A Hypercat-Enabled Semantic Internet of Things Data Hub

Ilias Tachmazidis[1(✉)], Sotiris Batsakis[1], John Davies[2], Alistair Duke[2], Mauro Vallati[1], Grigoris Antoniou[1], and Sandra Stincic Clarke[2]

[1] University of Huddersfield, Huddersfield, UK
{i.tachmazidis,s.batsakis}@hud.ac.uk
[2] British Telecommunications, Ipswich, UK

Abstract. An increasing amount of information is generated from the rapidly increasing number of sensor networks and smart devices. A wide variety of sources generate and publish information in different formats, thus highlighting interoperability as one of the key prerequisites for the success of Internet of Things (IoT). The *BT Hypercat Data Hub* provides a focal point for the sharing and consumption of available datasets from a wide range of sources. In this work, we propose a semantic enrichment of the *BT Hypercat Data Hub*, using well-accepted Semantic Web standards and tools. We propose an ontology that captures the semantics of the imported data and present the *BT SPARQL Endpoint* by means of a mapping between SPARQL and SQL queries. Furthermore, federated SPARQL queries allow queries over multiple hub-based and external data sources. Finally, we provide two use cases in order to illustrate the advantages afforded by our semantic approach.

1 Introduction

The emerging notion of a smart city is based on the use of technology in order to improve the efficiency, effectiveness and capability of various city services, thus improving the quality of the inhabitants' lives [15]. A fundamental difference between smart cities and similar uses of technology in other areas, such as business, government or education, is the vast variety of the technologies used, the types and volumes of data, and the services and applications targeted [5]. Thus, developing successful smart city solutions requires the collection and maintenance of relevant data in the form of IoT data.

Over the past few years, eight industry-led projects were funded by Innovate UK[1] (the UK's innovation agency) to deliver IoT 'clusters', each centred around a data hub to aggregate and expose data feeds from multiple sensor types. The system that has come to be known as the *BT Hypercat Data Hub* was part of the Internet of Things Ecosystem Demonstrator[2] programme.

Addressing interoperability by focusing on how interoperability could be achieved between data hubs in different domains was a major objective of the

---

[1] https://www.gov.uk/government/organisations/innovate-uk.
[2] https://connect.innovateuk.org/web/internet-of-things-ecosystem-demonstrator/overview.

programme. Hence, Hypercat [1] was developed, which is a standard for representing and exposing Internet of Things data hub catalogues [6] over web technologies, to improve data discoverability and interoperability. Recent work [13], proposed a semantic enrichment for the core of the Hypercat specification, namely an RDF-based [8] equivalent for a JSON-based catalogue. Other IoT / smart city projects include Barcelona[3], MK:Smart[4] which uses the *BT Hypercat Data Hub* that is Hypercat-enabled but not semantically enriched, and the D-CAT[5] catalogue approach from W3C.

The main objective of this work is to achieve the semantic enrichment [2] of the data in the *BT Hypercat Data Hub* and to provide access to the enriched data through a SPARQL endpoint [11]. Furthermore, adding reasoning capabilities and the ability to combine external data sources using federated queries are important aspects of the implemented system.

The *BT Hypercat Data Hub* provides a focal point for the sharing and consumption of available datasets from a wide range of sources. In order to enable rapid responses, data in the *BT Hypercat Data Hub* is stored in relational databases. In this work, sensor, event, and location databases, i.e., databases containing information about sensor readings, events and locations are used. In order to provide a semantically richer mechanism of accessing the available datasets, the *BT Hypercat Ontology* was developed in order to lift semantically data stored within the relational databases. In addition, data translation through output adapters and SPARQL endpoints was defined. Thus, the semantically enriched data can be queried by accessing the developed *BT SPARQL Endpoint*.

Triplestores contain the information in RDF format combined with a built-in SPARQL endpoint. Thus, triplestores are commonly used for providing SPARQL endpoints. However, as data in the *BT Hypercat Data Hub* is stored in relational databases and this data is frequently updated, a more dynamic solution has been adopted. Thus, instead of copying the existing data into a triplestore, submitted SPARQL queries are dynamically translated into a set of SQL queries on top of the existing relational databases. In this way, a fully functioning SPARQL endpoint is provided, while during query execution, not only the SPARQL query itself is taken into consideration, but also the implicit information that is derived through reasoning over the developed ontology.

This work is organized as follows: Sect. 2 contains background information about the *BT Hypercat Data Hub* prior to its semantic enrichment. Section 3 contains a description of the *BT Hypercat Ontology* which was developed in this work in order to define the semantic representation of existing data. The corresponding mapping of data from a relational database to the semantic representation is described in Sect. 4. The *BT SPARQL Endpoint* is presented in Sect. 5 and the capability to combine information from external data sources by means of federated queries is presented in Sect. 6. Example use cases for the *BT Hypercat Data Hub* are illustrated in Sect. 7, while conclusions and future work are discussed in Sect. 8.

---

2   Background

The role of the *BT Hypercat Data Hub* is to enable information from a wide range of sources to be brought onto a common platform and presented to users and developers in a consistent way. Its portal provides a direct interface through which data consumers, such as app developers, can browse a data catalogue and select and subscribe to data feeds that they want to use. In addition, a JSON-based Hypercat [1] machine-readable catalogue, described further below, is also provided (as well as a recently proposed RDF-based Hypercat [13] catalogue). An API enables access to data feeds, secured by API keys, from browsers or within computer programs, while a relational, GIS capable, database enables complex queries that data can be filtered according to a wide range of criteria.

A set of edge adapters enables information coming onto the hub to be converted to a standard format for use inside the platform's core. It also provides a consistent API to end users and developers. The hub provides a consistent approach to integration between data exposed by sensors, systems and individuals via communication networks and the applications that can use derived information to improve decision making, e.g., in control systems. It includes a set of adapters for ingress (input) and egress (output). These are potentially specific to each data source or application feed and may be implemented on a case by case basis. There is therefore a need to translate data between arbitrary external formats and the data formats used internally.

In addition, as mentioned above, a Hypercat catalogue is implemented which is included via the Hypercat API. Hypercat is in essence a standard for representing and exposing Internet of Things data hub catalogues over web technologies, to improve data discoverability and interoperability. The idea is to enable distributed data repositories (data hubs) to be used jointly by applications through making it possible to query their catalogues in a uniform machine readable format. This enables the creation of knowledge graphs of available datasets across multiple hubs that applications can exploit and query to identify and access the data they need, whatever the data hub in which they are held.

From this perspective, Hypercat represents a pragmatic starting point to solving the issues of managing multiple data sources, aggregated into multiple data hubs, through linked data and semantic web approaches. It incorporates a lightweight, JSON-based approach based on a technology stack used by a large population of web developers and as such offers a low barrier to entry. Hypercat allows a server (IoT hub) to provide a set of resources to a client, each with a set of metadata annotations. There are a small set of core mandatory metadata relations which a valid Hypercat catalogue must include; beyond this, implementers are free to use any set of annotations to suit their needs.

3   BT Hypercat Ontology

In our previous work [13], we proposed a semantic enrichment for the core of the Hypercat specification, namely an RDF-based equivalent for a JSON-based catalogue. While Hypercat offers a syntactic first step, providing
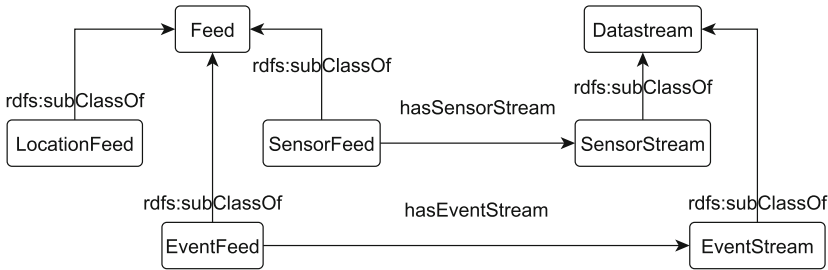
Fig. 1. BT Hypercat Ontology.

semantically enriched data goes further by allowing the unique identification of existing resources, interoperability across various domains and further enrichment by combining internally stored data with the Linked Open Data (LOD) cloud[6]. Data enrichment in the *BT Hypercat Data Hub* is achieved by representing data in RDF using concepts and properties defined in an OWL ontology [9]. Figure 1 shows the top level concepts of the *BT Hypercat Ontology*.

Feed is the top level class for any data feed that is asserted in the knowledge base. It contains the semantic properties of feeds. These include the feed id, creator, update date, title, url, status, description, location name, domain and disposition. There are also subclasses of class *Feed*, namely: *SensorFeed*, *EventFeed* and *LocationFeed* representing feeds for sensors, events and locations respectively.

The modelled data has been incorporated in the *BT Hypercat Data Hub* as one of the following feed types: (a) *SensorFeed*, (b) *EventFeed*, and (c) *Location-Feed*. Practically, each data source can advertise available information through the *BT Hypercat Data Hub* by providing a feed. A feed should be understood as a source of sensor readings, events or locations. Within each feed, data is available through datastreams (a class *Datastream* is defined, which has two subclasses namely: *SensorStream* and *EventStream* representing datastreams for sensors and events respectively). Thus, a given feed may provide a range of datastreams that are closely related e.g., for a weather data feed, different datastreams may provide sensor readings for temperature, humidity and visibility. Considering information about locations, a feed (of type *LocationFeed*) provides information directly by returning locations, namely locations are attached to and provided by a given feed.

A Hypercat online catalogue[7,8] contains details of feeds and information sources along with additional metadata such as tags, which allow improved search and discovery. The developed semantic model enables a semantic annotation and linkage of available feeds and datastreams. The *BT Hypercat Ontology* has been developed and made available with the uri (further details of the *BT Hypercat Ontology* can be found in [14]):

http://portal.bt-hypercat.com/ontologies/bt-hypercat

---

[6] http://lod-cloud.net/.
[7] http://portal.bt-hypercat.com/cat.
[8] http://portal.bt-hypercat.com/cat-rdf.

4   Data Translation

In this section we describe how data that is stored in a relational database within the *BT Hypercat Data Hub*, is made available in RDF.

4.1   RDF Adapter

By defining an ontology, semantically enriched data can be provided in RDF format. Note that prior to the semantic enrichment only XML and JSON formats were available. RDF data is represented in N-Triples format since such a format facilitates both storage and processing of data. Thus, each RDF triple is provided within a single line, in the following format: "<subject> <predicate> <object>.", while a collection of RDF triples is stored as a collection of lines. Note that N-Triples format can easily be transformed into other valid RDF formats, such as RDF/XML. In addition, the generated knowledge base can also be loaded in any given triplestore, namely any given RDF knowledge base, in order to facilitate operations such as query answering. Thus, by following W3C standards interoperability is ensured and the utilization of existing tools and applications is enabled.

The *BT Hypercat Data Hub* includes additional adapters for egress (output) in order to provide data in RDF format. In the following an example of how subject, predicate and object are generated for a *SensorFeed* is presented. Initially, the URI of each *SensorFeed* is generated, namely

<http://api.bt-hypercat.com/sensors/feeds/feedID>

Note that "http://api.bt-hypercat.com/" is the prefix URI for any data provided by the *BT Hypercat Data Hub*. In addition, "/sensors" provides information about the type of the feed (here *SensorFeed*), followed by "/feeds", which indicates that this URI belongs to a resource describing a feed, and finally "/feedID" is an id that uniquely identifies the given feed. For each *SensorFeed*, the *BT Hypercat Data Hub* provides its type, namely:

Subject	<http://api.bt-hypercat.com/sensors/feeds/feedID>
Predicate	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
Object	<http://portal.bt-hypercat.com/ontologies/bt-hypercat#SensorFeed>

Note that the generation of other triples follows a similar rationale. However, a detailed description of triple generation for each given concept and property is omitted due to space limitations.

4.2   SPARQL to SQL

In order to develop a *SPARQL to SQL* endpoint, Ontop[9] [3] was used as an external library. Ontop comes with a Protege[10] plug-in that allows the creation of mappings of SPARQL patterns to SQL queries (described below), see Fig. 2. In addition, it provides a reasoner that parses the mappings and the ontology, and handles the translation of SPARQL queries into a set of SQL queries in order to return the corresponding results (for the SPARQL query). A key advantage of using Ontop is that implicit information that is extracted from the ontology through reasoning is taken into consideration. In this way, semantically richer information compared to the knowledge that is stored in the relational database is provided. A description of how mappings can be created is presented below.
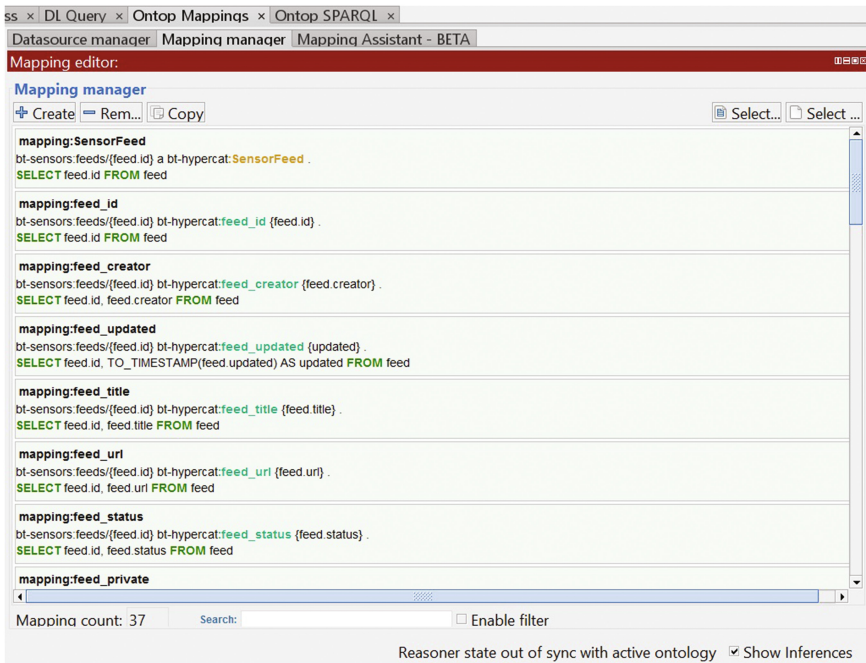

Fig. 2. Protege mapping editor.

In the following, an example of how a SPARQL triple pattern is mapped into a corresponding SQL query is described, and how the retrieved SQL results are used in order to construct RDF triples. *Mapping ID* corresponds to a unique id for a given mapping, *Target* (Triple Template) is the RDF triple pattern to be generated (note that SQL variables are given in braces, such as {feed.id}), and *Source* (SQL Query) is the SQL query to be submitted to the database.

First, the prefixes that are used are defined in order to shorten URIs, for example:

bt-sensors: http://api.bt-hypercat.com/sensors/
bt-hypercat: http://portal.bt-hypercat.com/ontologies/bt-hypercat

Then mappings are defined. For example, the following mapping maps the class *SensorFeed*. Note that class *SensorFeed* is subclass of *Feed*, and thus is a valid assertion, while providing semantically richer information:

Mapping ID	mapping:SensorFeed
Target (Triple Template)	bt-sensors:feeds/{feed.id} a bt-hypercat:SensorFeed .
Source (SQL Query)	SELECT feed.id FROM feed

Note that Fig. 2 contains additional mappings for the class *SensorFeed*.

The following query can be submitted to a *SPARQL to SQL* endpoint in order to retrieve *Feed*s:

```
PREFIX hypercat: <http://portal.bt-hypercat.com/ontologies/bt-hypercat#>
SELECT DISTINCT ?s
WHERE{ ?s a hypercat:Feed . }
```

Thus, Ontop will match the triple pattern "?s a hypercat:Feed" with the mapping "mapping:SensorFeed" since class *SensorFeed* is subclass of *Feed*. An SQL query (see Source) will be submitted to the relational database, while the retrieved *id*s (*feed.id*) will be used in order to generate RDF triples following the triple template (see Target).

Note that the generation of other triples follows a similar rational, while a more detailed description of triple generation for a given concept or property can be found in [14].

5   BT SPARQL Endpoint

In the following, a description of the high level architecture for the developed *BT SPARQL Endpoint* is presented. As shown in Fig. 3, two levels of abstraction are applied. At the lower level, there is a *SPARQL to SQL* endpoint for each relational database in the system, namely each *SPARQL to SQL* endpoint provides a SPARQL endpoint on top of the given relational database. In this way, the system administrator can add or remove a *SPARQL to SQL* endpoint at any time.

At the moment a *SPARQL to SQL* component is supporting the translation of SPARQL queries to PostgreSQL[11] relational databases that contain information about sensors or events. At the higher level, there is only one *SPARQL to*

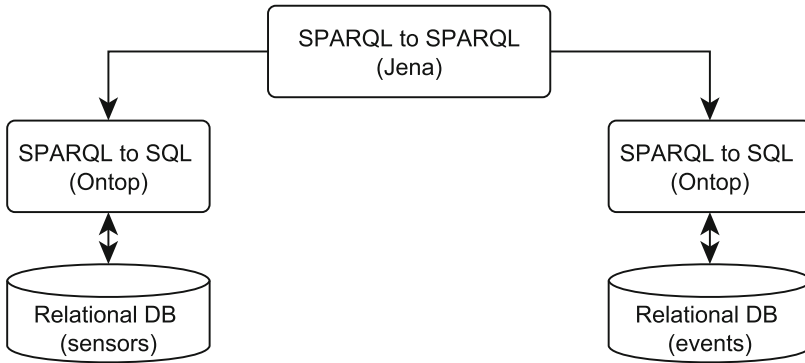---

[11] https://www.postgresql.org/.

Fig. 3. BT SPARQL endpoint.

SPARQL component (based on the query engine of Apache Jena[12] [4]), which is made available to end users. The underlying functionality indicates that end users submit SPARQL queries to the *SPARQL to SPARQL* endpoint, while the system queries internally all available *SPARQL to SQL* endpoints in order to extract the relevant information from existing relational databases. At any given point, the system administrator can add or remove a *SPARQL to SQL* endpoint depending on the available PostgreSQL databases.

Both *SPARQL to SPARQL* and *SPARQL to SQL* endpoints can be accessed using the *BT SPARQL Query Editor*, which is available for each endpoint. Users can provide the query text, namely the SPARQL query, using a graphic interface. In addition, the *BT SPARQL Query Editor* supports five results formats: HTML, XML, JSON, CSV and TSV.

One of the key advantages of SPARQL queries over SQL queries is that SPARQL queries incorporate semantic reasoning within the returned results. For example, classes *SensorFeed* and *EventFeed* are subclasses of class *Feed*. Thus, the reasoner classifies all objects that belong to either *SensorFeed* and *EventFeed* as *Feed*. The SPARQL query of Sect. 4.2 can also be submitted to a *SPARQL to SPARQL* endpoint in order to retrieve *Feed*s. Note that Ontop supports reasoning over RDFS[13] and OWL 2 QL[14].

6   Federated Querying

As described above, a *Federated SPARQL* endpoint has been added in order to enable federated queries over both the *BT SPARQL Endpoint* and other external SPARQL endpoints that are available through the LOD cloud. Such external

---

[12] https://jena.apache.org/index.html.
[13] http://www.w3.org/TR/rdf-schema/.
[14] https://www.w3.org/TR/owl-profiles/#OWL_2_QL.

Fig. 4. Federated SPARQL endpoint.

SPARQL endpoints that are part of the LOD cloud are for example: DBPedia[15], FactForge[16], OpenUpLabs[17] and the European Environment Agency[18].

The LOD cloud is expanding and new SPARQL endpoints are added (and removed) allowing for access to new data. Since the *Federated SPARQL* endpoint does not contain any information itself, it serves as a middleware that combines information coming from other SPARQL endpoints, as depicted in Fig. 4.

The *Federated SPARQL* endpoint extends further the functionality of the *BT SPARQL Endpoint* since external SPARQL endpoints can be used in order to retrieve information about events or social and economic information that can be combined with data from the *BT SPARQL Endpoint* for complex data analytics. Examples can be the extraction of data about natural disasters from external datasets combined with related sensor and event data from the *BT SPARQL Endpoint*. Other types of data extracted from external datasets can be, for example, social data related to housing projects and their correlation with sensor and event data from the *BT SPARQL Endpoint*.

Reasoning capabilities and spatiotemporal queries can be combined with external datasets (LOD) in order to retrieve information which is not directly represented in the *BT Hypercat Data Hub*. This can be achieved by means of federated queries spanning over different internal and external SPARQL endpoints.

For example, the following federated query retrieves sensor measurements from the *BT Hypercat Data Hub* related to a specific active bus stop, extracted from an external SPARQL endpoint (OpenUpLabs):

```
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX hypercat: <http://portal.bt-hypercat.com/ontologies/bt-hypercat#>
PREFIX naptan: <{http://transport.data.gov.uk/def/naptan/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT distinct ?d ?at_time ?western_longitude ?southern_latitude
       ?eastern_longitude ?northern_latitude ?stop ?lat ?long
WHERE {
   SERVICE <http://gov.tso.co.uk/transport/sparql>
   {
      ?stop a naptan:CustomBusStop;
            naptan:naptanCode ?naptanCode;
            naptan:stopValidity ?stopValidity;
            naptan:street"Kingswood Road";
```

---

[15] http://dbpedia.org/sparql.

[16] http://factforge.net/sparql.

[17] http://gov.tso.co.uk/transport/sparql.

[18] http://semantic.eea.europa.eu/sparql.

```
            geo:lat ?lat;
            geo:long ?long.
     ?stopValidity naptan:stopStatus ?stopStatus.
     ?stopStatus skos:prefLabel "Active"@en.
   }
   SERVICE <http://portal.bt-hypercat.com/BT-SPARQL-Endpoint/sparql>
    {
     ?d a hypercat:Datapoint.
     ?d hypercat:datapoint_at_time ?at_time.
     ?d hypercat:datapoint_western_longitude ?western_longitude.
     ?d hypercat:datapoint_southern_latitude ?southern_latitude.
     ?d hypercat:datapoint_eastern_longitude ?eastern_longitude.
     ?d hypercat:datapoint_northern_latitude ?northern_latitude.
     FILTER (?western_longitude > ?long - 0.1)
     FILTER (?southern_latitude > ?lat - 0.1)
     FILTER (?eastern_longitude < ?long + 0.1)
     FILTER (?northern_latitude < ?lat + 0.1)
   }
   FILTER(BOUND(?d))
}
```

7   Use Cases

This section is devoted to the description of two example use cases of the *BT Hypercat Data Hub*.

7.1   The SimplifAI Project

Urban traffic management and control is a primary concern of any city, and urban traffic transport operators often have at their disposal a disparate variety of real time and historical data, traffic controls (the most common of which are traffic signals) and controlling software. Software systems used for traffic management have a vertical design: they are not integrated at a horizontal level and cannot therefore easily share their data, or exploit data provided from other software/sources.

For achieving a higher level of data integration, and to better capture and exploit real-time and historical urban data sources, the SimplifAI project was carried out by a consortium consisting of the University of Huddersfield, British Telecommunications, Transport for Greater Manchester, and two other SMEs. In particular, the project focussed on exploiting the real-time and historical data sources to pursue better congestion control. As study area, a region of greater Manchester, UK was selected.

The overall concept in the improvement of traffic management was to utilise the semantically enriched data to enable the use of an intelligent function which requires both the integration of traffic data from disparate sources, and the transformation of the data into a predicate logic level, in order to operate. The intelligent function was to create traffic signal strategies in real time to solve challenges caused by exceptional or unexpected conditions.

The initial steps of the SimplifAI project concentrated on the semantic enrichment of traffic data. The raw data was taken from a large number of transport and environment sources and integrated into the *BT Hypercat Data Hub*, using the mapping of Sect. 4. After that, the focus was put on the utilisation of semantic data for generating traffic control strategies.

By enriching semantically the imported data, the unique identification of imported data is enabled. This is orthogonal to the problem solved by planning, as planning can also deal with ad hoc data. However, once the study area expands, using semantically enriched data will allow a systematic way of identifying resources that are mentioned in the generated plans. In addition, federated queries allow the developed system to extract data from the LOD cloud and combine it with data stored in the *BT Hypercat Data Hub* (e.g., the federated query of Sect. 6 combines bus stop information from an external source with internally stored data).

The intelligent function was based on an Automated Planning [7] approach [16], that is able to generate traffic control strategies (actions which change signals at a specified time) to alleviate traffic congestion caused by exceptional circumstances. The initial state of the modelled urban area, and information about available traffic lights and the structure of the network, were provided to the planning approach by the *BT Hypercat Data Hub*. The planner was then executed in order to generate control strategies for a number of test scenarios.

The quality of the strategies output from the planner was evaluated firstly by hand, inspecting the strategies to check that they were sensible, and by simulating their execution using traffic simulation software. Experts verified that strategies are sensible, and follow what would be expected when using "common sense". Simulations confirmed that generated strategies can effectively deal with unexpected conditions better than standard urban traffic control approaches: on average, the area is de-congested 20% faster, and tail-pipe emissions are reduced by 2.5%.

7.2 City Concierge

CityVerve is a Manchester, UK based IoT Demonstrator project, established in July 2016 with a two-year focus on demonstrating the capability of IoT applications for smart cities. One of the use cases of the CityVerve project, City Concierge, is aiming to increase uptake of walking and cycling as a preferred travel mode in Greater Manchester. Currently, Greater Manchester lacks integrated, consistent wayfinding services that can be accessed through a variety of media, including digital and print.

The City Concierge aims to develop a city user interface for the city region, integrating transportation and visitor services, allowing users to make informed choices regarding the way they travel. The scope of the use case includes improvements in the way people navigate around the city with a digital solution in conjunction with physical wayfinding assets, see Figs. 5 and 6.

Currently, it has been established that the *BT Hypercat Data Hub* provides the required infrastructure and functionality in order to enable the City

Fig. 5. Interaction between end users and city wayfinding assets.
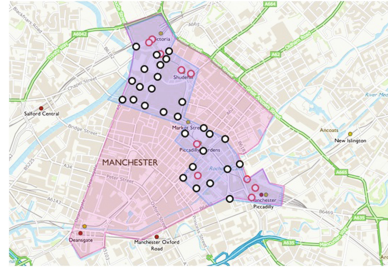

Fig. 6. Locations of wayfinding infrastructure.

Concierge. Translating data into RDF enables additional query capabilities such as SPARQL queries on top of the developed system and its combination with the LOD cloud through federated queries. Such queries are vital in order to achieve project's objectives, which include the deployment of IoT and digital software solutions that seek to address current challenges, while having the flexibility for future solutions to be developed on the network deployed as part of the CityVerve project.

8   Conclusion

In this work, the semantic enrichment of the *BT Hypercat Data Hub* has been presented. More specifically, the *BT Hypercat Ontology* has been introduced, which is the basis for the translation of existing data into an RDF representation. In addition, the *BT SPARQL Endpoint* has been implemented as a set of SPARQL endpoints and an additional endpoint, called *Federated SPARQL* endpoint, has been provided in order to allow the execution of federated queries. Moreover, an example federated query illustrates how the *BT Hypercat Data Hub* can be connected to the LOD cloud. Finally, two use cases are illustrating the extended functionality of the system, thus highlighting the benefits of the semantic enrichment.

Future work includes further semantic enrichment of the implemented system. Specifically, current support for SPARQL queries can be extended in order to enable GeoSPARQL queries [10] so as to provide direct access to spatial information that is currently available in the *BT Hypercat Data Hub*. In addition, spatiotemporal reasoning [12] is a prominent direction that could provide richer knowledge by reasoning over data that is coming from both the *BT Hypercat Data Hub* and the LOD cloud.

References

1. Beart, P.: Hypercat 3.00 Specification (2016)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Sci. Am. **284**, 29–37 (2001)
3. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: answering SPARQL queries over relational databases. Semant. Web **8**(3), 471–487 (2017)
4. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, pp. 74–83. ACM (2004)
5. d'Aquin, M., Davies, J., Motta, E.: Smart cities' data: challenges and opportunities for semantic technologies. IEEE Internet Comput. **19**(6), 66–70 (2015)
6. Davies, J., Fisher, M.: Internet of things - why now? Jnl Inst. Telecommun. Prof. **7**(3), 36–42 (2015)
7. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory & Practice. Elsevier, Amsterdam (2004)
8. Hayes, P.: RDF Semantics. In: W3C Recommendation (2004)
9. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.) OWL 2 web ontology language: primer. W3C Recommendation, vol. 27, October 2009. http://www.w3.org/TR/owl2-primer/
10. Perry, M., Herring, J.: OGC GeoSPARQL-a geographic query language for RDF data. OGC Implementation Standard, September 2012
11. PrudHommeaux, E., Seaborne, A., et al.: SPARQL query language for RDF. W3C Recommendation, vol. 15 (2008)
12. Stravoskoufos, K., Petrakis, E.G.M., Mainas, N., Batsakis, S., Samoladas, V.: SOWL QL: querying spatio-temporal ontologies in OWL. J. Data Semant. **5**, 249–269 (2016)
13. Tachmazidis, I., Davies, J., Batsakis, S., Antoniou, G., Duke, A., Clarke, S.S., Hypercat, R.D.F.: Semantic enrichment for IoT. In: Semantic Technology - 6th Joint International Conference, JIST 2016, Singapore, Singapore, November 2–4, 2016, Revised Selected Papers, pp. 273–286 (2016)
14. Tachmazidis, I., Batsakis, S., Davies, J., Duke, A., Vallati, M., Antoniou, G., Clarke, S.S.: A hypercat-enabled semantic internet of things data hub, Technical report, March 2017. https://arxiv.org/abs/1703.00391
15. Townsend, A.M., Cities, S.: Big Data, Civic Hackers, and the Quest for a New Utopia. WW Norton & Company, Cambridge (2013)
16. Vallati, M., Magazzeni, D., De Schutter, B., Chrpa, L., McCluskey, T.L.: Efficient macroscopic urban traffic models for reducing congestion: a PDDL+ planning approach. In: The Thirtieth AAAI Conference on Artificial Intelligence (AAAI) (2016)