Query Interface for Smart City Internet of Things Data Marketplaces: A Case Study

NAEIMA HAMED, Cardiff University, United Kingdom
ANDREA GAGLIONE, Digital Catapult, United Kingdom
ALEX GLUHAK, Digital Catapult, United Kingdom
OMER RANA, Cardiff University, United Kingdom
CHARITH PERERA*, Cardiff University, United Kingdom

Cities are increasingly getting augmented with sensors through public, private, and academic sector initiatives. Most of the time, these sensors are deployed with a primary purpose (objective) in mind (e.g., deploy sensors to understand noise pollution) by a sensor owner (i.e., the organization that invests in sensing hardware, for example, a city council). Over the last few years, communities undertaking smart city development projects have understood the importance of making the sensor data available to a wider community – beyond their primary usage. Different business models have been proposed to achieve this, including creating data marketplaces. The vision is to encourage new start-ups and small and medium-scale businesses to create novel products and services using sensor data to generate additional economic value. Currently, data are sold as pre-defined independent datasets (e.g., noise level and parking status data may be sold separately). This approach creates several challenges, such as (i) difficulties in pricing, which leads to higher prices (per dataset), (ii) higher network communication and bandwidth requirements, and (iii) information overload for data consumers (i.e., those who purchase data). We investigate the benefit of semantic representation and its reasoning capabilities towards creating a business model that offers data on-demand within smart city Internet of Things (IoT) data marketplaces. The objective is to help data consumers (i.e., small and medium enterprises (SMEs)) acquire the most relevant data they need. We demonstrate the utility of our approach by integrating it into a real-world IoT data marketplace (developed by *synchronicity-iot.eu* project). We discuss design decisions and their consequences (i.e., trade-offs) on the choice and selection of datasets. Subsequently, we present a series of data modeling principles and recommendations for implementing IoT data marketplaces.

CCS Concepts: • **Information systems** → **Web data description languages**; *Graph-based database models*; • **Human-centered computing** → **Ubiquitous and mobile computing**; *Human computer interaction (HCI).*

Additional Key Words and Phrases: Internet of Things, Semantic Interoperability, Data Discovery, Multi-dimensional Querying, Linked Data, Knowledge Management

ACM Reference Format:
Naeima Hamed, Andrea Gaglione, Alex Gluhak, Omer Rana, and Charith Perera. 2023. Query Interface for Smart City Internet of Things Data Marketplaces: A Case Study. *ACM Trans. Internet Things* 1, 1, Article 1 (January 2023), 39 pages. https://doi.org/10.1145/3609336

*This work is conducted as part of the Researcher in Residence (RiR) programme in collaboration with Digital Catapult.

Authors' addresses: Naeima Hamed, hamednh@cardiff.ac.uk, Cardiff University, Cardiff, United Kingdom; Andrea Gaglione, Digital Catapult, London, United Kingdom, andrea.gaglione@digicatapult.org.uk; Alex Gluhak, Digital Catapult, London, United Kingdom, alex.gluhak@digicatapult.org.uk; Omer Rana, ranaof@cardiff.ac.uk, Cardiff University, United Kingdom, Cardiff; Charith Perera, charith.perera@acm.org, Cardiff University, Cardiff, United Kingdom.

2577-6207/2023/1-ART1 $15.00
https://doi.org/10.1145/3609336

1 INTRODUCTION

Over the last decade, many cities have initiated projects that deploy different sensors for various reasons. One popular application domain is environmental monitoring. After accomplishing the primary objectives, such data are often discarded or stored somewhere where access can be difficult outside the initial project. There often needs to be a mechanism (or motivation) to share data with outside parties (other than the organization that deploys the sensing infrastructure). This approach leads to a waste of resources and can limit the potential benefits derived from such data. Internet of Things (IoT) data marketplaces for smart cities are being proposed as a solution to address this challenge. *Urban data Exchange*[1] is one such data marketplace aimed at facilitating businesses to develop IoT and AI-enabled services to improve citizens' lives and grow local economies. Data marketplaces have received limited attention in the academic community. However, the buying and selling of data have taken place for a long time, especially within the business-to-business (B2B) context. Initially, these data transactions took place offline between companies and their alliances. Data have been widely sold across various domains, such as travel, advertising, and insurance. Even though we are not focusing on personal data marketplaces in this paper, we will briefly introduce and discuss them in the Related Work section. Typically, a data marketplace comprises three types of stakeholders: (i) data buyers (who can also combine multiple data sources), (ii) data consumers, and (iii) data brokers. The key contributions of this work are as follows:

- We propose an ontology aggregating other well-known ontologies to model sensor data in IoT marketplaces. We made design decisions during the ontology engineering process that led to different trade-offs. We discuss the trade-offs of each decision and highlight good practices observed in existing efforts.
- We propose a unique on-demand data offer creation technique. Buyers can create their custom data requests (i.e., data order) by considering four aspects: location, data type, date/time, and service level agreement.
- Through a series of use cases, we demonstrate the utility of knowledge engineering (including reasoning/ inferencing) in the context of data marketplaces. We also present different levels (dataset level, market level, buyer level) of knowledge engineering approaches and their utility and related costs (e.g., computational complexity).
- We evaluate the performance of the proposed approach in three different data marketplace setups. We measured some parameters and extracted several recommendations for future marketplace deployments.

2 MOTIVATION AND THE PROBLEM DEFINITION

Currently, IoT data marketplaces sell data per entire dataset, as shown in Figure 1. For example, potential buyers could buy weather forecast data and parking status data in bulk. Each of the datasets may contain multiple pieces of data packed together in a pre-defined manner (e.g., `temperature`, `relativeHumidity` may be included in the Weather Forecast data offer). There are multiple problems with this approach.

Problem 1: Higher network communication and bandwidth requirement: ($\uparrow D_i^p \propto D_i^b \uparrow$) In the current approach, data offers are sold as pre-defined data bundles. There is no way to limit the number of data within a bundle that a buyer acquires (whether the buyer may request past (archival) data or future data (as a subscription)). The consequence of this approach is higher network download time and cost. For example, if a data consumer wants bicycle docking station data in Santander, they will need to buy the entire `Docking Stations - SAN`

[1] https://urbandata.exchange/

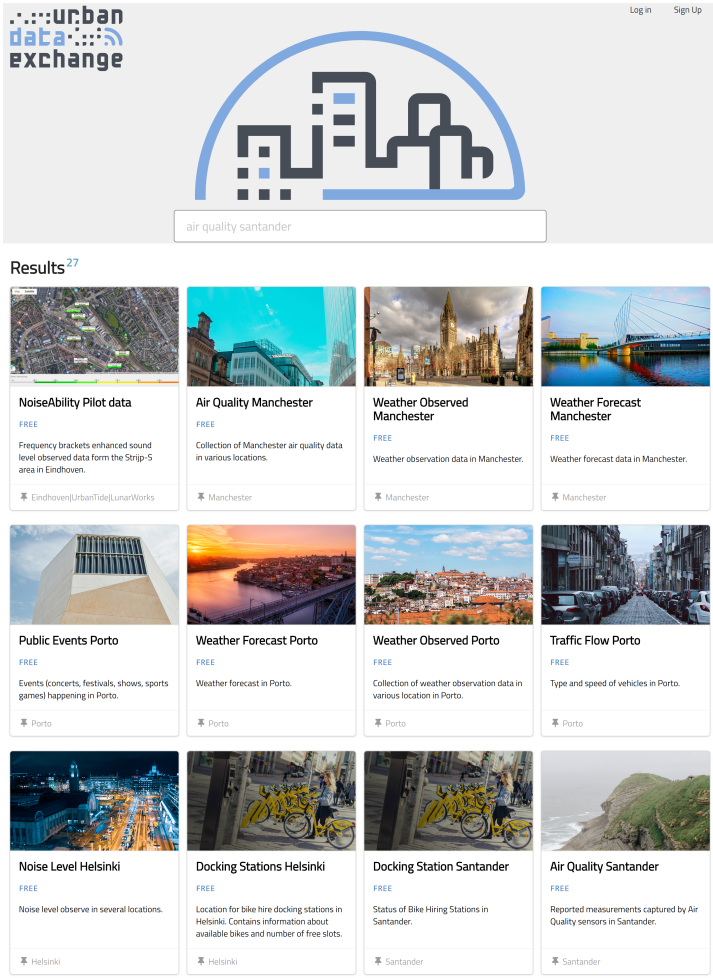


Fig. 1. Urban Data Marketplace (Current Approach): FIWARE data models are used to organize the data into datasets. They have been harmonized to enable data portability for different applications, including Smart Cities, Smart Agrifood, Smart Environment, Smart Energy, Smart Water, and others. The key weakness of this approach is that data buyers need to buy the entire dataset (e.g., Noise Level Helsinki) whether they need the entire dataset or not. This approach leads to higher data prices.

dataset whether they need the entire dataset or not. Therefore, there is a positive correlation between network communication, bandwidth requirement and volume of data. Given that the price of a data offer i is D_i^p and the bandwidth required is D_i^b, the price is proportional to bandwidth.

Problem 2: Difficulties in pricing which leads to higher prices: ($\uparrow D_i^p \propto D_i^v \uparrow$) Currently, each data bundle comprises large volumes of data. The cost of acquisition for a large amount of data is high. Therefore, the cost of the bundle has to be high as well. In a data marketplace, the price of a data offer has to cover the cost of data acquisition plus a profit margin. Therefore, there is a positive correlation between data prices and volume. Given that the price of a data offer i is D_i^p and volume is D_i^v, price is proportional to volume.

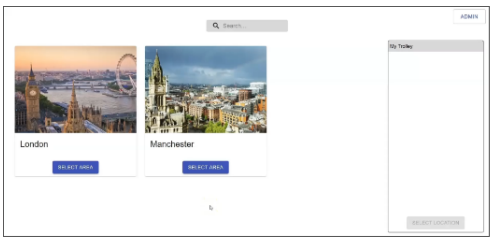
Problem 3: Information overload for data consumers: ($\uparrow D_i^v \propto D_i^{pp} \uparrow$) .
In data science projects, 80% of time and effort is often devoted towards preparing the data (i.e., acquiring, cleaning, transforming, etc.), and only 20% is used to do the actual analysis. Therefore, the larger the buyers' dataset, the more effort they need to prepare and filter the relevant data. Assume that a given data analysis task is related to `weekends` data (e.g., parking slot status). First, data scientists need to query the entire dataset, remove the data related to `weekdays` and select only the data related to `weekends`. Therefore, the current bulk pre-defined data offering approach unnecessarily increases data scientists' workload (i.e., data consumers'). Given the size of a data offer, i is D_i^v, the cost of data pre-processing is D_i^{pp} – the cost of data pre-processing is proportional to volume.

Problem 4: Limited data discovery capabilities: Currently, IoT data marketplaces organize datasets by type (broadly) and location. For example, it is usually up to the data seller to bundle the data into an offering as they see fit, as shown in Figure 1. Here, data offers are pre-defined and static without any mechanism to request customized data. Data search primarily relies on location. There is no way for data consumers to acquire `traffic data` in `London` on `rainy` days over the `last three years` in the current data marketplace scenario.

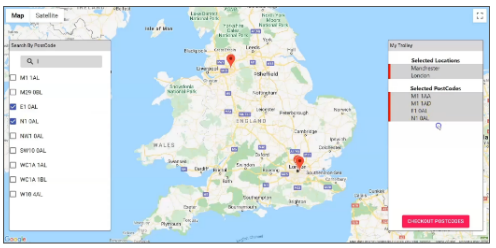
2.1 Design Principles and Architecture

We suggest design principles to help data consumers communicate the data they need. These efforts yielded Competency Questions (CQs) for our data model.

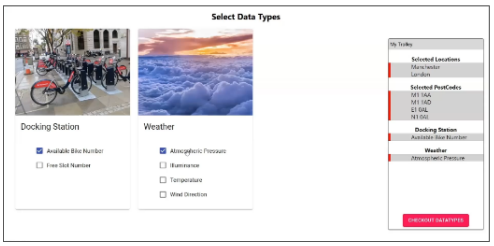
2.1.1 Design Principles. We enlisted twenty-one participants with backgrounds in computer and data science to extract the following design principles. The identified expressed their priorities using question terms: (1) where, (2) what, (3) when, and (4) how. Let us explain each of these design principles with a concrete example. As illustrated in a series of Figures (2(a), 2(b), 2(c), 2(d)), we have implemented the proposed IoT data marketplace by following these design principles.



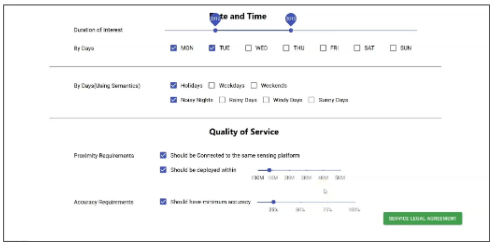
(a) Starter Page of the Custom Data Request Builder



(b) Data consumers are provided with a UI to select the areas of interest



(c) Data consumers are provided with a UI to select the data types they need



(d) Data consumers are provided with a UI to express their Time requirements

Fig. 2. Custom Data Request Builder

We considered the selected participants for their expertise in three key areas: (i) their understanding of the IoT domain, (ii) their knowledge of semantic web technology for the IoT from an

end-user perspective, and (iii) their proficiency in semantic data modelling. We reviewed their qualifications, expertise, and past experiences, sourcing them from professional networks, academic institutions, conferences, and events. The participants' questionnaire responses supplemented us with the necessary data for these filters to make deductions. For instance, one of the questions asked, "*Do you believe that bikes available for hire in London city will be less accessible than usual on a sunny day?*" The response showed that over 70% of the participants agreed with this statement. Consequently, we established logical rules to infer sunny days based on the decreased availability of bikes for hire. Moreover, the participants' responses revealed interesting insights about their demographics. Most participants have achieved a minimum of a bachelor's degree, with approximately 30% of them having pursued further education at the postgraduate level. The majority of participants, as such, have direct experience in applying semantic web technology to handle and analyze IoT data (e.g., database management, AI and machine learning). The distribution of years of experience in the current field is quite diverse, with 25% having 1-2 years, 30% having 3-5 years, and a sizeable proportion having more than ten years. Participants also have a wide range of ages, with the majority in the 25-34 and 35-44 age groups, respectively. Figure 3 and 4 show the study participants' education background and demographic information.

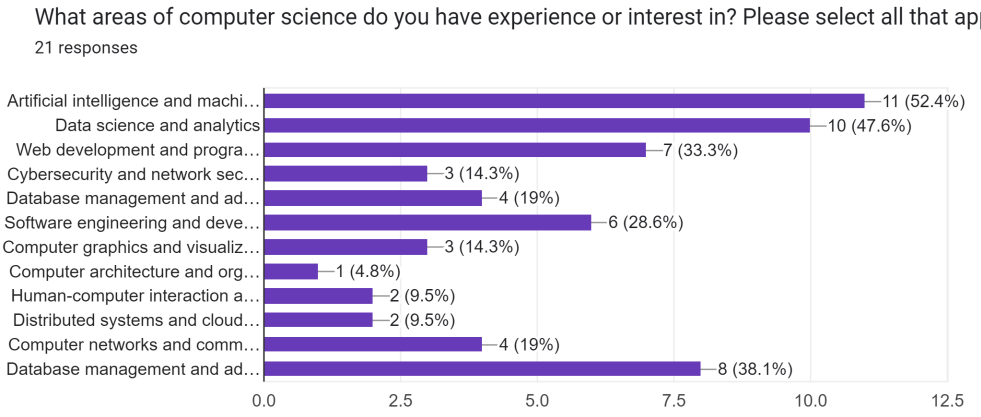


Fig. 3. Participants education and experience background in Computer Science.

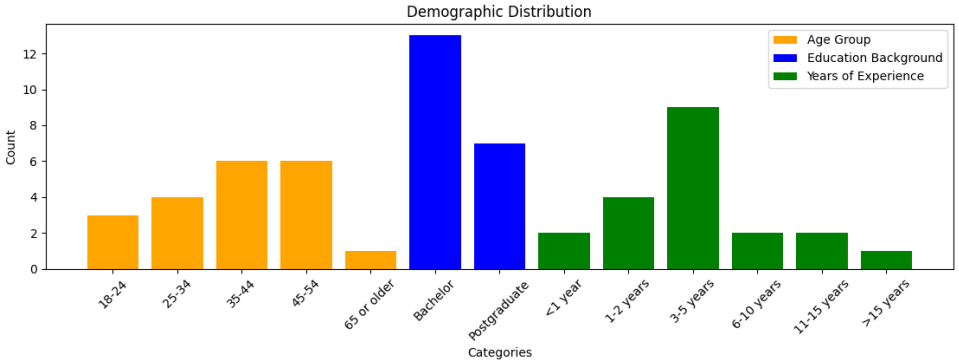


Fig. 4. Participants demographic information.

To guide potential data consumers, we have designed a user interface and developed a questionnaire to gather the necessary information as shown in Figure 2(a). Based on our preliminary investigation, the most critical decision data scientists make is to choose the area of interest (where). As illustrated in Figure 2(b), data consumers need to express the area of interest, and then available cities within the desired area will appear (e.g., London, Manchester). They can subsequently narrow down the search in a more granular way. For instance, they may select specific postcodes and zones or mark an area by drawing a polygon(s) on the provided map. As illustrated in Figure 2(c), consumers can select which data types they want. Data are categorized into logical groups. To organize the data types, we follow the data categories provided by FIWARE[2].

As shown in 2(d), the next stage allows data consumers to express their *'Time'* window of interest. First, data consumers must scope their requests for the overall duration (e.g., last ten years, following four years). Secondly, data consumers are offered to narrow down their requests using an expressive set of semantic terms (e.g., weekdays, weekends, rainy days, and sunny days). These filtering terms are only possible due to Semantic Web technologies. We will discuss this capability (as well as the principles and technologies behind facilitating such capability) in detail later in this paper. The next stage allows data consumers to express their Service Level Agreement (SLA) requirements, such as their *'co-location'* requirements. For example, a data consumer who has a focus on developing a *park and cycle* app. might want to understand `air` and `noise pollution`. In order for data to be meaningful, data consumers need to gather `air` and `noise pollution` data near `car parks` and `bicycle stations` (e.g., co-located within half a mile).

2.1.2 Architecture. The IoT data marketplaces need to be distributed in nature. Data owners are expected to store and manage data items and only share their metadata with brokers such as the IoT data marketplaces. Consequently, when a broker receives a request, it knows from where to gather the data (or to decide whether it is possible or not to fulfill the request). Figure 5 depicts the architecture of the IoT data marketplace, and details are presented here (https://gitlab.com/synchronicity-iot). In summary, we have developed a user interface that allows data consumers to build their data requests. We then organize each data request using a standardized JSON schema and send it to the validation engine. The validation engine determines whether the IoT data marketplace can fulfill a given request based on the available metadata. Then, one or multiple SPARQL queries will be generated based on the requirements of the data request (and depending on where the actual data reside).

3 DATA MARKETPLACE DESIGN

To address the requirements stated in 2.1.1, we propose an IoT data marketplace that allows potential data consumers to buy *only the data points/records they need to solve a given problem.* Pre-defining many data offerings is not feasible; therefore, the best approach is to allow consumers to create their data offerings (i.e., data requests). To illustrate the notion, assume a new tourism company is interested in purchasing various data entities to build an AI decision support system. These entities may contain specific observations about local attractions such as beaches, museums, parking spots, and bike docking stations. Thus, the organization prepares a single order that includes the data records from the datasets instead of acquiring the entire dataset for each entity separately. Nonetheless, this method has several drawbacks in terms of the following: (see Table 1).

- Data pricing could be complicated because each data source may price its observations-depending on their size and novelty. In addition, the broker fees and any variable extra charges have to be carefully calculated and added to the total bill.

[2]https://www.fiware.org/smart-data-models/

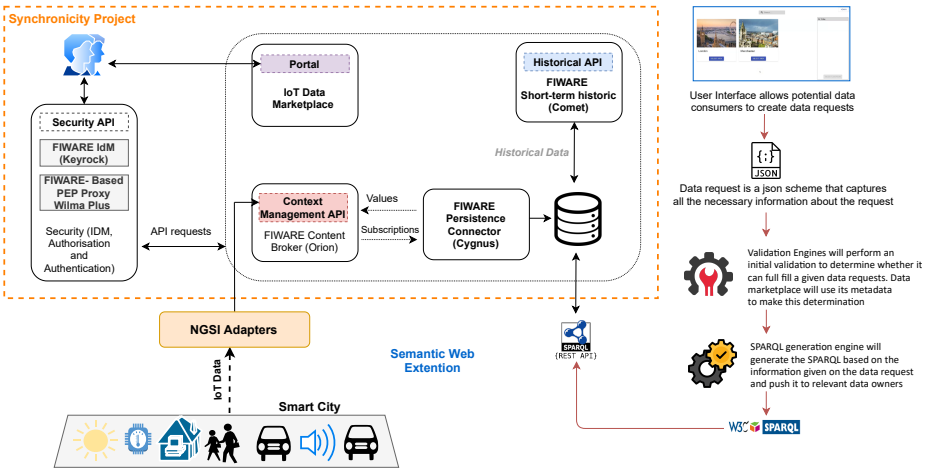


Fig. 5. Semantically Enhanced IoT Data Marketplace Architecture

- **Data publishing** could raise privacy and ownership concerns because data providers may have different privacy policies and credit preferences. Therefore, appropriately tailored privacy and data ownership agreements must exist to satisfy all parties.

Table 1. Possible Scenarios

Criteria	Current Pre-Defined Data Offering Approach	Proposed On-Demand Data Offering Approach
• Pricing Structure	Simple	Could be complicated
• Pricing Fairness	Less fair	More fair
• Data Discovery	Less discoverable	More discoverable
• Publishing Complexity	Easy and simple to publish	Could be complicated to publish
• Data Preparation Complexity (from a data consumer perspective)	Higher (as large datasets need to be processed and filtered)	Less (as data is already processed and filtered)

3.1 Data Model

Our data model comprised a foundational ontology instantiated with six heterogeneous datasets. The ontology aims to describe sensor data in the IoT data marketplace. We followed the NeOn methodology [1–3] to develop the ontology. Although there are numerous other ontology development methodologies [4–8], we selected NeOn as it has multiple modular scenarios to choose from and adapt to our current requirement. Figure 6 shows the ontology development's life cycle. We adopted NeOn's first, second and third scenarios. The first scenario outputs the Ontology Requirement Specification Document (ORSD). Then, we identified the non-functional requirements from the second scenario based on the ORSD (see Appendix A). We followed the process of reusing existing ontological resources from the third scenario. Following that, we implemented and evaluated the ontology using various tools. Figure 7 depicts the proposed core ontology, Table 2 discusses the key characteristics of this ontology, and Figure 8 shows the ontology instantiated with six sensor datasets.

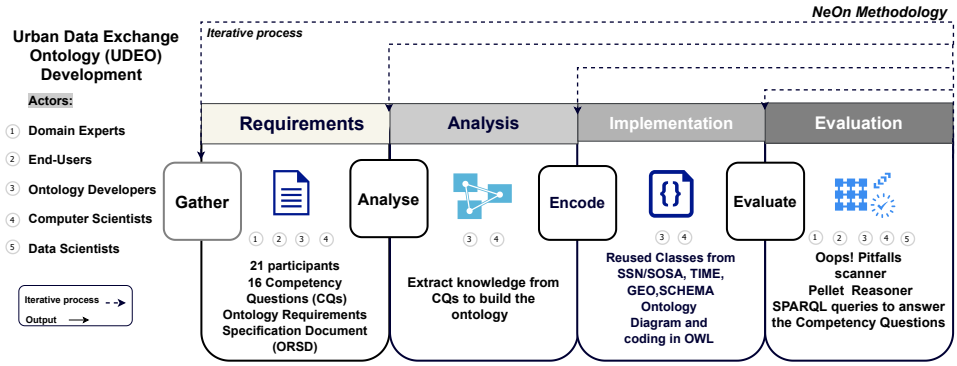


Fig. 6. Neon Methodology for developing the proposed Urban Data Exchange Ontology

3.2 Ontology Requirements

The first step in developing the ontology was to gather the required information. Here, we used the information collected at the design stage (see section 2.1.1- formulating Competency Questions (CQs) to develop the ontology. During this step, we produced the ORSD (see Appendix A), which contains the conceptual building blocks for the ontology as follows:

- Ontology purpose: To describe the sensor data.
- Ontology scope: The Internet of Things (IoT).
- Ontology implementation language: OWL2 Web Ontology Language.
- Ontology intended users: Small to medium businesses (SMEs).
- Ontology non-functional requirements: To List elements that must be included in the ontology, such as IoT geospatial and time classes.
- Ontology functional requirements: To contain the Competency Questions (CQs) that build and validate the ontology.

3.3 Ontology Analysis

In this step, we revised the Competency Questions (CQs) from the requirements phase and extracted knowledge to implement the ontology. We decided to reuse classes from mature ontologies to develop a new ontology that models the following concepts: 1) sensor data observations, 2) sensing infrastructure, 3) location, 4) temporal aspects, and 5) units of data.

To find suitable state-of-the-art ontologies, we searched Google Scholar [9] and the BioPortal repository [10], as well as other scholarly websites and ontology repositories, with inclusion criteria that the publication date had to be between 2015 and 2020. Different search terms were used to perform the search: "sensor data ontology", "semantic modeling for sensor data", "semantic IoT data", and "IoT ontology" [11, 12]. The outcome of the search yielded six ontologies that are commonly used to model sensor data, such as the Semantic Sensor Network Ontology (SSN) [13].

Among the shortlisted ontologies is the SAREF [14] ontology that describes smart appliances and related IoT devices and services, which may not be the most suitable ontology for modeling sensor data observation. Additionally, the IoT-Lite ontology [15] provides a basic set of classes and properties for describing IoT devices, sensors, and actuators. However, we may need more than this for our use cases, as we require classes to model the sensor's observation and properties rather

than the sensor alone. The W3C Web of Things (WoT) ontology[3] is a flexible, modular ontology that can be changed to fit different use cases and allow different IoT systems and domains to work together. WoT focuses on different aspects of IoT devices and services. However, its flexibility and generality can also make it challenging to adapt to our requirements. For example, one of the WoT classes, "Thing", models the IoT device, the service, or the data source. In contrast, the class "Sensor" is more suitable for modeling the sensor's observation.

The FIESTA-IoT ontology [4], as such, models IoT-related concepts but has more entities than what we need. It borrows classes from the SSN ontology (Version 1) [16], the W3C Web of Things (WoT) Thing Description, and the oneM2M standard[5]. The IoT-Semantics Ontology is another flexible ontology that lacks sufficient documentation, making it challenging for developers to adapt. After scouring and comparing the state-of-the-art ontologies, we decided to reuse concepts from the SSN ontology (Version 2) [13], mainly for its modular property. SSN ontology (Version 2) integrates three distinct ontologies. That is the SSN ontology (Version 1), the Sensor, Observation, Sample, and Actuator (SOSA) ontology [17], and the Quantities, Units, Dimensions, and Types (QUDT) ontology [18], qualifying it to be the optimal choice for our use case. Further, we named our new ontology the Urban Data Exchange Ontology (UDEO).

Sensor data observations: From SSN (Version 2), we re-used the SOSA ontology. The SOSA ontology is designed to be interoperable with other Semantic Web standards like RDF, OWL, and SPARQL, making it easy to integrate with other data sources and applications. Moreover, the SOSA ontology is continuously maintained and updated by a community of researchers and developers, ensuring that it stays relevant to current IoT applications. The ontology is also modular and extensible, making it easy to customize for our use cases.

Sensing infrastructure: SOSA provides sufficient facilities to model sensing infrastructure. Therefore, we also reused the concepts and properties for this purpose.

Location: This work focuses on outdoor locations, easily identified using GPS Coordinates.

Temporal aspects: We decided to store the timestamp of each observation using XML Date-Time data type (i.e., xsd:dateTime). We considered OWL-Time [19] to be incorporated into our ontology. However, we concluded that SPARQL's Time function also provides most of the OWL-Time capabilities. Therefore, we omitted it from our ontology.

Units of data: We adopted concepts from the Quantities, Units, Dimensions, and Types Ontology (QUDT) Ontology to model data units. QUDT is part of SSN ontology (Version 2) and provides a modular approach to ontology development, which allows for compatibility and interoperability with other systems that use QUDT or a similar ontology.

3.4 Ontology Implementation

As mentioned in the analysis step, we mainly adopted most of the UDEO classes from the SOSA ontology as shown in figure 7. The conceptual ontology model, or the lightweight version, was designed as a UML diagram in draw.io software[6]. The ontology diagram in 8 was discussed between UDEO stakeholders before its digitization. Each of the concepts and relationships modelled in UDEO is listed in Table 2. We encoded the digital version in the Protege ontology editor and exported it to a dedicated knowledge graph platform [20].

[3]https://www.w3.org/TR/wot-thing-description11/
[4]http://iot.ee.surrey.ac.uk/ontology/fiesta-iot.owl
[5]https://www.onem2m.org/
[6]https://app.diagrams.net/

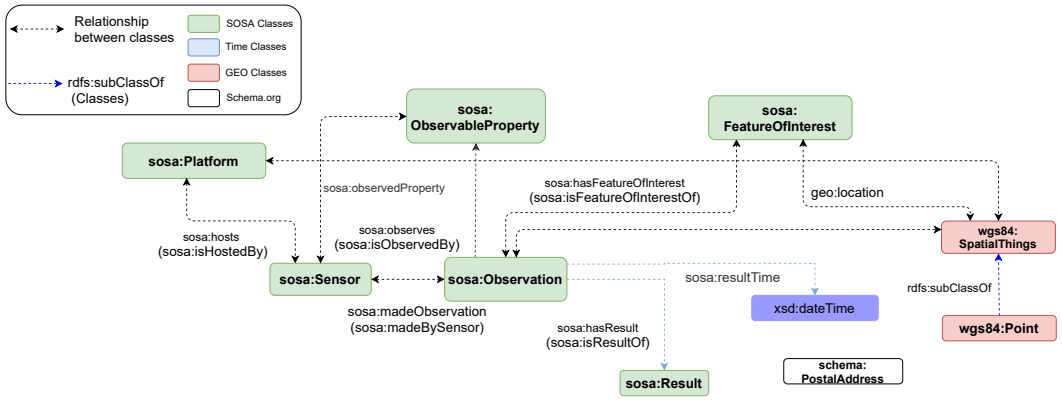


Fig. 7. The adopted classes for the proposed Urban Data Exchange Ontology (UDEO)

Table 2. Proposed data model: Concepts and relationships. [Official definitions are in *Italics*]

Concept	Description
Observation (OWL Class) [sosa]	*Act of carrying out an (Observation) Procedure to estimate or calculate a value of a property of a FeatureOfInterest (e.g., Room).* Observation can be seen as a placeholder that links relevant information together. As illustrated in Figure 8, observation can be considered an ID for each data record in our data model. Each raw depicts a data record.
ObservableProperty (OWL Class) [sosa]	*An observable quality (property, characteristic) of a FeatureOfInterest.* (e.g., temperature, humidity, presence)
Sensor (OWL Class) [sosa]	*Device, agent (including humans), or software (simulation) involved in, or implementing, a Procedure.* (e.g., Temperature sensor, humidity sensor, motion sensor). In our model, we have created a unique ID for each sensor based on its hosted platform.
Platform (OWL Class) [sosa]	*A Platform is an entity that hosts other entities, particularly Sensors, Actuators, Samplers, and other Platforms.* In UDEO, sensors are attached to different types of platforms, as shown in Figure8. We do not necessarily keep track of the exact location of the platform. However, location can be approximately identified by using the feature of interest.
FeatureOfInterest (OWL Class) [sosa]	*The thing whose property is being estimated or calculated in the course of an Observation to arrive at a result, or whose property is being manipulated by an Actuator, or which is being sampled or transformed in the act of Sampling.* In the context of UDEO, *BuidlingSpaces* are the *FeatureOfInterest* (e.g., offices, zones, floors). Most of the sensors are used to observe a property (phenomenon) of a location (e.g., the temperature in a room) [21].
Result (OWL Class) [sosa]	*The Result of an Observation, Actuation, or act of Sampling. To store an observation's simple result value, one can use the hasSimpleResult property.* Result is a placeholder to link related information, such as values and units. The UDEO model stores the data value and its unit type.
resultTime (Datatype Property) [sosa]	*The result time is the instant of time when the Observation, Actuation or Sampling activity was completed.* Each data record in the UDEO system comes with a time stamp. We attach a timestamp to each observation using this data property.
SpatialThing (OWL Class) [WGS84]	*A class for representing anything with a spatial extent, i.e., size, shape or position.*

3.5 Ontology Evaluation

This step evaluates the ontology quality in terms of *structure, semantic representation, and interoperability*. To evaluate the structure and semantic representation, we used the open-source online scanner, Oops! [22] and Pellet reasoner [23] built-in to Protege. Following that, we executed

SPARQL queries against the knowledge graph (i.e., UDEO instantiated with IoT datasets) to answer the Competency Questions (CQs) listed in the Appendix.

3.6 Experimentation Plan

Our experimentation plan consists of three layers, as shown in Figure 19, (i) data sources, (ii) adaptor layer; and (iii) evaluation. We aim to simulate a data marketplace using the most practical solution that fits the purpose.

- Data Sources: We expanded the UDEO to accommodate six data sources: docking stations, air quality, noise level, parking status, museums, and beaches. Further, we generated synthetic datasets for each data source that adhered to the FIWARE data model structure.
- Adaptor Layer: We mapped the six datasets into the Resource Description Framework (RDF) graph - referencing UDEO.
- Evaluation Layer: We stored the data from the adaptive layer in triple-store databases under three different architectures and evaluated each one's performance.

SynchroniCity IoT Data Marketplace receives data as JSON files (modeled using FIWARE standards) from data owners/publishers. SynchroniCity IoT Data Marketplace currently has a limited amount of data. It was not sufficient for us to conduct an experiment to uncover the utility of Semantic Web technologies and their impact on computing infrastructure. Therefore, we developed an algorithm to produce the required number of synthetic sensor data observations in JSON format (depending on the specific experiment). Then we transformed JSON data into Resource Description Framework (RDF) graph and loaded graph data into a triple-store database. We meant our algorithm could run and update concurrently or partially by modules. For example, the user may run the algorithm's first part to generate JSON data and then transform the output JSON file into RDF using the second part as an independent code. Our experience found that running the algorithm in stages consumed less time when generating many data observations (e.g., 1000K+). Code Snippet 1 explains the technique used. The UDEO and datasets were connected to create a knowledge graph. We employed Stardog [20], a knowledge graph platform that integrates heterogeneous and isolated data sources. Stardog hosts the triple-store databases and has an IDE (Stardog Studio) capable of performing numerous operations, including SPARQL, GraphQL, artificial intelligence, and machine learning. We wrote complex SPARQL queries to test the efficiency of retrieving information and inferring new phenomena. Further details are in the evaluation section.

4 EVALUATION

The characteristics exhibited in Figure 2 (d), such as sunny days and weekdays, have been modeled using SWRL rules. To illustrate the assessment, we inserted the rule into the database. We executed a query that conveyed, for instance, the condition of a sunny day, assuming a decrease in bike availability. To evaluate our knowledge graph's performance in terms of utility and response time, we evaluated it using three different architectures, each of which differs in how it stores and queries data. In the first instance, we used Code Snippet 1 to generate three incremental series of synthetic datasets. The number of generated observations was equal for each one. However, the volume varied when serialized into RDF graphs. Table 3 shows the kilobyte (KB) size for the generated RDF graphs. Furthermore, Figure 10 compares the volume of each data model. It was evident that the Noise Level dataset with more than 1K triples is the largest. That is due to the increased amount of metrics (e.g., CO, NOx) in a single observation compared to other data sources. The objective is to estimate how much data can be stored on a disk for each data model. It helps to understand how much data storage is required for a given use case, depending on the frequency of the observations

Algorithm 1: Data Generate, Transform and Load

```
Part 1– Generate
Function GenerateModel():
    Function GenerateId(size, chars + string):
        return join (chars for i in range(size));
        model = DataModel(Id ,type);
        n = name ;
        v = value ;
        model.add(n,v) ;
    return model ;
x = observation number (integer only) ;
for i in range(x) do
    data= GenerateModel();
    JSON.dump(data);
end for;
Part 2 –Transform ;
Data = ReadJSON(data) ;
for i, r in Data: do
    RDFData=Graph.add(subject,property,object);
    Graph.serialise(RDFData,format=turtle);
end for;
Part 3 – Load ;
StardogConnection=(endpoint,username,password);
DatabaseName = NewDatabase(RDFData) ;
connection = ConnectStardog(RDFData, StardogConnection);
connection.add(RDFData, format=turtle);
connection.commit
```

captured and the number of metrics modeled within each observation. We then interrogated the data stores to answer Competency Questions (CQs) such as *where can I park and ride?*. We developed complex SPARQL queries and executed them across the databases in question. Then, we inserted Semantic Web Rule Language (SWRL) into the databases and reran the queries. We assessed each approach by checking the correctness of each query's result (i.e., the answer to the competency question) and comparing response time on databases with and without SWRL. Subsequently, we reflected on the impact of complex queries and reasoning processes on data, highlighting some strengths and weaknesses. Furthermore, we analyzed the collected data to compare the results and determine if response time is faster after inserting SWRL rules. In other words, to examine if reasoning and setup reduce the query response time. Finally, key findings suggested the most suitable approach for the data marketplace. The accumulated response time observations were analyzed with the following steps to detect the difference between the two groups:

- Line graph to visualize and compare the data.
- Testing the data distribution with the Shapiro-Wilk normality test to determine the appropriate statistical test for detecting the difference between two groups (i.e., parametric or non-parametric). Here, the histograms and test p-value suggested that the observations were not found to be not normally distributed. Accordingly, we conducted non-parametric tests.

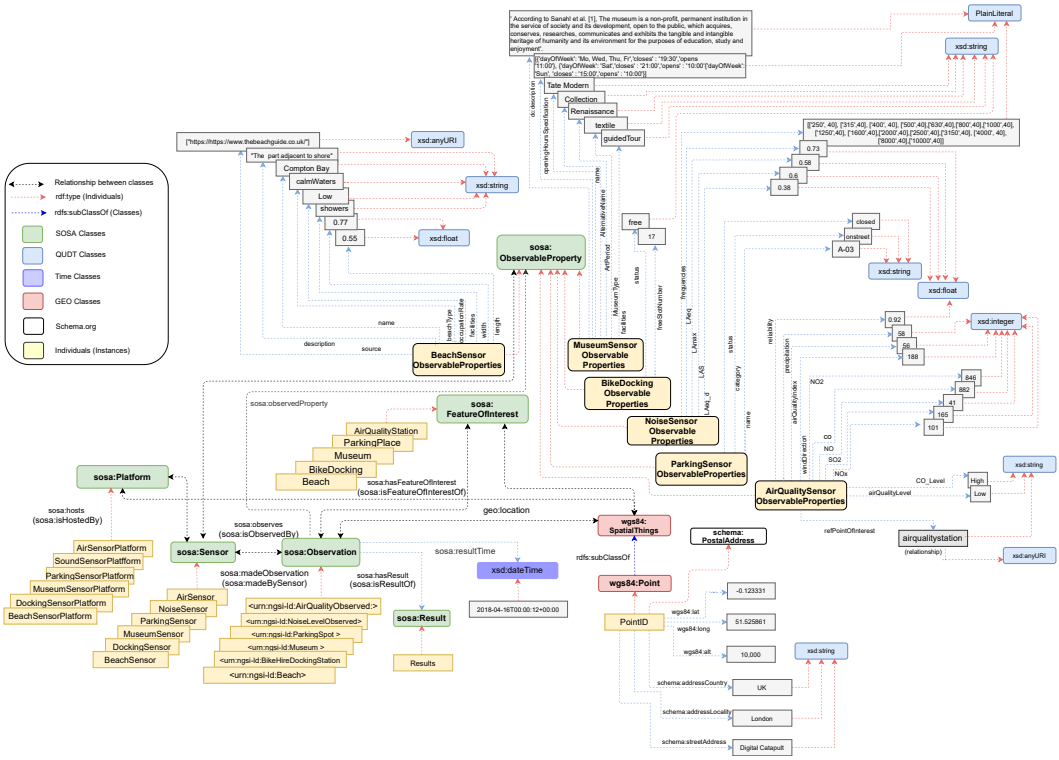


Fig. 8. The proposed Urban Data Exchange Ontology (UDEO) instantiated with six different sensors' datasets

- Using Kruskal-Wallis to check if the two groups of observations (No-rule-SWRL) are related (i.e., sampled from the same distribution)
- Finally, Mann-Whitney, the statistical test proved if there is a significant difference between the two groups. In other words, to examine if SWRL increased or reduced the query efficiency and response time.

Therefore, the claimed hypotheses can be formulated as:

Hypothesis 1 (H0). *SWRL reduces query response*

Hypothesis 2 (H1). *SWRL does not reduce query response time.*

The significance level of

$$\alpha = 0.05$$

We retain the null hypothesis if the p-value is greater than the significance level.

4.1 Rule-based Reasoning

In the context of the Semantic Web in general and RDF graphs in particular, reasoning, also referred to as inferencing, derives a new phenomenon from a given dataset based on named axioms, applicable rules, and definitions in the data model. Reasoning rules are declarative and represent proven knowledge or concepts modelled by experts. Rule-based inferring uses conditional IF-Then entailment rules. The logical consequences in the IF clause are inferred in the statement of Then. For example, outdoor activities are busier than usual on sunny days. Reasoning can reshape and align data, creating new views of data and connections. More importantly, it validates domain

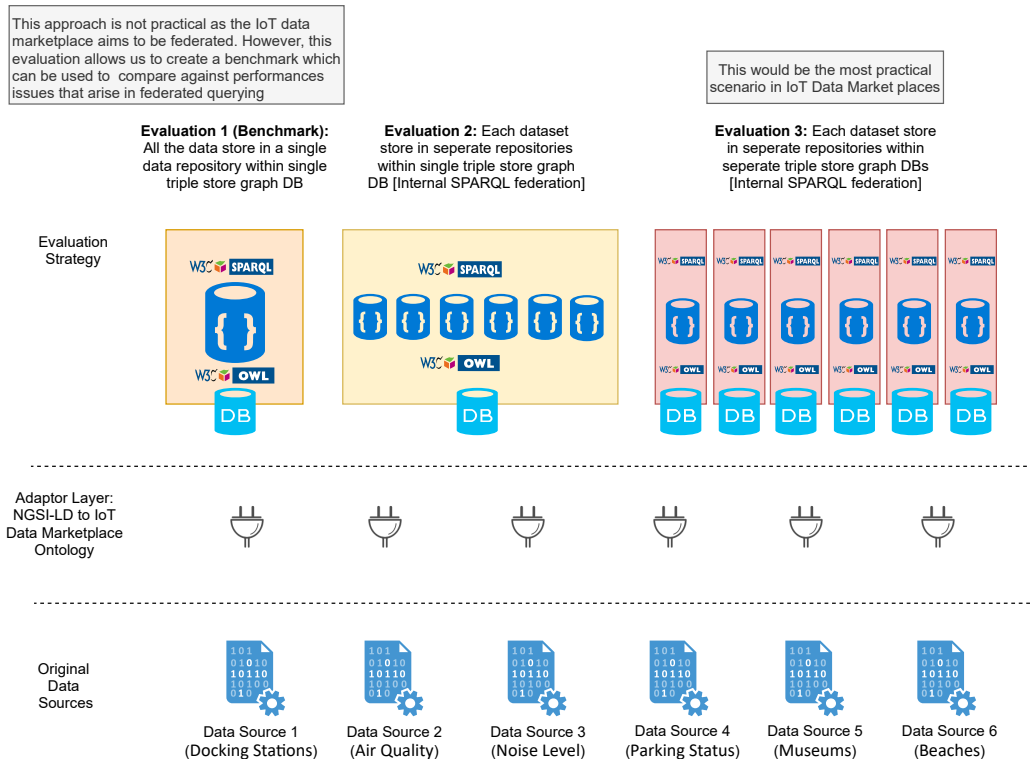


Fig. 9. Experimental Setup

Data Model	1K	10K	100K
Air Quality	90	903	9023
Beaches	108	859	8580
Docking Stations	119	1188	11877
Museums	153	1215	12141
Noise Level	135	1347	13471
Parking Status	99	918	9180

Table 3. Data Model RDF Graph Size (KB)

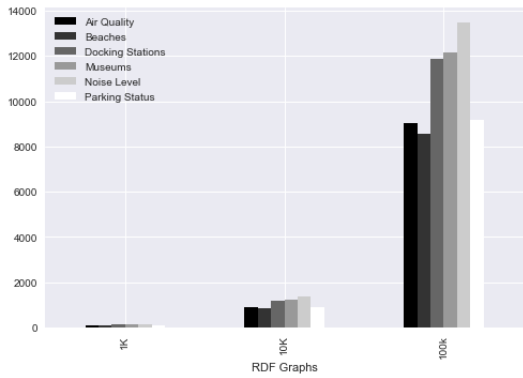


Fig. 10. Comparison between Data Model Sizes(KB)

modeling and detects violations. One of the features is the reasoning at query time. Besides the excellent performance, it allows users to specify the type and pay only for its reasoning usage. Reasoning can be enabled or disabled via a simple boolean command. When enabled, rules or axioms are triggered, and reasoning executes according to its value in the database. In this case study, we created a rule that assumes low rental bike availability during sunny days. We sketched this assumption to prove a concept that may not reflect objective reality. As discussed in the coming

sections, the sunny days rule is injected into our database and used to evaluate its performance in three different scenarios.

4.2 Evaluation One

The first experiment loaded all six datasets and our ontology (UDEO) into a single database. We executed the SPARQL query scripts at Code Snippet 3 and 4 a hundred times consecutively and respectively. The queries yield vacant car parking spots and available bikes near a geographical location. We recorded the time taken by each script in milliseconds (ms). We repeated the experiment after inserting SWRL, which could infer a sunny day. The SPARQL queries were executed 100 times before and after reasoning (i.e., inserting SWRL). Figure 11 shows the query responses for each query type (i.e., No-rule and SWRl), and the histograms in Figure 14 unveil the data distribution. The obtained p-values from Shapiro-Wilk, Kruskal-Wallis and Mann Whitney, as listed in Table 26, were far below the significance level of 0.05. Therefore, we reject the null hypothesis and conclude that SWRL does not reduce response time in this query setup.

Listing 1: SunnyDays Rule

```
Prefix rule: <tag:stardog:api:rule:>
[] a rule:SPARQLRule ;
rule:content """
PREFIX fiware:
<https://uri.fiware.org/ns/data-models#>
IF
 {?id a fiware:BikeHireDockingStation;
       fiware:AvailableBikeNumber;
       ?AvialableBikeNumber;
BIND(xsd:integer(AvialableBikeNumber) <5
AS ?SunnyDays)}
THEN
 {?id  fiware:SunnyDays  ?SunnyDays}""".
```

Listing 2: PREFIXES for *"Where can I park my car and ride a bicycle?"*

```
PREFIX fiware://<https://uri.fiware.org/ns/data-models#>
PREFIX ngsi: <https://uri.etsi.org/ngsi-ld/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX schema: <https://schema.org/>

PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geof:<http://www.opengis.net/def/function/geosparql>

PREFIX unit: <http://qudt.org/vocab/unit#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

Listing 3: Evaluation One NoRule Result Response Time = 237 ms

```
Select *
{?id a  fiware:ParkingSpot;
      fiware:category ?category;
      fiware:dataProvider ?dataProvider;
      ngsi:status ?status;
      ngsi:location ?location;
      ngsi:ParkingPoint ?ParkingPoint.
?id2 a  fiware:BikeHireDockingStation;
      fiware:availableBikeNumber ?availableBikeNumber;
      schema:address ?address;
      ngsi:status ?Bikestatus.
sosa:PointID a pos:Point;
        pos:SOSAPoint  ?SOSAPoint.
BIND (geof:distance
  (?ParkingPoint, ?SOSAPoint, unit:Kilometer)
  as ?Distance).
FILTER(xsd:integer(?Distance < 500))
FILTER(REGEX(?Bikestatus, "free"))
FILTER(REGEX(?availableBikeNumber, "1"))
FILTER(REGEX(?category, "offstreet")).}}
LIMIT 1
```

Listing 4: Evaluation One-SWRL-Result Response Time= 571 ms

```
Select *
{?id a  fiware:ParkingSpot;
      fiware:category ?category;
      fiware:dataProvider ?dataProvider;
      ngsi:status ?status;
      ngsi:location ?location;
      ngsi:ParkingPoint ?ParkingPoint.
?id2 a  fiware:BikeHireDockingStation;
      fiware:availableBikeNumber ?availableBikeNumber;
      schema:address ?address;
      ngsi:status ?Bikestatus;
      fiware:SunnyDays ?SunnyDays.
sosa:PointID a pos:Point;
        pos:SOSAPoint  ?SOSAPoint.
BIND (geof:distance
  (?ParkingPoint, ?SOSAPoint, unit:Kilometer)
  as ?Distance).
FILTER(xsd:integer(?Distance < 500))
FILTER(REGEX(?Bikestatus, "free"))
FILTER(REGEX(?availableBikeNumber, "1"))
FILTER(REGEX(?category, "offstreet")).}}
LIMIT 1
```

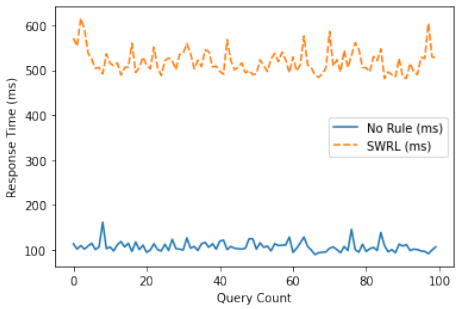


Fig. 11. Evaluation 1 visualisation

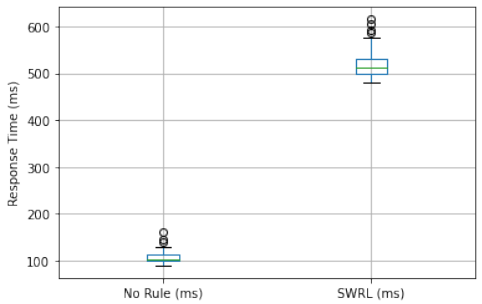


Fig. 12. Evaluation 1 boxplot

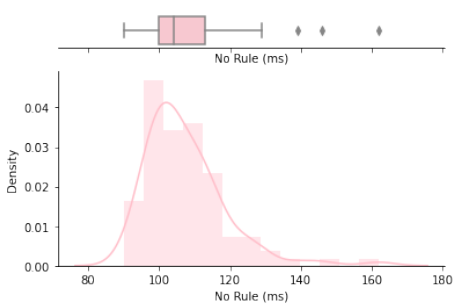


Fig. 13. Evaluation 1 No-Rule distribution plot

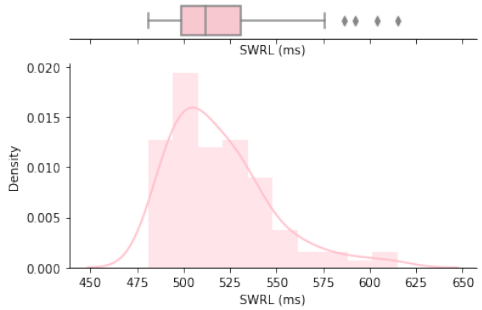


Fig. 14. Evaluation 1 SWRL distribution plot

Listing 5: Evaluation Two NoRule Result Response Time= 32 ms

```
SELECT *
{SERVICE <db://BikeHireDockingStation100k>
   {?id a  fiware:BikeHireDockingStation;
         fiware:availableBikeNumber ?availableBikeNumber;
         schema:address ?address;
         ngsi:status ?Bikestatus.}
{SERVICE <db://Parking>
   {?id2 a fiware:ParkingSpot;
         fiware:category ?category;
         fiware:dataProvider ?dataProvider;
         ngsi:status ?status;
         ngsi:location ?location;
         ngsi:ParkingPoint ?ParkingPoint.
sosa:PointID a pos:Point;
pos:SOSAPoint  ?SOSAPoint.
BIND (geof:distance
    (?SOSAPoint,?ParkingPoint, unit:Kilometer)
   as ?Distance).
FILTER(xsd:integer(?Distance < 290))
FILTER(REGEX(?Bikestatus, "free"))
FILTER(REGEX(?availableBikeNumber, "1"))
FILTER(REGEX(?category, "offstreet")).}}}
LIMIT 1
```

Listing 6: Evaluation Two-SWRL Result Response Time= 47 ms

```
SELECT *
{SERVICE <db://Parking>
{?id2 a  fiware:ParkingSpot;
      fiware:category ?category;
      fiware:dataProvider ?dataProvider;
      ngsi:status ?status;
      ngsi:location ?location.}
      # ngsi:ParkingPoint ?ParkingPoint.
{?id a fiware:BikeHireDockingStation;
      fiware:availableBikeNumber ?availableBikeNumber;
      fiware:AvailableBikeNumber ?AvialableBikeNumbe;
      schema:address ?address;
      ngsi:status ?Bikestatus;
      fiware:SunnyDays  ?SunnyDays.
sosa:PointID a pos:Point;
pos:SOSAPoint  ?SOSAPoint.
# BIND (geof:distance
#?SOSAPoint, ?ParkingPoint, unit:Kilometer)
#as ?Distance).}
# FILTER(xsd:integer(?Distance < 290))
FILTER(REGEX(?Bikestatus, "free"))
FILTER(REGEX(?availableBikeNumber, "1"))}
LIMIT 1
```

4.3 Evaluation Two

The second experiment stored each dataset with the UDEO locally but in separate databases using the same Stardog instance. Here, the SPARQL federation answered the competency question. For

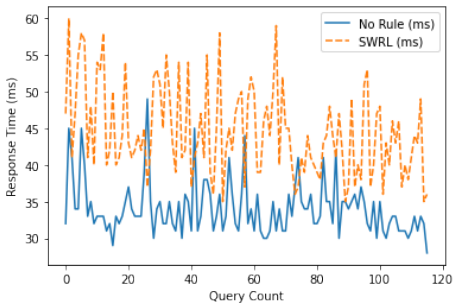


Fig. 15. Evaluation 2 visualisation

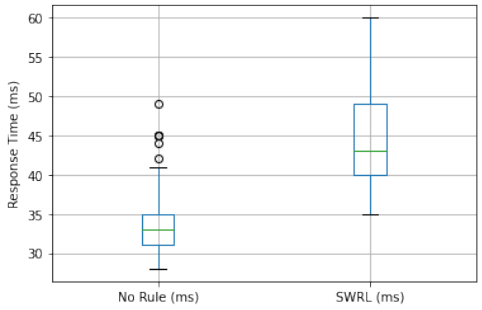


Fig. 16. Evaluation 2 boxplot

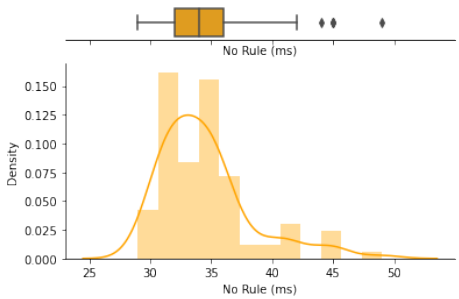


Fig. 17. Evaluation 2 No-Rule distribution plot

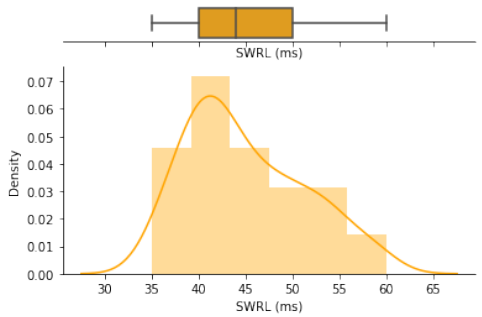
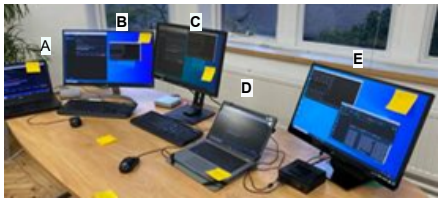


Fig. 18. Evaluation 2 SWRL distribution plot

instance, the query targeted the data source internally by using the SERVICE keyword with the URI typed as *db://database* instead of the SPARQL endpoint URL. For further analysis, we executed the SPARQL queries shown in the Code Snippet 5 and 6 hundred times, with and without SWRL-recording observations. In the same manner, the line plots 15 compared the two variables, while the histograms in 17 suggest that the observations do not follow the normal distribution. Noticeably, this setup (locally separated databases) has a quicker response time than the unified database in evaluation one. Similar to evaluation one 4.2, the p-values from Shapiro-Wilk, Kruskal-Wallis and Mann-Whitney, as listed in Table 26, were less than the significance level of 0.05. Once again, we reject the null hypothesis and recommend that injecting SWRL into the internally distributed databases does not accelerate response time.

4.4 Evaluation Three

In the last experiment, we stored each database on a separate computer node under different Stardog instances. Every machine acted as an independent data provider. We executed federated SPARQL queries, as illustrated in Code Snippet 7 and 8 to answer our competency question from the desired database without moving or copying data. This time, the query targeted a SPARQL endpoint on a remote machine. Therefore, an IP address was required along with the port number to reference the SERVICE Keyword. Nevertheless, HTTP authentication was also necessary to access the reference SPARQL endpoint. It can be achieved by disabling Stardog security on startup or storing password credentials in the Stardog directory. The same query for all other evaluations was executed concurrently, in the same manner, with and without SWRL. Thus, response times were recorded for analysis (i.e., Table 26) and comparison (i.e., Table 20) purposes. Astonishingly,



Name	Data Provider	CPU	SSD (GB)	RAM (GB)
A	Laptop 1	Intel® Core™ i7-6Gen@2.60GHz(8CPUs)	480	16
B	MiniPc3	Quad-Core Intel Atom Z8350 @1.44 GHz	128	8
C	MiniPc2	Quad-Core Intel Atom Z8350 @1.44 GHz	128	8
D	Laptop2	Intel Core i7 1.8GHz 10Gen	512	16
E	MiniPc1	Intel® Core™ i3 5Gen@2.00GHz (4CPUs)	60	8

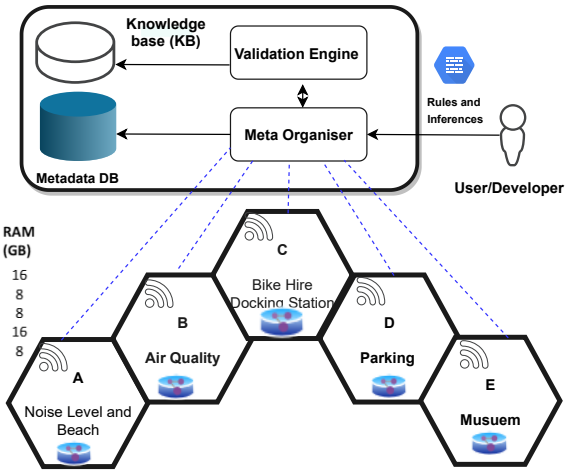


Fig. 19. Evaluation Three Setup, A to E represent the different data providers' machines. Data providers may store their data on edge using mini pcs or dedicated computers. Here, the user/developer places a request. It gets verified by the validation engine and passed to the meta organizer that knows the data provider that has the requested information.

the average response time with SWRL was faster than no-rule. Accordingly, we retain the null hypothesis and conclude that SWRL reduces the query response time in the decentralized data storage manner.

Listing 7: Evaluation Three NoRule Result Response Time = 99 ms

```
SELECT *
 {SERVICE <http://192.168.0.128:5820/Parking/query>

    {?id2 a fiware:ParkingSpot;
          fiware:category ?category;
          fiware:dataProvider ?dataProvider;
          ngsi:status ?status;
          ngsi:location ?location.}
{SERVICE <http://192.168.0.128:5820/Bike/query>

    {?id a  fiware:BikeHireDockingStation;
          fiware:availableBikeNumber ?availableBikeNumber;
          fiware:AvailableBikeNumber ?AvialableBikeNumber;
          schema:address ?address;
          ngsi:status ?Bikestatus;}}
LIMIT 1
```

Listing 8: Evaluation Three SWRL Result Response Time = 86 ms

```
SELECT *
{SERVICE <http://192.168.0.128:5820/Parking/query>
{?id2 a fiware:ParkingSpot;
      fiware:category ?category;
      fiware:dataProvider ?dataProvider;
      ngsi:status ?status;
      ngsi:location ?location.}

{?id a  fiware:BikeHireDockingStation;
      fiware:availableBikeNumber ?availableBikeNumber;
      fiware:AvailableBikeNumber ?AvialableBikeNumber;
      schema:address ?address;
      ngsi:status ?Bikestatus;
      fiware:SunnyDays  ?SunnyDays.}}
LIMIT 1
```

4.5 Results

Data aggregated (n=100) for each evaluation was fed to a Python code for visualization, exploration, and statistical testing. Figure 24 are line graphs that visualize the flow of each trail. Querying the database after inserting SWRL took more time than querying the database without a rule. We performed a descriptive analysis to understand the data. Then, Shapiro-Wilk, Kruskal-Wallis, and Mann-Whitney statistical tests were used to check the distribution of the samples and detect and compare differences between the two independent samples (no-rule/SWRL) for each experiment, respectively. *Descriptive Analysis* in Figures 25 explored the datasets to help understand the data content and characteristics. For example, the line graphs for evaluation one indicated a considerable

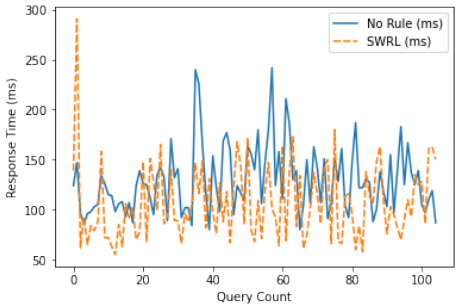


Fig. 20. Evaluation 3 visualisation

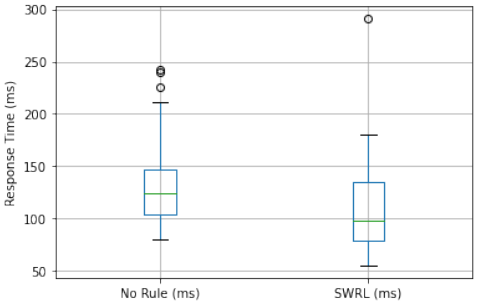


Fig. 21. Evaluation 3 boxplot

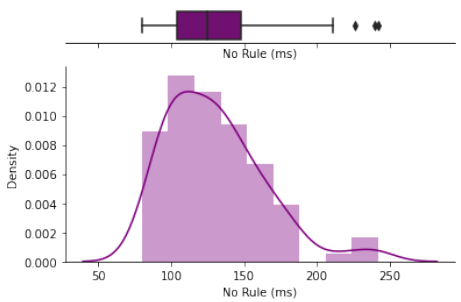


Fig. 22. Evaluation 3 No-Rule distribution plot

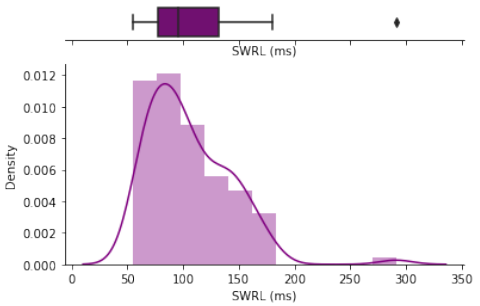
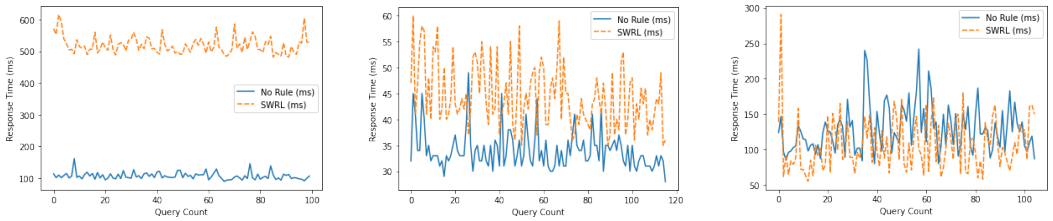


Fig. 23. Evaluation 3 SWRL distribution plot

difference between query time responses for the dataset with and without SWRL. Unlike evaluation two, where the gap narrowed dramatically, it was surprising that querying with SWRL was slightly faster in evaluation 3. This result could be due to the distribution of the datasets over disparate nodes and the remote access, which distinguished evaluation three from the others. The mini-PC nodes that hosted the datasets had relatively small disk spaces ranging from 32GB to 128 GB and no less than 8GB of RAM. Noticeably, the number of observations was equal in all three experiments, the mean fluctuated between approximately 34.45 and 519.13, and the standard deviation was at its lowest point of 3.78 in evaluation two. To further discover the data distribution, the histograms and *Shapiro-Wilk* decided the most appropriate statistical tests. It was clear from the histograms that the data in the three evaluation datasets did not resemble bell curves. Shapiro-Wilk test confirmed the non-normality further through the p-values (<0.05). Therefore, we rejected H0 with a 95 percent confidence interval and concluded that all datasets do not follow the normal distribution. *Kruskal-Wallis test* was the non-parametric test that suited our data distribution. It suggested that the two samples (no-rule/SWRL) for each experiment came from different distributions with p-values of less than 0.05. Figure 26 explained in detail the hypotheses result of the evaluations based on their p-values, and the final test, *Mann-Whitney*, was used to determine if the response time was different after inserting SWRL. The result also suggested a significant difference between the samples of each experiment (no-rule/SWRL).

4.6 Cost of Adapting Linked Data

The feasibility of linking distant, semantically modeled data sources and reasoning over them using SWRL has been established. It facilitates quick access to diverse data sources. Therefore, data owners can retain and manage their data while sharing only their metadata with the IoT data



(a) (b) (c)

Fig. 24. Compare the three evaluations of query response time (ms), with and without reasoning rules inserted. (a) Evaluation One (all datasets and the ontology are stored in a single database). (b) Evaluation Two (each dataset and ontology are stored locally in separate databases). (c) Evaluation Three (each dataset and ontology stored remotely with autonomous data providers)

	No Rule (ms)	SWRL (ms)
count	100.00	100.00
mean	107.40	519.13
std	11.44	27.17
min	90.00	481.00
25%	100.00	498.75
50%	104.00	512.00
75%	113.00	530.75
max	162.00	615.00

(a)

	No Rule (ms)	SWRL (ms)
count	100.00	100.00
mean	34.45	45.00
std	3.78	6.28
min	29.00	35.00
25%	32.00	40.00
50%	34.00	44.00
75%	36.00	50.00
max	49.00	60.00

(b)

	No Rule (ms)	SWRL (ms)
count	100.00	100.00
mean	130.04	105.68
std	33.95	37.55
min	80.00	55.00
25%	104.00	77.50
50%	125.00	96.00
75%	147.75	131.75
max	242.00	291.00

(c)

Fig. 25. Descriptive analysis for the three evaluations of query response time (ms)—with and without reasoning rules inserted. Tables explore the datasets' nature and summarize their contents.

Evaluations	Shapiro-Wilk H0 : Data follow a normal distribution. H1 : Data do not follow a normal distribution.				Kruskal-Wallis H0 :Two samples are related H1 :Two samples are not related		Mann Whitney H0 : Sample distributions are equal H1 :Sample distributions are not equal	
	P-value = 0.05							
	No Rule	Result	SWRL	Result	No Rule /SWRL	Result	No Rule /SWRL	Result
One	4.70508E-08	*Reject H0*	5.02E-06	*Reject H0*	0.000	*Reject H0*	0.000	*Reject H0*
Two	3.52513E-08	*Reject H0*	0.0006118	*Reject H0*	0.000	*Reject H0*	0.000	*Reject H0*
Three	4.14385E-05	*Reject H0*	4.94E-07	*Reject H0*	0.000	*Reject H0*	0.000	*Reject H0*

Fig. 26. Evaluation Statistical Analysis

marketplace. The results indicate that semantic data sources efficiently send small packets through the communication network. For instance, when reasoning is performed, the query response time decreases. However, the information overhead and implementation cost must be evaluated before system deployment.

- Typically, information overhead stems from the data structure, runtime, and data exchange. In our methodology, the semantic data sources are database engines whose data are represented

Evaluations	Average Query Time (ms)				
	SPARQL Query		Difference	%	
	No Rule	SWRL			
One	107.4	519.13	411.73	383	↑
Two	34.45	45	10.55	31	↑
Three	130.04	105.68	-24.36	-19	↓

Fig. 27. Evaluations query average time comparison

in RDF and queried via their SPARQL endpoints. The most significant expense is the creation of an ontology for each data source, as query resolution necessitates a complete merging ontology. Here, ontologies are created by a large consortium of academics and industry professionals [17, 24]. Occasionally, different consortiums may have differing views on the data type, making it difficult to model the data in a single format. Simultaneously, Semantic Web technologies enable the merging and reasoning over ontologies, allowing different classes, for instance, to be defined as equivalent.

- Costs associated with the system implementation may be related to the need for dedicated computers for data storage and retrieval. As depicted in Figure 19, These computers had between 8GB and 16GB of RAM, and the smallest solid-state disc (SSD) capable of storing the relevant data source was 60GB in size. Table 3 illustrates how much RDF-formatted data can be stored on a single storage disc. For example, an air quality sensor that generates one observation per minute would require approximately one KB of storage space on a disc for 90 observations. Thus, the data provider can determine the amount of information stored on a 60GB SSD. As mentioned, calculating the overhead could determine the total cost before implementation. Although using inexpensive computing devices contributes to the efficiency of this system, the cost may be increased by the labour time of human resources.

4.7 Use Cases

Our proposed on-demand data model is applicable in various industries. It offers practical and cost-effective solutions for SMEs. Businesses can build various innovative services enriched with machine learning and AI that respond to end-users personally. In comparison, legacy data trading constrained businesses to acquire bulky datasets that may incur more charges and require high maintenance. (i.e., filtering to process relevant data records). This study's hypothetical use cases concern the tourism and housing industries. The former is a small company that enables consumers to browse and book trips to local attractions, promoting sustainable travel. The latter is a state agency recommending properties with considerably clean air features (i.e., properties in less polluted and quiet areas).

4.7.1 ***Use Case 1***. **TripRecomender** is a small business that enables end-users to plan green trips to local attractions. It predominantly aims for high-rated customer satisfaction by leveraging AI's self-learning competencies. The company adopted the online chat application, called bots or chatbots. An intelligent program relies on actual data to carry out a specific task. Here, TripRecommender is planning to train its chatbot to search for and suggest local tourist destinations and sustainable travel solutions, increasing customer satisfaction and boosting the company's revenue. Well-trained bots can reduce the time and effort spent on monotonous trip planning. TripRecommender considers users' requests and suggests tailored options based on previously known information. The company's chatbot requires sufficient relevant data to analyze and turn into meaningful information to accomplish this task. Relevant data records may exist in separate

data sources, making aggregating challenging. With our on-demand data model, TripRecommender can query and retrieve granular data records that are fit to train the bot. (e.g., ten years of data for either the local beaches occupation rate or the availability of rental bikes on weekends at the local docking stations). Subsequently, it recommends customized offers and infers new events. For example, the SPARQL query in **Code Snippet 9** expresses how our model retrieves certain weekend information about (i) a local beach's name, services, and occupancy rate, (ii) a museum's opening hours, and (iii) an available rental bike location.

Listing 9: Use Case 1 - TripRecommender

```
PREFIX : <http://api.stardog.com/>
PREFIX stardog: <tag:stardog:api:>
PREFIX schema: <https://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22–rdf–syntax–ns#>
PREFIX ngsi–ld:<https://uri.etsi.org/ngsi–ld/default–context/>
PREFIX fiware:<https://uri.fiware.org/ns/data–models#>
PREFIX ngsi: <https://uri.etsi.org/ngsi–ld/>
SELECT *
   {?id a fiware:Beach;
   ngsi:name ?Name;
   fiware:facilities ?Facilities;
   fiware:occupationRate ?OccupationRate;
   fiware:Weekends   ?Weekends.
{SERVICE   <http://192.168.0.78:5820/Museum/query>
   {?id2 a fiware:Museum;
   fiware:openingHoursSpecification   ?OpeningHours.
{SERVICE   <http://192.168.0.128:5820/Bike/query>
   {?id3 a fiware:BikeHireDockingStation;
   fiware:availableBikeNumber ?availableBikeNumber;
   schema:address  ?address;
   ngsi:status ?Bikestatus. }}}}}
```

4.7.2 ***Use Case 2.*** **CleanAir Housing** is a medium-sized business that sells and rents out residential properties. Recently, the company noticed a staleness and price drop for properties in highly polluted areas based on its sales records. Conversely, homes in quiet and less polluted areas will most likely sell in a year. Air pollution could negatively impact health. It happens when particular gases and liquid particles are released into the atmosphere, forming PM2.5 and PM10 particles and elevating levels of carbon monoxide (CO) and nitrogen dioxide (NO2) pollutants above the clean air legal limit. Sources of these toxic gases include vehicle exhaust, factories, and domestic combustion. Measuring the Air Quality Index (AQI) can assess the air quality in areas of interest. As a result, CleanAir Housing decided to integrate AI-powered services to (i) predict sales forecasts based on historical data and air pollution levels, (ii) recommend the optimal price to match the expected value, and (iii) hunt for local fast-selling homes. Several environmental data records are needed to calculate air quality and noise levels. Traditional ways to acquire such data are by deploying thousands of sensors or purchasing bulky ecological datasets. Both options demand time and funds. Hence, further data mining, processing, and analysis are required to extract valuable insights. Integrating our on-demand data model enables the filtering and extraction of the needed metrics from multiple datasets, forming search data that are fit to train the company's AI. For instance, the SPARQL query script in **Code Snippet 10** combined metrics from air quality and noise level datasets to filter out the addresses in areas with low noise levels and good AQI.

Listing 10: Use Case 2 - CleanAir Housing

```
SELECT*
{SERVICE  <http://192.168.0.128:5820/AirQuality/query>;
   {?id a  fiware:AirQualityObserved;
      schema:address  ?address;
      fiware:dateObserved ?date;
      fiware:Precipitation ?Precipitation;
      fiware:Reliability  ?Reliability;
      fiware:WindDirection?WindDirection;
      fiware:AirQualityIndex ?AirQualityIndex;
      ngsi−ld:Co  ?Co;
      ngsi−ld:CO_Level ?CO_Level;
      ngsi−ld:No?No;
      ngsi−ld:Nox ?Nox;
      ngsi−ld:No2 ?No2;
      ngsi−ld:So2 ?So2.
 {SERVICE  <http://192.168.0.78:5820/Noise/query>
   ?id2 a  fiware:NoiseLevelObserved;
      fiware:DateObservedFrom ?DateObservedFrom;
      fiware:DateObservedTo ?DateObservedTo;
      fiware:frequencies ?frequencies;
      fiware:DataProvider ?DataProvider;
      ngsi:location  ?location;
      ngsi−ld:lAeq_d ?lAeq_d;
      ngsi−ld:lAmax ?lAmax.
      FILTER(xsd:float(?lAmax) > 0.72)
      FILTER(REGEX(?address, "A"))
      FILTER(REGEX(?CO_Level,"Low"))
      FILTER(xsd:integer(?AirQualityIndex) < 100)
      FILTER(xsd:integer(?Nox) < 100)}}}
   Limit10
```

5 RELATED WORK

5.1 Data Trading and Marketplaces

The IoT data trading field has seen significant attention in recent years. Researchers have focused on the challenges and opportunities associated with buying and selling datasets online. Liang et al. highlighted the value of appropriately sharing IoT data between data owners and consumers [25]. Schomm et al. discussed the trend of monetizing data sharing through data marketplaces [26]. These marketplaces have the potential to streamline the process of accessing desired data and contribute to the development of more efficient city services [27]. However, data trading has its challenges. The granularity of data availability, market awareness, and privacy concerns have discouraged data traders [28–30]. Data consumers often face the issue of having to purchase entire datasets even if they only need a small portion of the data, leading to unnecessary costs and computational burden [31]. Liu et al. pointed out that there is a need for alternative options for purchasing data, and slow-changing datasets are significant drawbacks of existing data marketplaces [32]. To address these challenges, they proposed a collaborative environment that aims to maximize fairness in data trading through the computation of data costs using a bespoke algorithm. Similarly, Patrizi et al. focused on optimizing data gain and rewards between data providers and consumers through contractual agreements [33]. Several related studies have explored different aspects of data trading in the IoT context. Ren et al. developed an architecture called Datum, which focuses on minimizing the total cost of IoT data through efficient buying and deployment decisions [34]. Jung et al. introduced PRIVATA, a private environment that enables users to place data orders with intended prices and negotiate ceilings [35]. While these approaches address specific aspects of

data trading, they provide a different level of integration of disparate data sources and flexibility than the proposed customized data ordering mechanism. Anthony [36] put forward a design for an ecosystem that combines Distributed Ledger Technology (DLT) and the MQTT protocol to simplify the process of exchanging and trading data in smart cities. By implementing this ecosystem, the study aimed to create a more efficient and seamless data marketplace that supports the development of smarter cities. While these approaches offer valuable insights into data trading, they need to fully address the integration of disparate data sources and the flexibility needed for cost-effective data retrieval. Our study takes a different approach by proposing a customized data ordering mechanism. Rather than purchasing entire datasets, data consumers can retrieve specific data from diverse datasets and negotiate costs based on their needs. This approach aims to reduce unnecessary data acquisition, lower costs, and enhance the efficiency of data trading.

5.2 Smart City Data Modeling

Sensor datasets are often aggregated and showcased in online data marketplaces using various methods. Data models [37] and other initiatives such as in [38] emphasized the openness to innovation in FIWARE, specifically in semantic data integration. NYAYA introduced by De Virgilio et al., focused on managing large volumes of semantically modeled datasets, offering scalable storage and powerful querying capabilities [39]. Fernandez et al. developed a system explicitly addressing data discovery, integration, and sharing challenges within isolated environments [30]. Our study, as such, employed semantic data modeling. Our datasets which followed FIWARE model were converted into RDF format, enabling remote querying through SPARQL endpoints. Notably, our study demonstrates faster response times compared to NYAYA [39]. One of the challenges in accessing semantically integrated information lies in connecting to remotely distributed systems and obtaining accurate and timely responses to user queries. To tackle this, Le-Phuoc et al. [40] proposed using Linked Stream Middleware, which leveraged Semantic Web technologies and SPARQL queries to achieve system interoperability and simplify the integration of time-sensitive information with relevant linked data.

In our study, access to semantically curated data is provided in their original sources, eliminating the need for data duplication or movement. Evaluating semantically modeled data often involves competency questions formulated as SPARQL queries. Lopez et al. focused on evaluating the competency of answers obtained through SPARQL queries and reasoning engines, with a particular emphasis on transforming user questions expressed in natural language into accurate and prompt semantic queries [41]. In contrast, our study tackles the challenge of constructing complex SPARQL queries that incorporate reasoning rules to respond to user inquiries effectively.

The efficiency of this approach is evaluated by considering the response time of complex queries that retrieve and reason over data from diverse sources. Middleware solutions have also been explored in the literature. Fisteus et al. developed Ztreamy, a scalable and high-speed middleware that includes data sharing and semantic filtering services, enabling efficient publishing of real-time information to many clients [42]. Platforms like SPUD have been developed to incrementally discover, model, and link dynamic and static data, with a particular focus on real-time traffic information processing [43]. Lécué et al. studied STAR-CITY, a semantic traffic system deployed in multiple cities, emphasizing its scalability and ability to be replicated in other locations [44]. Le-Phuoc et al. introduced the Graph of Things (GoT) system, which ingests massive streams of heterogeneous data and allows access and querying through a SPARQL endpoint [45]. Ali et al. developed a semantically enriched system for event detection and data analytics using distributed and heterogeneous data streams from IoT-enabled communication systems [46]. Cirillo et al. [47] examined three real-world use cases (global IoT market, smart city analytics, and IoT augmented autonomous driving) to showcase how the FIWARE platform addresses their specific

requirements. The study also emphasized FIWARE's commitment to innovation, particularly in the areas of semantics and privacy. Lujic [48] addresses the challenges of near real-time edge analytics and proposes a framework called SEA-LEAP that improves data replication and placement for on-demand analytics. The experiments demonstrate its effectiveness in reducing latency and optimizing decision-making processes in edge environments. On the other hand, our study focuses on enhancing the SynchroniCity data marketplace by using Semantic Web technologies. It allows consumers to pay for specific sensor information instead of purchasing entire datasets. The study also highlights the benefits of semantic modeling and reasoning, interoperability through SPARQL, and the advantages of edge computing.

In comparison to these works, our study focuses on leveraging semantic modeling and the capabilities of FIWARE to facilitate the trade of sensor data in smart cities, mainly targeting semantic data developers, including small and medium enterprises (SMEs).

5.3 Semantic Web and Linked Data Applications

The Semantic Web, which uses standards like RDF, OWL, and SKOS, facilitates the integration of data from various sources. Researchers have explored the use of Semantic Web technologies in different domains [37, 38], highlighting their benefits and addressing various challenges [49–51]. Hitzler argued that these technologies provide a cost-effective approach to data discovery, publishing, and reuse [52]. Stephan et al. emphasized their efficiency in achieving transparency [53], while Shadbolt et al. suggested that semantically integrating heterogeneous data using RDF can enhance their usefulness [54]. Mahmud et al. implemented a semantic model that converts CSV data into RDF with rich semantics, adhering to Linked Open Data principles [55]. Fabian et al. extended the RDF model to address challenges in open data publication, including API limitations, search and browsing, information quality, security, and licensing standards [56].

In the field of education, Wu et al. [57] and Bashir et al. [58] applied Semantic Web technologies. Stephan and Tchouanguem Djuedja [53, 59] have explored their use in the environmental domain, while Moreno et al. [60] and Margan et al. [61] have applied them in the context of transportation. The health sector has also witnessed the benefits of Semantic Web technologies. Dragoni et al. developed PerkApp, a platform for monitoring citizens' lifestyles in real-time and promoting healthy living [62]. Linked open data has been successfully used in the development of context-aware recommender systems, as demonstrated by Fogli et al. [63].

Furthermore, Semantic Web technologies have made an impact in industry, particularly in cloud environments. Xie et al. [64] and Petersen et al. [65] have used these technologies to model, unify, and share heterogeneous resources. Turning to IoT data marketplaces, Serrano et al. discussed the FIESTA project, which focused on providing tools and techniques for data access, testbed integration, and authentication and authorization services [66]. On the other hand, our study extends the SynchroniCity data marketplace by incorporating Semantic Web technologies. Our emphasis is on enabling consumers to selectively pay for specific sensor information rather than purchasing entire datasets. We proposed a semantic model, conducted experiments, and evaluated the effectiveness of semantic modeling and reasoning using SPARQL queries. Our work also explored interoperability through SPARQL endpoints and highlighted the advantages of edge computing for distributed RDF datasets. These studies collectively demonstrated the potential of Semantic Web technologies in improving data marketplaces. They offered advantages such as data integration, customization, transparency, and efficient data access.

6 DISCUSSION, LIMITATIONS AND FUTURE RECOMMENDATIONS

SynchroniCity data marketplace sells sensor data in bulk. Consumers interested in specific observations from different sensors (e.g., air quality and noise level) must purchase each sensor's dataset

separately. Such a practice may incur more charges and cause a high-latency network. Our study extended SynchroniCity data marketplace with Semantic Web technologies to allow consumers to pay for the sensor's information they need - instead of buying the entire dataset. Consumers can acquire multiple observations from various data providers to fulfill their orders. For example, finding nearby parking spaces and museum opening hours on a particular date and time.

Our end-product consists of a user-friendly interface [67] with an interactive map and a semantic data model. More specifically, we (i) built a novel semantic model. It encompassed an Urban Data Exchange Ontology (UDEO) and FIWARE synthetic datasets for six different providers. (ii) conducted three different experiments, as shown in 19 to determine the most practical modeling and storage solutions for the IoT data marketplaces. (iii) evaluated the experiments to demonstrate the effectiveness of the semantic modeling and SWRL - using different SPARQL queries to answer related competency questions. The evaluation results support the hypothesis that reasoning over distributed data sources could be the ideal architecture for the IoT data marketplace. Evidently, in evaluation one 4.2, querying after inserting rules took a long time, and the time-lapse between queries with and without rules is relatively slow. In evaluation two 4.3, the query time response was dramatically reduced compared to evaluation one. It is worth mentioning that to make SWRL rules work, we got to insert the rule independently in each database, unlike the evaluation one, where we dealt with a single database. Even when we inserted SWRL in each database, it did not activate with the SERVICE keyword. The query line had to be executed within the rules inserted in the database. Query line targeting rules and reasoning responded when calling the database internally in the Stardog Studio workspace. In evaluation three 4.4, the requested data can be obtained remotely via HTTP, using the host's IP address, port, and database name. The user query breaks into triple patterns that interrogate data sources SPARQL endpoints for results. Figure 27 compared the average evaluation time between querying databases with and without SWRL. Evaluation One showed the highest difference in query response time among the evaluations. Unexpectedly, the average response time on SWRL databases was lower than without rules. Therefore, we can draw valuable insights from our semantic model results as follows:

Semantic modelling and reasoning: Extracting explicit information from IoT SynchroniCity datasets was challenging since these data lacked formal definitions for widely shared standards. Our semantic model transformed them into queriable triplestores. The adapter (code) mapped the data to RDF while referencing the UDEO. We inserted abstract rules into the databases to trigger reasoning such as *Sunnydays and weekends*. Sunny days rule sets available rental bikes level low, assuming higher demand on such days. While the weekends' rule deduced high occupancy rates on local attractions such as beaches and museums. Reasoning quickened query response time in experiment three by reducing the search space while filtering out information adhering to the rules. SPARQL queries retrieved granular and semantically enriched and reasoned information from different datasets, stored locally and remotely. As a result, customized data requests can be achieved at low costs.

Interoperability: We suggest that experiment three's approach is interoperable. SPARQL allowed remote access through its endpoints, achieving seamless data sharing between different RDF databases stored on heterogeneous machines.

***Edge computing*:** In experiment three, we distributed the RDF datasets on separate computers operating independently. Executing data on these edge computers satisfied the horizontal scaling property, provided storage capacity, allowed computational flexibility (i.e., semantic modelling), and maintained low network latency (i.e., transmitting query results instead of the whole dataset).

Limitations: Despite that, our semantic model slashed data price, reduced network latency, and cut down information overload in the SynchroniCity data marketplace- yet this approach has

some drawbacks. In particular, *the pricing structure, data platform security, data quality, and safe dissemination.* The pricing structure of our model allows consumers to pay for desired information instead of an entire dataset. Although it costs less, working out the total price of an order could be a complex task. We deal with different data providers with different data tariffs, broker fees, and taxes, if any. Each of these has independent calculations and may change over time. Therefore, offering fixed and competitive charges is an open challenge. We recommend adding a self-configuring pricing model that standardizes and price-marks data records across independent stores. For example, set one reasonable price for each data record- automatically updating to match the data market's supply and demand, then adjust the broker fees to be a fair percentage of the total bill. Regarding security, accessing and querying data stored in remote machines via HTTP pose risks. Stardog offers security options such as authentication and password encryption; so far, its default security settings are considered minimal for network communications. Therefore, we recommend using the Secure Sockets Layer (SSL) encryption when deploying Stardog in production mode. Concerning data quality, *Synthetic data* used in this study are consistent with good quality, while real sensors data may have errors and missing values. Hence, data quality should be carefully addressed to replicate our real-life study. We recommend using accurate machine learning algorithms and artificial intelligence to detect and automatically correct errors. The information retrieved to fulfil consumers' requests creates new datasets. These datasets have diverse sources collected by sensors owned by different stakeholders. Thus, publishing them may raise data ownership and privacy concerns. A remedy could be building a tool that (i) traces the data lineage and accurately identifies the owner. (ii) applies a GDPR-compliant privacy policy agreed upon by all parties (data buyer, seller/owner and broker).

7 CONCLUSIONS

Data marketplaces are a new category of online marketplaces. Therefore, they are not well-researched within the academic community or well-implemented within the industry. SynchroniCity represents the first attempt to deliver a Single Digital City Market for Europe by piloting its foundations at scale in 11 reference zones - 8 European cities and 3 more worldwide cities - connecting 34 partners from 11 countries across 4 continents. The primary goal is to meet the data needs of consumers. Data marketplaces also emphasize vital challenges around data acquisition. Data marketplaces incentivize owners to share the gathered data and recover part of the acquisition costs. A fundamental issue of syntactic data marketplaces such as SynchroniCity is that they do not selectively provide a mechanism to buy data. It means data consumers have to buy the entire datasets that data owners offer. We demonstrated the utility of annotating IoT data with semantics through experiments, allowing highly selective data querying. As a result, we converted the IoT data marketplace into a queryable data store. Moreover, we reduced data prices by allowing consumers to request less data selectively—only the relevant data they needed to achieve the task at hand. We used Semantic Web technologies to address the critical challenges in IoT data marketplaces without significantly burdening the compute infrastructure.

ACKNOWLEDGMENTS

We acknowledge the support received by Research in Residence Program (RiR) (EP/T517203/1) and EU H2020 funded Synchronicity project (Grant agreement ID: 732240).

REFERENCES

[1] Authors Asunción Gómez-pérez, Enrico Motta, and Mari Carmen Suárez-figueroa. Introduction to the NeOn Methodology Introduction to the NeOn Methodology. *Cycle*, (c), 2005.

[2] Asunción Gómez-Pérez and Mari Carmen Suárez-Figueroa. Scenarios for building ontology networks within the NeOn Methodology. *K-CAP'09 - Proceedings of the 5th International Conference on Knowledge Capture*, pages 183–184, 2009.

[3] The NeOn Methodology framework: Ascenario-based methodology for ontologydevelopment. *Applied Ontology*, 10(2):107–145, 2015.

[4] Helena Sofia Pinto, Steffen Staab, and Christoph Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *ECAI*, 2004.

[5] Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. Methontology: from ontological art towards ontological engineering. 1997.

[6] York Sure, Steffen Staab, and Rudi Studer. *On-To-Knowledge Methodology (OTKM)*, pages 117–132. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[7] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[8] María Poveda-Villalón, Alba Fernández-Izquierdo, Mariano Fernández-López, and Raúl García-Castro. Lot: An industrial oriented ontology engineering framework. *Engineering Applications of Artificial Intelligence*, 111:104755, 2022.

[9] Google. Google Scholar. https://scholar.google.com, Accessed March 31, 2023.

[10] National Center for Biomedical Ontology. BioPortal. https://bioportal.bioontology.org, Accessed March 31, 2023.

[11] Garvita Bajaj, Rachit Agarwal, Pushpendra Singh, Nikolaos Georgantas, and Valérie Issarny. 4w1h in iot semantics. *IEEE Access*, 6:65488–65506, 2018.

[12] Amit Sheth. Internet of things to smart iot through semantic, cognitive, and perceptual computing. *IEEE Intelligent Systems*, 31(2):108–112, 2016.

[13] W3C Semantic Sensor Networks Working Group. Semantic sensor network ontology (ssn) version 2. https://www.w3.org/TR/vocab-ssn/, 2019. [Online; accessed March 24, 2023].

[14] Laura Daniele, Frank den Hartog, and Jasper Roes. Created in close interaction with the industry: the smart appliances reference (saref) ontology. In *International Workshop Formal Ontologies Meet Industries*, pages 100–112. Springer, 2015.

[15] Maria Bermudez-Edo, Tarek Elsaleh, Payam Barnaghi, and Kerry Taylor. Iot-lite: A lightweight semantic model for the internet of things. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 90–97, 2016.

[16] Michael Compton, Payam Barnaghi, Luis Bermudez, Raúl Garcia-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. The ssn ontology of the semantic sensor network. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 97–113. Springer, 2012.

[17] Krzysztof Janowicz, Armin Haller, Simon J.D. Cox, Danh Le Phuoc, and Maxime Lefrancois. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 56:1–10, 2019.

[18] National Institute of Standards and Technology. QUDT - quantities, units, dimensions and data types ontology. http://qudt.org/, 2010.

[19] Jerry R Hobbs and Feng Pan. Time ontology in owl. *W3C working draft*, 27:133, 2006.

[20] The Enterprise Knowledge Graph Platform | Stardog, 2022.

[21] Jie Jiang, Riccardo Pozza, Nigel Gilbert, and Klaus Moessner. Makesense: An iot testbed for social research of indoor activities. *ACM Trans. Internet Things*, 1(3), jun 2020.

[22] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34, 2014.

[23] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[24] Rachit Agarwal, David Gomez Fernandez, Tarek Elsaleh, Amelie Gyrard, Jorge Lanza, Luis Sanchez, Nikolaos Georgantas, and Valerie Issarny. Unified iot ontology to enable interoperability and federation of testbeds. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 70–75. IEEE, 2016.

[25] Fan Liang, Wei Yu, Dou An, Qingyu Yang, Xinwen Fu, and Wei Zhao. A Survey on Big Data Market: Pricing, Trading and Protection. *IEEE Access*, 6:15132–15154, 2018.

[26] Fabian Schomm, Florian Stahl, and Gottfried Vossen. Marketplaces for data: An initial survey. *SIGMOD Record*, 42(1):15–26, 2013.

[27] Ruiming Tang, Huayu Wu, Xiuqiang He, and Stephane Bressan. Valuating Queries for Data Trading in Modern Cities. *Proceedings - 15th IEEE International Conference on Data Mining Workshop, ICDMW 2015*, pages 414–421, 2016.

[28] Youssef Khazbak, Junpeng Qiu, Tianxiang Tan, and Guohong Cao. Targetfinder: A privacy preserving system for locating targets through iot cameras. *ACM Trans. Internet Things*, 1(3), jun 2020.

[29] Igor Bilogrevic and Martin Ortlieb. "If you put all the pieces together..." - Attitudes towards data combination and sharing across services and companies. *Conference on Human Factors in Computing Systems - Proceedings*, pages 5215–5227, 2016.

[30] Raul Castro Fernandez, Pranav Subramaniam, and Michael J. Franklin. Data market platforms: Trading data assets to solve data problems. *Proceedings of the VLDB Endowment*, 13(11):1933–1947, 2020.

[31] Allison Woodruff, Vasyl Pihur, Sunny Consolvo, Laura Brandimarte, and Alessandro Acquisti. Would a privacy fundamentalist sell their DNA for $1000... if nothing bad happened as a result? The Westin categories, behavioral intentions, and consequences. *SOUPS '14: Proceedings of the Tenth Symposium On Usable Privacy and Security*, pages 1–18, 2014.

[32] Ziyang Liu and Hakan Hacigümüş. Online optimization and fair costing for dynamic data sharing in a cloud data market. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1359–1370, 2014.

[33] Nathan Patrizi, Eirini Eleni Tsiropoulou, and Symeon Papavassiliou. Health data acquisition from wearable devices during a pandemic: A techno-economics approach. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[34] Xiaoqi Ren, Palma London, Juba Ziani, and Adam Wierman. Datum: Managing Data Purchasing and Data Placement in a Geo-Distributed Data Market. *IEEE/ACM Transactions on Networking*, 26(2):893–905, 2018.

[35] Kangsoo Jung, Junkyu Lee, Kunyoung Park, and Seog Park. Privata. *Schriftwechsel mit Metternich. Erster Teil: 1803–1819*, pages 309–326, 2019.

[36] Bokolo Anthony. Decentralized brokered enabled ecosystem for data marketplace in smart cities towards a data sharing economy. *Environment Systems and Decisions*, pages 1–19, 2023.

[37] Antonio J Jara, Martin Serrano, Andrea Gómez, David Fernández, Germán Molina, Yann Bocchi, and Ramon Alcarria. Smart cities semantics and data models. In *Proceedings of the International Conference on Information Technology & Systems (ICITS 2018)*, pages 77–85. Springer, 2018.

[38] Flavio Cirillo, David Gómez, Luis Diez, Ignacio Elicegui Maestro, Thomas Barrie Juel Gilbert, and Reza Akhavan. Smart city iot services creation through large-scale collaboration. *IEEE Internet of Things Journal*, 7(6):5267–5275, 2020.

[39] Roberto De Virgilio, Giorgio Orsi, Letizia Tanca, and Riccardo Torlone. Semantic data markets: A flexible environment for knowledge management. *International Conference on Information and Knowledge Management, Proceedings*, pages 1559–1564, 2011.

[40] Danh Le-Phuoc, Hoan Quoc Nguyen-Mau, Josiane Xavier Parreira, and Manfred Hauswirth. A middleware framework for scalable management of linked streams. *Journal of Web Semantics*, 16:42–51, 2012.

[41] Vanessa Lopez, Spyros Kotoulas, Marco Luca Sbodio, Martin Stephenson, Aris Gkoulalas-Divanis, and Pól Mac Aonghusa. QuerioCity: A linked data platform for urban information management. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7650 LNCS(PART 2):148–163, 2012.

[42] Jesus Arias Fisteus, Norberto Fernández García, Luis Sánchez Fernández, and Damaris Fuentes-Lorenzo. Ztreamy: A middleware for publishing semantic streams on the Web. *Journal of Web Semantics*, 25:16–23, 2014.

[43] Spyros Kotoulas, Vanessa Lopez, Raymond Lloyd, Marco Luca Sbodio, Freddy Lecue, Martin Stephenson, Elizabeth Daly, Veli Bicer, Aris Gkoulalas-Divanis, Giusy Di Lorenzo, Anika Schumann, and Pol Mac Aonghusa. SPUD - Semantic processing of urban data. *Journal of Web Semantics*, 24:11–17, 2014.

[44] Freddy Lécué, Robert Tucker, Simone Tallevi-Diotallevi, Rahul Nair, Yiannis Gkoufas, Giuseppe Liguori, Mauro Borioni, Alexandre Rademaker, and Luciano Barbosa. Semantic traffic diagnosis with STAR-CITY: Architecture and lessons learned from deployment in Dublin, Bologna, Miami and Rio. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8797:292–307, 2014.

[45] Danh Le-Phuoc, Hoan Nguyen Mau Quoc, Hung Ngo Quoc, Tuan Tran Nhat, and Manfred Hauswirth. The Graph of Things: A step towards the Live Knowledge Graph of connected things. *Journal of Web Semantics*, 37-38(2016):25–35, 2016.

[46] Muhammad Intizar Ali, Naomi Ono, Mahedi Kaysar, Zia Ush Shamszaman, Thu Le Pham, Feng Gao, Keith Griffin, and Alessandra Mileo. Real-time data analytics and event detection for IoT-enabled communication systems. *Journal of Web Semantics*, 42:19–37, 2017.

[47] Flavio Cirillo, Gürkan Solmaz, Everton Luís Berz, Martin Bauer, Bin Cheng, and Ernö Kovacs. A standard-based open source iot platform: Fiware. *IEEE Internet of Things Magazine*, 2(3):12–18, 2019.

[48] Ivan Lujic, Vincenzo De Maio, Srikumar Venugopal, and Ivona Brandic. Sea-leap: Self-adaptive and locality-aware edge analytics placement. *IEEE Transactions on Services Computing*, 15(2):602–613, 2022.

[49] Sergio Gustavo Guillén, Pilar Sala, Giuseppe Fico, María Teresa Arredondo, Alicia Cano, Jorge Posada, German Gutiérrez, Carlos Palau, Konstantinos Votis, Cor N Verdouw, et al. Iot european large-scale pilots–integration, experimentation and testing. *Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution*, 2017.

[50] Subashini Raghavan, Boung Yew Lau Simon, Ying Loong Lee, Wei Lun Tan, and Keh Kim Kee. Data integration for smart cities: opportunities and challenges. *Computational Science and Technology: 6th ICCST 2019, Kota Kinabalu, Malaysia, 29-30 August 2019*, pages 393–403, 2020.

[51] Sergey Kozhevnikov, Miroslav Svitek, and Shuo-Yan Chou. Smart services ecosystem concept for cooperative its. In *Proceedings of the 11th Euro American Conference on Telematics and Information Systems*, pages 1–8, 2022.
[52] Pascal Hitzler. A Review of the Semantic Web Field. *Commun. ACM*, 64(2):76–83, jan 2021.
[53] E G Stephan, T O Elsethagen, L K Berg, M C Macduff, P R Paulson, W J Shaw, C Sivaraman, W P Smith, and A Wynne. Semantic catalog of things, services, and data to support a wind data management facility. *Information systems frontiers*, 18(4):679–691, 2016.
[54] Open Government Data and the Linked Data Web: Lessons from data. gov. uk. *IEEE Intelligent Systems*, 27(3):16–24, 2012.
[55] S. M. Hasan Mahmud, Md. Altab Hossin, Md. Rezwan Hasan, Hosney Jahan, Sheak Rashed Haider Noori, and Md. Razu Ahmed. Publishing csv data as linked data on the web. In Pradeep Kumar Singh, Bijaya Ketan Panigrahi, Nagender Kumar Suryadevara, Sudhir Kumar Sharma, and Amit Prakash Singh, editors, *Proceedings of ICETIT 2019*, pages 805–817, Cham, 2020. Springer International Publishing.
[56] Fabian Kirstein, Kyriakos Stefanidis, Benjamin Dittwald, Simon Dutkowski, Sebastian Urbanek, and Manfred Hauswirth. *Piveau: A Large-Scale Open Data Management Platform Based on Semantic Web Technologies*, volume 12123 LNCS. Springer International Publishing, 2020.
[57] Pengfei Wu, Fengjuan Ma, and Shengquan Yu. Using a linked data-based knowledge navigation system to improve teaching effectiveness, 2021.
[58] Faiza Bashir and Nosheen Fatima Warraich. Systematic literature review of Semantic Web for distance learning, 2020.
[59] An integrated Linked Building Data system: AEC industry case. *Advances in engineering software (1992)*, 152:102930–, 2021.
[60] Asier Moreno, Asier Perallos, Diego López-De-Ipiña, Enrique Onieva, Itziar Salaberria, and Antonio D Masegosa. A novel software architecture for the provision of context-aware semantic transport information. *Sensors (Basel, Switzerland)*, 15(6):12299–12322, 2015.
[61] Integration of mobile gis and linked data technology for spatio-temporal transport data model. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(4):721–724, 2019.
[62] Mauro Dragoni, Marco Rospocher, Tania Bailoni, Rosa Maimone, and Claudio Eccher. *Semantic technologies for healthy lifestyle monitoring*, volume 11137 LNCS. Springer International Publishing, 2018.
[63] Alessandro Fogli and Giuseppe Sansonetti. Exploiting semantics for context-aware itinerary recommendation. *Personal and Ubiquitous Computing*, 23(2):215–231, 2019.
[64] Cheng Xie, Hongming Cai, Lida Xu, Lihong Jiang, and Fenglin Bu. Resource Service Toward Cloud Manufacturing. 13(6):3338–3349, 2017.
[65] Niklas Petersen, Lavdim Halilaj, Irlán Grangel-González, Steffen Lohmann, Christoph Lange, and Sören Auer. Realizing an RDF-based information model for, a manufacturing company – A case study. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10588 LNCS:350–366, 2017.
[66] Martin Serrano, Amelie Gyrard, Michael Boniface, Paul Grace, Nikolaos Georgantas, Rachit Agarwal, Payam Barnaghi, Francois Carrez, Bruno Almeida, Tiago Teixeira, et al. Cross-domain interoperability using federated interoperable semantic iot/cloud testbeds and applications: The fiesta-iot approach, 2017.
[67] IOT Garage / Quarriable Smart City Data Market - User Interface · GitLab, 2022.

A APPENDIX

B ONTOLOGY REQUIREMENTS SPECIFICATION DOCUMENT(ORSD)

B.1 Purpose

The Urban Data Exchange Ontology (UDEO) aims to describe sensor data in IoT marketplaces.

B.2 Scope

Internet of Things (IoT).

B.3 Implementation Language

The Web Ontology Language (OWL2).

B.4 Intended End-Users

- Small to Medium Enterprise (SME).

- Data Scientists.
- Computer Scientists.

B.5 Intended Uses

- To build a linked data that offers data on-demand (i.e., granular data retrieval from disparate sources).
- For reasoning about the data of interest.
- Build Artificial Intelligence (AI) models.

B.6 Ontology Requirements

B.6.1 Non-Functional Requirements.

- UDEO must include IoT concepts such as sensor.
- UDEO must include relationships between IoT concepts and features of interest in space and time.

B.6.2 Functional Requirements.

- Nineteen Competency Questions (CQs) formulated as SPARQL queries.

CQ1: *What is the temperature of the room on a given date and time?*

```
SELECT (AVG(?result) as ?averageHeartRate)
 {GRAPH <http://smarthome> {
        ?s rdf:type sosa:Observation;
        sosa:hasFeatureOfInterest ?fi;
        sosa:madeBySensor :BpmCore;
        sosa:observedProperty :heartRate;
        sosa:resultTime ?dt;
        sosa:hasResult [
             rdf:type qudt-1-1:QuantityValue ;
             qudt-1-1:numericValue ?result ;
             qudt-1-1:unit qudt-unit-1-1:HeartBeatsPerMinute ].
    ?fi bot:hasStorey :Level1.
    :Level1 bot:hasSpace :Kitchen, :LivingRoom.
    {
        SELECT ?building (COUNT(?storey) as ?storeyCount)
    {
            ?building a bot:Building ;
                      bot:hasStorey ?storey.}
                      GROUPBY ?building}
FILTER(?fi = ?building && ?storeyCount = 3)
BIND(SUBSTR(str(?dt), 0, 11) as ?date)}}
```

Listing 11. SPARQL Query for Scenario 1

CQ 2: *What are the air quality data in certain addresses at given date and time?*

```
INSERT {?id a   fiware:AirQualityObserved;
                fiware:AirQualityIndex ?AirQualityIndex;
                fiware:Precipitation ?Precipitation;
                fiware:Reliability  ?Reliability;
                fiware:WindDirection ?WindDirection;
```

```
                ngsi-ld:Co  ?Co;        ngsi-ld:No ?No;
                ngsi-ld:Nox ?Nox;       ngsi-ld:No2 ?No2;
                ngsi-ld:So2 ?So2.}
Where {
        SELECT *
          {?id a    fiware:AirQualityObserved;
                fiware:airQualityIndex ?airQualityIndex;
                fiware:precipitation ?precipitation;
                fiware:reliability  ?reliability;
                fiware:windDirection ?windDirection;
                fiware:airQualityLevel ?airQualityLevel;
                fiware:dateObserved  ?date;
                ngsi-ld:CO  ?CO;       ngsi-ld:CO_Level ?CO_Level;
                ngsi-ld:NO ?NO;        ngsi-ld:NOx ?NOx;
                ngsi-ld:NO2 ?NO2;      ngsi-ld:SO2 ?SO2.

BIND(STRAFTER(str(?airQualityIndex), ":") as ?Index)
BIND(STRBEFORE(str(?Index), ",") as ?AirQualityIndex)
BIND(STRAFTER(str(?precipitation), ":") as ?prec)
BIND(STRBEFORE(str(?prec), ",") as ?Precipitation)
BIND(STRAFTER(str(?reliability), ":") as ?rel)   BIND(STRBEFORE(str(?rel), ",") as ?Reliability)
BIND(STRAFTER(str(?windDirection), ":") as ?win) BIND(STRBEFORE(str(?win), ",") as ?WindDirection)
BIND(STRAFTER(str(?CO), ":") as ?carbon)         BIND(STRBEFORE(str(?carbon), ",") as ?Co)
BIND(STRAFTER(str(?NO), ":") as ?nitrogen)       BIND(STRBEFORE(str(?nitrogen), ",") as ?No)
BIND(STRAFTER(str(?NOx), ":") as ?nitOx)         BIND(STRBEFORE(str(?nitOx), ",") as ?Nox)
BIND(STRAFTER(str(?NO2), ":") as ?nitdi)         BIND(STRBEFORE(str(?nitdi), ",") as ?No2)
BIND(STRAFTER(str(?SO2), ":") as ?sulpher)       BIND(STRBEFORE(str(?sulpher), ",") as ?So2)}}
************************************************
SELECT *
        {?id a  fiware:AirQualityObserved;
            schema:address  ?address;
            fiware:dateObserved ?date;
            fiware:Precipitation ?Precipitation;
            fiware:Reliability  ?Reliability;
            fiware:WindDirection ?WindDirection;
            fiware:AirQualityIndex ?AirQualityIndex;
            ngsi-ld:Co  ?Co;          ngsi-ld:CO_Level ?CO_Level;
            ngsi-ld:No ?No;           ngsi-ld:Nox ?Nox;
            ngsi-ld:No2 ?No2;         ngsi-ld:So2 ?So2.
     ?id2 a  fiware:NoiseLevelObserved;
            fiware:DateObservedFrom ?DateObservedFrom;
            fiware:DateObservedTo ?DateObservedTo;
            fiware:frequencies ?frequencies;
            fiware:DataProvider ?DataProvider;
            ngsi:location  ?location;
            ngsi-ld:lAeq_d ?lAeq_d;
            ngsi-ld:lAmax ?lAmax.

FILTER(xsd:float(?lAmax) > 0.72)   FILTER(REGEX(?address, "A"))
FILTER(REGEX(?CO_Level, "Low"))    FILTER(xsd:integer(?AirQualityIndex) < 100)
FILTER(xsd:integer(?Nox) < 100)}   Limit 10
```

Listing 12. SPARQL Query for Scenario 2

CQ 3: *Help me plan a day out!*

```
Select DISTINCT ?Type ?Facilities  ?Days
                ?ParkingName ?Category ?Temperature ?Distance ?PlacesNearby
{GRAPH <urn:ngsi-ld:Museum:121513>
{?id    ns1:museumType ?type.
 ?type          ngsi-ld:hasValue   ?Type.
 ?id            ns1:facilities     ?facilities.
 ?facilities    ngsi-ld:hasValue   ?Facilities.
 ?openingHours  ngsi-ld:hasValue   ?OpeningHours.
 ?OpeningHours  :dayOfWeek         ?Days;
                :opens             ?Opens;
                :closes            ?closes.}
GRAPH <urn:ngsi-ld:ParkingSpot:123456>
{ ?id2          ngsi-ld:name       ?name.
  ?name         ngsi-ld:hasValue   ?ParkingName.
  ?id3          ns1:category       ?catergory.
  ?catergory    ngsi-ld:hasValue   ?Category.}

sosa:Temperature sosa:hasResult     ?Temperature.
{ ?MuseumPoint   ns4:long          ?long;
                 ns4:lat           ?lat.
?PlacesNearby    geof:nearby (?MuseumPoint 10 unit:Kilometer);
                 ns4:long ?p_long;
                 ns4:lat ?p_lat;
BIND (geof:distance(sosa:MuseumPoint, ?PlacesNearby, unit:Kilometer) as ?Distance)}}
```

Listing 13. SPARQL Query for Scenario 3

CQ 4: *Where can I park and ride near a certain GPS location?*

```
SELECT *
    {?id a  fiware:ParkingSpot;
            fiware:category ?category;
            fiware:dataProvider ?dataProvider;
            ngsi:status ?status;
            ngsi:location ?location;
            ngsi:ParkingPoint ?ParkingPoint.
    ?id2 a  fiware:BikeHireDockingStation;
            fiware:availableBikeNumber ?availableBikeNumber;
            fiware:AvailableBikeNumber ?AvialableBikeNumber;
            fiware:SunnyDays ?SunnyDays;
            schema:address ?address;
            ngsi:status ?Bikestatus.
sosa:PointID a              pos:Point;
             pos:SOSAPoint  ?SOSAPoint.
BIND (geof:distance(?ParkingPoint, ?SOSAPoint, unit:Kilometer) as ?Distance)
FILTER(xsd:integer(?Distance < 500))
```

```
FILTER(REGEX(?Bikestatus, "free"))
FILTER(REGEX(?availableBikeNumber, "1"))
FILTER(REGEX(?category, "offstreet")).}
```

Listing 14. SPARQL Query for Scenario 4

CQ 5: *WeekDays Rule*

```
prefix rule: <tag:stardog:api:rule:>
[] a rule:SPARQLRule ;
    rule:content
    IF {
        ?id a fiware:AirQualityObserved;
            fiware:AirQualityIndex ?AirQualityIndex;
        BIND (?AirQualityIndex > 100  AS ?LowAirQuality)}
    THEN {
        Weekdays fiware:LowAirQuality ?LowAirQuality}.
```

Listing 15. SPARQL Query for Scenario 5

CQ 6: *we cycle everywhere! if the local beach is busy this weekend, we will head to the museum?*

```
SELECT *
        { ?id a fiware:Beach;
            ngsi:name ?Name;
            fiware:facilities ?Facilities;
            fiware:occupationRate ?OccupationRate;
            fiware:Weekends   ?Weekends.

{SERVICE <db://museum100k>
        {?id2 a fiware:Museum;
            fiware:openingHoursSpecification  ?OpeningHours.

{SERVICE <db://BikeHireDockingStation100k>
        {?id3 a fiware:BikeHireDockingStation;
             fiware:availableBikeNumber ?availableBikeNumber;
             schema:address             ?address;
             ngsi:status                ?Bikestatus.}}}}}
```

Listing 16. SPARQL Query for Scenario 6

CQ 7: *Where are the locations of available bikes?*

```
PREFIX ns1: <https://uri.fiware.org/ns/data-models#>
PREFIX ns2: <https://uri.etsi.org/ngsi-ld/>
PREFIX ns3: <https://schema.org/>
```

```
SELECT *
       {  ?BikeAvailable   ns1:freeSlotNumber ?availabenumber;
                           ns1:freeSlotNumber ?value.
          ?s               ns2:location       ?loc}
```

Listing 17. SPARQL Query for Scenario 7

CQ 8: *Where are the geographical information for available bikes near me?*

```
SELECT ?Result
        {udeo:availableBikeNumber sosa:hasResult ?AvailabeBikeNumber.
        udeo:freeSlotNumber sosa:hasResult ?FreeSlotNumber.
        ?location a ngsi-ld:GeoProperty.
        ?address  :streetAddress   ?StreetAddress;
                  :addressLocality ?AddressLocality;
                  :adressCountry  ?Country .
       ?coordinate_node ns4:lat ?Lat;
                   ns4:long      ?Long.
BIND(CONCAT(STR(?StreetAddress),",", STR(?AddressLocality),",",STR(?Country)) AS ?Address).
BIND(CONCAT("POINT(",STR(?Long),"", STR(?Lat),")") AS ?Coordinates).
BIND(CONCAT("Available_Bike_No.","", STR(?AvailabeBikeNumber),"",
"Slot:", STR(?FreeSlotNumber),"Located_at","","Address:",
STR(?Address),",",  "coordinates :", STR(?Coordinates)) AS ?Result)}
```

Listing 18. SPARQL Query for Scenario 8

CQ 9: *What is Barry Island beach profile and how can I get there?*

```
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX ns4: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX ngsi-ld: <https://uri.etsi.org/ngsi-ld/>
PREFIX udeo: <http://www.w3id.org/udeo/>

SELECT ?BeachInfo  ?Result

   {udeo:availableBikeNumber  sosa:hasResult ?AvailabeBikeNumber.
    udeo:freeSlotNumber       sosa:hasResult  ?FreeSlotNumber.
    ?location a               ngsi-ld:GeoProperty.
    ?address   :streetAddress   ?StreetAddress;
               :addressLocality ?AddressLocality;
               :adressCountry   ?Country.
    ?coordinate_node   pos:lat  ?Lat;
                       pos:long ?Long.
    udeo:accesssType (sosa:hasResult|ngsi-ld:hasValue)     ?accessType.
    udeo:beachType   (sosa:hasResult|ngsi-ld:hasValue)     ?beachType.
    udeo:facilities  (sosa:hasResult|ngsi-ld:hasValue)     ?facilities.
    udeo:occupationRate  (sosa:hasResult|ngsi-ld:hasValue) ?occupationRate .
    udeo:name        (sosa:hasResult|ngsi-ld:hasValue)     ?name.
    udeo:width        sosa:hasResult                       ?w.
    udeo:length       sosa:hasResult                       ?l.
```

```
    BIND(CONCAT(STR(?beachType),",", STR(?accessType)
    ,",", STR(?facilities),",", STR(?occupationRate)) AS ?BeachInfo).
    BIND(?w * ?l AS ?area).
    FILTER (?area >1000).
    FILTER REGEX (?name ,"(Barry)").
    FILTER REGEX (?BeachInfo, "(lifeGuard)").
    BIND(CONCAT(STR(?StreetAddress),",", STR(?AddressLocality)
    ,"," , STR(?Country)) AS ?Address).
    BIND(CONCAT(STR(?Long),":", STR(?Lat)) AS ?Coordinates).
    BIND(CONCAT("Available_Bike_No.","", STR(?AvailabeBikeNumber),"",
    "Slot:", STR(?FreeSlotNumber),"Locatedat","","Address:", STR(?Address)
    ,",",  "atacoordinates :", STR(?Coordinates)) AS ?Result )}
```

Listing 19. SPARQL Query for Scenario 9

CQ 10: *What is the hunmanby Gap beach profile? (e.g., type, accessibility, facilities, occupation rate and area)*

```
Prefix ngsi-ld: <https://uri.etsi.org/ngsi-ld/>
Prefix def: <https://smart-data-models.github.io/data-models/terms.jsonld#/definitions/>

SELECT ?name ?BeachInfo ?area {
        ?Access  def:accessType  ?AccessType.
        ?AccessType ngsi-ld:hasValue  ?accessType.
        ?Beach  def:beachType  ?BeachType.
        ?BeachType ngsi-ld:hasValue  ?beachType .
        ?BeachFacilities  def:facilities ?Facilities.
        ?Facilities ngsi-ld:hasValue ?facilities.
        ?Occupation  def:occupationRate ?OccupationRate.
        ?OccupationRate ngsi-ld:hasValue ?occupationRate.
        ?BeachName  ngsi-ld:name ?Name.
        ?Name   ngsi-ld:hasValue ?name.
        ?BeachWidth  def:width ?width.
        ?width ngsi-ld:hasValue ?w .
        ?BeachLength def:length ?length.
        ?length ngsi-ld:hasValue ?l .
BIND(CONCAT(STR(?beachType), ",", STR(?accessType),
", ", STR(?facilities), ",", STR(?occupationRate)) AS ?BeachInfo).
BIND(?w * ?l  AS ?area) .
FILTER REGEX (?name , "(a)").}
```

Listing 20. SPARQL Query for Scenario 10

CQ 11: *Provide me with a parking name, status, location and the service provider from this knowledge base.*

```
Prefix ngsi-ld: <https://uri.etsi.org/ngsi-ld/>
Prefix ns1: <https://uri.fiware.org/ns/data-models#>

SELECT ?name ?ParkingInfo ?coord{
    ?ParkingName ngsi-ld:name  "A-01"^^xsd:string.
    ?Name ngsi-ld:hasValue  ?name.
```

```
    ?ParkingStatus  ngsi-ld:status  ?Status.
    ?Status ngsi-ld:hasValue  ?status .
    ?ParkingLocation  ngsi-ld:location ?Location.
    ?Location ngsi-ld:hasValue ?location.
    ?location ngsi-ld:coordinates ?coord.
    ?ParkingProvider  ns1:dataProvider  ?Provider.
    ?Provider ngsi-ld:hasValue ?dataProvider.
    ?ParkingRef ns1:refParkingSite ?RefParkingSite.
    ?RefParkingSite   ?p ?refParkingSite .
BIND(CONCAT(STR(?status), ",", STR(?coord),
", ", STR(?dataProvider), ",", STR(?refParkingSite)) AS ?ParkingInfo).}
```

Listing 21. SPARQL Query for Scenario 11

CQ 12: *What are the air quality levels at Digital Cataplut?*

```
Prefix : <https://uri.etsi.org/ngsi-ld/default-context/>
Prefix ngsi-ld: <https://uri.etsi.org/ngsi-ld/>
Prefix ns1: <https://uri.fiware.org/ns/data-models#>
Prefix ns2: <https://schema.org/>

SELECT *
{?id a  ns1:AirQualityObserved ;
        ns2:address [ a ngsi-ld:Property ;
        ngsi-ld:hasValue [ :addressLocality ?City ;
        :adressCountry ?Country ;
        :streetAddress "Digital Catapult" ] ] ;
        :CO [ a ngsi-ld:Property ;
        ngsi-ld:hasValue ?CO ;
        ngsi-ld:unitCode ?COCode ] ;
        :CO_Level [ a ngsi-ld:Property ;
        ngsi-ld:hasValue ?CO_Level ] ;
        :NO [ a ngsi-ld:Property ;
        ngsi-ld:hasValue ?NO ;
        ngsi-ld:unitCode ?NOCode ] ;
        :NO2 [ a ngsi-ld:Property ;
        ngsi-ld:hasValue ?NO2 ;
        ngsi-ld:unitCode ?NO2Code ] ;
        :NOx [ a ngsi-ld:Property ;
        ngsi-ld:hasValue ?NOx ;
        ngsi-ld:unitCode ?NOxCode ] ;
        :SO2 [ a ngsi-ld:Property ;
        ngsi-ld:hasValue ?SO2 ;
        ngsi-ld:unitCode ?SO2Code ]

BIND(CONCAT( " ", STR(?City), " ", STR(?Country)) AS ?Address)}
```

Listing 22. SPARQL Query for Scenario 12

CQ 13: *When does V&A South Kensington open and close on Sunday?*

```
PREFIX : <https://uri.etsi.org/ngsi-ld/default-context/>
PREFIX ngsi-ld: <https://uri.etsi.org/ngsi-ld/>
PREFIX ns1: <https://uri.fiware.org/ns/data-models#>

SELECT  ?Opens ?Closes {
        ?Museum ngsi-ld:name ?name.
        ?name ngsi-ld:hasValue "V&A South Kensington".
        ?Museum ns1:openingHoursSpecification ?value.
        ?value ?p ?OpeningHours.
        ?OpeningHours :dayOfWeek "Sun";
        :opens ?Opens;
        :closes ?Closes.}
```

Listing 23. SPARQL Query for Scenario 13

CQ 14: *What is the distance between the museum and nearby parking spot/docking station?*

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX ns4: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX ngsi-ld: <https://uri.etsi.org/ngsi-ld/>
PREFIX udeo: <http://www.w3id.org/udeo/>
 SELECT DISTINCT * {
 udeo:MuseumPoint ns4:long     ?long;
                  ns4:lat      ?lat .
 ?PlacesNearby
        geof:nearby (udeo:MuseumPoint 10 unit:Kilometer);
        ns4:long ?p_long;
        ns4:lat ?p_lat;
BIND (geof:distance( udeo:MuseumPoint, ?PlacesNearby, unit:Kilometer) as ?Distance). }}
```

Listing 24. SPARQL Query for Scenario 14

CQ 15: *How many bikes available within 250 km from my location (51.52586 -0.123331)?*

```
PREFIX: ngsi-ld: <https://uri.etsi.org/ngsi-ld/>
PREFIX: ns2: <https://schema.org/>
PREFIX: ns4: <http://www.w3.org/2003/01/geo/wgs84_pos#>

SELECT ?AvailabeBikes   ?Distance{

?BikeAvailable      ns2:availableBikeNumber ?availabenumber.
?availabenumber     ngsi-ld:hasValue        ?AvailabeBikes.

?s      ns4:long           ?long ;
        ns4:lat            ?lat ;
geof:nearby (51.52586 -0.123331 250 unit:Kilometer ) .

? s1    geof:nearby (?s 250 unit:Kilometer);
```

```
        ns4:lat ?poi_lat;
        ns4:long ?poi_long.
BIND (geof:distance(?s, ?s1, unit:Kilometer) as ?Distance). }
```

Listing 25. SPARQL Query for Scenario 15

CQ 16: *Where has the air quality been observed?*

```
PREFIX : <https://uri.etsi.org/ngsi-ld/default-context/>
PREFIX ngsi-ld: <https://uri.etsi.org/ngsi-ld/>
PREFIX ns1: <https://uri.fiware.org/ns/data-models#>
PREFIX ns2: <https://schema.org/>

SELECT ?Address
{?id a ns1:AirQualityObserved ;
            ns2:address [ a ngsi-ld:Property ;
            ngsi-ld:hasValue [ :addressLocality ?City ;
                            :adressCountry ?Country ;
                            :streetAddress ?Street ] ] ;
BIND(CONCAT(STR(?Street), " ", STR(?City), " ", STR(?Country)) AS ?Address)}
```

Listing 26. SPARQL Query for Scenario 16