Using automotive property graph-based data models in a knowledge graph

Aidan O'Mahony[0000−0001−8413−9656], Alan Barnett[0000−0001−9658−2380], and Merry Globin[0000−0002−0573−3962]

Dell Technologies, Cork, Ireland {`firstname.surname`}`@dell.com`

Abstract. Data vocabularies facilitate the organization and retrieval of knowledge. As the volumes of data being generated in the internet age is exploding, the need for structuring data such that automated organization can occur is becoming even more crucial to manage this data. A popular method of searching this data in the context of a vocabulary is through the use of graph theory. This paper describes the scenario where, through the use of a data vocabulary for describing automotive data based on labeled property graphs, data is modified such that it is stored in a knowledge graph. The difficulties that were encountered in this effort and how they were overcome are also discussed.

Keywords: Vocabulary · Triple Store · Property Graph · Knowledge Graph.

1 Introduction

Knowledge graphs can be defined as databases in which the data is modeled as graphs, relying on schemas and vocabularies, and data operations are expressed through graph-oriented operations [1] [2]. The two most mature types of graph technologies are labeled property graphs and semantic graphs [3] and in order to make use of these there is also the need for a data model, syntax, vocabulary and entity IDs. One of the primary providers of data models is the FIWARE foundation [4], which provides smart data models for property graphs, and Schema.org, which provides semantic graph-based data models. The challenge is when one graph-based data model is more expressive than the equivalent model in the other graph type. This problem was encountered as part of the Horizon 2020 project MOSAICrOWN [5] when attempting to make use of vehicle data in a structured manner. The goal of MOSAICrOWN is the creation of a privacy preserving data market. Vehicle and driver information is ingested into the data market where metadata is stored in a graph and the non-metadata is stored in an object store. A high-level diagram of the data market is presented in Figure 1. In this paper the problem of associating the metadata with context and storing it in a graph is also described.

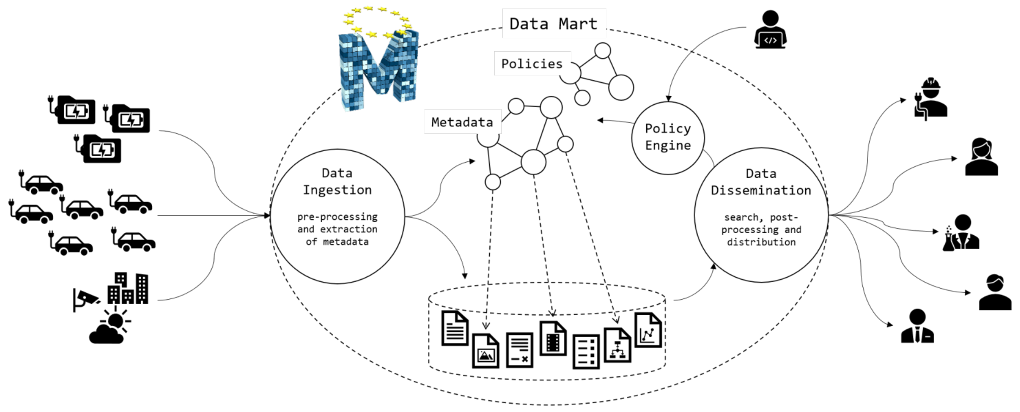

Data Mart
Policies
Metadata
Policy Engine
Data Ingestion
pre-processing and extraction of metadata
Data Dissemination
search, post-processing and distribution

Fig. 1. The MOSAICrOWN data market concept

Section 2 presents the related work already carried out in the area of property graph and semantic graph ontologies. Section 3 discusses in more detail what the technologies are, such as NGSI-LD and JSON-LD. In Section 4, the problem is discussed in more detail and also how the problem was overcome. Finally, in Section 5 conclusions are discussed.

2 Related Work

2.1 Knowledge Graphs

There are two main knowledge graph types: labeled property graphs and semantic graphs. Both types are comprised of nodes and edges, where nodes represent objects and edges represent those object's relationships to each other. Property graph nodes contain key-value pairs (properties) which are used to represent an entity/object, and use node traversal and querying languages such as "Gremlin" or "Cypher". The edges of both types of graph are represented as directed labeled edges between nodes; an arrow indicates the direction of the relationship. The supported datatypes of properties can vary depending on the property graph implementation and can range from primitive datatypes to nested objects and arrays. Additionally, some property graphs support property annotations, sometimes referred to as property metadata. Semantic graphs, such as Resource Description Framework (RDF), which uses open-standard SPARQL [6] query language. RDF graphs use a "triple" structure to represent the graph. The effect of this structure is that vertices on the graph are used to represent entities and literals, and edges are the relationships between them.

Labeled Property Graphs A simple explanation of using an ontology to build a property graph schema is to imitate the process of mapping a relational schema from an entity relationship diagram. To do this one must map all concepts within the ontology to a node of the schema, as well as map the relationships to edges of the schema [7]. The labeled property graphs data model is commonly used for the purpose of encoding data into knowledge graphs due to its speed of graph traversals, data storage mechanism efficiency and its adaptability in modeling domains. Real-time queries can be particularly resource consuming and such queries in the context of knowledge graphs are no exception. Labeled property graphs are a method of facilitating analytical operations on graphs at scale [7]. Edges and nodes are used here to describe cross-entity relations. Labeled Property Graphs use a property value structure corresponding to entities which models them in a concise format [6]. Whilst both property graphs and semantic graphs have directed labeled edges, only certain property graphs allow annotations to be associated with the label. Figure 2 presents an example of a labeled property graph.

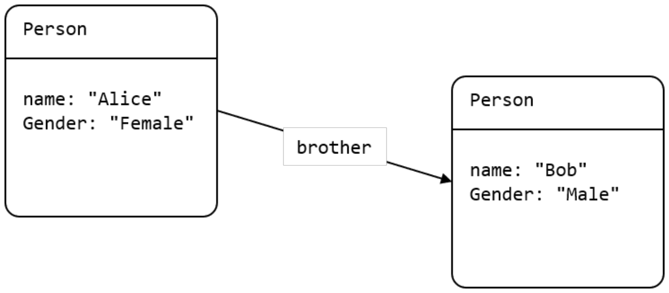


Fig. 2. Labeled Property Graph

Semantic Graphs Semantic graph nodes can be internationalized resource identifiers (IRIs), blank nodes, or literals : IRIs are a generalization of universal resource identifiers (URIs), and only simple literal datatypes such as strings or dates are supported. Properties of an object are implemented in the graph as individual linked nodes. Semantic graphs, like property graphs, have directed edge labels. Edge labels in semantic graphs are IRI defined predicates that allow inferences to be drawn between nodes based on rules defined in the ontology. In the example illustrated in Figure 3, the "brother" predicate linking the subject Alice to the object Bob, the "gender" predicate linking the subject Alice to the object Female, and rules from a suitable ontology, can be used to infer that Alice is also the sister of Bob and that Bob is Male.

2.2 Resource Description Framework

Resource description framework (RDF) is a common standard for defining Semantic Graphs, the alternative graph data model to Labeled Property Graphs described earlier in this section. RDF enables the exchange of data, particularly

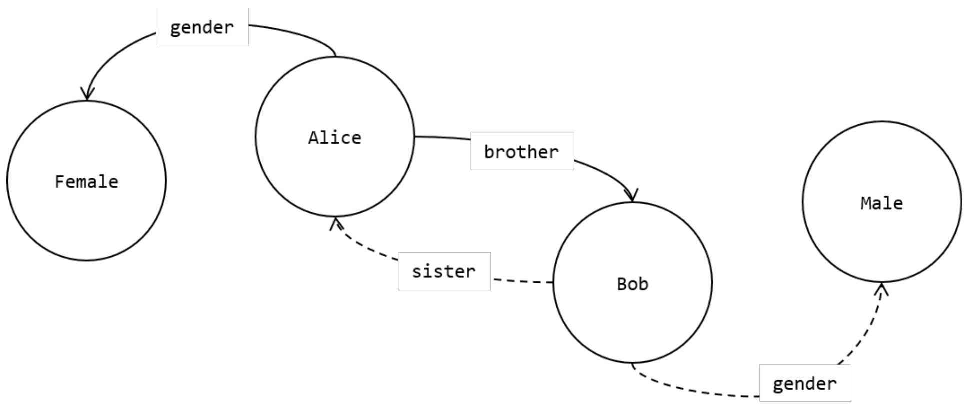


Fig. 3. Semantic Relationship Between Nodes

used for describing metadata regarding objects, system structures or almost any entity - concrete or abstract. The transfer and usage of these descriptions, 'resources' in RDF terminology, is enabled in machine-interpretable form while preserving the meaning of the descriptions. Edges and nodes are used in RDF for entity relation description [6]. RDF describes structures called "triples" which are information consisting of an object and a subject, which are linked by a predicate. The order of a triple is subject-object-predicate. RDF uses triples together with 'internationalized resource identifiers' (IRIs). The subjects, objects and predicates within triple structures are given IRIs for identification, but also to denote a link between items via a shared IRI. In scenarios where triples are grouped into separate named graphs a fourth parameter which denotes a graph name is added to the triple structure. The structure is then named a 'quad' due to the presence of four parameters.

Within RDF there is a concept called reification. Reification entails adding metadata to resources. Four primary types of reification are detailed in [8] – "Reifying RDF: What Works Well With Wikidata?". Standard reification involves denoting a triple via another RDF resource, whose attributes are the subject, object and predicate of the triple being denoted. Metadata can be added to these triples to provide more detailed descriptions. N-ary relations are a type of reification that leverages intermediate resources, on what originally would have been an edge describing a relationship. These intermediate resources are annotated with metadata for increased detail over a standard edge. Singleton properties enable the addition of metadata by using unique, per-statement, predicates which describe relationships by linking with "high-level predicates" [8]. Named Graphs reification is the process of adding a graph IRI to a triple, thus creating a quad structure which metadata can be added to [8].

2.3 RDF-star: Use Case: Ontologies and RDF-star for Knowledge Management

RDF-star is an extension to the standard RDF model which facilitates the association of annotations, or metadata, to RDF triple structures – this is achieved by adding descriptive data to the graph edges which describe relationships. This enables additional properties to describe resources to be added without requiring reification – which aims to provide a querying performance boost in bypassing the reification process. The performance impact of RDF-star is described in [9] – "Use Case: Ontologies and RDF* for Knowledge Management" where the statement is made that an elegant graph database can halve the loading times of data-sets containing sizable quantities of the metadata. Reasonably, this performance increase is tied to the amount of such metadata existing in the dataset [9]. The expansion from RDF to RDF-star necessitated an expansion of the SPARQL query language's semantic and syntax, called SPARQL-star.

2.4 Semantic Property Graphs

Semantic Property Graphs (SPGs) are an interesting emergent technology, although it is presently not quite mature enough for this use case. SPGs are a means of achieving "model-attributed semantic graphs". The creators describe it as a fusion between the labeled Property Graph and Resource Description Framework models for graph data description. The SPG provides adaptability while retaining efficiency in storage and compute usage when performing analysis and traversals of graphs. The SPG approach is underpinned by the RDF ontology – with a substantial reduction in the structure of the graph, coupled with file size serializing that preserves information pertinent to graph analysis. The concept of an SPG is described as a "framework to project reified RDF graphs into the property graph model" [6].

2.5 Summary

RDF with semantic graphs was the technology selected, due to several factors; RDF is based on an open standard and is highly interoperable. The requirement to integrate with a policy engine whose policies are written in RDF was of significant influence. Aside from its inherent usefulness as a description framework, RDF also has a range of useful open source tooling, such as JSON-LD conversion, which can be leveraged. Lastly, RDF can be used to implement the W3C policy schema "Open Digital Rights Language" (ODRL). RDF-star was a recent technological development at the time of specifying this system and, moreover the system had no specific requirement for RDF-star. Future, more expressive, requirements of the system may require a move to RDF-star, as RDF-star is built upon RDF, there is a clear upgrade path.

3 Data Formats

This section gives a brief description about the data formats that were evaluated as part of this project, i.e, JSON-LD, NGSI-LD and N-Triples.

3.1 JSON-LD

JSON-LD (JavaScript Object Notation for Linked Data) is a light-weight syntax based on JSON to serialize linked data in JSON and it provides a format readable by both machines and humans. One drawback of JSON is its ambiguity, i.e., as JSON represents data in key-value pairs, where the key used in one context may have different meaning in another context. JSON-LD overcomes this ambiguity issue with mapping the keys to IRIs via a context. Also, JSON-LD provides a universal identifier via the use of IRIs to the JSON object [10]. In other words, JSON-LD gives meaning to the JSON data. An example of JSON-LD is given in Figure 4.

```
{
  "@context":"https://schema.org/",
  "@type":"Car",
  "name":"Exemplary Car Model Name",
  "manufacturer":"Exemplary Car Manufacturer",
  "steeringPosition":{"@id":"https://schema.org/
     LeftHandDriving"}
}
```

Fig. 4. An example of JSON-LD representation of a Car

The “@context” keyword defines the semantic structure of the message and is the one which maps key, name, to the IRI. The “@id” keyword is the universal identifier to the JSON object. Additionally, JSON-LD can have other keywords. For example, “@type” keyword is the type of object/data. The value of @type will be explained in the context of JSON-LD data where it expands into full URL.

3.2 NGSI-LD

NGSI-LD (Next Generation Service Interfaces for Linked Data) is an open specification released by ETSI (European Telecommunications Standardization Institute) for context information/data modeling and APIs to produce, consume and subscribe context information [11]. NGSI-LD has Entity, Property and Relationship. Entity can be the subject and properties are represented as type-value pairs. Relationship is the association between the entities [12]. The NGSI Vehicle data model has three different formats: normalized, key-value pair, and linked data. The linked data format is based on JSON-LD [13].

An example of NGSI-LD is given in Figure 5: As in JSON-LD, "@context"

```
{
  "@context":  [
    "https://smart-data-models.github.io/dataModel.Transportation/Vehicle/schema.json",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
  ],
  "id": "urn:ngsi-ld:Vehicle:1fa179a6-b507-4857-ad72-eb5513ef05c6",
  "type": "Vehicle",
  "location": {
    "type": "GeoProperty",
    "value": {
      "type": "Point",
      "coordinates": [13.3986, 52.5547]
    }
  },
  "fuelType": {
    "type": "Property",
    "value": "Diesel"
  },
  ...
}
```

Fig. 5. An example of NGSI-LD representation of a car

keyword defines the semantic structure of the message and maps the entity to the IRI. The "id" keyword is the universal identifier to the entity, Santa Claus. The "type" keyword specifies the type of the object. As mentioned, the property, "name", is expressed as type-value pair. The entity has a relationship, "spouse", which in turn is another entity.

3.3 N-Triples

N-Triples is a data format for encoding RDF graphs and is line-based. As mentioned in Section 2.2, RDF triples consists of subject, predicate, and object. N-Triples triples is a set of RDF triples [14].

An example of N-Triples is given in Figure 6. The example can be read as subject, "<urn:ngsi-ld:Vehicle:1fa179a6-b507-4857-ad72-eb5513ef05c6>", has predicate, "http://schema.org/bodyType", and object, "hatchback". As in JSON-LD and NGSI-LD, IRIs are used as term identifiers in N-Triples as well.

```
<urn:ngsi-ld:Vehicle:1fa179a6-b507-4857-ad72-eb5513ef05c6> <
    http://schema.org/bodyType> "hatchback" .
<urn:ngsi-ld:Vehicle:1fa179a6-b507-4857-ad72-eb5513ef05c6> <
    http://schema.org/color> "White" .
<urn:ngsi-ld:Vehicle:1fa179a6-b507-4857-ad72-eb5513ef05c6> <
    http://schema.org/name> "Exemplary Car Model Name" .
```

Fig. 6. An example of N-Triples representation of car

4 Automotive FIWARE Data Models Stored in Knowledge Graphs

The choice of the data model used was guided primarily by the source of the automotive data. In this case, the data source was provided by the third-party automotive Application Programming Interface (API) provider "High-Mobility" [15]. Using this API mechanism allows for data access via a REpresentational State Transfer (REST) API and the data is provided as "application/json" response content type. After a review of the available data models, the FIWARE vehicle data model [16] had the most data points represented as required by the "High-Mobility" response. Figure 7 shows an illustration of the process involved in ingesting the JSON data from the vehicle, associating it with the FIWARE data model, using a custom ICV data model, converting the NGSI-LD to RDF Triples, and then storing the RDF Triples in a knowledge graph. The custom ICV schema facilitates the integrating of vehicle attributes such that the metadata can be associated with the policy language designed as part of the MOSAICrOWN Horizon 2020 project. The input vehicle data depends on the fact that every vehicle has a unique Vehicle Identification Number (VIN) [17] which allows for the connection of each node in the graph to a unique node for each vehicle.

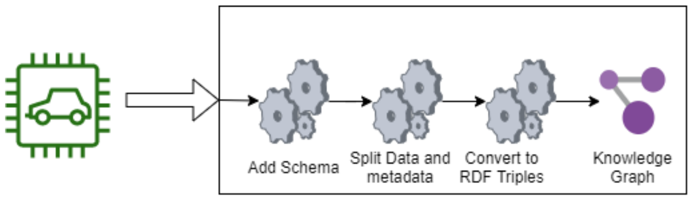


Fig. 7. Ingestion process taking JSON data, converting to NGSI-LD and then converting to RDF Triples

4.1 Detailed Vehicle Data Ingestion

The data supplied by the “High-Mobility” API is in JSON format without any context information. In this format, illustrated in Figure 8, it is difficult to gain the benefits of an information model and any graph technology which might be suitable. Therefore, modification of the incoming data such that it adheres to the “Vehicle” data model is required. The transformation is carried out using the JSON to JSON transformation library “Jolt” [18]. An example of the post-transform is presented in Figure 9.

```
{
  "vin": {
    "value": "7CKRXPHSZE"
  },
  "modelName": {
    "value": "Mercedes-Benz EQC"
  },
  "name": {
    "value": "GOHGTXXM"
  },
  "licensePlate": {
    "value": "BMW-7447"
  },
  "modelYear": {
    "value": 2013
  },
  ...
}
```

Fig. 8. JSON data from High-Mobility API

Another feature of note is the treatment of metadata. In our scenario, we make use of the fact that certain data points remain static throughout the data lifecycle, e.g, colour or VIN. We treat this metadata separately to the discrete data points by storing the metadata in the knowledge graph and linking the metadata to the data via a unique id. This is critical as without this linking concept there is no way to search the metadata index and then retrieve the required data object.

Using the Dell EMC schema, there now exists an NGSI-LD compliant document however the need to make use of a semantic knowledge graph makes further conversion necessary. This conversion is carried out in a similar fashion, i.e. via “Jolt”, resulting in a JSON-LD document similar to that presented in Figure 10. Data and metadata is also split just prior to this conversion and the data is stored in the object store as a dataset, an example of which is also in Figure 10. Vehicle metadata is defined as vehicle data describing the vehicle, e.g.

```
{
  "type": "Vehicle",
  "category": {
    "value": [
      "private"
    ]
  },
  "refVehicleModel": {
    "type": "Property",
    "value": "Mercedes-Benz EQC"
  },
  "speed": {
    "type": "Property",
    "value": "76",
    "observedAt": "2020-04-23 15:00:55.219231"
  },
  ...
}
```

Fig. 9. JSON data to NGSI-LD

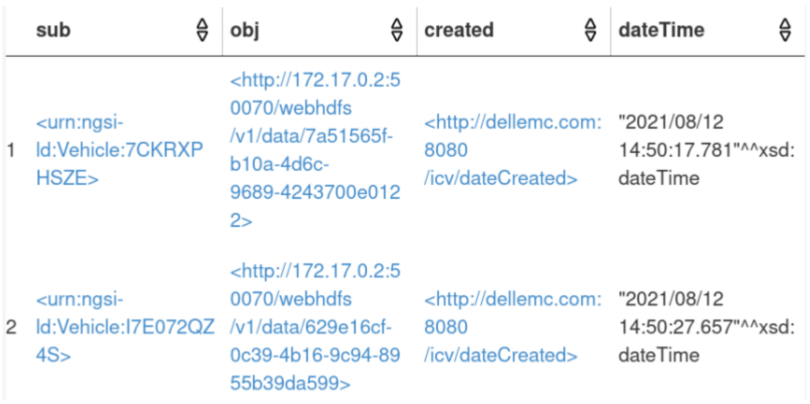
colorName. Vehicle attributes which regularly change (e.g. speed or location) are treated differently and are linked via a URI and are stored in the object store. Figure 11 and Figure 12 show how this link between the metadata and the object store is constructed. At this point the document is converted to N-Triples which is suitable for ingestion into the data markets metadata index (Apache Jena).

4.2 MOSAICrOWN Policy Specification Language

The MOSAICrOWN policy specification language [19] is a deliverable of the MOSAICrOWN project. The automotive use case described in MOSAICrOWN relates to the automotive domain and aims at addressing the data privacy issues related to electric vehicles. The policy model and language designed to address these issues is concerned with data wrapping and sanitization in order to protect sensitive portions of private data while facilitating the use of this data in a collaborative data market. The automotive scenario includes three main parties: i) data owners (drivers) ingesting their data into the data market; ii) consumers accessing data in the data market; iii) the data market provider offering storage and computation services to data owners and consumers. From an initial analysis, subjects, transformations, datasets, metadata, operations, and purposes were identified as basic elements of the policy model that also needs to be captured by the policy language. An example of a policy is presented in Figure 13

```
{
  "id": "urn:ngsi-ld:Vehicle:5FSQC8LARN",
  "@context": [
  {
    "id": "@id",
    "type": "@type",
    "dataCreated": {
      "@id": "http://purl.org/dc/terms/modified",
      "@type": "http://www.w3.org/2001/XMLSchema#dateTime"
    }
  },
  "http://dellemc.com:8080/icv/schema.json"
  ],
  "dataset": {
    "id": "http://172.17.0.2:50070/webhdfs/v1/data/cce23594-2
       744-401b-9f9c-37c703d2e925",
    "dateCreated": "2021-07-06T10:44:46.612Z"
  },
  "vin":"5FSQC8LARN",
  "name":"USWUZXZE",
  "privacyPolicy": "http://dellemc.com/policy/
     leastPrivatePolicies",
  "modelName":"Mercedes-Benz EQC"
  ...
}
```

Fig. 10. NGSI-LD data to JSON-LD



	sub	obj	created	dateTime
1	<urn:ngsi-ld:Vehicle:7CKRXPHSZE>	<http://172.17.0.2:50070/webhdfs/v1/data/7a51565f-b10a-4d6c-9689-4243700e0122>	<http://dellemc.com:8080/icv/dateCreated>	"2021/08/12 14:50:17.781"^^xsd:dateTime
2	<urn:ngsi-ld:Vehicle:I7E072QZ4S>	<http://172.17.0.2:50070/webhdfs/v1/data/629e16cf-0c39-4b16-9c94-8955b39da599>	<http://dellemc.com:8080/icv/dateCreated>	"2021/08/12 14:50:27.657"^^xsd:dateTime

Fig. 11. Use of additional node to allow for internal structure in a dataset

```
{
  "dataset": {
    "@id": "http://dellemc.com:8080/icv/dataset",
    "@type": "@id"
  },
}
```

Fig. 12. Additional property needed to represent data of vehicle

```
  "@context": [
    "http://www.w3.org/ns/odrl.jsonld",
    "http://localhost:8000/ns/mosaicrown/namespace.jsonld"
  ],
  "@type": "Set",
  "uid": "https://dellemc.com/policy/leastPrivatePolicies",
  "permission": [
  {
    "uid": "https://dellemc.com/policy/
        leastPrivatePolicies_perm",
    "assignee": "https://dellemc.com/user/fleetmanager",
    "target": [
          "https://dellemc.com/icv/licensePlate",
      "https://dellemc.com/icv/vin",
      "https://dellemc.com/icv/name"
    ],
    "action": ["odrl:read","odrl:use","odrl:write","odrl:sell
        ","odrl:sellReport"],
    "purpose": ["statistical", "marketing"]
  }
  ]
```

Fig. 13. Example MOSAICrOWN policy for automotive data which allows allows <user> to <action> on <target> for <purpose>

4.3 Application of RDF-star to Vehicle Data

RDF-star is designed to assist in the expression of more complex relationships when compared to RDF[20]. It achieves this through allowing for the annotation of information to edges within the graph. Furthermore, a growing number of tools, e.g., Ontotext GraphDB [9], have support for RDF-star and SPARQL-star. At the time of writing this paper, policy enforcement is the responsibility of the policy engine [19] developed for MOSAICrOWN, however it is envisaged that a scenario in the future where the association of a specific permission with a triple as an annotation might be required. As MOSAICrOWN uses some of the W3C's ODRL data policy model to construct the policy language, it aligns well with the approach of using linked data and RDF. An example of such an annotation is presented in Figure 14, where an RDF-star annotation is added to indicate that at the speed of 50 kilometers per hour the driver of the vehicle allows the speed to be used for odrl:statistical purposes, but not odrl:marketing. This indicates that the data owner allows for their speed to be aggregated for analysis, but not for specifically targeted commercial purposes.

```
:7GYY97R55FC :speed "50"^^xs:integer .

<<:7GYY97R55FC :speed "50"^^xs:integer>> :permission#purpose odrl:statistics .
```

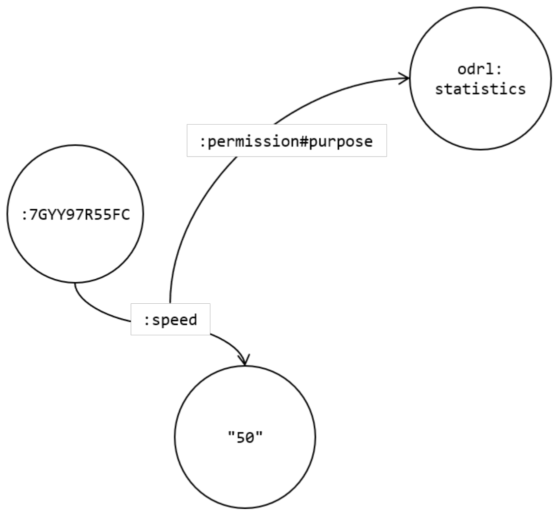


Fig. 14. RDF-star example using MOSAICrOWN policy language to annotate RDF

At the time of development, it was deemed that the tool support and status of the RDF-star standard was not sufficiently stable for use in the MOSAICrOWN project, however this will be revisited in the future as new, more expressive, metadata requirements are identified.

5 Conclusions

In this paper an automotive use case which uses a property graph-based data model in conjunction with a data market metadata index which is implemented as an RDF semantic graph is presented. The various graph and data formats which were evaluated for suitability were also discussed. The method for interchanging between these models is useful for the situation where one data model better represents the real-world data in a more complete fashion than the other however challenges in this interchanging had to be overcome. The reasons, problems, and solutions encountered in this effort were presented and discussed. Finally, the benefit of RDF-star for the automotive use case combined with data policy was considered, and this extension appears to have use for policy annotation in a privacy-centric data market.

References

1. Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008.
2. GO Blog. Introducing the knowledge graph: thing, not strings. *Introducing the Knowledge Graph: things, not strings*, 2012.
3. Nico Baken. Linked Data for Smart Homes: Comparing RDF and Labeled Property Graphs. In *LDAC2020—8th Linked Data in Architecture and Construction Workshop*, pages 23–36, 2020.
4. Álvaro Alonso, Alejandro Pozo, José Manuel Cantera, Francisco De la Vega, and Juan José Hierro. Industrial data space architecture implementation using FIWARE. *Sensors*, 18(7):2226, 2018.
5. MOSAICrOWN – Multi-Owner data Sharing for Analytics and Integration respecting Confidentiality and OWNer control. https://mosaicrown.eu/. (Accessed on 07/06/2021).
6. Sumit Purohit, Nhuy Van, and George Chin. Semantic property graph for scalable knowledge graph analytics. *arXiv preprint arXiv:2009.07410*, 2020.
7. Rana Alotaibi, Chuan Lei, Abdul Quamar, Vasilis Efthymiou, and Fatma Özcan. Property graph schema optimization for domain-specific knowledge graphs. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 924–935. IEEE, 2021.
8. Daniel Hernández, Aidan Hogan, and Markus Krötzsch. Reifying rdf: What works well with wikidata? *SSWS@ ISWC*, 1457:32–47, 2015.
9. Ahren Edgar Lehnert and Robert Thomas Kasenchak. Use Case: Ontologies and RDF* for Knowledge Management. *European Semantic Web Conference ESWC*, 2021.
10. A JSON-based Serialization for Linked Data. https://www.w3.org/TR/json-ld11/.
11. Nikos Kalatzis, George Routis, Yiorgos Marinellis, Marios Avgeris, Ioanna Roussaki, Symeon Papavassiliou, and Miltiades Anagnostou. Semantic interoperability for iot platforms in support of decision making: an experiment on early wildfire detection. *Sensors*, 19(3):528, 2019.

12. What is NGSI-LD? https://github.com/FIWARE/tutorials.Linked-Data.
13. NGSI-LD HOWTO. https://fiware-datamodels.readthedocs.io/en/latest/ngsi-ld_howto/index.html.
14. A line-based syntax for an RDF graph. https://www.w3.org/TR/n-triples/.
15. High mobility · privacy-centric car data. https://about.high-mobility.com/. (Accessed on 06/29/2021).
16. datamodel.transportation/vehicle at master · smart-data-models/datamodel.transportation · github. https://github.com/smart-data-models/dataModel.Transportation/tree/master/Vehicle. (Accessed on 06/29/2021).
17. ISO - ISO 3779:2009 - Road vehicles — Vehicle identification number (VIN) — Content and structure. https://www.iso.org/standard/52200.html. (Accessed on 08/25/2021).
18. Elias Al-Tai. An evaluation of the expressive power and performance of JSON-to-JSON transformation languages, 2018.
19. Pierangela Samarati and Rigo Wenning W3C. First Version of Policy Specification Language and Model. MOSAICrOWN Project Deliverable 3.3, June 2020.
20. Bob Kasenchak, Ahren Lehnert, and Gene Loh. Use Case: Ontologies and RDF-Star for Knowledge Management. In Ruben Verborgh, Anastasia Dimou, Aidan Hogan, Claudia d'Amato, Ilaria Tiddi, Arne Bröring, Simon Maier, Femke Ongenae, Riccardo Tommasini, and Mehwish Alam, editors, *The Semantic Web: ESWC 2021 Satellite Events*, pages 254–260, Cham, 2021. Springer International Publishing.