Semantic Technologies for Describing Measurement Data in Databases

Ulf Noyer, Dirk Beckmann, and Frank Köster

Institute of Transportation Systems, German Aerospace Center
{ulf.noyer,dirk.beckmann,frank.koester}@dlr.de

Abstract. Exploration and analysis of vast empirical data is a cornerstone of the development and assessment of driver assistance systems. A common challenge is to apply the domain specific knowledge to the (mechanised) data handling, pre-processing and analysis process.

Ontologies can describe domain specific knowledge in a structured way that is manageable for both humans and algorithms. This paper outlines an architecture to support an ontology based analysis process for data stored in databases. Build on these concepts and architecture, a prototype that handles semantic data annotations is presented. Finally, the concept is demonstrated in a realistic example. The usage of exchangeable ontologies generally allows the adaption of presented methods for different domains.

1 Introduction

Reliable and safe automation is one foundation for modern traffic systems and part of concepts for assistance and automation systems. Therefore, for the analysis and reflection of users in automated environments, experimental systems (i.e. vehicles and driving simulators) play an important role. They produce exhaustive amounts of data, which can be used for an evaluation of the considered system in a realistic environment. Long term studies and naturalistic driving studies [7] result in similar datasets. Much data means a lot of information to interpret and potential results to discover. Therefore our motivation is, to store derived meta data closely connected with its original data for an integrated processing. By using semantic technologies a continuous technical and semantic process can be provided.

As a starting point, experimental data is to be considered as already stored in the relational database and should not be changed in the process. As many of the following principles not only match for measurement data, but in general for tables in relational databases, also the term bulk data is used as an equivalent for measurement data. So, it is aspired, to use semantic technologies for describing the database elements to support their interpretation [15]. They allow to formally handle complex data structures very flexible and schema knowledge can be extended easily. This is a demand resulting from the fact, that experimental systems are in continuous development and projects touch different knowledge domains.

G. Antoniou et al. (Eds.): ESWC 2011, Part II, LNCS 6644, pp. 198–211, 2011.

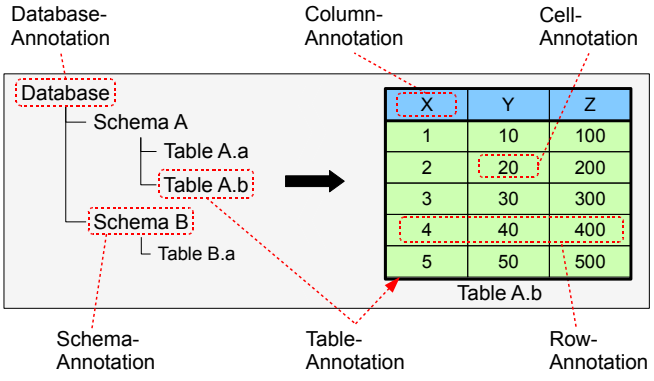

Fig. 1. Tables of a relational database with possible annotations

Figure 1 shows the intended database elements for semantic annotations. Tables containing sensor data of an experiment can be tagged with experiment information. Columns can have a semantic annotation about physical unit of containing elements and used sensor for recording. Rows or cells can get an annotation about bad quality or detected events in the considered elements. A complex application case is presented in Sect. 5. More meaningful examples of generic metadata annotations are also discussed in [16,12].

In some of the following considerations it is assumed, that table records have an order, so the statement *row* is preferred to record or tuple. Also rows are considered with the restriction that they must be uniquely identifiably. The needed database elements from Fig. 1 are transformed in unique URIs to be used in RDF statements. However, data itself is not transformed, since the value itself is not used for the presented purposes. Based on this, annotations are used similar to a formal memo or notepad mechanism. As domain knowledge is solely modelled in exchangeable ontologies, the presented concepts can be also applied for other domains.

2 Background and Related Work

This section introduces other work, which is needed for this paper or is relevant.

2.1 Usage of Semantic Technologies

In the last years several metadata technologies evolved, with the aim to make them better processable by machines. Especially the Resource Description Framework (RDF) is important in this context. Using RDF, a graph with relations of the individual metadata elements can be modelled. Resources to be described in RDF get annotated using triples, consisting of a subject, predicate and an object. On top of this, statements can be flexibly constructed. For serializing i.e. RDF/XML or Turtle syntax can be used.

The Web Ontology Language (OWL) is based on principles of RDF and can be used to formally describe complete ontologies. In an ontology the terms and their relations and rules are described. Especially for plain rules there is the Semantic Web Rule Language (SWRL) as an extension of OWL [13].

2.2 RDF-Stores

A RDF-repository [18,1] (triple-store) is used to manage RDF-statements. It supports adding, manipulating and searching statements in the repository. Such repositories can have different backends to store managed statements. Using relational databases as backend is in focus of this document. The statements consisting of subject, predicate and object are stored in a separate relation in the underlying database. Advantages of different optimization strategies are exhaustively discussed in [22]. Current repositories often directly support the Simple Protocol and RDF Query Language (SPARQL). It has for RDF-stores similar importance as SQL for relational databases.

2.3 Mapping of Relational Databases into RDF

There have been already early approaches to map relational data into RDF for the integration of relational databases and its contents [5]. In principle, a predefined scheme or rules for mapping the schema of the database and its content into RDF statements are used. There are projects ([3,6]), which are concerned with this functionality. However, both projects are not used for this work, because they would be strongly oversized. As shown in Sect. 3.3 just the mapping of database schema is needed (and not of data itself), what is comparatively easy. This work also only requires a mapping of the actually used database elements. If in the future also data itself managed in a database should be included in RDF mapping, usage of [6] could be a valuable option. This work is based on the principles discussed in [5].

2.4 Scientific Workflow Systems

The presented paper is distantly related with the field of scientific workflow systems (SWS, i.e. [2,20]). In SWS a process for evaluation of a data product can be specified. At execution, each processing step runs consecutively on data. The result of one step is input of the next one. Often SWS offer a graphical interface for composition of workflow steps using directed graphs. These systems support processing by offering metadata about processing steps and data lineage. Exchanged data products are considered as black boxes by SWS, which are not interpreted by the system.

But that is the focus of the presented paper, to concentrate on the content level of exchanged data. For that reason micro-annotations are utilized in a similar way as they are used with web pages [9] or multimedia files [11]. The aspect of data flow and its composition has only a minor role. In this work it is assumed that time structured data is stored in a database, an unusual restriction for classical SWS.

3 Data Management

The recorded time structured bulk data is stored in a relational database. All other data either describes these time series or is deduced from it. As a result, all data besides time series is considered as metadata.

3.1 Use Cases

This section presents the identified use cases for semantic annotations.

– Manual examination: Using an interface for interaction with annotations a user can directly interact and manipulate them (cf. Sect. 4). Resources can be referenced for documentation purposes. It also allows exporting tables with annotations into a flat file format for usage in other applications.

– Services: A service is an automated piece of software, which analyses some given data/annotations in the database and creates new data/annotations by applying specific domain knowledge. It has a web service interface for being externally triggered. Web services can be easily reused and orchestrated for more complex workflow setups or automatically started, after new data has been gathered. Furthermore, web services can be also reused by many SWS.

– Application integration: Any arbitrary application can also integrate support for semantic annotations. *Ærogator* is used for explorative data analysis and can display annotations parallel to time series data [12]. For data mining in time series *EAMole* is used and could respect annotations under data mining aspects [12].

3.2 Common Architecture

An overview of data storage and access are shown in Fig. 2 [14]. In the bottom layer, there is a relational database with the internal name Dominion-DataStore. It stores relational bulk data and RDF annotations in completely separated data structures. All elements in the figure with a horizontal pattern are related to bulk data. Elements with a vertical pattern describe elements connected with semantic annotations. Measurement data from an experiment is always stored in a new table, so relational database schema is quite simple. In the database there are *stored procedures* available to directly work on RDF annotations with SQL.

Access to RDF statements is more abstracted for an efficient data handling. The *triple store abstraction* implements the concrete RDF storage. The native RDF handling of an Oracle database is used. An alternative would be a database product independent schema layout (cf. Sect. 2.2). On top of the RDF storage abstraction a common *semantic web framework* handles RDF graphs [18]. Depending on the concrete RDF-graph, also RDFS- or OWL- models can be handled. Another layer is responsible for mapping required database elements into resource URIs, to be referenced in RDF statements. Concrete mapping implementation is based on the principles discussed in Sect. 3.3. The RDF store allows managing strictly divided models. So for each new bulk data table a new RDF

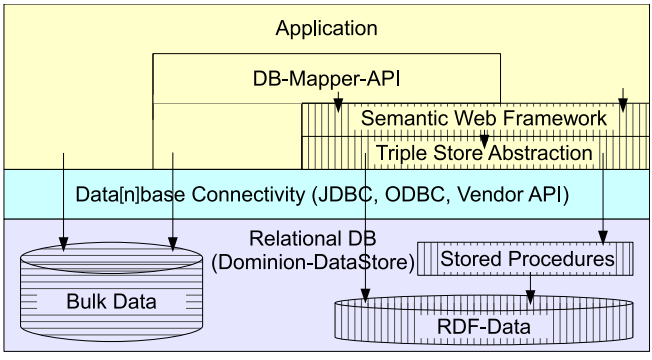

Fig. 2. Overview of architecture used for data access

model is used. In this way a good scalability can be reached. Generic information is stored in a jointly used RDF model.

Strict separation of bulk data and annotations also reflects their different usage. Afterwards measurement data is not modified anymore. Its meaning is hidden in recorded number values to be interpreted. On top of the bulk data, there is an abstract layer of interpretations. Descriptions (i.e. for columns) or identified events (i.e. overtaking manoeuvre) from human imaginations are to be seen there. In contrast to bulk data, there are much less elements. This abstraction layer is used during analysis, so that there is a high dynamic in the statements. These statements are usually assertional (ABox). Schema knowledge (TBox) can be edited using Protégé [19] or a Semantic Wiki [17] as known for RDFS/OWL and then be integrated (cf. Sect. 4). The just presented approach is the adaption of commonly discussed multimedia annotations [9,11] for measurement data. The term micro-annotations is also adapted from this domain.

3.3 Annotation of Relational Data

In the database, bulk data is organized in relations. RDF-annotations can reference single attribute instances, tuples, attributes, relations, schemas or databases (cf. Fig. 1). Theoretically, same mechanisms could be used to annotate other relational database concepts (i.e. keys, constraints). However, those are not considered, since they are not of interest for discussed use cases. For that purpose, an integration of relational elements in RDF is needed (cf. Sect. 2.3). Since RDF requires an unique specifier for every resource, database elements must be mapped into URIs. The used mapping approach builds on principles of [5].

Relations are uniquely identified by their name *table_name*. Also attributes of a table have an unique name *column_name*. For simplicity, slash (/) is not allowed in *table_name* and *column_name*. To identify tuples of a relation, the considered relation must have an *unique key*. Currently, an attribute *row_id* is added to every relation, which contains unique integers in context of a relation and serves as a surrogate key. By combining the identification of a tuple and an

attribute, single attribute instances can be referenced. Based on those assumptions, Table 1 shows building instructions for URIs to identify different database elements.

Table 1. Transformation rules for a database element into an URI

DB element	Transformation into URI
Relation	http://www.dlr.de/ts/dominion-datastore/ ↪ table/*table_name*
Attribute of a relation	http://www.dlr.de/ts/dominion-datastore/ ↪ table/*table_name*/ ↪ column/*column_name*
Tuple of a relation	http://www.dlr.de/ts/dominion-datastore/ ↪ table/*table_name*/row/*row_id*
Attribute instance of a relation	http://www.dlr.de/ts/dominion-datastore/ ↪ table/*table_name*/ ↪ column/*column_name*/ ↪ row/*row_id*

An example of an URI, that references an attribute instance of attribute VELOCITY with a unique key 48 in relation `20100716_105523_drvstate`, would have the following form: `http://www.dlr.de/ts/dominion-datastore/table/20100716_105523_drvstate/column/VELOCITY/row/48`

In the same way, before table name the name of the database instance and the schema name are also included (Fig. 3). The inversion is well-defined and therefore URIs can be assigned to their original database elements. Described approach does not integrate values of relational bulk data itself into RDF, since they are not required (cf. Sect. 2.3).

4 User Interface

The graphical user interface allows to visualize semantic annotations. During sequent processing steps it helps to understand temporary results. Fig. 3 shows a screenshot of the application called *Semantic Database Browser*. In the left window a schema, table or column can be chosen. The main window shows selected tables. Database elements, which have annotations, are highlighted with a colour. When such an annotated element is selected, the bottom window shows the annotations as a tree. In Fig. 3 the column VELOCITY is selected in the left window.

The tree representation was selected, as it is very compact, especially compared to graph visualizations. Since only the really needed sub-tree is fetched from memory model, this approach preserves resources. However, it must be considered, that only outgoing edges of the graph model are drawn. If there are cycles in the graph and the tree is expanded, nodes can appear multiple times. In front of each tree element is a symbol, which indicates the type of the node

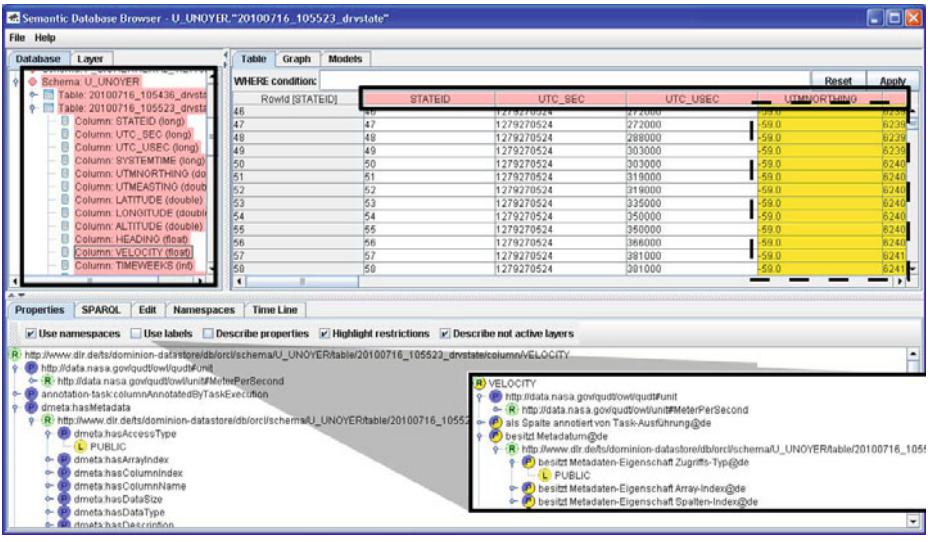

Fig. 3. User interface for interactive working with annotations including highlighted elements and a montage showing nationalized labels

(named resource, blank node, property or literal). The option *describe properties* lets the tree also show properties of properties. In this way i.e. domain- and range-restrictions of properties can be viewed. Another feature is to optionally show labels (identified by `rdfs:label` [10]) instead of resource names (URIs or anonymous ids). In this case a literal is chosen, which best matches the language chosen in the user interface (montage Fig. 3). Using the context menu it is also possible to directly open URI-resources with the associated program, i.e. a web-page in a semantic wiki with the web browser.

Since annotations are stored in different models for each bulk data table (cf. Sect. 3.2), they are loaded on demand for the currently shown table. For that reason interactions with annotations in the interface are generally performed on the base graph model G_B. This graph is the union of the actually used graphs. That at least includes the generic graph G_G, containing annotations for database instance, schemas and table schema. Annotations on bulk data, which are always stored in new tables, are managed in separate RDF models. For each table T_i there exists a graph model G_{T_i}. Graph G_B is manipulated during runtime and is the union of generic annotations G_G and currently viewed table T_j: $G_B = G_G \cup G_{T_\text{j}}$. Since G_G only growths very moderately and most annotations are managed in the different G_{T_i}, the scalability of the approach for many experiments with their measurement tables is assured. If one table can be processed, this is also possible for a great many of them.

It can be necessary to reference external knowledgebases G_{OWL_i} in form of OWL documents. Therefor $\bigcup_{i \in I} G_{OWL_i}$ with $i \in I$ is the set of relevant OWL documents. As consequence G_{OWL_i} can be added as a sub-graph to G_B, with the

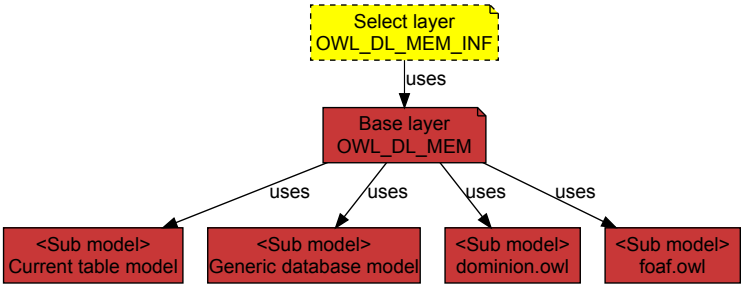

Fig. 4. Relations of a layer with the base graph and sub-graphs

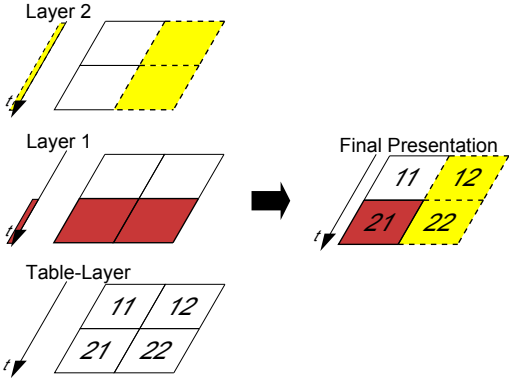

Fig. 5. Overlay of layers for the presentation of database elements

result $G_B = G_G \cup G_{T_j} \cup \bigcup_{i \in I} G_{OWL_i}$. In the case of measurement data handling an additional OWL document primarily contains additional schema knowledge, as assertional knowledge concerning database elements is maintained in G_{T_i}.

The resulting graph of graphs is shown in Fig. 4 in the bottom half, where sub-graphs are labelled with full text names. This graphical presentation is available in the main window accessible by a tab. The user can interact with the graph, as he can remove sub-graphs or request some statistics about them. For further support of visualization *layers* are offered, which are configured by the user in the left window (accessed by a tab). A layer is either the graph G_B as complete union of the included knowledge bases (base layer) or a view derived from G_B with an additional layer name and colour (Fig. 4 in the upper half).

If a database element is contained in a layer, it is highlighted in the specific colour of the layer. Furthermore, layers have an order defined by the user. An element contained in several layers is shown in the colour of the layer with the greatest index (Fig. 5). In this way, a layer concept is realized similar to be found in geo information systems. Derived graphs for layers are created using a reasoner or SPARQL queries (CONSTRUCT-, DESCRIBE- or SELECT) on the

base layer. The result set of a SELECT-query is handled as a graph consisting only of nodes and without edges. In Fig. 3 the base layer with all annotated database elements is shown in the background and results in a (red) colouring of schemas, tables and columns (highlighted by two solid boxes). Cells in the UTMNORTHING-column are contained in a layer based on a SELECT-query and are therefore highlighted (yellow, surrounded by a dashed box).

Other features in the user interface cover manually editing annotations and management of namespaces. By using SPARQL individual layers can be searched. The corresponding graph of a selected resource and neighbouring nodes in a given radius can be visualized. A in this way displayed graph also reflects the common colouring of the different layers.

For another kind of annotation presentation the ordering of database rows is necessary. The user interface therefore uses the natural order of the unique key for the tables (cf. Sect. 3.3). In case of measurement data, rows are ordered by time. This kind of presentation is also in terms of colour indicated in Fig. 5 for layers 1 and 2 along t-axis. At this, for every layer row- and cell-annotations are projected on t-axis. As a result, for each layer a kind of bar chart is drawn, to visualize in which intervals annotations are existent (Fig. 6). Since layers can be freely chosen, visual comparisons are possible.

The visual interface, which is described in this section, is in a prototypic but fully operational state.

5 Example Application Cases

This section presents two usage examples for the previously introduced annotations for measurement data.

5.1 Data Quality Management

Data quality has an essential impact for the considered analysis process. An automated service checks recorded measurement data against its specification. The specification is provided by the Dominion vehicular middleware [8] during experiments and is stored as annotations in the Dominion-DataStore. Fig. 3 partially shows these specifications as annotations in namespace `dmeta`.

As result of the quality checks, there are (micro-)annotations generated exactly for those cells, which violate their specification. Those cells are shown in the same figure in the UTMNORTHING column and highlighted by a separate layer (yellow). These annotations are linking the violated specifications, to allow searching and presenting different kinds of violations.

5.2 Description of Manoeuvres

The second application case is inferred from the IMoST project [4]. This project focuses on merging into traffic on highways. Basis for analysis are test drives from driving studies of DLR.

An annotation service is triggered to analyse measurement data of a test drive and extract relevant information. In this way (micro-)annotations *blink left*, *near centreline* and *vehicle ahead lost* are created for complete rows. These annotations themselves are divided in the categories *action* and *environmental event*. Annotation *blink left* is always generated, when the appropriate measurement variable shows a 1 to indicate, that the direction indicator has been activated. If the variable for the distance of the vehicle to the centreline is less than 0.3 meters, the annotation *near centreline* is created. When the distance detection to a leading vehicle suddenly looses the vehicle ahead, the service creates the *vehicle ahead lost* annotation. In this case it can be expected, that either the own or the foreign vehicle has performed a lane change. Generally, automated services are just one possibility to create annotations. Alternatives are the manually creation using a video rating or completely automated approaches like data mining.

In this way, just few of the rows get annotations, since most time nothing important happens (depending on the experiment scenario). So described annotations are used as a memo or notepad mechanism, to tag relevant database elements with information for further processing. Fig. 6 shows a time line representation (cf. Sect. 4) for selected events of an experiment. In this picture *blink left* and *near centreline* are layers containing corresponding annotations, which are selected by a SPARQL-query from base layer.

Fig. 6. Time line representation from user interface for the base layer and two layers containing selected events

In the following steps of semantic enrichment every analysis step builds on the previous ones. Thus, in every step the point of view gets more abstract and more comprehensible for domain specialists. Below, the concept *overtaking* is defined. An overtaking manoeuvre occurs, if the events *blink left*, *near centreline* and *vehicle ahead lost* arise at the same time. Based on this, an analyst can search for all these events in an analysis tool and then directly visualize the searched situation. Technically this search is implemented using SPARQL-queries. In the formerly presented user interface PREFIX statements for registered namespaces are optional:

```
PREFIX ts: <http://www.dlr.de/ts/>
SELECT ?x WHERE {
  ?x ts:hasProperty ts:BlinkLeft .
  ?x ts:hasProperty ts:NearCentreline .
  ?x ts:hasProperty ts:VehicleAheadLost .
}
```

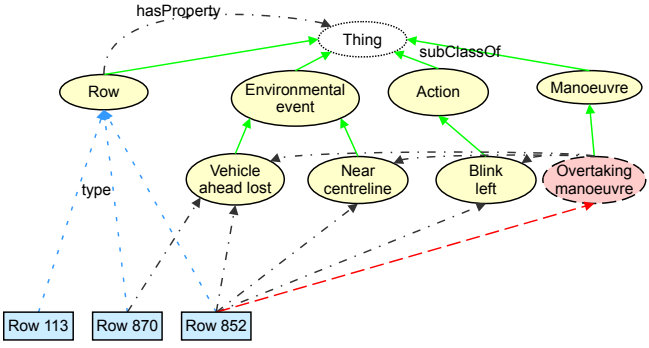

Fig. 7. Extract from the ontology to illustrate concepts of inheritance hierarchy

By using an ontology the concepts and their relationships are systemized. Fig. 7 shows an ontology, which covers concepts for events introduced in this section. Circles mark classes, which are organized in an inheritance hierarchy (green, solid arrows). The rectangular subjects are instances of type *row*. There are only three tuples (row identifier 113, 852 and 870) shown in this simplification. Black arrows with alternating dashes and dots as pattern mark rows with identified events.

Beside introduced sub-concepts in practice there can be much more (actions: i.e. brake, steer; environmental events: i.e. near to other vehicle, off the road). On base of these basic concepts, the concept class *manoeuvre* is introduced. Concrete actions and environmental events can form more complex manoeuvres (cf. [24], i.e. overtaking, turn off) and a more abstract view on street incidents is created. In the shown ontology, there is the newly introduced concept *overtaking manoeuvre* as a subclass of manoeuvre. Of course an algorithm in a service could identify the tuples, where the defined conditions for an overtaking manoeuvre are met, and annotate them. Alternatively, concept overtaking can be modelled in the ontology as an intersection of restrictions on necessary conditions. In Fig. 7 this is illustrated by the three arrows with alternating dashes and dots as pattern and overtaking as a subject. Modelling of restrictions is not shown in the figure. By using this modelling approach a reasoner can be applied accordingly. Its inference mechanisms deduce, in case that all conditions are complied to, an overtaking manoeuvre is existent. In Fig. 7, that is indicated by a red, dashed arrow. Hence, analysts can now directly identify overtaking manoeuvres using SPARQL.

As domain experts decide to redefine the concept overtaking, just the ontology must be changed and results can be inferred. So ontology can serve as an important and central pool of certain expert knowledge. Furthermore, it can formalize used vocabulary and concepts to help avoiding ambiguities and redundancies. Another advantage is, that ontologies can be modularized in sub-ontologies, so experts of a very specific sub-domain can be responsible for a sub-ontology. From the knowledge management's point of view this is very helpful. Bulk data of time

series is very static and remains unchanged, so it is stored in traditional, relational tables. But because metadata and domain knowledge are heavily evolving during analysis process and often have complicated structures, RDF is a suitable modelling approach for them. RDF storage isn't as efficient as relational storage, but as long as metadata is less in volume than bulk data, handling and storage is usually not a demanding task.

Modelling of temporal aspects in ontologies is still a question of research (i.e. [21,23]). But that doesn't matter for the presented application cases, since just aspects without temporal aspects are modelled in the ontology. Temporal aspects can be covered using other analysis techniques. In the presented case an algorithm identifies, where radar looses tracking of a vehicle ahead, and creates appropriate annotations.

Nevertheless, the use cases are currently under heavy development, to support current results in this area of interest. In future, continuous row annotations will be modelled using time points and intervals. In this way, number of annotations can be reduced a lot, by just annotating beginning and end of an interval for an occurred event or action. That feature was already used to create Fig. 6, since the two upper layers in the picture are rendered using just start and end points. SWRL will be used to define a finite state machine on top of these events and detect abstract manoeuvres. Advantage of SWRL rules is that they are much more expressive for reasoning purposes than plain ontologies. So, more complex reasoning can be performed on event relationships, what is challenging to cover just using conventional database technologies.

In the discussed use-case, created annotations are only used to search them using SPARQL. But annotations represent common results and should be also reused by other kinds of processing steps. In this way, working with more abstract concepts is iteratively introduced and analysts can work on a more semantic level.

6 Summary and Outlook

The presented article discusses an approach for data management of measurement data and semantic metadata to support an iterative data analysis process. Aim of this iterative process is to provide the results of each processing step as transparent as possible. The data in the data store can be flexibly annotated to support the continuous and iterative process of knowledge discovery, by using RDF annotations and micro-annotations for semantic enrichment incorporating ontologies.

The presented approach is a combination of advantages of classical relational databases with semantic annotations and ontologies for the analysis of sensor data. Using ontologies, concepts and vocabulary of the metadata can be systemized. For database elements annotations serve as a kind of memo or notepad mechanism, which help both analysts and algorithms processing the time series data. In this way table data can be arbitrarily linked and become a part of the Semantic Web. Also the user benefits of the possibility to use inferencing

software. The presented user interface allows an intuitive workflow with introduced annotations. Semantic annotations are flexible and well suited for this application of knowledge integration. By storing time structured data using a relational database, it can be easily accessed using standard database connectivity of used applications without any adaptations.

On the contrary there are also challenges in the presented approach. Using two technologies instead of one for accessing specific data increases complexity. That also influences modelling of considered data. The developer must consider whether to store data in a relational way or as annotations. Moreover, the speed of accessing annotations can be slower than a solely relational storage. A RDF-store allows to handle separate data models for the annotation instances. So, separate models can be used for different time structured bulk data tables to ensure scalability. Generally we promote to use a relational design for bulk data, which has a fix schema. For light weighted data and metadata, which constantly evolves and where the interpretation is in focus, we prefer annotations.

The use cases and ontologies are currently refined with a stronger focus on temporal aspects. Furthermore, the integration of automated services in a SWS has to be realized. A further medium-term objective is to polish the user interface and port it to the Eclipse Rich Client Platform for a productive deployment.

References

1. Aduna-Software: OpenRDF.org — ... home of Sesame. WWW (October 2008), `http://www.openrdf.org/`
2. Altintas, I., et.al.: Kepler: An extensible system for design and execution of scientific workflows. In: Scientific and Statistical Database Management (2004)
3. Barrasa, J., Óscar Corcho, Gómez-Pérez, A.: R2O, an Extensible and Semantically Based Database-to-ontology Mapping Language. In: Workshop on Semantic Web and Databases (2004)
4. Baumann, M., et al.: Integrated modelling for safe transportation — driver modeling and driver experiments. In: 2te Fachtagung Fahrermodellierung (2008)
5. Berners-Lee, T.: Relational Databases on the Semantic Web (1998), `http://www.w3.org/DesignIssues/RDB-RDF.html`
6. Bizer, C.: D2R MAP — A Database to RDF Mapping Language. In: World Wide Web Conference (2003)
7. Fancher, P., et al.: Intelligent cruise control field operational test (final report). Tech. rep., University of Michigan (1998)
8. Gačnik, J., et al.: DESCAS — Design Process for the Development of Safety-Critical Advanced Driver Assistance Systems. In: FORMS (2008)
9. Gertz, M., Sattler, K.U.: Integrating scientific data through external, concept-based annotations. In: Data Integration over the Web (2002)
10. Hayes, P.: RDF Semantics. W3C Recommendation (February 2004)
11. Herrera, P., et al.: Mucosa: A music content semantic annotator. In: Music Information Retrieval (2005)
12. Köster, F.: Datenbasierte Kompetenz- und Verhaltensanalyse — Anwendungsbeispiele im selbstorganisierten eLearning. In: OlWIR (2007)

13. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 1, 41–60 (2005)
14. Noyer, U., Beckmann, D., Köster, F.: Semantic annotation of sensor data to support data analysis processes. In: Semantic Authoring, Annotation and Knowledge Markup Workshop (SAAKM) (2009)
15. Noyer, U., Beckmann, D., Köster, F.: Semantic technologies and metadata systematisation for evaluating time series in the context of driving experiments. In: 11th International Protégé Conference, pp. 17 – 18 (2009)
16. Rothenberg, J.: Metadata to support data quality and longevity. In: Proceedings of the 1st IEEE Metadata Conference (1996)
17. Schaffert, S., Francois Bry, J.B., Kiesel, M.: Semantic wiki. Informatik-Spektrum 30, 434–439 (2007)
18. SourceForge.net. Jena — A Semantic Web Framework for Java (October 2008), http://jena.sourceforge.net/
19. Stanford Center for Biomedical Informatics Research: The protégé ontology editor and knowledge acquisition system. WWW (April 2009), http://protege.stanford.edu/
20. myGrid team. Taverna workbench project webseite (November 2009), http://taverna.sourceforge.net/
21. Tusch, G., Huang, X., O'Connor, M., Das, A.: Exploring microarray time series with Protégé. In: Protégé Conference (2009)
22. Velegrakis, Y.: Relational Technologies, Metadata and RDF, ch. 4, pp. 41–66. Springer, Heidelberg (2010)
23. Virgilio, R.D., Giunchiglia, F., Tanca, L. (eds.): Semantic Web Information Management: A Model-Based Perspective, ch. 11, pp. 225–246. Springer, Heidelberg (2010)
24. Vollrath, M., et al.: Erkennung von Fahrmanövern als Indikator für die Belastung des Fahrers. In: Fahrer im 21. Jahrhundert (2005)