FAIMUSS: Flexible Data Transformation to RDF from Multiple Streaming Sources

Georgios M. Santipantakis
Department of Digital Systems
University of Piraeus
18534 Piraeus, Greece
gsant@unipi.gr

Apostolos Glenis
Department of Digital Systems
University of Piraeus
18534 Piraeus, Greece
apostglen46@gmail.com

Nikolaos Kalaitzian
Department of Digital Systems
University of Piraeus
18534 Piraeus, Greece
nikoskalai@gmail.com

Akrivi Vlachou
Department of Digital Systems
University of Piraeus
18534 Piraeus, Greece
avlachou@aueb.gr

Christos Doulkeridis
Department of Digital Systems
University of Piraeus
18534 Piraeus, Greece
cdoulk@unipi.gr

George A. Vouros
Department of Digital Systems
University of Piraeus
18534 Piraeus, Greece
georgev@unipi.gr

ABSTRACT

In this paper, we present *FAIMUSS* a tool for data transformation from a wide variety of heterogeneous streaming and archival sources to RDF. This is a typical situation in the analysis of mobility data, such as maritime and aviation, where streaming position data of moving objects need to be associated with static information (such as crossing sectors, protected geographical areas, weather, etc.) in order to provide semantically enriched trajectories. *FAIMUSS* is designed to perform "near-to-the-sources" integration, by interaction between a streaming source and an archival source, thus generating linked RDF graph fragments. Most of the existing approaches operate either on streaming or on static sources, thus fail to address our problem setting. In addition, *FAIMUSS* supports reusable user-defined functions that are applied to input data and achieve the desired data transformations and cleaning. We demonstrate our prototype by using data from the maritime and aviation domains.

1 INTRODUCTION

The Resource Description Framework (RDF) enables the description of physical entities as resources, in favor of data interoperability, integration and exchange. As a result, a wide range of solutions exists for converting a data source to RDF, based on a schema specified by RDFS or OWL profiles. For instance, R2RML [5] is a mapping language that translates SPARQL queries to SQL (and vice-versa). Ontology Based Data Access (OBDA) [2, 3] also focuses on relational data sources, but again requires advanced knowledge to define the (usually complicated) mappings between the data source and the ontology schema. SPARQL-Generate [10] provides an extension of SPARQL 1.1 that allows generation of RDF fragments from documents. Solutions tailored for specific RDF stores also exist, for example Virtuoso Cartridge[1], however they are tied to a particular product.

Moreover, the amount of streaming data sources has increased rapidly in recent years, and such sources pose new challenges for data integration [13]. Apart from the obvious performance challenge raised by high stream rate, streaming data are typically semi-structured, and contain noisy data. As such, the problem of integrating streaming data sources is still open, and there is a lack of flexible tools that transform streaming data to RDF, integrate it with external sources, and are easily configurable to new sources.

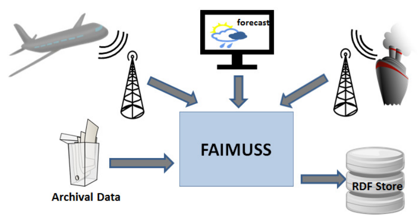


Figure 1: *FAIMUSS* consumes streaming and archival sources and produces (linked) RDF data.

Our work is motivated by the need to produce enriched representations of moving object trajectories expressed in RDF, by ingesting streaming surveillance data in real-time and associating it with other sources, both streaming and archival. Figure 1 illustrates this process, focusing on mobility data in maritime [4] and aviation domains recording positions of moving objects, such as vessels and aircrafts. However, this is simply for demonstration reasons, as our work is readily applicable to any other category of streaming data, including social data consumed through an API (e.g., Twitter API).

In this paper, we present *FAIMUSS* (Flexible dAta TransformatIon to RDF from MUltiple Streaming Sources), a flexible, user-friendly system for end-to-end data transformation of streaming data to RDF and integration with external sources. Given an ontology, *FAIMUSS* converts streaming data (but also other data sources) into RDF triples, which are also integrated with other archival data sources. We employ the datAcron ontology [12] for representing trajectories at multiple levels of analysis. However, the system imposes no constraints on the use of a specific ontology, while it supports a wide range of input source data formats. Salient features of *FAIMUSS* include its flexibility and user-friendliness: the process of triple generation is determined by a *Graph Template* that is easily edited by the user with the support of a rich editor. Furthermore, *FAIMUSS* supports "near-to-the-sources" integration, by generating *linked* RDF graph fragments during the process of data transformation. Finally, the implementation of *FAIMUSS* also addresses scalability issues and aims at high performance.

[1]https://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtProgrammerGuideRDFCartridge

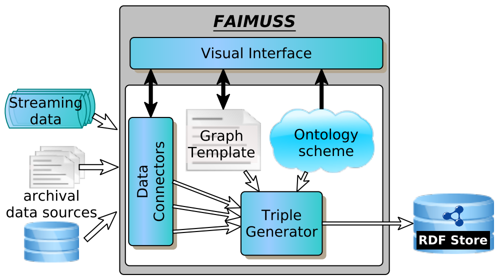
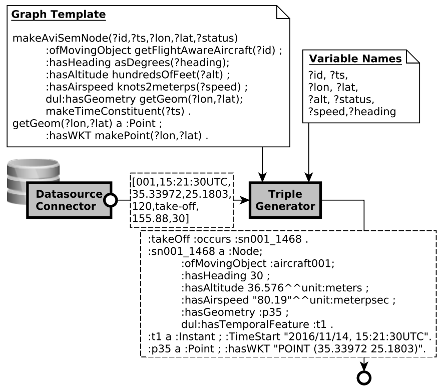


Figure 2: System architecture of *FAIMUSS*. White arrows indicate data flows and black arrows indicate user interactions.



```
makeAviSemNode(?id,?ts,?lon,?lat,?status)
        :ofMovingObject getFlightAwareAircraft(?id) ;
        :hasHeading asDegrees(?heading);
        :hasAltitude hundredsOfFeet(?alt) ;
        :hasAirspeed knots2meterps(?speed) ;
        dul:hasGeometry getGeom(?lon,?lat);
        makeTimeConstituent(?ts) .
getGeom(?lon,?lat) a :Point ;
        :hasWKT makePoint(?lon,?lat) .
```

```
?id, ?ts,
?lon, ?lat,
?alt, ?status,
?speed,?heading
```

```
[001,15:21:30UTC,
35.33972,25.1803,
120,take-off,
155.88,30]
```

```
:takeOff :occurs :sn001_1468 .
:sn001_1468 a :Node;
        :ofMovingObject :aircraft001;
        :hasHeading 30 ;
        :hasAltitude 36.576^^unit:meters ;
        :hasAirspeed "80.19"^^unit:meterpsec ;
        :hasGeometry :p35 ;
        dul:hasTemporalFeature :t1 .
:t1 a :Instant ; :TimeStart "2016/11/14, 15:21:30UTC".
:p35 a :Point ; :hasWKT "POINT (35.33972 25.1803)".
```

Figure 3: A Graph Template and Variable Vector example in *Triple Generator*.

2 SYSTEM ARCHITECTURE

In this section, we describe the system architecture that consumes data from a variety of data sources and outputs RDF graph fragments. This process is designed in a generic fashion, so as to accomplish multiple objectives, including flexibility, extensibility and scalability. *FAIMUSS* comprises the following main components, a) the *Data Connector*, b) the *Triple Generator*, and c) the *Visual Interface*. Figure 2 illustrates the overall system and the individual components and their interactions, which will be described in detail. The prototype system has been implemented in Oracle Java 8 (64-bit) and is platform-independent.

2.1 Data Connector

The *Data Connector* component implements the functions that accept data from an individual data source. Moreover, it performs data conversion on values of specific fields as provided by the source and basic data cleaning operations. As the data sources can vary significantly in terms of representation, size, rate of data access, and noise, *FAIMUSS* provides a Data Connector for each type of source. The implemented system currently supports data consumption from a wide range of sources/formats, such as: a) CSV format, b) direct access to databases, c) JSON messages, d) XML files, e) METAR/SPECI weather reports from offline/continuous feeds from online services, f) binary (GRIB2) files for weather reports, g) SPARQL endpoints to online open data, h) ESRI shape-files to convert information about spatial object to entities and relations of the ontology, and i) proprietary formats of streaming. The list of supported data formats can be easily extended, to support the inclusion of any other data format required in the future.

As an abstraction of the data source in hand, the Data Connector considers all data sources as streams, i.e. it consumes data record-by-record (or tuple-by-tuple), to be processed with minimal latency. This data access model makes no distinction on the nature of data (e.g. archival data or streaming data). In addition, it minimizes the memory footprint of Data Connectors, and enables scalability and parallelization as a multi-threaded process, both for a single source and across sources.

2.2 Triple Generator

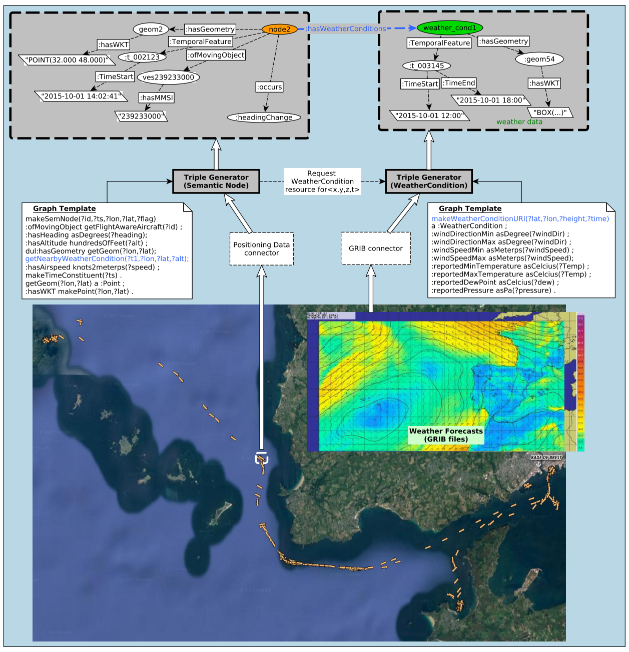
The role of *Triple Generator* is to consume the data provided by the Data Connector and generate the corresponding set of triples. This procedure depends on configuration files providing a Graph Template $\mathbf{G}_T$, and a vector of variable names $\mathbf{V}$. The vector $\mathbf{V}$ contains the variables that appear in the Graph Template, and binds the values of the input record to the variables in $\mathbf{G}_T$. The Graph Template consists of a set of *triple patterns*, i.e. any of the three elements in the triple (subject, predicate, object) can be replaced by a variable. The Graph Template differs from the standard RDF Graph Pattern, in that variables can be used as arguments in functions, to compute dynamic values at runtime. Furthermore, the implementation of *Triple Generator* is parallelized in a multi-threaded process, due to the record-by-record conversion to triples.

Figure 3 illustrates the application of a Graph Template on a single record from a stream of aircraft surveillance data to generate the corresponding triples. Specifically, the Triple Generator consumes the data provided from the Data Connector and constructs a new resource (:sn001_1468) of type :Node, based on the Graph Template depicted at the top of the figure. This new resource is the spatio-temporal representation of the moving object (aircraft). The Triple Generator also relates the :sn001_1468 to the resource representing an aircraft (using the property :ofMovingObject), as well as information regarding the status, the altitude and position of the moving object.

Figure 4 shows another example of data transformation where two data sources are processed and integrated, by having two instances of Triple Generator interact with each other. The positioning data connector provides surveillance data of vessels to a Triple Generator instance, similar to the previous example. The GRIB connector accesses binary files that contain weather forecasts and can extract the related weather attributes for a given spatio-temporal position. As soon as a new position is received, a request is made to the Triple Generator instance that retrieves the weather information and provides it in RDF representation based on the respective Graph Template. This allows data transformation of selected weather information, namely this corresponding to the area defined by the positioning information. More interestingly, the weather Triple Generator returns the URI of the weather condition to the positioning Triple Generator, which can then create the *:hasWeatherCondition* property and associate the position with the weather. This is an example of lightweight, "near-to-the-sources" integration.

2.3 Visual Interface

The *Visual Interface* enables the configuration of Data Connector and Triple Generator, the preview of input data, the visualization of output triples on a map, and output validation. Specifically, the user can select the ontology to be used, which will provide



Graph Template

```
makeSemNode(?id,?ts,?lon,?lat,?flag)
:ofMovingObject getFlightAwareAircraft(?id) ;
:hasHeading asDegrees(?heading);
:hasAltitude hundredsOfFeet(?alt) ;
dul:hasGeometry getGeom(?lon,?lat);
getNearbyWeatherCondition(?t1,?lon,?lat,?alt);
:hasAirspeed knots2meterps(?speed) ;
makeTimeConstituent(?ts) .
getGeom(?lon,?lat) a :Point ;
:hasWKT makePoint(?lon,?lat) .
```

Graph Template

```
makeWeatherConditionURI(?lat,?lon,?height,?time)
a :WeatherCondition ;
:windDirectionMin asDegree(?windDir) ;
:windDirectionMax asDegree(?windDir) ;
:windSpeedMin asMeterps(?windSpeed) ;
:windSpeedMax asMeterps(?windSpeed);
:reportedMinTemperature asCelcius(?Temp) ;
:reportedMaxTemperature asCelcius(?Temp) ;
:reportedDewPoint asCelcius(?dew) ;
:reportedPressure asPa(?pressure) .
```

Figure 4: Example of data transformation and "near-to-the-sources" integration (maritime domain).

the vocabulary for the Triple Generator. Obviously, the ontology should be taken into consideration when editing a Graph Template, to guarantee consistency of the generated triples.

Figures 5 and 6 provide screenshots of the Visual Interface using data from the aviation domain. As illustrated in Figure 5(a), the user can select and configure the data source that will be processed (e.g., specify the IP and port of remote endpoints, or folder/file for local access), and the type of Data Connectors that should be used for consuming the data. A sample of the data available in the configured data source, for visual inspection. The user can also select the fields of the input data source that should be converted to triples, and specify the *Variables Vector* to be used in the Graph Template. Figure 5(b) shows the editor for composing the Graph Template. On the right side, the set of available functions is provided, from which the user can insert in the editor by a double click. Also, the Variables corresponding to the input source are also available. The editor uses different font color for functions and variables to improve readability. In addition, import (export) of Graph Templates from (to) disk is supported. It should also be mentioned that automatic validation of a sample (or the entire output) of triples generated w.r.t. the provided Graph Template is supported. Figure 6 depicts a map-based interface provided by *FAIMUSS*, which enables the illustration of spatial and spatio-temporal RDF data. In this example, it depicts trajectories of moving objects (aircrafts), which correspond to RDF data generated by our system.

3 DEMONSTRATION SCENARIOS

We have used *FAIMUSS* in two domains, maritime and aviation, thus showing the generic nature of our system and its applicability in different domains. The selection of these domains originates from our involvement in the H2020 research project datAcron (http://datacron-project.eu/), where one of the objectives is to *integrate heterogeneous and voluminous archival data with streaming data sources.*

Scenario 1: Flexible Integration of New Sources. We intend to show in the demo that a new streaming data source can be added to *FAIMUSS*, after a small set of steps that can be performed by selections from the GUI. The aim is to demonstrate the ease of use of *FAIMUSS* in practice. In more detail, the streaming data source is going to be surveillance data from vessels travelling in the Mediterranean Sea. Each record in the stream contains the spatio-temporal position of a moving object, along with its unique identifier. From the GUI of *FAIMUSS*, we are going to input the necessary information for establishing a connection (IP address, port, and authentication details). Thereafter, a Graph Template is going to be specified from an editor of the GUI, which guides the Triple Generator and determines the transformation of stream records into RDF triples. As already mentioned, the editor facilitates the composition of Graph Templates even for moderately familiar users, not only by loading existing Graph Templates that can be modified, but also by providing access to lists of available functions and variables that generate Graph Template code in the editor.

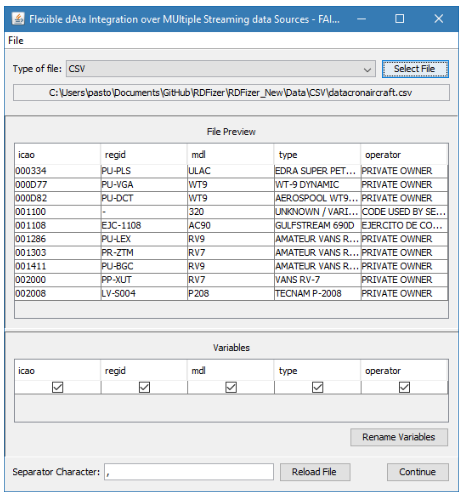
Scenario 2: Trajectory Enrichment. We will show in the demo how we can generate enriched trajectories from moving objects' positions, where the positional information can be integrated with arbitrary data sources. Continuing scenario 1, the spatio-temporal position of each vessel is going to be integrated with weather conditions. Weather conditions are available as forecasts for a temporal period (e.g. three hours) in the form of large binary files (GRIB2 format), and mainly define 3-dimensional cells that contain multiple variables describing weather condition (e.g., temperature, humidity, pressure, etc.). In this scenario, upon receipt of a spatio-temporal position of a moving object, the Triple Generator instance responsible for surveillance data is going to produce RDF triples. By communication with a Triple Generator instance for weather data, which produces weather-related triples, the resulting positional RDF data is linked with weather information. In this way, we generate enriched trajectories of moving objects represented in RDF, which will be illustrated on a map for visual inspection, and provide additional (weather-related) information for each spatio-temporal position of a moving object.

4 RELATED WORK

A wide list of data transformation and conversion tools exists, and such tools[2] have been evaluated during the design of our system, however these are mostly ad-hoc solutions tailored for converting archival data in specific file formats. For example, Omnidator [1] converts any CSV or HTML online file into RDF triples. GeoTriples [8] employs automatically generated R2RML mappings given a source and a configuration, but demands the use of a relational database, which is not applicable for streams. SPARQL-Generate [10] provides an extension of SPARQL 1.1 that allows generation of RDF fragments from documents.

Data integration over streaming data poses new challenges compared to traditional data integration [13]. The Graph of Things [9] targets an IoT setting where many sources provide data for integration and querying, and supports spatial and temporal data, but it is not optimized for mobility data. Also related to our work is StreamLoader [11] which provides a user-friendly,

[2]https://www.w3.org/wiki/ConverterToRdf


Flexible dAta Integration over MUltiple Streaming data Sources - FAI...
File
Select File
File Preview
Variables
Rename Variables
Reload File
Continue

Type of file: CSV

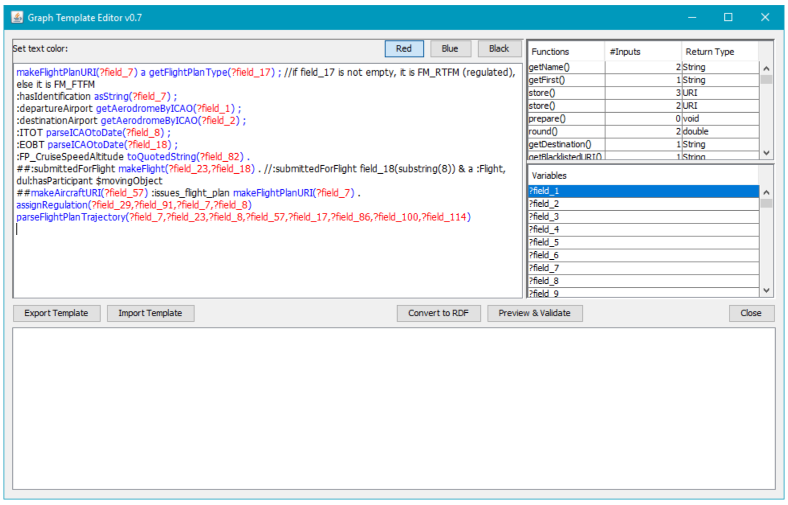
C:\Users\pasto\Documents\GitHub\RDFizer\RDFizer_New\Data\CSV\datacronaircraft.csv

icao	regid	mdl	type	operator
000334	PU-PLS	ULAC	EDRA SUPER PET...	PRIVATE OWNER
000D77	PU-VGA	WT9	WT-9 DYNAMIC	PRIVATE OWNER
000D82	PU-DCT	WT9	AEROSPOOL WT9...	PRIVATE OWNER
001100	-	320	UNKNOWN / VARI...	CODE USED BY SE...
001108	EJC-1108	AC90	GULFSTREAM 690D	EJERCITO DE CO...
001286	PU-LEX	RV9	AMATEUR VANS R...	PRIVATE OWNER
001303	PR-ZTM	RV7	AMATEUR VANS R...	PRIVATE OWNER
001411	PU-BGC	RV9	AMATEUR VANS R...	PRIVATE OWNER
002000	PP-XUT	RV7	VANS RV-7	PRIVATE OWNER
002008	LV-S004	P208	TECNAM P-2008	PRIVATE OWNER

icao	regid	mdl	type	operator
☑	☑	☑	☑	☑

Separator Character: ,

(a) Configuration of Data Connector



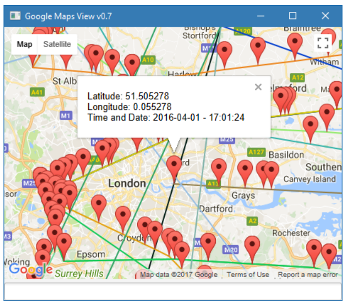
```
makeFlightPlanURI(?field_7) a getFlightPlanType(?field_17) ; //if field_17 is not empty, it is FM_RTFM (regulated),
else it is FM_FTFM
:hasIdentification asString(?field_7) ;
:departureAirport getAerodromeByICAO(?field_1) ;
:destinationAirport getAerodromeByICAO(?field_2) ;
:ITOT parseICAOtoDate(?field_8) ;
:EOBT parseICAOtoDate(?field_18) ;
:FP_CruiseSpeedAltitude toQuotedString(?field_82) .
##:submittedForFlight makeFlight(?field_23,?field_18) . //:submittedForFlight field_18(substring(8)) & a :Flight,
dul:hasParticipant $movingObject
##makeAircraftURI(?field_57) :issues_flight_plan makeFlightPlanURI(?field_7) .
assignRegulation(?field_29,?field_91,?field_7,?field_8)
parseFlightPlanTrajectory(?field_7,?field_23,?field_8,?field_57,?field_17,?field_86,?field_100,?field_114)
```

Functions	#Inputs	Return Type
getName()	2	String
getFirst()	1	String
store()	3	URI
store()	2	URI
prepare()	0	void
round()	2	double
getDestination()	1	String
getBlacklistedURI()	1	String

Variables
?field_1
?field_2
?field_3
?field_4
?field_5
?field_6
?field_7
?field_8
?field_9

(b) Editor for Graph Templates

Figure 5: Screenshots of the Visual Interface of *FAIMUSS* (aviation domain).



Latitude: 51.505278
Longitude: 0.055278
Time and Date: 2016-04-01 - 17:01:24

Figure 6: Map-based visualization.

web-based environment for ETL (Extract-Transform-Load) of streaming data from heterogeneous sensors. However, it deviates from our objective, namely to provide representations of streaming data in RDF, thus allowing linking with other external (web) sources. Moreover, it does not explicitly address the domain of moving objects and trajectories thereof, which is the motivation of our work. Only recently, Optique [6, 7] proposes an approach for integration of streaming with static relational data. Again, this problem is much narrower than the one addressed by our work, since we make no assumptions on the availability of a relational database nor do we impose such requirements. None of the existing approaches targets streaming mobility data of vessels or aircrafts explicitly. Moreover, our work goes one step further, by introducing a system for flexible data transformation to RDF, with a particular focus on mobility data, also supporting linking of generated RDF graph fragments.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we present the *FAIMUSS* system for flexible data transformation of streaming and archival data sources to RDF. Despite the generic and extensible design of our system, the focus of our attention is on moving objects and integrating their spatio-temporal positions with a variety of heterogeneous data sources, including weather conditions, spatial areas of interest, as well as external databases. In our future work, we intend to fully parallelize our system, thus providing a scalable solution to the problem of data integration from different data sources.

ACKNOWLEDGMENTS

This work has been supported by project datAcron, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 687591.

REFERENCES

[1] Omnidator. http://omnidator.appspot.com/.
[2] C. Bizer and A. Seaborne. D2RQ-treating non-RDF databases as virtual RDF graphs. In *Proc. of ISWC*, 2004.
[3] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, pages 1–17, 2016.
[4] C. Claramunt, C. Ray, E. Camossi, A. Jousselme, M. Hadzagic, G. L. Andrienko, N. V. Andrienko, Y. Theodoridis, G. A. Vouros, and L. Salmon. Maritime data integration and analysis: recent progress and research challenges. In *Proc. of EDBT*, pages 192–197, 2017.
[5] S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language. https://www.w3.org/TR/r2rml/, 2012.
[6] E. Kharlamov, S. Brandt, E. Jiménez-Ruiz, Y. Kotidis, S. Lamparter, T. Mailis, C. Neuenstadt, Ö. L. Özçep, C. Pinkel, C. Svingos, D. Zheleznyakov, I. Horrocks, Y. E. Ioannidis, and R. Möller. Ontology-based integration of streaming and static relational data with optique. In *Proc. of SIGMOD*, pages 2109–2112, 2016.
[7] E. Kharlamov, Y. Kotidis, T. Mailis, C. Neuenstadt, C. Nikolaou, Ö. L. Özçep, C. Svingos, D. Zheleznyakov, S. Brandt, I. Horrocks, Y. E. Ioannidis, S. Lamparter, and R. Möller. Towards analytics aware ontology based access to static and streaming data. In *Proc. of ISWC*, pages 344–362, 2016.
[8] K. Kyzirakos, I. Vlachopoulos, D. Savva, S. Manegold, and M. Koubarakis. GeoTriples: a tool for publishing geospatial data as RDF graphs using R2RML mappings. In *TC/SSN@ISWC*, pages 33–44, 2014.
[9] D. Le-Phuoc, H. N. M. Quoc, H. N. Quoc, T. T. Nhat, and M. Hauswirth. The Graph of Things: A step towards the Live Knowledge Graph of connected things. *Web Semantics: Science, Services and Agents on the World Wide Web*, 37:25–35, 2016.
[10] M. Lefrançois, A. Zimmermann, and N. Bakerally. A SPARQL extension for generating RDF from heterogeneous formats. In *Proc. of ESWC*, pages 35–50, 2017.
[11] M. Mesiti, L. Ferrari, S. Valtolina, G. Licari, G. L. Galliani, M. Dao, and K. Zettsu. StreamLoader: An Event-Driven ETL System for the On-line Processing of Heterogeneous Sensor Data. In *Proc. of EDBT*, pages 628–631, 2016.
[12] G. Santipantakis, G. Vouros, C. Doulkeridis, A. Vlachou, G. Andrienko, N. Andrienko, G. Fuchs, J. M. C. Garcia, and M. G. Martinez. Specification of semantic trajectories supporting data transformations for analytics: The datAcron ontology. In *Proc. of Semantics*, 2017.
[13] N. Tatbul. Streaming data integration: Challenges and opportunities. In *Proc. of ICDE*, pages 155–158, 2010.