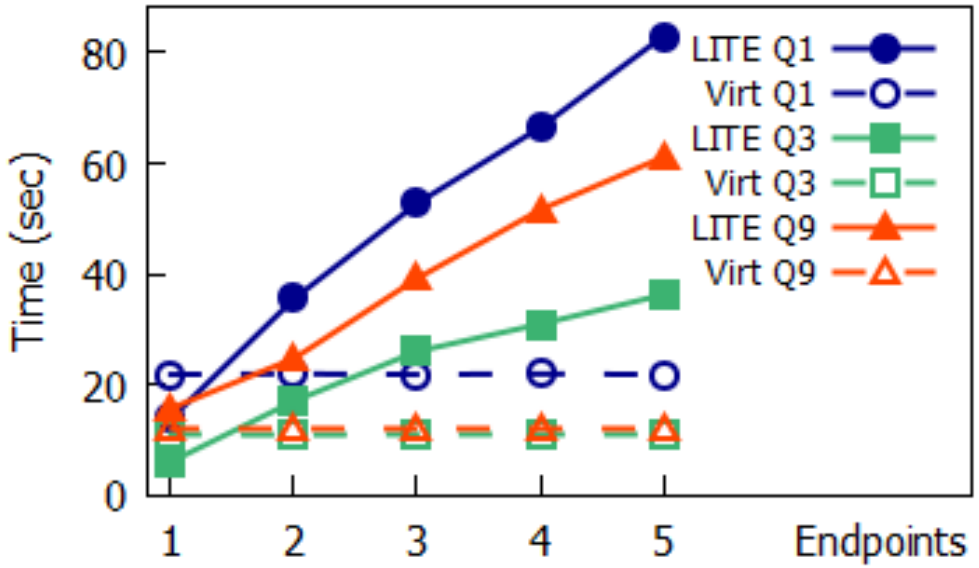
- For Scale Factor 1 – on average 4 times faster than Virtuoso
- For Virtuoso queries time out due to the increased amount of data to process



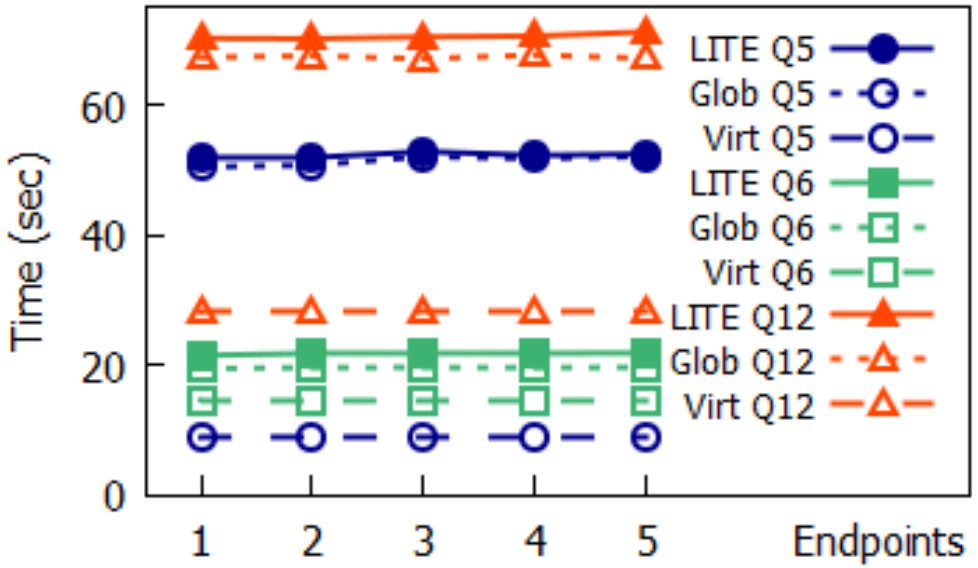


Comparing Performance over the Number of Endpoints

- Fig.1 – Virtuoso needs to query more endpoints while LITE's execution is in parallel
- Fig.2 – FILTER expression apply restrictions on the continent to limit the results to one endpoint only – thus no growth



Queries 1, 3,and 9



Queries 5, 6, and 12

Optimizing performance on a single endpoint

Issues with Standalone Endpoints

- Analytical queries are responsible for high load in SPARQL endpoints – they often time out http://cofog01.data.scotland.gov.uk/sparql

```
PREFIX gol: <http://cofog01.data.scotland.gov.uk/def/golspie/>
PREFIX qb:  <http://purl.org/linked-data/cube#>
PREFIX util: <http://cofog01.data.scotland.gov.uk/def/golspie/>
PREFIX org: <http://www.w3.org/ns/org#>
PREFIX vc: <http://cofog01.data.scotland.gov.uk/id/vcard/>

SELECT ?tm ?plc (SUM(?v) as ?value) WHERE {
  ?slc gol:refBuilding ?bld ; gol:reportDateTime ?tm ; qb:observation ?ob .
  ?ob gol:utilityConsumption ?v ; gol:refUtility util:electricity .
  ?bld org:siteAddress/vc:adr/vc:locality ?plc .
} GROUP BY ?tm ?plc
```

- Need special techniques to speed up execution, like materialized views

MARVEL – MAterialized RDF Views with Entailment and incompLetness

- Speed up the analytical SPARQL queries by using materialized views
  - A SPARQL syntax for defining aggregate views
  - Cost model to select RDF views for materialization
  - An algorithm for rewriting SPARQL aggregate queries using these views

Creating Materialized Views

- Aggregation – summarized data in tabular format
- Need to create new triples to store it as RDF
- Use *CONSTRUCT* query
- New IRI for generated triples - grouping combination - creating unique IRI

```
CONSTRUCT { ?id pred1 ?a .   ?id pred2 ?b .   ?id pred3 ?sum . }
WHERE {
   SELECT ?id ?a ?b (SUM(?c) AS ?sum)
   WHERE{    ?a ?p1 ?b .    ?a ?p2 ?c .
       BIND(IRI( 'http://ex.org/id#', CONCAT(STR(?a), STR(?b)) ) AS ?id).
   } GROUP BY ?a ?b ?id
}
```

Cost Model – Calculating View Benefit

- Goal - select *N* views with highest benefit
- Benefit of view *w* relative to *v* :

  $B_{w,v} = Cost(v) - Size(w)$ if $Cost(v) > Size(w)$

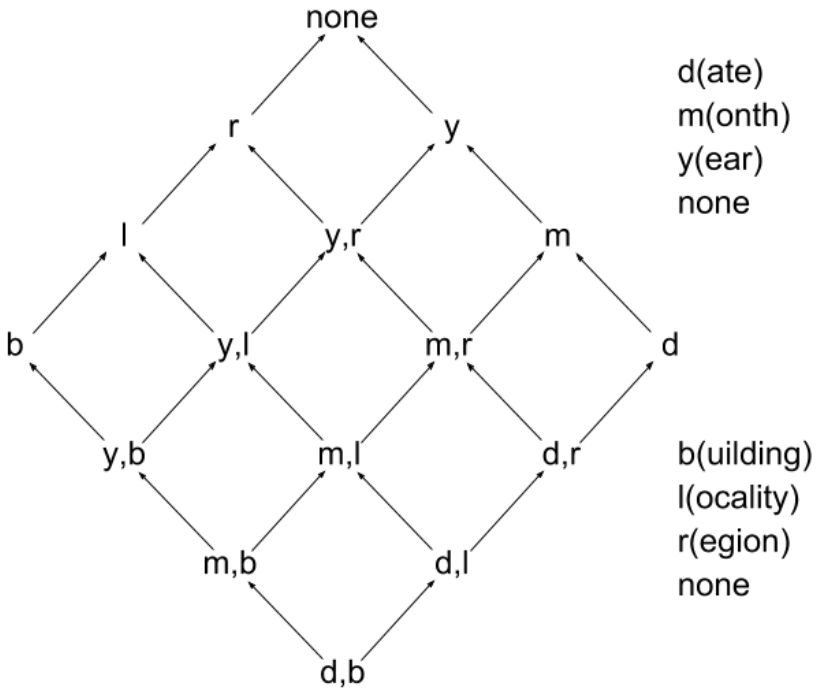
  $B_{w,v} = 0$ otherwise
- Number of triples in an observation:

  $(n + m)$, n - # of dimensions, m - # of measures
- Size of a view *w:*

  $Size(w) = (n + m) * K$, K - # of observations
- If *v* is computed from *w*:

  Cost(v) = Size(w)
- Total benefit of view *w*:

  $B_w = \sum B_{w,v_i}$



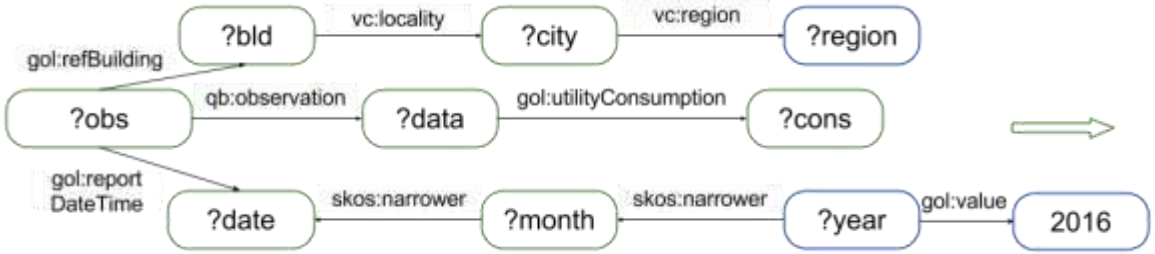
- Incomplete views, complex and indirect hierarchies
- View materialization - deriving implicit information (complete answer)

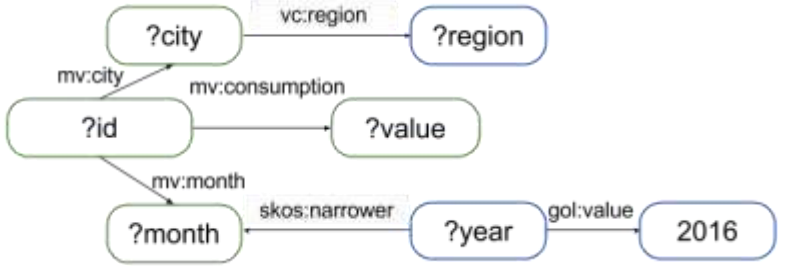
Query Rewriting Algorithm

- Replacing user query over data using view

```
SELECT ?year ?region (SUM(?value) as ?aggValue)
FROM <http://data.scotland.gov.uk>
FROM NAMED <http://data.scotland.gov.uk/matview1>
WHERE {
  GRAPH <http://data.scotland.gov.uk/matview1> {
    ?id mv:month ?month; mv:city ?vCity;
    mv:consumption ?value.
  }
  ?month skos:narrower ?year . ?city vc:region ?region .
  ?year gol:value ?yearV .
  FILTER(?yearV = 2016)
}
GROUP BY ?year ?region
```

User Query





Rewritten Query

Evaluation

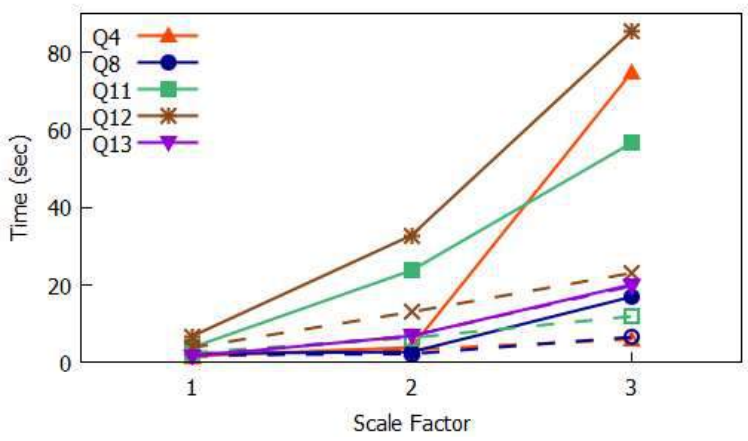
- Benchmark data generators – no option to generate required data
- Adapted data generators for 2 different datasets. Generated data for three scale factors (mln triples):

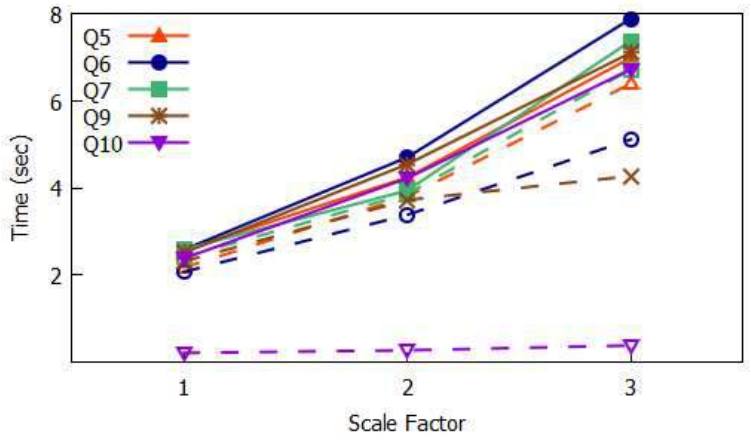
	SF1	SF2	SF3
LUBM	5M	15M	33M
SSB	104M	191M	277M

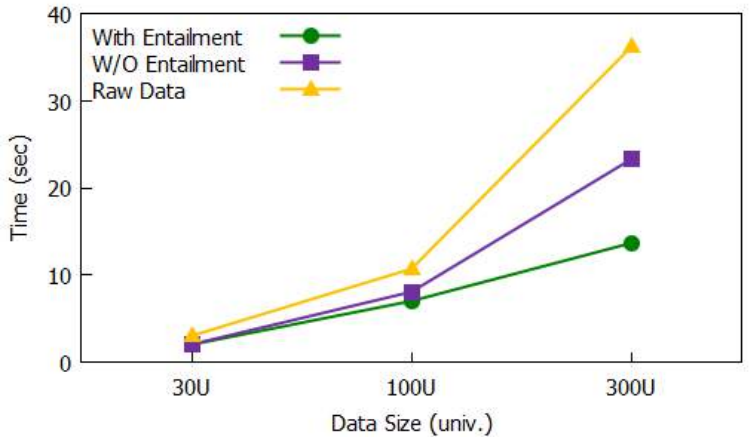
- 64-bit Ubuntu 14.04 LTS with CPU Intel(R) Core(TM)i7-950, 24GB RAM, 600GB HDD; Virtuoso v07.10.3207; dotNetRDF
- Executed 5 times after a single warm-up run; average runtime reported

Comparing Queries in Adapted Datasets

- With MARVEL, execution of queries is on average 5 times faster (up to 18 times for some queries)



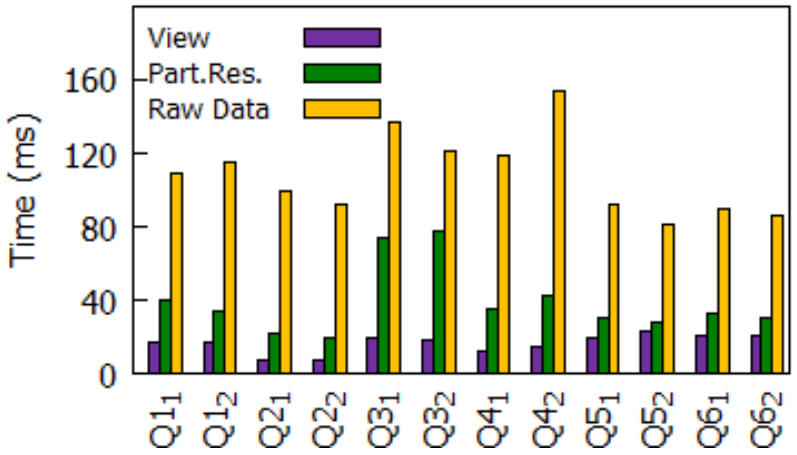




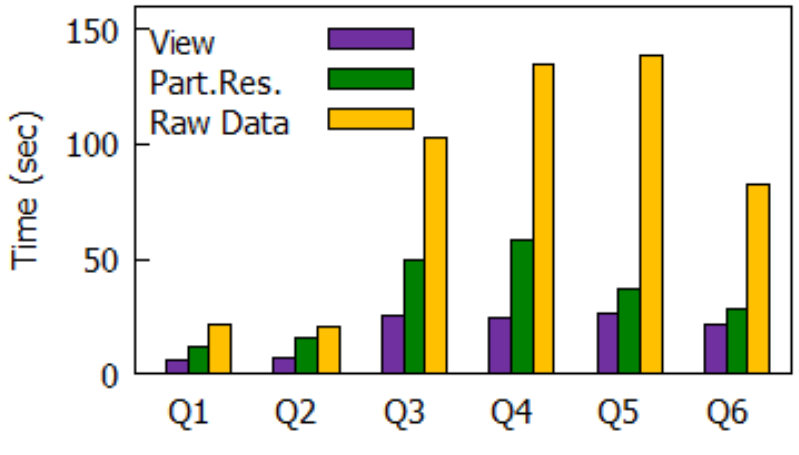
- MARVEL is 1.7 times faster for views with entailment

Comparing with Partial Materialization Approach

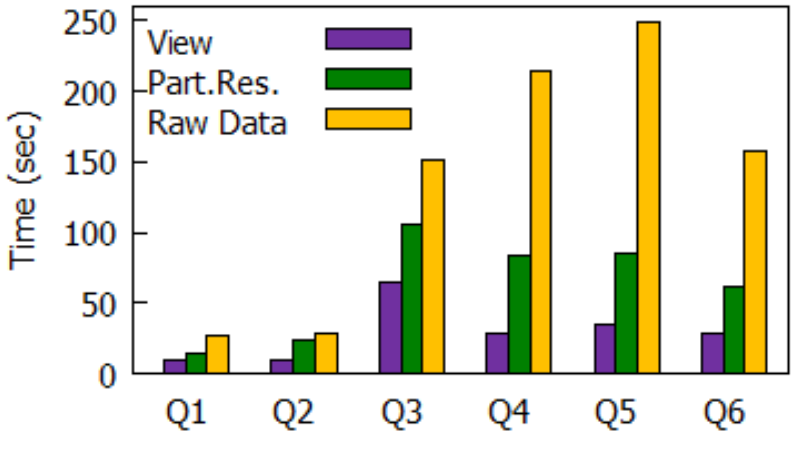
- Partial results of user queries to answer subsequent queries
- MARVEL on average more than twice as fast as partial result materialization



Slice



Dice



Roll-up

Conclusion

- Virtual integration of several endpoints into a federated system to perform analytical queries
  - RDF-based schema mappings
  - Algorithms for rewriting and optimization in consideration of RDF specifics
- Efficiently processing aggregate queries in a federation of SPARQL endpoints
  - Cost models, strategies and algorithms for query processing optimization
  - 7x times performance gain for analytical queries in a federation
- Optimizing performance of standalone endpoints in a federation using materialized views
  - Cost models and algorithm for rewriting aggregate queries using views
  - 3-11 times speedup for aggregate queries on a single endpoint

Future Work

- In federated query optimization:
  - Using more complex statistics for better cardinality estimation, optimizing more complex queries (optional patterns, complex aggregation functions and complex subqueries)
  - A cost model for optimizing queries on a global level
  - Accounting for data overlap in sources
- For queries on standalone endpoints:
  - Optimizing cost-model for particular endpoints
  - Refining view selection based on logs
- Interesting directions:
  - Considering entailment rules in federated settings (inferred data in one endpoint affects results of query answering in others)
  - Answering analytical federated queries using distributed materialized views
  - Integration of data from heterogeneous data management systems

Publications

- Chapter 2
  - D. Ibragimov, K. Hose, T. B. Pedersen, E. Zimányi. Towards Exploratory OLAP Over Linked Open Data - A Case Study. In *BIRTE 2014*, pages 114–132
- Chapter 3
  - D. Ibragimov, K. Hose, T. B. Pedersen, E. Zimányi. Processing Aggregate Queries in a Federation of SPARQL Endpoints. In *ESWC 2015*, pages 269–285
- Chapter 4
  - D. Ibragimov, K. Hose, T. B. Pedersen, E. Zimányi. Efficient Support of Analytical SPARQL Queries in Federated Systems. *In preparation for a conference submission*
- Chapter 5
  - D. Ibragimov, K. Hose, T. B. Pedersen, E. Zimányi. Optimizing Aggregate SPARQL Queries Using Materialized RDF Views. In *ISWC 2016*, pages 341–359(1)

Realted Work

- [71] A. Halevy, “Answering queries using views: A survey,” VLDB Journal, vol. 10, no. 4, pp. 270–294, 2001
- [68] D. Colazzo, F. Goasdoué, I. Manolescu, and A. Roatis, “RDF analytics: lenses over semantic graphs,” in WWW, 2014, pp. 467–478.
- [25] L. Etcheverry, A. Vaisman, and E. Zimányi, “Modeling and querying data warehouses on the semantic web using QB4OLAP,” in DaWaK, 2014, pp. 45–56.
- [26] B. Kämpgen and A. Harth, “No size fits all - running the star schema benchmark with SPARQL and RDF aggregate views,” in ESWC, 2013, pp. 290–304.
- [22] A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J. Mazón, F. Naumann, T. B. Pedersen, S. Rizzi, J. Trujillo, P. Vassiliadis, and G. Vossen, “Fusion cubes: Towards self-business intelligence,” IJDWM, vol. 9, no. 2, pp. 66–88, 2013.
- [5] A. Abelló, O. Romero, T. B. Pedersen, R. Berlanga, V. Nebot, M. J. Aramburu, and A. Simitsis, “Using semantic web technologies for exploratory OLAP: A survey,” TKDE, vol. 99, 2014.
- [28] V. Nebot and R. Berlanga, “Building data warehouses with semantic web data,” Decision Support Systems, vol. 52, no. 4, pp. 853–868, 2012.