Efficient Processing of Semantically Represented Sensor Data

Farah Karim[1], Maria-Esther Vidal[2] and Sören Auer[1,2]
[1]*Enterprise Information Systems (EIS), University of Bonn, Germany*
[2]*Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS), Germany*

Keywords: Sensor, Linked Data, Data Factorization, Query Optimization and Execution.

Abstract: Large collections of sensor data are semantically described using ontologies, e.g., the Semantic Sensor Network (SSN) ontology. Semantic sensor data are RDF descriptions of sensor observations from related sampling frames or sensors at multiple points in time, e.g., climate sensor data. Sensor values can be repeated in a sampling frame, e.g., a particular temperature value can be repeated several times, resulting in a considerable increase in data volume. We devise a factorized compact representation of semantic sensor data using linked data technologies to reduce repetition of same sensor values, and propose algorithms to generate collections of factorized semantic sensor data that can be managed by existing RDF triple stores. We empirically study the effectiveness of the proposed factorized representation of semantic sensor data. We show that the size of semantic sensor data is reduced by more than 50% on average without loss of information. Further, we have evaluated the impact of this factorized representation of semantic sensor data on query execution. Results suggest that query optimizers can be empowered with semantics from factorized representations to generate query plans that effectively speed up query execution time on factorized semantic sensor data.

1 INTRODUCTION

The increasing use of semantic technologies in industry, health care, and other domains has brought attention to scalability and performance improvements. One crucial dimension of semantic technologies is the semantic enrichment and integration of sensor data. Repetition of sensor measurement values of real-world phenomena causes extraordinary expansion of data volumes. Particularly, duplicated measurement values impact on the size of datasets when sensor data is represented using RDF and ontologies, e.g., the Semantic Sensor Network (SSN) ontology.

RDF data compression is a possible solution for managing large volumes of RDF data efficiently. However, compressed RDF data can only be processed and queried by RDF query engines customized for a specific compression tool, e.g., *HDT* (Fernández et al., 2013) binary serialization format for storing and exchange of RDF data, and the RDF-3X (Neumann and Weikum, 2010) compression schema for RDF. Further, data semantics or ontological knowledge are not utilized by these compression techniques.

Inspired by existing work on factorized databases (Bakibayev et al., 2013), we propose the *Factorizing Semantic Sensor Data (FSSD)* approach to factorize semantic sensor data. FSSD exploits ontological knowledge to: *i*) generate a compact representation of sensor observations where repeated measurement values are reduced, and *ii*) rewrite and execute queries over factorized data.

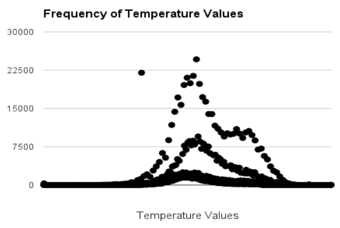
We empirically evaluate the effectiveness of the FSSD framework on a collection of semantic sensor data of different sizes. These datasets contain observations of different climate phenomena during the hurricane and blizzard seasons in the United States in the years 2003, 2004, and 2005. The experimental results show that the proposed factorization method reduces the amount of semantic sensor data (i.e., number of RDF triples) by more than 50% without loss of information. Moreover, observed results show that query processing over factorized sensor data is boosted by up to two orders of magnitude.

This paper comprises seven additional sections. Using a real-world example, the need for efficient representation of semantic sensor data is motivated in 2. Preliminary definitions are presented in 3. We then define our approach in 4 and 5. Results of our empirical evaluation are reported in 6. Existing approaches are reviewed in 7. We conclude and present an outlook to future work in 8.

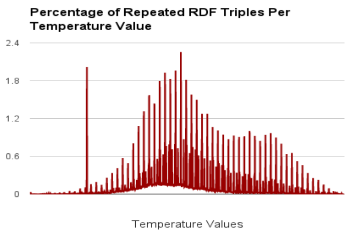
Karim, F., Vidal, M-E. and Auer, S.
Efficient Processing of Semantically Represented Sensor Data.
DOI: 10.5220/0006287002520259
In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST 2017)*, pages 252-259
ISBN: 978-989-758-246-2

```
SELECT ?val, COUNT(?val) as ?vCount
WHERE {
  ?obs  a          :TempObservation;
       :result     ?res .
  ?res :floatValue ?val .
} GROUP BY ?val HAVING ?vCount > 9
```

(a) **Exemplar SPARQL query** to calculate the frequency distribution of temperature values repeated more than 9 times in observations in 2003 MesoWest dataset.



(b) **Frequency** Distribution of Temperature Values of Mesowest dataset.



(c) **Percentage** of Repeated Values.

Figure 1: **Motivating Example**: (a) SPARQL query against year 2003 MesoWest dataset with temperature observations. Prefixes are used as in https://www.w3.org/wiki/SRBench; (b) Frequency Distribution of temperature values from year 2003 Mesowest Linked Observation Data repeated more than 9 times observations; (c) Percentage of RDF triples per repeated temperature values with respect to Linked Observations of temperature from year 2003 Mesowest Data.

2 MOTIVATING EXAMPLE

The MesoWest RDF datasets[1] comprise sensor linked data describing hurricane and blizzard observations in the United States. Observations include measurements of different climate phenomena, e.g., temperature, precipitation, wind speed, and humidity. The SSN ontology is utilized to semantically describe these observations. Together these datasets contain almost two billion RDF triples comprehensively describing major storms in the United States since 2002.

The RDF dataset on the storm season in the year 2003 comprises 12,011,466 RDF triples about temperature, and 1,193,345 observations. The SPARQL query in Figure 1a calculates the frequency distribution of temperature values that are repeated more than nine times in the temperature observations. Figure 1b shows the results produced by the evaluation of this query. The results reveal that 1,191,218 observations out of 1,193,345 meet this condition. Additionally, temperature values are repeated on average more than 1,100 times, and the temperature value `32` is the *mode*, i.e., `32` is the most repeated value and is associated with 24,632 observations. Also, each observation is described using 11 RDF triples; Figure 1c illustrates the percentage RDF triples repeated per temperature value. As can be observed, some repeated temperature values are associated with up to 2.5% of the total number of temperature related RDF triples in the whole dataset. Similar frequency distributions can be observed for other climate phenomena, and corroborate the *natural intuition* that the number of *phenomenon distinct values* is much smaller than the number of *observations*.

We exploit these characteristics of semantic sensor data, and propose a compact representation where RDF triples of repeated measurement values are *factorized* from the observations and included in the dataset only once. Unlike other compression techniques, queries can be directly executed against a factorized representation of semantic sensor data. Further, semantics encoded in factorized representations can be utilized to guide the query optimizer to generate query plans able to speed up query processing.

3 PRELIMINARIES

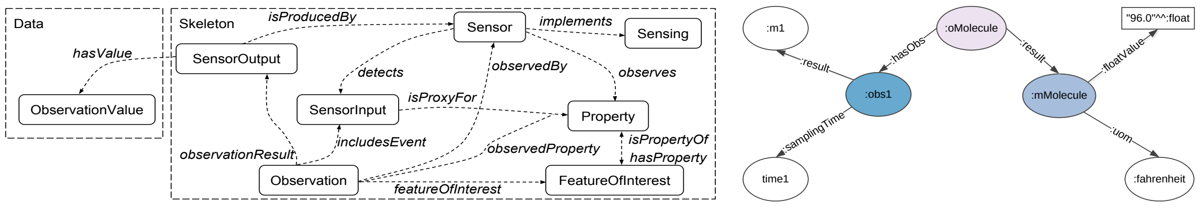
The Semantic Sensor Network (SSN) Ontology (Compton et al., 2012) developed by the W3C Semantic Sensor Network Incubator Group[2], has been extensively used to semantically describe sensor data (Henson et al., 2009; Gao et al., 2014; Ali et al., 2015). SSN comprises the *Skeleton* and *Data* modules, which provide RDF classes and properties to describe sensor data in terms of observations, feature of interest, observed property, measurement values and units. Figure 2a depicts classes and properties in these modules; an RDF graph describing an observation is presented in Figure 2b. Formally, an RDF graph is defined as follows:

Definition 3.1 (RDF triple and RDF graph)**.** (Arenas et al., 2009) Let **I**, **B**, **L** be disjoint infinite sets of URIs, blank nodes, and literals, respectively. A triple (s, p, o) $\in$ ($\mathbf{I} \cup \mathbf{B}$)$\times$**I**$\times$($\mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$) is denominated an RDF triple, where s is called the subject, p the predicate, and o the object. An RDF graph is a pair $G = (V, E)$, where V is a set of nodes in $\mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$, and E is a set of RDF triples.

Example 3.1. Figure 3 presents an RDF graph that corresponds to a portion of the RDF dataset from the storm season in year 2003. Nodes correspond to resources representing observations, measurements,

[1]http://wiki.knoesis.org/index.php/LinkedSensorData

[2]https://www.w3.org/2005/Incubator/ssn/



(a) **Portion** of the SSN Ontology. (b) **RDF graph** with three molecules.

Figure 2: **The SSN Ontology and RDF Molecule Example**. (a) Concepts and relationships in Skeleton and Data Modules from the SSN Ontology; (b) RDF graph with three subject molecules, each associated with two RDF triples.

and timestamps. Further, literals are also represented as nodes in the RDF graph and properties are from a variety of RDF vocabularies that include the Semantic Sensor Network (SSN) Ontology. We ignore prefixes and replace long URLs by short identifiers for clarity.

According to (Patni et al., 2010), an RDF graph of semantically described observations models the act of observing a real world phenomenon, and produce a value of the observed property. Figure 3 presents an RDF graph with two observations, `:obs1` and `:obs2`, from same sensor `:TR197` and of same type `:TempObservation` and the observed property `:AirTemp` in different timestamps `ts1` and `ts2`, having the same measurement value and unit: `"96.0"^^:float` and `:fahrenheit`, respectively.

RDF graphs usually comprise entity description sub-graphs, sometimes also referred to as Concise Bounded Descriptions (CBD). These subgraphs are named *RDF subject molecules* defined as follows:

Definition 3.2 (RDF Subject Molecule). (Fernández et al., 2014) Given an RDF graph G, an RDF subject-molecule $M \subseteq G$ is a set of triples t1, t2, ... , tn in which subject(t1) = subject(t2) = ... = subject(tn).

Example 3.2. Figure 2b presents an RDF data set with three RDF subject molecules, where the subjects of the molecules are `:obs1`, `:oMolecule` and `:mMolecule`, that describe resources in terms of their properties. For simplicity, we will refer to RDF subject molecules as *molecules* in the rest of the paper.

4 THE FSSD APPROACH

We propose FSSD, a framework that relies on a deductive database system to solve the problem of *factorizing semantic sensor data (FSSD)*. Figure 4 depicts the FSSD architecture. FSSD is composed of two main components: *the FSSD factorizer* and the *the FSSD query engine*. Given an RDF graph G_1 *the FSSD factorizer* identifies a factorized RDF graph G_2, such that, $G_1 \vdash_{\mathcal{FSSD}} G_2$. Input RDF graphs are represented as Datalog predicates using the *FSSD data model*; a deductive system relies on a set of rules that produce the factorized RDF graph G_2. The *FSSD query engine* is able to rewrite a SPARQL query Q against the original RDF graph G_1 into a SPARQL query Q', and perform query processing techniques to ensure that the answers of evaluating Q over G_1 and Q' over G_2 are the same, i.e., $[[Q]]_{G_1} = [[Q']]_{G_2}$. Optimization techniques are conducted to identify a bushy query execution plan for Q'. Thus, the *FSSD query engine* even evaluating Q' against the factorized RDF graph G_2 generates the same answers as if Q were evaluated over G_1. However, as we will report on our experimental study, query execution time can be reduced by up to two orders of magnitude.

4.1 The FSSD Factorizer

The FSSD factorizer provides a solution to the problem of *factorizing semantic sensor data (FSSD)* in RDF graph G_1 by generating a factorized graph G_2. Given an RDF graph G_1, the *FSSD factorizer* relies on the *FSSD data model* to represent G_1 as an extensional database (EDB) in Datalog. The knowledge required to generate the factorized is encoded in a set of intensional database (IDB) rules; a fixed point evaluation of the IDB on EDB allows for the generation of the instances of the *FSSD data model* that are used by the factorized RDF graph G_2. The *factorized graph creator* receives the instances of the *FSSD data model* and generates the factorized RDF graph G_2.

4.2 The FSSD Data Model

The FSSD factorizer resort to a deductive database system to solve the problem of *factorizing semantic sensor data (FSSD)*. RDF triples in an input RDF graph G_1 are formally represented as a Datalog extensional facts (EDB) containing three arguments. For example, the RDF triple (*:obs1 :procedure :TR197*) is formally represented as the following EDB fact:

$$triple(:obs1, :procedure, :TR197)$$

The *FSSD data model* also provides predicates: *measurementMolecule()* and *observationMolecule()*, which represent RDF molecules of measurements and

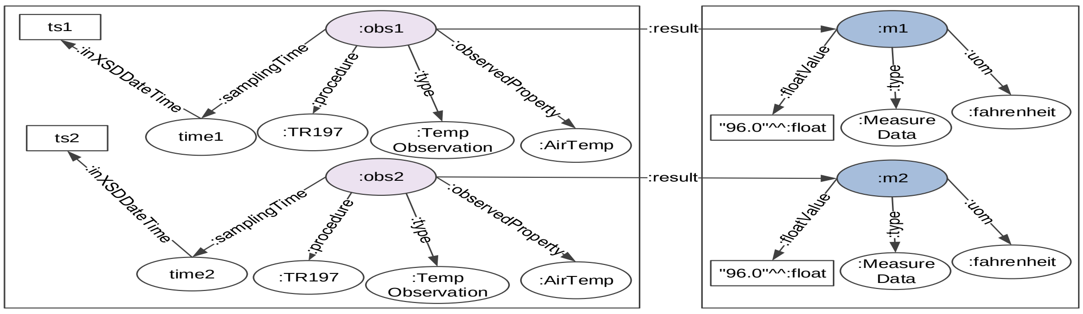


Figure 3: **An Example of the Repeated Measurement Values**: RDF graph G_1 has measurements, `:m1` and `:m2`, and observations, `:obs1` and `:obs2`, duplicated.

observations in a factorized RDF graph G_2. For example, an RDF molecule *:mMolecule* with value 96.0 and unit of metric Fahrenheit is modeled as follows: *measurementMolecule(:mMolecule,T,V,U)*, where

- T: *(:type,:MeasureData)*
- V: *(:floatValue,"96.0"^^:float)*
- U: *(:uom,:fahrenheit)*

4.3 Deductive System Engine

The deductive system engine relies on a Datalog representation of triples in an RDF graph G_1, as well as on Datalog intentional rules 1-3 (Figure 5) to generate a factorized RDF graph G_2. Rule 1 (Figure 5a) creates a measurement molecule for all the Datalog instantiations of measurements in G_1 such that all the measurements with same measurement unit and value and are mapped to the one measurement molecule. Thus, rule 1 ensures that repeated measurements are combined together in the form of a measurement molecule in G_2. Similarly, base and recursive cases of Rule 2 (Figure 5b and 5c) map all the Datalog instantiations of observations, with same sensor, observed phenomenon, property, measurement value and unit, in G_1 to the one observation molecule in G_2.

Rule 3 (Figure 5d) creates Datalog instantiations of relationships between measurement and observation molecules. Following a semi-naive approach deductive system engine computes Datalog representation of measurement and observation molecules in G_2 as a bottom-up evaluation of Rules 1-3 on Datalog representation of G_1 following a semi-naive algorithm that stops when the least fixed-point is reached. Built-in predicates are considered as EDB predicates that instead of being physically stored, are implemented as external programs and executed during the evaluation of a Datalog program. To ensure the safety condition of Datalog programs, built-in predicates are not evaluated until their input variables are bound (Ceri et al., 1989). The Datalog inferred facts are given to the *factorized graph creator* to generate the RDF graph G_2.

4.4 The Factorized Graph Creator

Facts generated by the deductive system engine correspond to the Datalog intentional predicates in the head of Rules 1-3 (Figure 5), i.e., these predicates represent measurement and observation molecules and relationship between them. The *molecule creator* component of the *factorized graph creator* creates RDF triples from these intentional predicates representing measurement and observation molecules. The *molecule linker* component of the *factorized graph creator* derives RDF triples from the instances of the *FSSD data model*, describing the relationships between the measurement and the observation molecules, as well as the rest of the RDF triples associated with them.

5 THE FSSD QUERY ENGINE

The *FSSD query engine* is described based on query rewriting over factorized RDF graphs, as well as optimization and execution of the rewritten query.

5.1 The Query Rewriter and Optimizer

Given RDF graphs G_1 and G_2, where $G_1 \vdash_{\mathcal{FSSD}} G_2$, the FSSD *query rewriter* reformulates a SPARQL query Q over G_1 into a SPARQL query Q' over G_2 according to σ, i.e., $qr(Q,\sigma)=Q'$. Then, query optimization is conducted to generate efficient query plans.

During query rewriting two BGPs, b_o and b_m, corresponding to observation and measurement molecules, respectively, are generated against each BGP b in Q'. Then query rewriter identifies a set of triple patterns that both BGPs share, and generates three new BGPs. One BGP b' includes shared triples patterns and the other two BGPs, b'_o and b'_m are generated as a result of eliminating shared triple patterns from b_o and b_m, respectively. Then, optimization techniques are executed to identify star-shaped groups in all the

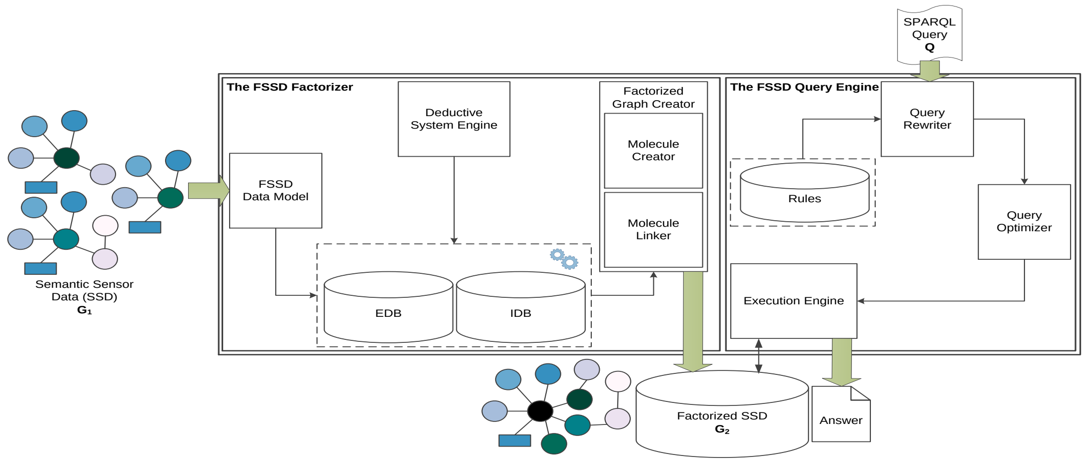


Figure 4: **The FSSD Architecture Components**: The *FSSD Factorizer* receives an RDF graph G_1 containing observations with repeated measurement values, and generates a factorized RDF graph G_2 where repeated observations and measurements are factorized. The *FSSD Data Model* internally represents RDF graphs as Datalog extensional facts (EDB); a set of Datalog rules (IDB) represent the transformations to produce G_2. A *Deductive System Engine* performs a fixed point to evaluate IDB rules on from EDB facts. Inferred facts are transformed into RDF graphs by the *RDF graph Creator*. The *FSSD Query Engine* allows for the execution of SPARQL queries against factorized RDF graphs and ensures efficient query processing.

BGPs, as well as to create corresponding bushy tree plans.

5.2 The FSSD Execution Engine

We implemented the FSSD *query engine* on top of the RDF triple store GH-RDF3X (Vidal et al., 2010) to exploit the benefits of the plan produced by the *query rewriter*. GH-RDF3X is an extension of RDF-3X (Neumann and Weikum, 2010), a *best-of-breed* RDF engine (Huang et al., 2011), to accept plans of any shape and assign particular operators to join star-shaped groups in a bushy plan. The features of RDF-3X and GH-RDF3X are crucial to allow for efficient executions of the reformulated queries; particularly, the cache system implemented by RDF-3X is able to load portions of data in resident memory, and thus, speed up execution time in query plans that produce small intermediate results.

6 EXPERIMENTAL STUDY

We empirically study the benefits of the proposed factorization techniques for semantically represented sensor data, and evaluate the impact on the size of the factorized RDF graphs as well as on query execution time. An extension of the *best-of-breed* RDF engine, RDF-3X is utilized in the evaluation. We empirically assessed the following research questions: **RQ1)** Are the proposed factorization techniques able to reduce redundancy of measurements and observations in sensor data? **RQ2)** Can queries against factorized RDF graphs speed up the execution time? **RQ3)** Is the performance of queries against factorized RDF graphs affected by the size of the factorized RDF graphs? The experimental configuration to evaluate the research questions mentioned above is as follows:
Datasets: We conduct the evaluation on three sensor datasets[3]. These datasets comprise observations of different climate phenomena e.g., temperature, visibility, precipitation, wind speed, and humidity, during the hurricane and blizzard seasons in the United States in the years 2003, 2004, and 2005. Table 1 describes the main characteristics of these datasets.
Queries: SPARQL queries range from simple queries with one triple pattern to complex queries having up to fourteen triple patterns with UNION, OPTIONAL and FILTER clauses, are used as baseline in our experimental testbed. [4].
Metrics: We report on the following metrics: **a) Number of Triples (NT)** in the semantic sensor data collection. **b) Query Execution Time (ET)** is the elapsed time between the submission of the query to RDF-3X engine and the complete output of the answer. **ET** is measured as the *real time* produced by the *time* command of the Linux operation system.
Implementation: Experiments were performed on Linux Debian 8 machine with an CPU Intel I7 980X 3.3GHz with 32GB RAM 1333MHz DDR3. Queries were run with cold and warm cache.[5] To run on warm

[3]Available at: http://wiki.knoesis.org/index.php/LinkedSensorData

[4]Details can be found at https://sites.google.com/site/fssdexperimets/

[5]To run cold cache, we clear the cache before running

```
measurementMolecule(:mMolecule,(:type,:MeasureData),(:floatValue,"96.0"^^:float),(:uom,:fahrenheit)):-
      triple(:m1,:type,:MeasureData) & triple(:m1,:floatValue,"96.0"^^:float) &
      triple(:m1,:uom,:fahrenheit) & uri(:mMolecule,[float("96.0"^^:float),localname(:fahrenheit)]).
```

(a) Datalog Rule 1. A measurement molecule with URI :mMolecule.

```
observationMolecule(:oMolecule,(:procedure,:TR197),(:type,:TempObservation),
(:observedProperty,:AirTemp),[(:hasObs,:obs1)]):-
      triple(:obs1,:procedure,:TR197) & triple(:obs1,:type,:TempObservation) &
      triple(:obs1,:observedProperty,:AirTemp) & triple(:obs1,:result,:m1) &
      triple(:m1,:floatValue,"96.0"^^:float)&triple(:m1,:uom,:fahrenheit)&uri(:oMolecule,[localname(:
           obs1).
```

(b) Base Case of Datalog Rule 2. An observation molecule with URI :oMolecule.

```
observationMolecule(:oMolecule,(:procedure,:TR197),(:type,:TempObservation),
(:observedProperty,:AirTemp),EM=[(:hasObs,:obs1),(:hasObs,:obs2)]):-
      observationMolecule(:o1,(:procedure,:TR197),(:type,:TempObservation),
      (:observedProperty,:AirTemp),[(:hasObs,:obs1)]) & observationMolecule(:o2,(:procedure,:TR197),
      (:type,:TempObservation),(:observedProperty,:AirTemp),[(ssd:hasObs,:obs2)]) &
      EM1=[(:hasObs,:obs1)|EM1],triple(:obs1,:result,:m1) & triple(:m1,:uom,:fahrenheit) &
      triple(:m1,:floatValue,"96.0"^^:float) & EM2=[(:hasObs,:obs2)|EM2] & triple(:obs2,:result,:m2) &
      triple(:m2,:uom, :fahrenheit) & triple(:m2,:floatValue,"96.0"^^:float) & :m1!=:m2 &
      uri(:oMolecule,[localname(:o1),localname(:o2)]) & concat([(:hasObs,:obs1)],[(:hasObs,:obs2)],EM)
           .
```

(c) Recursive Case of Datalog Rule 2. An observation molecule with URI :oMolecule.

```
triple(:oMolecule, :result, :mMolecule):-
      measurementMolecule(:mMolecule,(:type,:MeasureData),(:floatValue,"96.0"^^:float),
      (:uom,:fahrenheit)) & observationMolecule(:oMolecule,(:procedure,:TR197),
      (:type,:TempObservation),(:observedProperty,:AirTemp),EM=[(:hasObs,:obs1)|EM1],
      triple(:obs1,:result,:m1),triple(:m1,:uom,:fahrenheit),triple(:m1,:floatValue,"96.0"^^:float).
```

(d) Datalog Rule 3. Observation and Measurement Molecules are Linked.

Figure 5: **Example of Datalog Rules for Molecule Creation and Linking**: (a) Rule 1 creates a measurement molecule :mMolecule with value and unit. (b) Rule 2 is the base case to create the observation molecule :oMolecule. (c) Recursive case of Rule 2 combines observation molecules that comprise observation molecule :oMolecule, i.e., molecules such that all observation molecules with same sensor, observed phenomenon, observed property, measurement unit, and value; (d) Rule 3 links observation and measurement molecules. Molecules of measurements and observations are identified with URIs.

cache, we execute the same query five times by dropping the cache just before running the first iteration of the query; thus, data temporally stored in cache during the execution of iteration i can be used in iteration $i+1$. The RDF triple store GH-RDF-3X[6] is used to execute the SPARQL queries. GH-RDF3X is built on top of RDF-3X version 0.3.4., and it is tailored to execute star-shaped queries and bushy tree plans.

Effectiveness of the Semantic Sensor Data Factorization For evaluating the effectiveness of the proposed factorization techniques and answer research questions **RQ1**, we execute the factorized decomposer on datasets **D1**, **D2**, and **D3**. The effectiveness of the factorization approach is studied in terms of the reduction of RDF triples (**NT**). Table 1 shows the number of RDF triples (**NT**) in datasets **D1**, **D2**, and **D3** before and after the factorization. The results demonstrate that the proposed factorization techniques are capable of reducing the RDF triples by at least 53.24%. Moreover, the results report that the factorized representation of sensor observations requires in average a small number of RDF triples, i.e., five RDF triples instead of ten, while preserving all the information within the original RDF graph. These results allows us to positively answer research question **RQ1**, i.e., factorized RDF graphs effectively reduce redun-

each query by performing the command `sh -c "sync ; echo 3 > /proc/sys/vm/drop_caches"`.

[6]https://github.com/gh-rdf3x/gh-rdf3x. Downloaded on March 2016

Table 1: **Effectiveness of the Semantic Sensor Data Factorization (NT)**. Number of triples before and after factorization along with savings. Percentage of savings increases as size of the dataset (**NT**), while Avg. **NT** decreases.

Dataset ID	**NT before Factorization**	**NT after Factorization**	**Savings in %**	**Avg. NT per Obs. before Factorization**	**Avg. NT per Obs. after Factorization**
D1	38,054,493	17,795,661	53.24	9.29	4.35
D2	108,644,568	47,788,732	56.01	9.32	4.10
D3	179,128,407	78,421,471	**56.22**	9.31	**4.08**

dancy of measurements and observations.

Efficiency of Query Plans of Reformulated Queries To answer research questions **RQ2** and **RQ3**, we analyze the efficiency of generated query execution plans, and the benefits of running these queries on cold and warm caches. The original queries `Q` are compared to reformulated queries `Q′` with bushy query plans composed of star-shaped groups. Original queries (`Q`) are executed against the original datasets, while plans for reformulated queries (`Q′`) are run against gradually increasing factorized datasets. Figure 6 reports on the query execution time (milliseconds. log-scale) with cold cache, and the minimum value observed with warm cache. In all cases bushy query plans exhibit better performance whenever they are run on cold and warm caches. This observation supports the statement reported in the literature (Vidal et al., 2010) that bushy tree plans composed of star-shaped groups produce small intermediate results which can be maintained in resident memory and re-used in further executions. Thus, the performance of optimized query plans is considerable better with warm cache, overcoming other executions by up to two orders of magnitude, e.g., Q2. Q5 retrieves DISTINCT sensors against observation molecules, therefore, execution time in factorized dataset is reduced by 5 orders of magnitudes. Results also suggest that performance of optimized query plans is not affected by the RDF graph size, e.g., D1, D2, and D3 with 216,953,114 RDF triples. These results allow us to positively answer research questions **RQ2** and **RQ3**.

7 RELATED WORK

In (Joshi et al., 2013) RDF data is compressed using frequent pattern mining techniques. (Pan et al., 2014) implements data summarizing and image compression techniques to minimize RDF data redundancy. Similarly, Fernandez et al. (Fernández et al., 2013) represents RDF triples as collection of data identifiers instead of original subject, predicate, and object values. The compressed RDF data, generated by above techniques, can only be queried by customized query engines, and decompression techniques need to be executed to execute any data management task. Contrary, we device factorization techniques to generate a compact RDF representation, where query execution can be performed directly and efficiently by exploiting semantics encoded in the factorized representation to generate query execution plans.

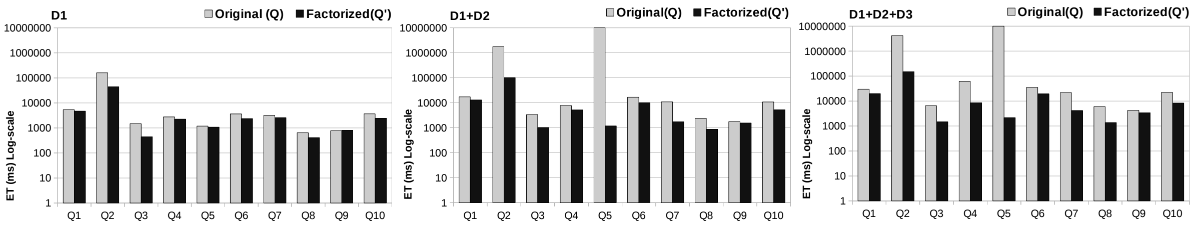
Factorization techniques have have been utilized for optimization of relational data and SQL query processing by applying logical axioms of relational algebra (Bakibayev et al., 2013; Bakibayev et al., 2012). Queries can be executed in factorized relational data, and efficient execution plans can be found to speed up execution time. We build on these experimental results and proposed factorization technique tailored for semantically described sensor data.

8 CONCLUSIONS AND FUTURE WORK

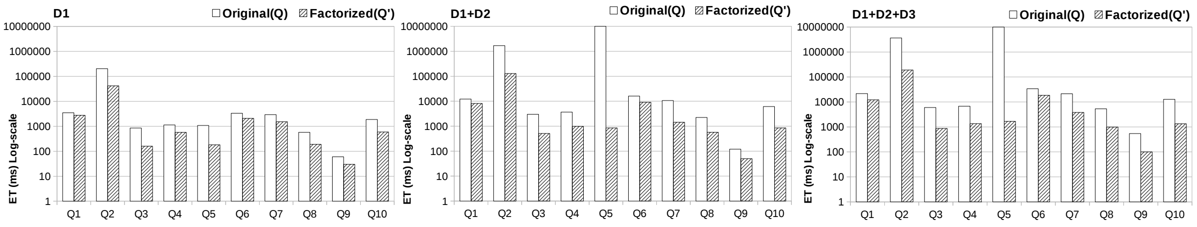
This paper presents factorization techniques for semantic sensor data to reduce redundancy, ensure correctness of query answers, and preserve complexity of query processing tasks. A set of Datalog rules provides the basis for a deductive system that allows for the creation of factorized RDF graphs. Additionally, these logical rules are utilized to transform SPARQL queries against factorized RDF graphs, and to generate query plans that speed up query execution time. We empirically evaluate the effectiveness of the proposed factorizations techniques and results confirm that exploiting semantics encoded in semantic sensor data allows for reducing redundancy by up to 50%. Our experiments confirm the efficiency of the query plans generated by our proposed approach. In summary, factorization techniques provide a feasible solution to the problem of reducing RDF redundancy. In the future, we will focus on extending the proposed factorization and query optimization techniques to streaming RDF data. Finally, we plan to incorporate these factorization techniques as part of existing RDF engines.

ACKNOWLEDGEMENTS

Farah Karim is supported by a scholarship of German Academic Exchange Service (DAAD), and part of this work is supported by the EU H2020 programme for the project BigDataEurope (GA 644564).



(a) Query Execution Time (ET) on Gradually Increasing Original and Factorized Semantic Sensor Data in **Cold** Cache.



(b) Query Execution Time (ET) on Gradually Increasing Original and Factorized Semantic Sensor Data in **Warm** Cache.

Figure 6: **Execution Time (ET in ms log-scale) of Testbed Queries (Q1-Q10) in GH-RDF3X**. Original queries Q against original RDF graphs of sensor data, and optimized queries Q' are run in cold (**First Row**) and warm caches (**Second Row**). Optimized query plans reduce execution time on factorized RDF graphs in cold and warm caches.

REFERENCES

Ali, M. I., Gao, F., and Mileo, A. (2015). Citybench: a configurable benchmark to evaluate rsp engines using smart city datasets. In *International Semantic Web Conference*, pages 374–389. Springer.

Arenas, M., Gutierrez, C., and Pérez, J. (2009). Foundations of rdf databases. In *Reasoning Web. Semantic Technologies for Information Systems*, pages 158–204. Springer.

Bakibayev, N., Kociský, T., Olteanu, D., and Zavodny, J. (2013). Aggregation and ordering in factorised databases. *PVLDB*, 6(14):1990–2001.

Bakibayev, N., Olteanu, D., and Zavodny, J. (2012). FDB: A query engine for factorised relational databases. *PVLDB*, 5(11):1232–1243.

Ceri, S., Gottlob, G., and Tanca, L. (1989). What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. Knowl. Data Eng.*, 1(1):146–166.

Compton, M., Barnaghi, P., Bermudez, L., GarcíA-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., et al. (2012). The ssn ontology of the w3c semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32.

Fernández, J. D., Llaves, A., and Corcho, Ó. (2014). Efficient RDF interchange (ERI) format for RDF data streams. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*, pages 244–259.

Fernández, J. D., Martínez-Prieto, M. A., Gutiérrez, C., Polleres, A., and Arias, M. (2013). Binary RDF representation for publication and exchange (HDT). *J. Web Sem.*, 19:22–41.

Gao, L., Bruenig, M., and Hunter, J. (2014). Semantic-based detection of segment outliers and unusual events for wireless sensor networks. *arXiv preprint arXiv:1411.2188*.

Henson, C. A., Neuhaus, H., Sheth, A. P., Thirunarayan, K., and Buyya, R. (2009). An ontological representation of time series observations on the semantic sensor web.

Huang, J., Abadi, D. J., and Ren, K. (2011). Scalable SPARQL querying of large RDF graphs. *PVLDB*, 4(11):1123–1134.

Joshi, A. K., Hitzler, P., and Dong, G. (2013). Logical linked data compression. In *10th Extended Semantic Web Conf. ESWC*, pages 170–184.

Neumann, T. and Weikum, G. (2010). The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1):91–113.

Pan, J. Z., Gómez-Pérez, J. M., Ren, Y., Wu, H., Wang, H., and Zhu, M. (2014). Graph pattern based RDF data compression. In *4th Joint Int. Conf. on Semantic Technology (JIST)*.

Patni, H., Henson, C., and Sheth, A. (2010). Linked sensor data. In *Collaborative Technologies and Systems (CTS), 2010 International Symposium on*, pages 362–370. IEEE.

Vidal, M., Ruckhaus, E., Lampo, T., Martínez, A., Sierra, J., and Polleres, A. (2010). Efficiently joining group patterns in SPARQL queries. In *7th Extended Semantic Web Conf. (ESWC)*.