Generating Data Wrapping Ontologies from Sensor Networks: a case study
Short Paper

Juan F. Sequeda[1], Oscar Corcho[2], Asunción Gómez-Pérez[2]

[1] Department of Computer Sciences, University of Texas at Austin
jsequeda@cs.utexas.edu
[2] Ontology Engineering Group, Departamento de Inteligencia Artificial,
Universidad Politécnica de Madrid
{ocorcho, asun}@fi.upm.es

Abstract. Information coming from sensor networks is being increasingly used in a variety of systems (decision support systems, information portals, etc), normally combined with information coming from more traditional sources (e.g., relational databases, web documents, etc). However, existing ontology-based information integration approaches cannot be easily used for this combination task since they are mainly focused on the integration of information coming from these traditional sources, and do not support sensor network data. In this paper we make a first step towards enabling the inclusion of sensor network data into these integration approaches, with the automatic generation of data wrapping ontologies for sensor networks. Our approach extends existing ones used for extracting data wrapping ontologies from relational databases, using the schema of sensor network queries and external ontology search and relation discovery services.

Keywords: Data Integration, Ontology Learning, Sensor Networks.

1 Introduction

Sensor networks generate large amounts of heterogeneous data that can be used in a range of systems, from providing information about temperature and humidity in a piece of land to monitoring the heart rate of a person, to give a few examples. With this increase in the amount of data available, and with the subsequent increase in the range of applications that make use of this data, new challenges in data access and integration arise. Traditionally, many data integration problems have been approached through the use of ontologies [1], which provide views over existing data sources or which are described according to how the sources cover them (global-as-view and local-as-view approaches, respectively). In all these approaches, the concepts of mediators and wrappers [2] are commonly used, and in some cases it is also common the usage of different layers of ontologies (global and local).

Ontology-based information integration approaches make use local ontologies, more commonly known as putative [3] or data wrapping ontologies, for each data source. However, manually generating these ontologies can be very costly. Hence, there is a need to automatically generate data wrapping ontologies to facilitate the information integration task. This automatic generation is done by applying a variety of ontology learning methods [4, 5], which allow the construction of lightweight ontologies from non-structured, semi-structured and structured data. It is worth noting that these types of ontologies depend on the structure of the underlying relational schema are prone to ambiguity and may be considered controversial.

None of these methods has focused though on the automatic construction of ontologies from data coming from heterogeneous sensor networks, even considering that the problem of automatically generating ontologies from sensors is similar to the case of extracting ontologies from relational databases. Hence, we propose to use and extend one of these approaches for the specific case of sensor data. More specifically, we use and extend Tirmizi et al's approach [6] because it is the only one that is formally described in first order logic.

In order to illustrate our approach, we present an example of how to derive a data wrapping ontology from a sensor network system based on a well-known benchmark (the Linear Road benchmark [7]). First, we apply techniques used to automatically generate ontologies from relational databases schemas to sensors. This is then complemented with an approach to generate ontologies from derived queries, which are a series of intermediate queries used to formulate a final query, and the use of external ontology search and relation discovery services. We finalize by presenting future challenges in how this process can be completely automated.

This paper is organized as follows: In Section 2 we present the Linear Road Benchmark, which is the running example, used throughout this paper. Section 3 describes how an ontology can be generated from a sensor network. This section is divided in several blocks according to the degree of complexity of the ontology that is being generated. Finally, Section 4 we present conclusions and future work

2 Running example: Linear Road Benchmark

The Linear Road Benchmark is a well-established benchmark for Data Stream Management Systems [7]. This benchmark specifies a variable tolling system by determining changing factors of car congestion on a highway. Each car on the highway is equipped with a sensor that emits the vehicle's exact location and speed every 30 seconds. The data emitted by the sensors are sent as streams to a central system where statistics are generated about traffic conditions on the highways. The central system aggregates the data received from the vehicles, computes the toll in real time, and transmits the tolls back to the vehicles. This tolling system is designed to discourage drivers to use already congested roads because they have an increased toll. Consequently, it would encourage drivers to use less congested roads because they would have decreased tolls.

Arasu et al describes the Linear Road benchmark, which is used as basis of a running example for queries in the Continuous Query Language (CQL) language [8].

This benchmark only has one base input stream, which contains measurements of speed and positions using the highway. The schema of this input stream[1] is shown in Figure 1.

```
PosSpeedStr(vehicle_id/* unique car identifier       */
          speed,     /* speed of the car            */
          x-pos);   /* coordinate in express way   */
```

Fig. 1. Schema of PosSpeedStr stream.

We also expand our running example by adding a new base relation Vehicles, which would be a relation with details of vehicles. This base relation is not part of the original benchmark, however we find the need to add this relation in order to show results in the following sections.

```
Vehicle(vehicle_id, /* primary key                  */
        model,       /* model of the vehicle        */
        license_plate); /* license plate of vehicle */
```

Fig. 2. Schema of PosSpeedStr stream.

This benchmark considers a 100 mile long highway divided in one hundred 1-mile segments. Vehicles only pay a toll when they enter a congested segment. A segment is congested when the average speed of all vehicles in a segment over the last 5 minutes is less than 40 mph. The objective of this benchmark is to calculate the toll that each vehicle is supposed to pay. Due to the fact that calculating tolls for each vehicle is fairly complex, the final desired query is expressed using several derived queries, as shown in Figure 2. For example, the query `TollStr` is created by a combination of queries on `VehicleSegEntryStr`, `SegVolRel` and `CongestedSegRel`.

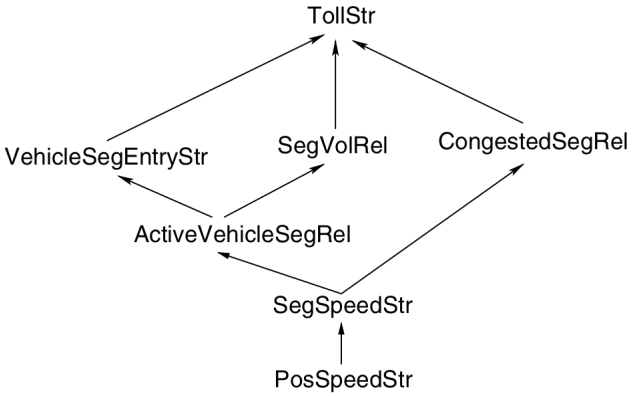

Fig. 3. Graph of the derived queries generated to create the final query TollStr.

[1] http://infolab.stanford.edu/stream/cql-benchmark.html

We describe the meaning of each query in Table 1. Each of these queries results in either a Stream or a Relation. The exact queries in CQL and the different type of Stream and relation operators can be found in [9].

Table 1. Description of each query from Fig. 3.

Query	Description
PosSpeedStr(vehicleID, speed, xPos)	A stream that contains the exact location of a vehicle and its current speed
SegSpeedStr(vehicleID, speed, segNo)	A stream that contains the segment in which a vehicle is located and its current speed
ActiveVechileSegRel(vehicleID, segNo)	At a time t, a relation that contains which vehicles are in which segments
VehicleSegEntryStr(vehicleID, segNo)	At a time t, a stream that contains the vehicles that are entering a specific segment
CongestedSegRel(segNo)	At a time t, a relation the contains the congested segments
SegVolRel(segNo, numVehicles)	At a time t, a relation that contains the current amount of vehicles in each segment
TollStr(vehicle, toll)	The final output stream which contains the toll that each vehicle should pay

3 Generating Ontologies from Sensors

This section presents our proposed method of generating an OWL data wrapping ontology from sensors. We assume the following formal definitions of Arasu et al [8].

Definition 1 (Stream) A stream S is a (possibly infinite) bag (multiset) of elements (s, t), where s is a tuple belonging to Q, the schema of S and $t \in T$ is the timestamp of the element.

Definition 2 (Relation) A relation R is a mapping from each time instant in T to a finite but unbounded bag of tuples belonging to Q, the schema of R. It is similar to the relation of the standard relation model.

Definition 3 (Base Streams) The source data stream that arrives at the Data Stream Management System (i.e `PosSpeedStr`)

Definition 4 (Derived Streams) The intermediate streams produced by operators in a query (i.e. `SegSpeedStr`, `VehicleSegEntryStr`, `TollStr`)

Definition 5 (Base Relations) The input relations (i.e. `Vehicle`)

Definition 6 (Derived Relations) The relations produced by query operators (i.e. `ActiveVechileSegRel`, `CongestedSegRel`, `SegVolRel`)

Definition 7 (Derivation of Queries) A derivation of queries (DoQ) is a directed graph D = (V, A) where V is a set of base or derived streams S and base or derived relations R, which act like nodes, and A is a set of ordered pairs of vertices. Figure 2 is a DoQ.

Our proposed method consists of three phases. First, we generate an ontology from the base streams and base relations. This phase is very similar to the generation of ontologies from relational databases, hence the same technique will be applied. Second, we generate another ontology from the set of derived streams and derived relations through a novel approach. Finally, both ontologies are combined into the final data wrapping ontology.

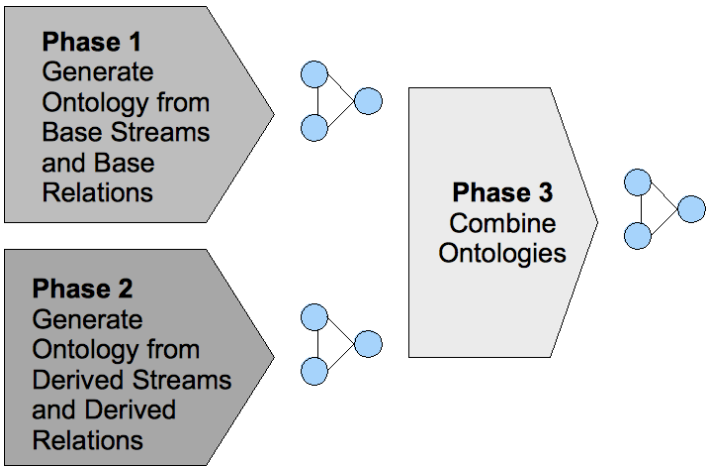

Fig. 4. Phases of the method to generate an ontology from sensors.

3.1 Phase 1: Generating Ontologies from Base Streams and Base Relations

We have defined a number of predicates to aid in the process of defining transformation from base streams and base relations to an ontology. There are two sets of predicates: Sensor predicates and Ontology predicates. The sensor predicates (see Table 2) determine whether an argument matches a construct in the domain of sensors.

Table 2. Sensor Predicates.

Predicate	Description
BaseStr(s)	s is a Base Stream
BaseRel(r)	r is a Base Relation
Attr(x, s)	x is an attribute in s and either BaseRel(s) or BaseStr(s) holds
FK(x,r,x,s)	x is the same attribute in r and s and either BaseRel(r) or BaseStr(r) holds and either BaseRel(s) or BaseStr(s) holds.
NonFK(x,s)	x is an attribute in s and either BaseRel(s) or BaseStr(s) holds and x does not as an attribute in any other Streams or Relations.

The ontology predicates (see Table 3) determine whether an argument matches a construct that can be represented in an OWL ontology.

Table 3. OWL Ontology Predicates.

Predicate	Description
Class(c)	c is an OWL Class
ObjP(p,d,r)	p is an OWL Object Property with domain d and range r
DataTP(p,d,r)	p is an OWL Datatype Property with domain d and range r

This phase consists of three steps. First, identifying OWL classes, then OWL Object Properties and finally OWL Datatype properties.

Step 1.1. Identifying OWL Classes

Following [7], we can consider that all base relations and base streams can be mapped to an OWL class, as stated in the following rules:

Rule 1.

$$\text{Class}(x) \leftarrow \text{BaseStr}(x)$$

Rule 2.

$$\text{Class}(x) \leftarrow \text{BaseRel}(x)$$

Example of Rules 1 and 2:

Rule 1 identifies the base stream `PosSpeedStr` as an OWL class.

Step 1.2. Identifying OWL Object Properties

An object property is a relationship between two classes in a particular direction. Because sets of sensors are not explicitly related, as it occurs with relational tables

through foreign keys, the existing rules from Tirmizi et al. cannot be similarly applied.

The primary key of the base relation `Vehicle` has the same identifiers used in the `PosSpeedStr` base stream. Taking this assumption into consideration, if base streams and base relations share the same attributes, then that attribute can be mapped to an OWL Object Property. The domain and range of the property could be any of the resulting OWL Classes.

Rule 2.

$$\text{ObjP}(x,r,s) \leftarrow \text{FK}(x,r,x,s)$$

Example of Rule 2:
Given the stream `PosSpeedStr` and the relation `Vehicle`, Rule 1 and 2 would map `PosSpeedStr` and `Vehicle` as OWL classes. Furthermore, because `PosSpeedStr` and `Vehicle` have the attribute `vehicleID` in common, this attribute would be mapped to an OWL Object Property. `PosSpeedStr` could be the domain and `Vehicle` the range, or vice-versa.

Step 1.3. Identifying OWL Datatype Properties
A datatype property is a relationship between a class and a literal. Every attribute of a base stream or base relation is mapped to an OWL Datatype Property.

Rule 3.

$$\text{DataTP}(x,s,\text{type}(x)) \leftarrow \text{NonFK}(x,s)$$

where type(x) is a function that maps an attribute x to its corresponding XML datatype.

Example of Rule 3:
The attributes speed and `xPos` from `PosSpeedStr` would get mapped as OWL Datatype Properties with domain `PosSpeedStr`.

The resulting ontology from this approach is shown in Figure 5. The ovals are OWL classes and the squares are the range of the OWL Datatype properties. These squares are blank because in this phase, it is not possible to identify the range (integer, float, etc) because it is not specified in the schemas. This issue will be dealt with in the following phases.

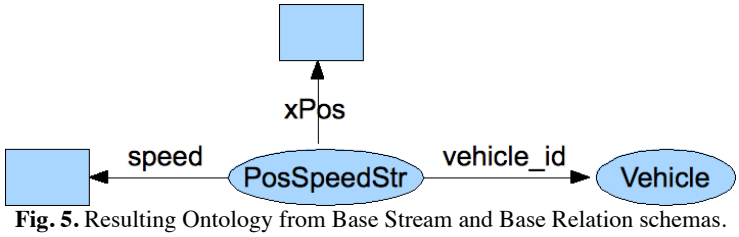

Fig. 5. Resulting Ontology from Base Stream and Base Relation schemas.

3.2 Phase 2: Generating Ontologies from Derived Streams and Derived Relations

Derived streams and derived relations are produced by query operators, and have been derived originally from base relations and/or base streams [8]. Applying the rules presented in the previous phase to the derived streams and derived relations would result in a repetition of classes and properties, because the elements of the queries are reoccurring. However, the rules presented in the previous phase are not sufficient to generate an expressive ontology because base relations and base streams of sensors do not have varied relationships in the schema.

In this second phase, we intend to identify more semantics through the derived streams and relations. This phase makes use of external ontology search such as Watson [9] and relation discovery services like Scarlet [10] to enhance the ontology. Usually complex queries can only be generated by a series of previous derived queries, which are originally derived from the base stream and base relations.

The ontology generated in phase 1 lacks important classes and relationships in this domain. For example, requirements such as: a vehicle has a speed, unit of speed (km/h or mph), a vehicle pays a toll, or a vehicle is located in a segment, are not represented in the ontology from phase 1. In this step, we present a heuristic that analyzes the derivation of queries in order to identify these types of missing classes and relationships.

Step 2.1. Identify attributes of the queries in a DoQ

Table 1 shows all the queries used in order to derive the final query. Figure 2 shows a graph that represents the derivation of the queries shown in Table 1. In this step, we proceed to extract the unique attributes across all the queries in a DoQ. For our running example, these attributes would be:

{ VehicleID, Speed, XPos, SegNo, NumVehicles, Toll}

Step 2.2. Consider each attribute as a generic entity

The previous step extracted all the unique attributes from the queries. However, at this moment, there is no way to identify the role of these attributes in an existing ontology because these could be either classes or properties. Therefore we consider these attributes as entities.

> ***Definition 7 (Entity)*** An entity is a unique attribute x extracted from a DoQ which may correspond to an OWL Class, OWL Object Property, OWL Datatype Property or instance.

Step 2.3. Identify relationships between attributes of each query

Each query in a DoQ is either a stream or a relation. The schema Q of the stream or relation has a series of attributes. We proceed to create binary relationships between all the attributes of each schema Q, which have at least two attributes.

Table 4. Relationships among the attributes of a DoQ.

Query	Description
PosSpeedStr(vehicleID, speed, xPos)	VehicleId ↔ Speed VehicleId ↔ xPos Speed ↔ xPos
SegSpeedStr(vehicleID, speed, segNo)	VehicleID ↔ Speed VehicleID ↔ segNo Speed ↔ segNo
ActiveVechileSegRel(vehicleID, segNo)	VehicleID ↔ segNo
VehicleSegEntryStr(vehicleID, segNo)	VehicleId ↔ segNo
CongestedSegRel(segNo)	No relationships
SegVolRel(segNo, numVehicles)	segNo ↔ numVehicle
TollStr(vehicle, toll)	vehicleID ↔ toll

Step 2.4. Add the unique relationships between the entities and create a graph

Given all the entities from Step 2.2 and the relationships between the entities from Step 2.3, we identify the unique relationships and consequently create an unlabeled bidirectional graph as shown in Figure 6.

Table 5. Unique relationships among attributes of a DoQ.

VehicleId	↔	Speed
VehicleId	↔	xPos
Speed	↔	xPos
VehicleID	↔	segNo
Speed	↔	segNo
segNo	↔	numVehicle
vehicleID	↔	toll

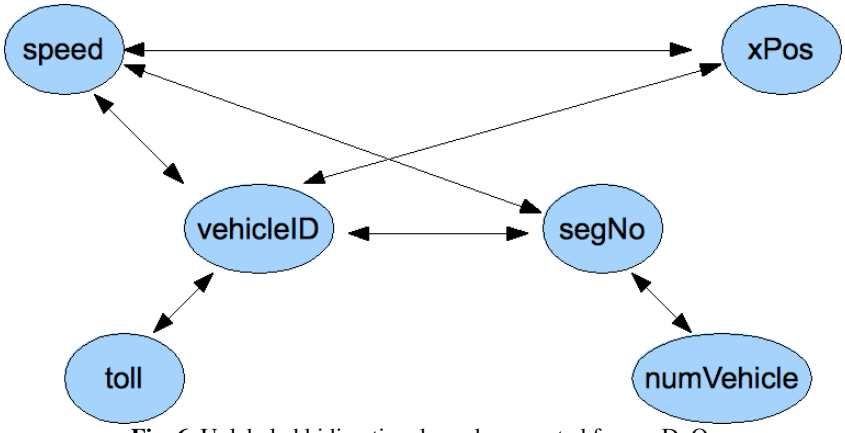

Fig. 6. Unlabeled bidirectional graph generated from a DoQ.

Step 2.5. Refine graph and convert into an Ontology

Ontologies can be considered as directed graphs with labeled edges. However, the graph in Figure 6 is not an ontology for several reasons. First, in order for it to be converted into an ontology, we need to determine what each node means. For example, the node `speed` makes sense being an OWL Datatype property with domain Vehicle. This would mean that the node `vehicleID` makes sense to become an OWL Class.

Second, the edges are bidirectional and unlabeled; therefore there is no knowledge about the domain and ranges. For example, the nodes `vehicleID` and `segNo` could be both considered as OWL Classes. Therefore the edge between the two nodes would be considered as an OWL Object Property. This property could then have a label `isLocatedIn` with domain `vehicleID` and range `segNo`. Identifying the labels of the edges is another issue. Either a relationship discovery system can give you a series of possible labels (hasSpeed, hasMaximumSpeed, etc), or a human user needs to label the edges manually.

Third, this graph presents nodes and edges that may not make sense in the real world to a user, for example the relationship between the nodes "segNo" and "speed".

In this step, these mismatches need to be identified and the graph refined into an ontology. In order to do so, we propose to identify the relationship between two entities relying on existing background knowledge and ontologies on the Semantic Web. One way to do this is using Scarlet [10], which is a service for discovering relations between two concepts by making use of ontologies on the Semantic Web. However, Scarlet is limited because it does not discover relationships between generic entities. On the other hand, the Watson Semantic Web search engine [9], which is the back-end for Scarlet, allows searching for entities inside ontologies.

Therefore, our future work is to take the lessons learned from Scarlet and extend the use case in order to discover relationships between two generic entities. For example, given two entities vehicle and speed, there is sufficient background knowledge on the Semantic Web to determine that vehicle is a concept and hasSpeed is a datatype property with domain vehicle and range float. Furthermore, it would be possible to discover the appropriate labels for classes and properties. This approach would then fully automate the process of refining the graph into an ontology. The final result of manually refining the ontology is shown in Figure 7, which we acknowledge can be automated by applying the techniques previously proposed.

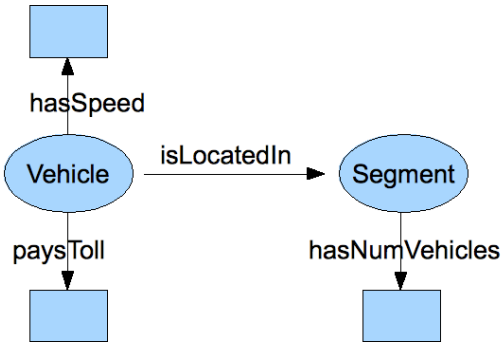

Fig. 7. Resulting Ontology from Phase 2.

3.3 Phase 3: Combining Ontologies

The outcomes of phase 1 and phase 2 are two different ontologies that complement each other. The final step of this process it to merge both of these ontologies. The result of this process is shown in Figure 8.

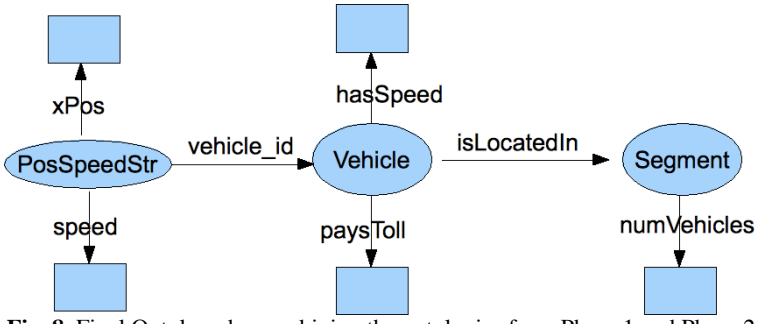

Fig. 8. Final Ontology by combining the ontologies from Phase 1 and Phase 2.

4 Conclusions and Future Work

We have described a method for the automatic generation of data wrapping ontologies from sensor network data. By automating this generation process during runtime instead of design time, it may become easier to integrate data from sensor networks that are not known a priori. This method extends one of the existing methods for the generation of data wrapping ontologies from relational data sources, making appropriate extensions that are related to some of the specific characteristics of sensor-based data. We have illustrated the application of our method with an example coming from a well-known benchmark for data stream management systems, the Linear Road Benchmark, so as to highlight the main differences with respect to ontology extraction from relational data sources and to provide examples of some of these characteristics.

This is still work in progress, hence there are many limitations and open challenges that need to be addressed in the future. First, our illustrative example has only shown the feasibility of applying our method in simple scenario. In order to be able to claim the adequacy of our method for a more generic kind of settings involving sensor networks, we need to make more extensive tests in other scenarios. For this reason, we will analyze data coming from sensor networks available in services like sensorbase[2] or pachube[3], and we will make more extensive tests that will allow us determine the quality of the generated ontologies, especially in terms of their adequacy for the task of enabling better data integration. The design (and subsequent execution) of these experiments is complex, due to the high variety in the amount and

[2] http://sensorbase.org/
[3] http://www.pachube.com/

complexity of data integration tasks that can involve these sensor networks. Besides, another open problem with this type of information is that it is still difficult to find good public sources for sensor network information.

Second, proposed method does not take advantage of the fact that it is dealing with sensor-based information. Although, as expressed in the introduction, the domains in which sensors are used are very broad, most of the measurements that they take are related to physical quantities (temperature, speed, humidity, etc.). Hence restricting or prioritizing somehow the domains in which the external ontology search and relation discovery services are used may increase the accuracy of the obtained results. This could be also done by allowing simple annotations from human users to be used also as inputs.

Spatio-temporal aspects of sensor-based information are also very relevant when it comes to processing (and integrating) information from them. Every tuple generated by a sensor network in response to a query will normally have a timestamp associated, together with, possibly, some additional spatial information (e.g., GPS coordinates). Hence every instance of the data wrapping ontology that may be generated from the sensor network should have that additional information associated to it, and this information may be used in the integration process. The ontological representation of time and space may be done in different ways, either by using specific spatio-temporal ontologies, or by using extensions of the RDF and OWL models, such as the one proposed for time by Gutiérrez et al [11].

Finally, measurement units are also a major source of heterogeneity when dealing with this type of information. It is not always easy to determine which is the measurement unit that a sensor network is using when providing data for an observation. In fact, this information comes in many occasions attached to the written specifications and manuals of the deployed sensors, and is not available as part of the schema or metadata that is attached to it. This problem would be easier to deal with if sensor networks provided a more homogeneous set of metadata to describe the measurement units that they use, together with any other type of useful information. Other possibilities would be related to the combination of data with existing relational data whose units are known, although this is not the most common case either.

Acknowledgments. This work has been supported by the European Commission projects SemSorGrid4Eng (FP7-ICT-223913).

References

1. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S., Ontology-based integration of information - a survey of existing approaches. In: Stuckenschmidt, H. (Ed.), IJCAI-01 Workshop: Ontologies and Information Sharing. 2001.
2. Wiederhold, G., Mediators in the Architecture of Future Information Systems. IEEE Computer. 1992
3. Sequeda, J.F., Tirmizi, S.H., Miranker, D.P., A Bootstrapping Architecture for Integration of Relational Databases to the Semantic Web. In Poster Proceedings of International Semantic Web Conference 2008

4. Gomez-Perez, A., Manzano-Macho, D., A survey of ontology learning methods and techniques. Deliverable 1.5 OntoWeb Project. 2003
5. Sequeda, J.F., Tirmizi, S.H., Corcho, O.. Miranker, D.P., Direct Mapping SQL Databases to the Semantic Web: a survey. Technical Report 09-04, Department of Computer Sciences, University of Texas at Austin. 2009
6. Tirmizi, S.H., Sequeda, J.F., Miranker, D.P.. Translating SQL Applications to the Semantic Web. In Proceedings of Database Experts Systems and Application 2008
7. Arasu, A., Cherniack, M., Galvez, E., Maier, D., Maskey, A., Ryvkina, E., Stonebracker, M., Tibbetts, R. Linear Road: A Stream Data Management Benchmark. In Proceedings of Very Large Databases Conference 2004.
8. Arasu, A., Babu, S., Widom, J., The CQL continuous query language: semantic foundations and query execution. VLDB Journal 15(2). 2006
9. d'Aquin, M., Grindinoc, L., Sabou, M., Angeletou, S., Motta, E., Watson: Supporting Next Generation Semantic Web Applications. In Proceedings of World Wide Web Conference 2007
10. Sabou, M., d'Aquin, M., Motta, E., Exploring the Semantic Web as Background Knowledge for Ontology Matching. Journal of Data Semantics. 2008
11. Gutierrez, C., Hurtado, C., Vaisman, A., Introductin Time into RDF. IEEE Transaction on Knowledge and Data Engineering. Vol 19. No. 2. 2007