Undefined 1 (2009) 1–5
IOS Press

Real-time Generation of Linked Sensor Data and Multidimensional Data Cubes for Smart Environments

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Muntazir Mehdi [a,b], Ratnesh Sahay [a] Wassim Derguech [a] Weiping Qu [b] Stefan Deßloch [b] and Edward Curry [a]

[a] *INSIGHT Centre for Data Analytics, National University of Ireland, Galway Ireland*
E-mail: {firstname.lastname}@insight-centre.org
[b] *Department of Computer Science, Technical University Kaiserslautern Germany*
E-mail: {lastname}@informatik.uni-kl.de

Abstract. Events represent a record of an activity in the system, are logged as soon as they happen, and are chronologically independent. In most Smart Environment settings, events usually refer to the sensor data. The dynamicity of sensor data sources and publishing real-time sensor data over a generalised infrastructure like the Web, pose a new set of integration challenges. Semantic Sensor Networks demand excessive expressivity for efficient formal analysis of sensor data. Topics like data-warehousing, event processing, and decision making are very well established in research and industry. However, with the frequently changing nature of data models, the challenge is to deal with data model specific processing. With inception of ideas like Internet of Things (IoT) and Web of Things (WoT), research today deems it important for efforts to be placed in processing sensor or event data. In this work, we present a methodology to deal with sensor data using Semantic Web technologies, in real-time, and for smart environments is proposed. The proposed methodology addresses two different problems: 1) Collection of sensor data, transformation, meta-data enrichment of sensed data and publishing to Linked Open Data (LOD) Cloud, and 2) Adopting data model specific or context-specific properties in automatic generation of multidimensional data cubes and publishing to LOD Cloud. This article also details an extensive evaluation and analysis of the results obtained during the study. The results are compared with state-of-the-art W3C recommended RDF Data Cube Vocabulary.

Keywords: Multi-dimensional Data Cubes, Web of Things (WoT), Linked Sensor Data, Linked Open Data (LOD), Semantic Sensor Networks (SSN)

1. Introduction

Event Processing is a method that works on combining event-data originating from different sources to identify or infer patterns that are meaningful, critical, and require an immediate response [29, 30]. An event represents a record of an activity in the system or a result of a particular function or business process, it is logged as soon as it happens, does not necessarily have a fixed data type, and is chronologically ordered [30]. It is often hard to keep track of multiple events generating from different enterprise-wide applications as these events can be correlated or captured independently. In order to address this problem, complex event processing has taken its place in research and

industry [44]. We observe a huge emergence of complex events based data within smart environments like Smart Buildings, Smart Enterprises and Smart Cities. The backbone of these smart environments is sensors that produce huge amounts of data.

Smart environments are thus complex systems demanding a solution to manage and visualize data of their operations. Additionally, complex systems need to support real-time and effective decision-making [13]. The recent initiative of "Web of Things (WoT)" is proposed to use Web standards for publishing and consuming data from the embedded devices (e.g., sensors) built into everyday smart devices. The application of traditional Online Analytical Processing (OLAP) data cubes (like budgeting, forecasting, prediction, reporting) over the WoT is a new line of research that could bring sensor data together from disparate sources and put its related information into a format that is conducive to analysis and decision making. Specifically, on a very basic level, a detailed summary report of energy consumption (in an energy management context), generated using OLAP tools over data cubes would be sufficient enough to predict or allocate the future energy usage in a building or apartment or a house.

In this article, a solution that: 1) collects, transforms and publishes sensor data (event-data) as linked data and, 2) generates multidimensional data cubes by consuming linked sensor data is proposed. Both, the linked sensor data and data cubes, once generated, are then stored in a database (Triple Store), and are published to the Linked Open Data (LOD) Cloud; therefore, supporting decision making and event processing on huge amounts of event-data and multidimensional event-data. The approach is validated within a real world smart building. Event-data generated from sensors, deployed in the building to record parameters like (light, temperature, heat, power, etc.) are used. The event-data is then enriched with any necessary and relevant meta-data (contextual information) using the W3C Semantic Sensor Network Ontology [21]. The necessary meta-data for enriching the events is stored in a triple store and is retrieved via a SPARQL [36] end-point. An ontology, based on general and basic concepts of W3C RDF Data Cube Vocabulary (RDF-DCV) [16] for generating multidimensional data cubes is proposed and used. Therefore, the main motivation behind the work presented in this article is to generate linked sensor data and multidimensional data cubes on-the-fly for WoT or Semantic Sensor Networks (SSN) to support decision making in smart environments.

The rest of the article is structured as follows: Section 2 gives a brief insight into the necessary background concepts and the foundations. Section 3 highlights the first part of our contribution describing how event-data is collected, transformed into linked data, enriched with relevant meta-data, how the data is broadcasted for data cube generation, and published to LOD Cloud. Sections 4 and 5 highlight the second part of the contribution by detailing the process of event registration and multidimensional data cubes generation. Section 6 presents a detailed evaluation of the proposed methodology along with result analysis. We conclude the presented work, compare our contribution to other related works, and give a brief insight into the prospective work in Section 7. But first, we introduce our motivating scenario in this Section.

1.1. Motivation

Today enterprises are producing more and more data, making its exchange, management, and decision making complex. Smart Environments rely on sensor data to provide necessary business intelligence to support decision making. A possible use-case in this scenario can be a smart building within an energy management application to control supply and demand of energy. In this context, an approach that supports energy related decision making on a real world use-case realized in the Digital Enterprise Research Institute (DERI) is proposed.

The DERI building has been retrofitted with energy sensors to monitor the energy consumption within the building. In total there are over 50 fixed energy consumption sensors covering office space, café, data centre, kitchens, conference and meeting rooms, computing museum along with over 20 mobile sensors for devices, light and heaters energy consumption as well as light, temperature and motion detection sensors. A building-specific aspect of the dataspace has been presented in [15] with a sensor network-based situation awareness scenario presented in [21].

An Event processing technique can be applied to process this real time sensor information to support energy management applications [14]. However, it has been observed that when it comes to event processing, there is a limited support for decision making, data mining and knowledge discovery [28]. Indeed, the identification of critical and meaningful patterns highly depends on the amount of data available. A collection of a huge amount of data that has been gener-

ated by these sensors would increase accuracy of results.

In order to build efficient energy management systems, which is solely based on semantic web technologies, in this context, two main challenges need to be handled: *heterogeneous* data management and data *cubes* in *Web of Things* settlement, specifically, in Semantic Sensor Networks and Smart Environments.

1.1.1. Heterogeneous Data Management

Sharing information and data is a big challenge in smart environments. Due to dynamic and heterogeneous nature of data being generated from different applications in different domains across different enterprises, there is a need to transform data in a format that is easily exchangeable and integrateable.

One possible use-case for this can be a smart city environment that relies on an application using combined data from different energy management applications running in different smart buildings. Therefore, an approach on converting application specific data, sensor data (event-data) into RDF and, publishing it into Linked Open Data (LOD) Cloud would support applications relying on combined data. Alternatively, in a case, where the data providers are cautious and instead of publishing their data as open data, they might prefer to keep data closed. In this case, the closed data, from different data providers can be easily combined using linked data techniques. Such a scenario is depicted in Figure 1, where linked building data clouds from i) *National University of Ireland, Galway (NUIG), Building-1*, ii) *NUIG, Building-2*, iii) *DERI Building* and iv) *Lehrgebiet Informationssysteme (LGIS) Building* are linked together to form a unified data cloud.

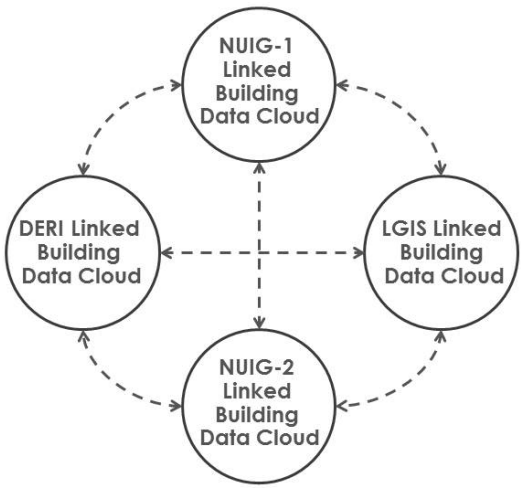


Fig. 1. Linking Building Data Clouds

Similarly, another possible use-case motivating the work presented in this article is adoption of large Internet of Things (IoT) and Smart City projects. SmartSantander smart city project have deployed tens of thousands of Internet-connected sensor devices in large cities among Europe [37]. The sensing capabilities of these devices are wide ranging, including solar radiation, wind speed and direction, temperature, water flow, noise, traffic, public transport, rainfall, parking, and others. A linked dataset, containing historical sensor data (event-data) and multidimensional data cubes, created using data generated from these sensors can sustain the idea of Smart City.

The approach presented in this article not only transforms sensor data into RDF, but it also generates contextual multidimensional data cubes and publishes both, linked sensor data and data cubes to the LOD Cloud.

Linked data is a set of best practices for representing information in RDF format and relating or connecting this information. The basic ingredient of linked data is structured data and links between structured data. The main philosophy of linked data is to create data that can be shared and reused. Linked data leverages Web standards and Web protocols to enable sharing of structured data thus supporting the creation of a global information space [24]. Linked data has been and is still facilitating publishing structured data on Web in large volumes thus creating a huge Linked Open Data (LOD) cloud. More details about linked data and LOD Cloud is given in Section 2.

1.1.2. Data Cubes & Web of Things

Since event processing involves providing an immediate response over a huge amount of data, a system that provides a fast response to complex data retrieval queries is needed. To overcome this obstacle, a system that uses a data warehouse for storing historical data (sensor data) and managing multidimensional data cubes is proposed in this article.

Contrary to a general database management system, data-warehouses are large stores of information containing historical data; additionally, they provide a multidimensional view of the data they contain. The main purpose and benefits behind keeping historical data is to support decision making, knowledge discovery, identification of hidden patterns and data mining. In the work reported here, we exploit the usage of historical data and data cubes in a WoT settlement and

present an approach for cubes generation to assist the aforementioned benefits.

The multidimensional shape of data inside a data-warehouse is also referred to as cubes. A cube structures information into dimensions and facts. A dimension characterizes and represents a context for the analysis of facts (e.g. location, type, time) and a fact contains interesting and relevant measures (e.g. power usage, electricity usage) [16].

The vision adopted in this work is similar to the one proposed by the Web of Things (WoT). The main requirement to realize WoT is to extend the existing Web such that real-world objects like electronic appliances, digitally enhanced everyday objects, sensors, and embedded devices can easily be blended in it. In the use-case reported in this article, the approach is limited to sensor data deployed within the DERI building. However, this approach can be easily extended to consider other smart devices and smart environments.

The power of WoT comes from light-weight HTTP servers embedded within devices to enable the use of HTTP as an application level protocol rather than a transport level protocol [19]. In the scenario reported in this work, CoAP (Constrained Application Protocol), which is built on top of HTTP is used. CoAP is a Web transfer protocol that provides a request/response model for interaction between endpoints [40]. Both WoT and CoAP are further detailed in Section 2.

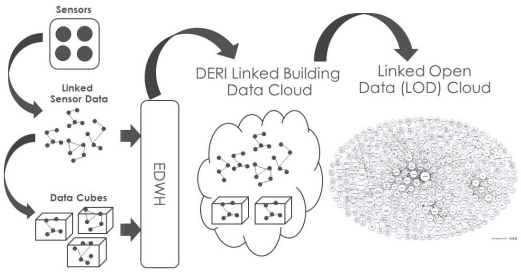


Fig. 2. Use-case Scenario

The overall use-case scenario presented and carried out in this study is shown in Figure 2, where it is clearly visible that the first part of the contributive work presented in this article is to generate linked sensor data from raw sensor data, and the second part involves creating multidimensional data cubes on the generated linked sensor data. Both linked sensor data and multidimensional data cubes are generated on-the-fly (real-time) and are stored in an Event Datawarehouse (EDWH), which is basically a triple store. The triple store is then hosted as a linked building data cloud (in this case - DERI Linked Building Data Cloud). The building data cloud can further be linked to the Linked Open Data (LOD) Cloud, depending on the need and specific use case scenario. Both Triple Stores and LOD Cloud are further discussed in Section 2.

2. Background and Foundations

A brief detail of the necessary concepts to build the foundations, to understand this article are presented in this Section. The background and foundations will help the reader to familiarize themselves with the domain and scope of the work.

2.1. Semantic Web

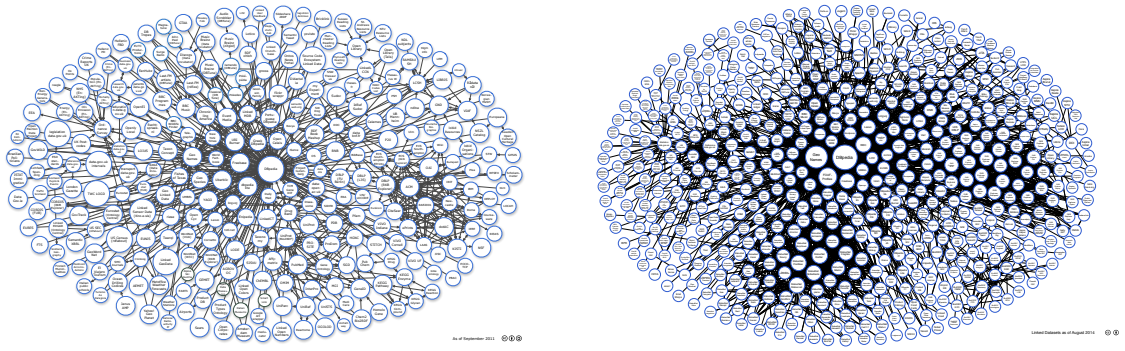
The Web (as in recent past) was viewed as a network of documents linked together in form of web-pages. The content of most of these web-pages was only human consumable. Even the content that was automatically generated from sophisticated database systems was usually presented without its original information structure [3]. Not until the term 'Semantic Web' was coined by Tim Berners-Lee who envisioned that the Semantic Web will bring structure to information contained in the web-pages [4]. The Semantic Web is basically a network of linked data in a form that is more easily machine processable [35].

2.2. Linked Data and Linked Open Data (LOD)

2.2.1. Linked Data

Various working groups from the World Wide Web Consortium (W3C) have developed many languages, tools, and standards to improve and extend the semantic web system. Due to dedication of these working groups, the Web has recently evolved into a Web where documents and data are linked and can be linked easily using variety of semantic web tools. This evolution has resulted in a set of best practices for publishing and linking (connecting) structured data on the Web [35].

Linked Data was introduced by Tim Berners-Lee to highlight a style of publishing and interlinking structured data on the Web. The basic assumption behind linked data is: the more the interlinked data, the more its value and usefulness increases. In simple words, linked data is about using the Web to connect data from different data sources. Technically, linked data refers to publishing data on the Web in a way that it can be



(a) The LOD Cloud (September 2011) [33]

(b) The LOD Cloud (August 2014) [38]

Fig. 3. LOD Cloud Evolution

processed by machines and it is linked to and linked from other external data sets [5, 6]. More specifically, linked data is about using the Resource Description Framework (RDF) and the Hypertext Transfer Protocol (HTTP) for publishing structured data on the Web and for connecting published data between different data sources [7]. The basic precepts of linked data are [5]:

- using RDF data model to publish structured data on the Web
- using RDF links to connect different data sources

In addition to the rules given above, Berners-Lee outlined a set of rules in 2006 for publishing data on the Web in order to ascertain that all published data becomes part of a single global data space. The set of rules are widely known as 'Linked Data Principles' and define the necessary blueprints to publish and link data using the infrastructure of the Web adhering to its standards. The principles are given as [6]:

- Use URIs for naming things
- Use dereferenceable HTTP URIs
- When a URI is dereferenced, use standards (RDF, SPARQL) for providing useful information
- Include links to other URIs, so additional information can be retrieved

2.2.2. *Linked Open Data (LOD)*

The Linking Open Data (LOD) Project[1] is the most significant application and adoption of linked data. The LOD project is a community initiative, supported by Semantic Web Education and Outreach (SWEO) Interest Group[2] with an aim of converting open data available in different data sources into RDF and creating links between common datasets [6]. The project started in early 2007, by October 2007, according to statistics provided by linkeddata.org [3] and lod-cloud.net[4], the participating datasets in LOD consisted of more than two billion RDF triples, interlinked by more than two million RDF links. As of September 2011, the datasets in LOD comprised 31 billion RDF triples, interlinked by 504 million RDF links. The recent evolution of LOD Cloud is given in Figures 3, where the resulting LOD Cloud as of September 2011 is depicted in Figure 3a, and as of August 2014 is shown in Figure 3b.

The LOD Cloud is mainly comprised of diverse data. For example, data about geographic locations, people, companies, books, scientific publications, films, music, television and radio programmes, genes, proteins, drugs and clinical trials, online communities, statistical data, census results, and reviews [6]. This diversity in data is further classified in different domains.

2.2.3. *Triple Stores*

Triple stores have widely gained popularity as tools for storing and publishing linked data, mainly in RDF format. A wide range of linked data publishing tools have been developed. The main aim of these tools is to present linked data on the Web or present non-RDF data in non-RDF legacy data sources as linked data. Almost all of the developed tools for publishing data are capable of supporting dereferencing URIs into RDF descriptions and many of these tools also pro-

[1] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[2] http://www.w3.org/blog/SWEO/

[3] http://linkeddata.org/home

[4] http://lod-cloud.net/state/

vide SPARQL endpoints. The SPARQL endpoints are used for providing query based manipulation of RDF data [6]. Among these tools, OpenLink Virtuoso[5,6] has gained a lot of popularity in both research and industry. OpenLink Virtuoso provides both SPARQL endpoint and linked data interface for serving RDF data. With OpenLink Virtuoso, RDF data can be directly saved in Virtuoso triple store or it can be generated based on the mapping from non-RDF relational databases. In this work, OpenLink Virtuoso is used as linked data publishing tool.

2.3. Semantic Sensor Networks (SSN)

In recent years, an increase in use of sensors and networks of sensing devices has been observed. A wide range of applications of sensors has been noted in meteorology, medical care, environmental monitoring, security, and surveillance. An increasing amount of adoption of sensor networks by different disciplines, like civic planning, satellite imaging, medical sciences, and homeland security has resulted in increasing volume of sensor data. This increase in data has in turn increased and introduced challenges like heterogeneity of devices, data formats, and measurement procedures. In an attempt to address these challenges, the use of semantic web technologies for annotating sensor data with semantic meta-data to increase and sustain interoperability, as well as, provide contextual information has proven valuable [41].

In simple words, Semantic Sensor Networks or Semantic Sensor Web refers to employing (to employ) Semantic Web technologies in sensor networks. As Semantic technologies can be practical in query, management, and combination of sensor and observational data, therefore, they provide users with the flexibility to operate at abstraction levels above the technical details of format and integration. This facilitates the user to not worry about domain concepts and restrictions on quality. Furthermore, autonomous or semi-autonomous agents can be used to assist in collection, processing, and reasoning on sensors and their observations.

The power of linked data can be used to interlink sensor data with external sources on the Web [9]. The W3C Semantic Sensor Network Incubator group[7], works with the aim to provide ontologies for sensor networks. In the proposed approach, "Semantic Sensor Network (SSN) Ontology" [9] is used for enriching sensor data with meta-data.

2.4. Web of Things (WoT)

With easy and high availability of Internet access, a new concept called "Internet of Things (IoT)" has evolved. IoT refers to Internet connectivity directly by objects (i.e. smart devices) and is an exemplification of Ubiquitous or Pervasive Computing vision. One of the major concerns in pervasive computing area has been the integration of smart devices with real world. Even though with introduction of IoT, many smart devices can get Internet connectivity but they still require dedicated software for controlling and monitoring purposes. This is where "Web of Things (WoT)" comes into the picture. WoT is an extension and refinement of IoT which refers to realization of IoT paradigm using Web technologies. WoT not only integrates smart devices into the Internet (the network) but also on the Web (the application layer) [20]. WoT introduces Constrained Application Protocol (CoAP) [40], which is an open Internet standard for the Web of Things. CoAP is based on HTTP, with a purpose to provide a framework for resource-oriented applications running on constrained IP networks [39]. In the proposed approach, CoAP and UDP (instead of TCP on transport layer) are used to communicate with sensors.

2.5. Smart Environments

Physical environments which are interlaced with computational elements, actuators, displays, and sensors, lodged in everyday environmental objects and connected via a network known as Smart Environments. Smart Environments have evolved from Ubiquitous computing and are a consequence of pervasive computing with an aim to make human-computer-interaction or human-system-interaction a pleasant experience [11].

The concept of 'Smart Building' is inherently among many types of Smart Environments with its roots connected to the concept of Building Automation Systems (BAS). Specifically, BAS is a realization of a control system that administers the building elements or other systems like security, heating, air conditioning, and electrical system. The main aim of Smart Buildings is to make life experience of the building inhabitants

[5] http://virtuoso.openlinksw.com/

[6] http://www.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSRDF

[7] http://www.w3.org/2005/Incubator/ssn/

more pleasant while managing optimum resource consumption [12].

Similarly, another concept 'Smart City' recommends adoption of Information and Communication technologies (ICT) to sense, analyze and integrate data from different core systems of the city. The idea specializes on application of ICT to a specfic region, defined as a city in order to facilitate and support livelihood, services, industrial and commercial activites of the region [42].

3. Collecting and Publishing Sensor Data

As discussed before, the first objective of our proposed approach is collection, transformation, enrichment and publishing sensor data to LOD Cloud. In this Section, the process for collecting sensor data, filtering out unnecessary information, conversion of sensor data into RDF (linked data), sensor data enrichment with necessary and relevant meta-data, and publishing sensor data to LOD Cloud is addressed.

3.1. Sensor Data Collection, Transformation, and Enrichment

The primary sources for data in our scenario are sensors. These sensors have been placed, deployed, and installed on multiple locations in the building. The main function of these sensors is to record values (like: heat, temperature, light, power usage etc.). The whole process on how this sensor data is collected, transformed, enriched, and published is depicted in Figure 4. Components of this process are briefly described in the remainder of this Section.

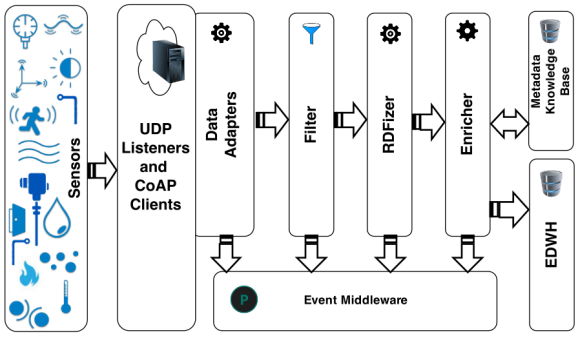


Fig. 4. Process for Sensor Data Collection and Publishing

3.1.1. Sensors, Listeners, and Clients

The first block in Figure 4 labeled as *Sensors*, represents the sensors deployed within the building. The sensed data streams are retrieved by the listeners and clients. In this article, the terms "sensed data stream", "sensor data" and "event-data" are used interchangeably. Most of these sensors generate event-data at a frequency of $1-2$ events per second.

In order to collect the data from sensors, UDP (Universal Datagram Protocol) listeners and CoAP clients were developed. Recall, CoAP (Constrained Application Protocol) is a Web transfer protocol that provides a request/response model for interaction between endpoints.

While UDP listeners listen on different ports where sensors are pushing their data, CoAP clients pull event-data streams periodically from sensors via HTTP. An example of an event-data can be seen in Listing 1 where `17:13:54.66` and `15/5/2013/17/13/54` represent the 'Time and Date', the sensor identifier code (SIC) is `000D6F0000945C77`, and `P`, `W`, `C`, `R` and `S` symbolize 'Power', 'Watt', 'Channel', 'Reading' and 'Sensor' respectively. This particular type of event provides information on power consumption.

```
17:13:54.66 0050C2F4C075:D2:W:15/5/2013/17/13/54; S
:000D6F0000945C77:P=171.872;C=4;R=13;
```

Listing 1: Raw Event-Data

The power consumption events are gathered by listener and client component at a frequency of 70 and 90 events per minute. Each event-data that records power consumption, encoded as $UTF-8$ has a size between 0.093 and 0.098 KB.

Data Adapters are a sub-component, connected to Listeners and Clients with an intent to gather data collected from UDP Listeners and CoAP Clients and publish the data to the Event Middleware component. Since some of the sensors associated with sensing of power and electricity values, deployed in the building, and configured to sense values at different channels - another responsibility of data adapters is to combine electricity and power values sensed on different channels for the same sensor within same sensing timeframe.

3.1.2. Filter: Cleaning Event-Data

The event-data collected by UDP listeners or CoAP clients and gathered by data adapters is forwarded to Filter component. Some incoming event-data contains some irrelevant or unnecessary information in

the form of noise, namely: empty values, repeating characters, unknown characters, etc. The filter component uses string processing and manipulation techniques (e.g. string or character elimination or replacement) for cleaning this event-data and filtering irrelevant information.

Apart from filtering the incoming event-data, the Filter component is also responsible for conversion of date and time values, as well as, conversion of symbols into representations. For example, filtering the previously introduced raw event-data of Listing 1 consists of, concatenating SIC and 'Channel', conversion of Date & Time to milliseconds, conversion of symbols to their corresponding representations, and discarding other information. A resulting example is shown in Listing 2.

```
000D6F0000945C77:4::READING::power
    ::1368638238530::171.872::watt
```

Listing 2: Filtered Event-Data

Once the necessary filtration has been done on the event-data, the data is forwarded to the Event Middleware the RDFizer components.

3.1.3. The RDFizer

The RDFizer component is responsible for converting and encoding the event-data into RDF data. As discussed before, the RDF data model requires data to be structured in the form of triples. Furthermore, the RDF data model requires each subject and predicate to have a URI. As per these requirements, the RDFizer component, first identifies possible triples in the event-data, assigns a URI to each identified subject/predicate and represents event-data in RDF format.

Listing 3 shows the event-data shown in Listing 2 converted to RDF and represented in $N3$. In this presented RDF data, the main subject is *sensorReading*, with predicates *sensorIdentifier, sensorType, sensorReadingType, timestamp, unit* and *value*.

```
@prefix r1: <http://energy.deri.ie/./sensorReading#>.
@prefix rdf: <http://w3.org/./22−rdf−syntax−ns#>.
@prefix xsd: <http://w3.org/2001/XMLSchema#>.

r1:1bc4d668−4ea1 a r1:sensorReading;
  r1:identifier "000D6F0000945C77:4"^^xsd:string;
  r1:sensorReadingType "READING"^^xsd:string;
  r1:sensorType "power"^^xsd:string;
  r1:timestamp "1368638238530"^^xsd:long;
  r1:unit "watt"^^xsd:string;
  r1:value "171.872"^^xsd:string.
```

Listing 3: Event-Data converted to RDF

3.1.4. Event Enricher and Sensor Meta-Data Knowledge Base

The meta-data information for each sensor serves as a knowledge base for enriching the event-data. The Sensor Meta-Data Knowledge Base represents the store that contains such information. An example of a meta-data information for a particular sensor, contains information like `consumerType` (i.e., type of the consumer e.g., database server, Web server, laptop, etc.), `consumer` (i.e., name or title of the consumer e.g., $Apache001$ or $Hadoop009$), `consumerLocation` (i.e., location of the consumer where sensor is deployed e.g., $Building1Floor2Room3$).

The Enricher is responsible for enriching event-data with necessary meta-data. The Enricher uses the Meta-Data Knowledge Base for retrieving meta-data. The meta-data of a specific sensor is retrieved using its identifier (i.e., SIC). The enrichment of event-data with its respective and relevant meta-data is done using the vocabulary and concepts of W3C Semantic Sensor Network Ontology.

Some of the major types of events that were dealt with in the scenario presented in this article are:

Electricity Usage Event:
titled: `ElectricityUsageEvent` - an event recording values of electricity usage.

Heat Usage Event:
titled: `HeatUsageEvent` - an event recording values of heat usage.

Weather Event:
titled: `WeatherEvent` - an event recording values of temperature.

Power Consumption Event:
titled: `PowerConsumptionEvent` - an event containing information on recorded and sensed values of power consumption.

Device State Change Event:
titled: `DeviceStateChangeEvent` - an event that provides information on the change in state of a device, states are on/off.

Following the example of the event-data presented in Listing 3, an instance of one type of event after enrichment is given in Listing 4. Notice that the event has RDF:Type *events:*

`PowerConsumptionEvent`, which is used to maintain and identify between different types of event. Listing 4 shows the instance of generated event-data, serialized in N3, after being enriched with meta-data information about consumer, consumerLocation, consumerDepartment and consumerType.

```
@prefix do: <http://energy.deri.ie/ontology#>
@prefix dr: <http://../deri/deri-rooms#>

:event-1026fd7b-0e5a a
  events:PowerConsumptionEvent ;
  do:consumer do:platform ;
  do:consumerType dr:Room01 ;
  do:consumerLocation dr:building01 ;
  do:powerUsage :usage-9739ccdd-c76d ;
  do:consumerDepartment "facilities"^^xsd:string;
  do:atTime :time-db2c0610-0b33.
:usage-9739ccdd-c76d a dul:Amount ;
  do:hasDataValue 171.87 ;
  do:isClassifiedBy dr:watt .
:time-db2c0610-0b33 a do:Instant ;
  do:inDDateTime
  "2013-05-15T18:17:18"^^xsd:dateTime.
```

Listing 4: Enriched RDF Event-Data

3.1.5. Event Middleware

The Event Middleware component is responsible for controlling the flow and managing event data at different stages, as visible in the Figure 4. It is visible from Figure 4 that the components of "Sensor Data Collection and Publishing" process - Data Adapters, Filter, RDFizer and Enricher are publishing their outputs to Event Middleware component. This is exercised to ensure data availability of sensed-data at each level, that is, after collection, filtration, RDFization, and enrichment. The sensed-data or event-data can be of interest to any other component in any specific output format.

3.2. Publishing Sensor Data

In addition to publishing sensor data to Event Middleware component, another part of the proposed methodology involves publishing the generated linked sensor data to Linked Open Data (LOD) Cloud. As it can be seen from Figure 4, the Enricher component, after enriching event-data with meta-data, initially publishes the event-data to the Event Middleware component. After publishing event-data to Event Middleware, the next responsibility of Enricher is to store sensor data in Event Data Warehouse (EDWH).

3.2.1. Event Data Warehouse (EDWH)

Event Data Warehouse (EDWH) is a central component for the proposed approach, responsible for storing sensor data, and storing the generated data cubes. Details about generation of data cubes are given in Section 5.

As opposed to conventional relational database systems, which are usually used as data warehouses to store historical and multidimensional data cubes, a Triple Store (OpenLink Virtuoso) is used in the methodology proposed in the proposed approach. Triple stores are popular in Semantic Web community, mainly because of their ability to support storage and querying of linked data. In addition to storing linked data, triple stores also facilitate exposing the stored linked data as open linked data by using a SPARQL endpoint. A SPARQL endpoint is used to query the linked data stored in triple store using SPARQL.

In this work, all the sensor data (for all event-types) after being filtered, transformed into RDF, and enriched with meta-data are stored in a triple store. A named-graph [8] is used for storage of event-data. There can be two different approaches for storing event-data, 1) Using different named-graphs for each different type of event or, 2) Using one specific named-graph for storing event-data. One named-graph is used for storage of all types of events, where the events are maintained and differentiated using the event-type information associated with each generated event-data. Additionally, the generated data cubes are also stored in the same triple store. There are usually two different techniques exercised to make linked data open and publicly available: 1) By making the SPARQL endpoint accessible via a publicly accessible URL and making the named-graph publicly accessible, and 2) By exporting (periodically, if necessary) an archive of dataset (named-graph) and posting the archive on a publicly accessible URL. Usually, both of these techniques are used together, in this proposed work, however, we propose and use the first technique.

Publishing the generated linked data not only achieves making the linked sensor data, and generated data cubes published as linked open data but also supports the idea of storing historical data (in case of linked sensor data) in a data warehouse as well as the concept of Data Marts or OLAP (in case of data cubes) in data warehouses. More details about EDWH is given in Section 5. In order to increase the effectiveness of the published linked open data, it can be further linked with other available datasets in the LOD Cloud. The linking can be a tricky and hectic job, however, there

are different linking frameworks to support linking local datasets (generated linked open datasets) with LOD Cloud datasets. Both Silk [45] and LIMES [34] frameworks are capable of linking local dataset with LOD Cloud, however, both of these frameworks are limited in identification and discovery of relevant datasets to link with. A separate set of techniques, proposed in [31] (to extract keywords or query terms) and [32] (to discover relevant SPARQL endpoints or LOD datasets) can be combined together to discover LOD datasets and link with using Silk or LIMES. Once linked, a query federation engine like [26] can be used to perform additional data analysis by gathering data from different LOD Cloud datasets.

4. Event Registration and EDWH Ontology

The second step of the proposed methodology involves generating multi-dimensional data cubes. To generate data cubes, first, an event needs to be registered in the system. This Section details the process of mapping necessary meta-data of an event to facilitate cube generation. As discussed before, an ontology to map necessary and relevant meta-data of an event during the event registration process is proposed. The details of the ontology are also given below. And the process of *Cube Generation* is detailed in Section 5.

4.1. Ontological Model

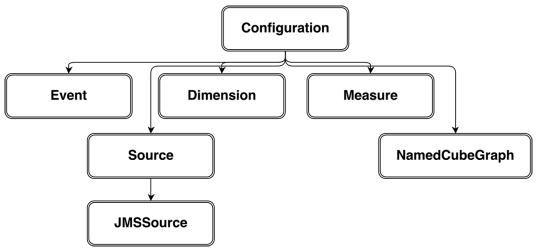
Due to lack of vocabularies that bridge W3C Semantic Sensor Network (SSN) Ontology and W3C Data Cube Vocabulary together [27], an ontology based on concepts of W3C RDF Data cube vocabulary to map meta-data (enriched using SSN ontology) for cube generation and to generate data cubes is defined and proposed in this article. Mainly due to the servicing characteristics of this ontology, we call it EDWH (Event Data Ware-House) Ontology. Two different models of the EDWH Ontology are given in Figure 5, where, the Class hierarchy of the ontology is shown in Figure 5a and a detailed semantic (conceptual) model is presented in Figure 5b.

Configuration is the highest level class in the hierarchy. This class corresponds to configuration information for an event when it is registered in the system for cube generation. Recall that, in order to generate data cubes for some particular event, the event needs to be registered first. The whole process of registering an event is detailed later in this Section.

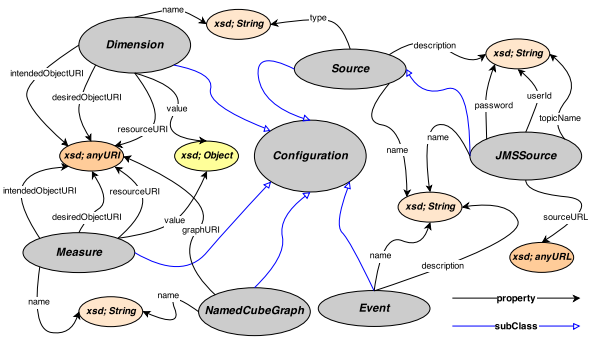
Dimension is a sub-class of configuration with an association of one-to-many. The Dimension class serves similar purpose of 'Dimension' concept in general data-warehousing. A Dimension identifies the observation or describes its characteristics (e.g., time, consumerLocation, consumer, consumerDepartment) [16]. In our case, an observation symbolizes one instance of an event-data value. Examples of dimensions can be observationTime and consumerLocation.

The *Measure* class is another sub-class of configuration class. The main concept of the Measure class is also derived from general data-warehousing concept of 'Measure' and serves the same concept of measure in RDF Data cube vocabulary; a measure represents the phenomenon being observed [16] (e.g., powerUsage).

The third child class of Configuration class is *Event*, which is responsible to contain information of an event-type that has been registered in the system by a user for creating data cubes. While registering an event, the user specifies the URI of the graph which should contain generated data cubes. *NamedCube-*



(a) EDWH Ontology Hierarchical Model



(b) EDWH Ontology Conceptual Model

Fig. 5. EDWH Ontology Models

Graph is also a sub class of configuration. The main purpose behind the definition of NamedCubeGraph class is storage of generated cubes.

The event data can be retrieved from more than one type of source. *Source* class is a generalized class that defines and contains information about the source of event data. Source is a sub-class of configuration and is a super class of *JMSSource* class. In our scenario, we use Java Message Service (JMS), *JMSSource* class is a specialized class of Source class to serve specific use of JMS Server.

4.2. Event Registration

Recall that, the sensors deployed in the DERI building sense different types of values, for example, heat, temperature, power consumption, etc. Each of the sensed data values, also known as event-data, represents a different type of event. Specifically, the registration of an event in the system symbolizes mapping the necessary information (meta-data) for generating cubes. The information specified during the registration should contain a set of dimensions and measures - which form a multidimensional cube, name of graph in triple store for storage and source of incoming events.

The Event Registration Process is depicted in Figure 6. The process is supported by a Web User Interface (UI). The first step of event registration on UI involves specification of the data source. In our scenario, source is a JMS-Source, hence in this step; the user specifies

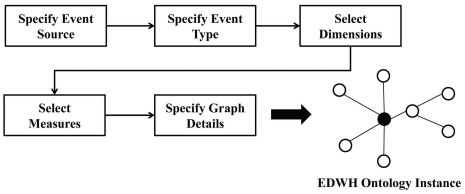


Fig. 6. Event Registration Process

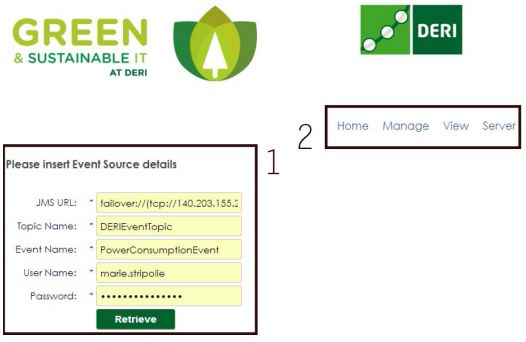


Fig. 7. Source Specification Screen

the URL of the JMS Server, the topic that is broadcasting events, the username and the password (if any) for accessing the server. The screenshot of data source specification on event registration system is given in Figure 7, where 7 (1) is used for specifying data source and 7 (2) is used for management of "Event Registration System" and "Event Data Ware-House (EDWH)" (detailed in this Section 5).

Since there can be different types of events broadcasted on the same topic on the same server, the next step is to select the event-type for which data cubes must be generated. Once the event-type has been specified, a single JMS message containing event-data of the specified event-type is retrieved. The retrieved message is in the RDF format and represents an instance of event-data. The subject/resource corresponds to an instance of event, and predicates of this subject characterize the event-data. Therefore, the RDF message is parsed and a set of predicates is formed. We call this set "the predicate set" and is denoted by *P*. The set contains items that are direct predicates of the main subject and are possible candidates of being dimensions or measure. In the next step of the registration process, the user is presented with the set *P*. The user first selects a set of dimensions from set *P*, then, the user selects a set of measures. Screenshots of possible (candidate) and selected predicates (properties) in case of both dimension and measure are given in Figures 8 and 9 respectively. The set of dimensions is denoted by *D* and the set of measures is denoted by *M* and selection of dimensions and measures is done according to following rules:

- P = $\{Pre_1, Pre_2, ... Pre_n\}$: Set of Predicates
- D $\neq \emptyset$ and D $\subset$ P: Set of Dimensions
- M $\neq \emptyset$ and M $\subset$ P: Set of Measures
- D $\cap$ M = $\emptyset$

While selecting dimensions and measures, the user also has to specify `intendedObjectURI` and `desiredObjectURI` properties of each selected dimension or measure. An example selection of dimensions and measures for `PowerConsumptionEvent` can be seen in Figures 10a and 10b respectively. Both `intendedObjectURI` and `desiredObjectURI` are data properties of dimension and measure classes of the ontology mentioned in Section 4.1, and both of these data properties contain values of the predicates. In the example shown in Figure 10a, the user has selected *consumer*, *consumerType* and *consumerDepartment* as

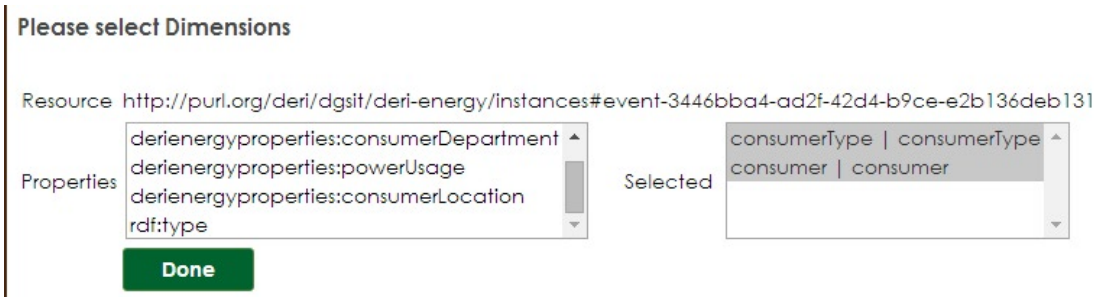


Fig. 8. Dimension Selection Screen

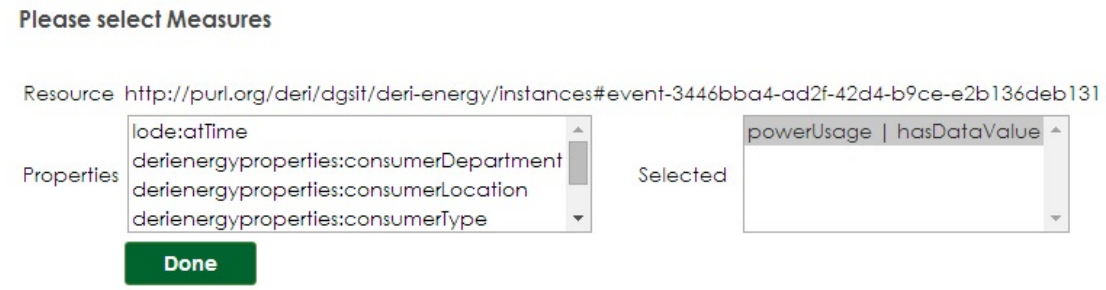
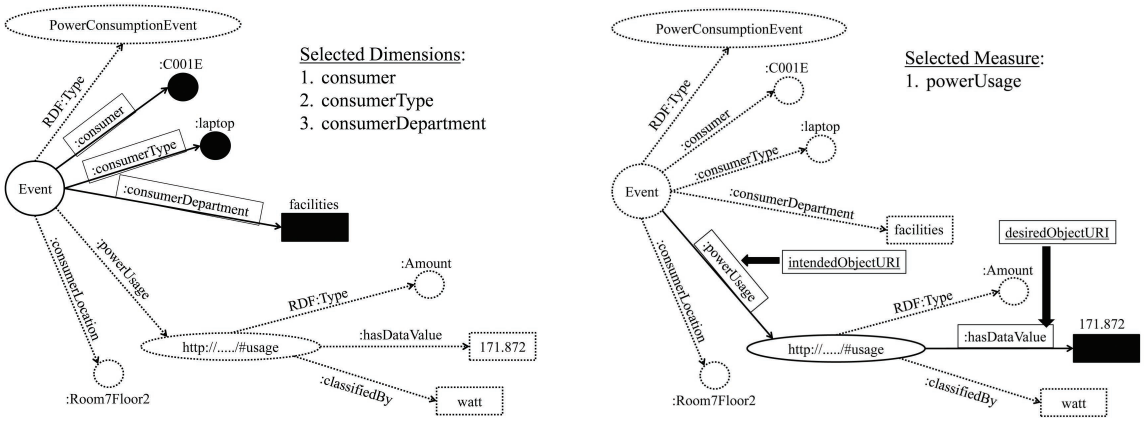


Fig. 9. Measure Selection Screen



(a) Example of selected dimensions

(b) Example of selected measures

Fig. 10. Example Dimension and Measures

possible set of dimensions for creating data cube. The `intendedObjectURI` and `desiredObjectURI` for each of the specified dimension is same since there is no nesting. In case of both *consumer* and *consumerType*, the corresponding values are URIs while *consumerDepartment* has literal value. In both Figures 10a and 10b URIs are represented using circles and literal values are depicted by rectangles. In Figure 10b, the user selected *powerUsage* as measure. Notice that `intendedObjectURI` and `desiredObjectURI` are different since RDF event-data has a nested resource for `powerUsage` i.e. "`:Amount`". In this case `intendedObjectURI` is `powerUsage` and `desiredObjectURI` is `hasDataValue`.

After selecting the dimensions and measures, the next step in the event registration process requires specification of graph details. This graph detail is used to store generated data cubes. While inserting values for graph details, the user specifies the name, detail

and comments. The name of the graph is used to create a named-graph in the triple store. After the process of getting all necessary information from the user for registering an event is completed, the system will generate an instance of EDWH ontology. A complete instance of the ontology contains the information on the selected dimensions, measures, graph detail and source of incoming events.

Once generated, the EDWH ontology instance is stored in the triple store. Specifically, the instance is stored in a named-graph in the Triple store. The named-graph thus contains information of registered events and is therefore called "Registered Events Knowledge Base (REKB)". The REKB is used by EDWH to retrieve registered events and their corresponding configurations. Generation and storage (in triple store) of multidimensional data cubes is responsibility of an Event Data Warehouse (EDWH) agent. The EDWH agent is a software agent that is introduced in Section 5.

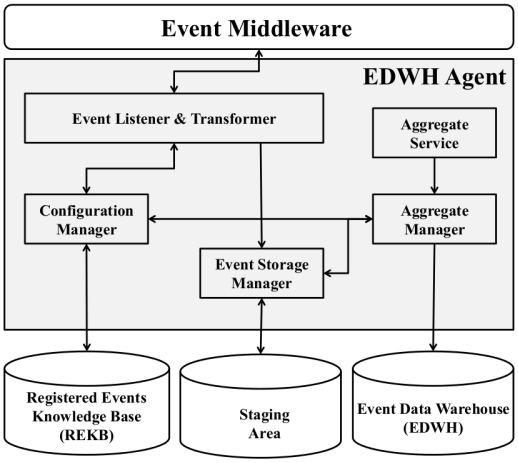


Fig. 11. EDWH Agent structure

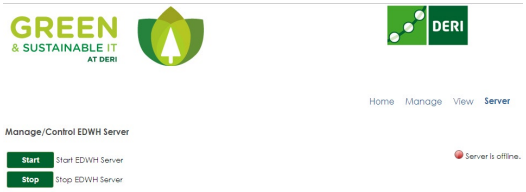


Fig. 12. EDWH Agent Screen

5. Cube Generation

As mentioned before, the generation of multidimensional cubes is another core element of the proposed work. Multidimensional data cubes provide support to decision making tools and applications to run complex queries on multidimensional data. In the proposed system implementation, an agent called the EDWH agent is developed. The main responsibility of this agent is to aggregate data, generate data cubes and store them in the triple store. The basic structure of the EDWH agent is given in Figure 11. As discussed before, the EDWH is controlled by a GUI - the screenshot of turning the EDWH agent on/off is given in Figure 12. It is pertinent to mention here that, the generation of data cubes is live only when the EDWH agent is turned on. If the EDWH agent is not live (i.e. off), the data cubes for the respective time-frame will not be generated and hence will not exist in the system.

5.1. EDWH Agent: Generating Cubes

In order to generate aggregates on data and create multidimensional cubes, all incoming real-time events need to be stored in a primary staging area. On starting the EDWH agent, the "Configuration Manager" component of the EDWH agent retrieves the events that have been registered for cube generation from Registered Events Knowledge Base (REKB) along with their corresponding configurations (registered event configuration information). The configuration of one particular event contains information on source, dimensions, measures and graph detail. This information is then used by the Event Listener and Transformer components.

The source information in the configuration is used to subscribe to topics and to start listening on those topics in order to retrieve events. The dimension and measure information is used to transform events. In addition to selected dimensions for an event, a set of `Time` dimensions (For example, Year, Month, Day, Hour, Minute, and Second Dimensions) are also considered. The transformation of an event is done by adding values of only selected dimensions and measures; the remaining information is discarded. Continuing with our previous examples of event shown in Listing 4, an example of transformed event can be seen in Listing 5. This event, after transformation is now called an observation with RDF:Type 'Observation' (because the event is now candidate for cube generation) and '`PowerConsumptionEvent`' (since the corresponding event has similar type).

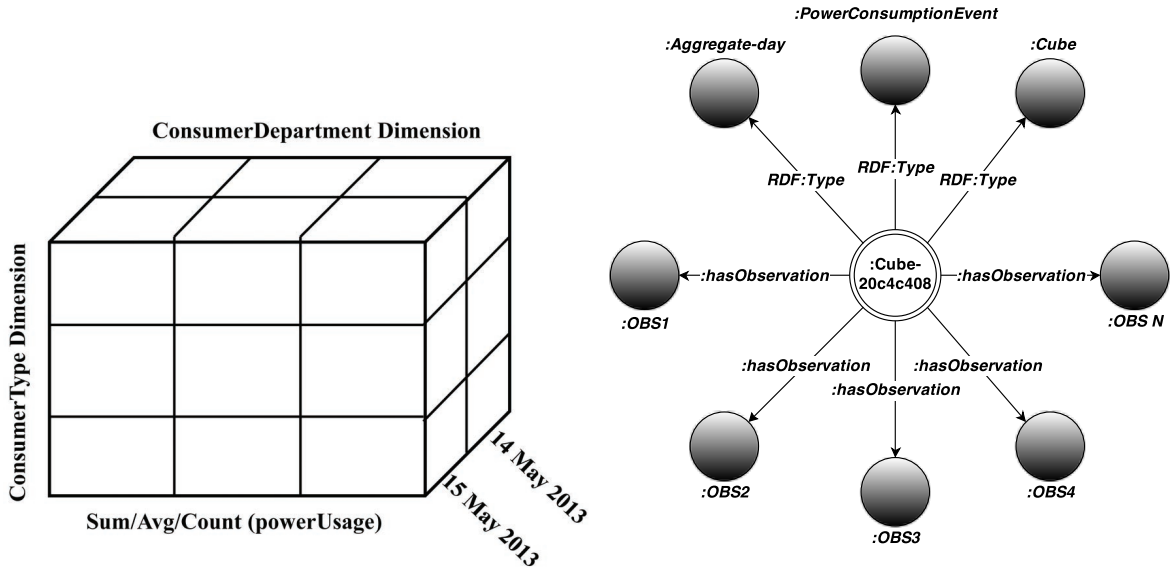
@prefix edwh: <http://energy.deri.ie/edwh#>.

```
:Observation−58m5c716 a do:Observation,
  events:PowerConsumptionEvent;
  edwh:consumer dr:platform;
  edwh:consumerDepartment "facilities"^^xsd:string;
  edwh:consumerType dr:Room01;
  edwh:day "15"^^xsd:long;
  edwh:hour "18"^^xsd:long;
  edwh:minute "17"^^xsd:long;
  edwh:month "5"^^xsd:long;
  edwh:second "18"^^xsd:long;
  edwh:year "2013"^^xsd:long;
  edwh:powerUsage "171.872"^^xsd:long.
```

Listing 5: Transformed Event Data Observation

Once transformed, the observation is forwarded to the "Event Storage Manager". The "Event Storage Manager" component is responsible for performing CRUD (Create, Read, Update, Delete) operations on the staging area. When this component receives transformed events from the "Event Listener and Transformer" component, it inserts or updates values in the staging area. When it receives a data request from the "Aggregates Manager", it performs data retrieval and sends a response back. The "Aggregates Manager" is responsible for performing aggregate operations (Sum, Average, Count, etc.) on the data received from the "Event Storage Manager" component and is also responsible for creating multidimensional cubes. This manager generates cubes for each registered event.

The cubes are generated based on information contained in the configuration of registered events. As of current implementation of the "Aggregates Manager" component, cubes are generated based on `Time` dimension and grouped together by values of the rest of the specified dimensions (contained in the configuration information of the registered event). The "Aggregate Service" component triggers cube creation routine in "Aggregates Manager". The service executes each 15 minutes (quarter), hour, day and month. The cubes it generates are called Quarter Cube, Hour Cube, Day Cube and Month Cube respectively. Each quarter cube is generated from data contained in the staging area. Once quarter cube is generated, it is stored in Event Data-Warehouse (EDWH). The triple store is used for both EDWH and staging area. The hour cubes are generated from data of quarter cubes (i.e. four quarter cubes are aggregated to generate an hour cube). A day cube is generated from hour cubes (24 hour cubes aggregated = 1 day cube) and similarly a month cube is generated from day cubes. Since, we are generating cubes on-the-fly, the approach of generating cubes from cubes has proven to be useful in our scenario. An example of a generated cube can be seen in Figure 13a and its corresponding RDF graph is shown in Figure 13b.



(a) An Example Cube

(b) An Example RDF-Graph of Cube

Fig. 13. An Example Cube and RDF Graph

The example cube that is shown in Figure 13a is created using `Time`, `ConsumerDepartment` and `ConsumerType` dimensions. This example cube is generated by applying any of the aggregate function (SUM, AVG, COUNT) over a period of one day. As shown in Figure 13b, the main subject of the RDF graph is "Cube-20c4c408" with RDF:Type *`:Cube`*, RDF:Type *`:Aggregate-day`* and RDF:Type *`:PowerConsumptionEvent`*. The RDF:Type `:Cube` signifies that the generated instance is of type Cube, similarly, RDF:Type `:PowerConsumptionEvent` signifies that the generated cube instance is for `PowerConsumptionEvent`. The RDF:Type `:Aggregate-day` is used to differentiate between other type of generated cube instances of `PowerConsumptionEvent`. Each generated cube has their respective aggregate-type in addition to 'Cube' and 'Event' type (e.g. `PowerConsumptionEvent`). A cube generated over a quarter has an aggregate-type: *`:Aggregate-quarter`*, similarly for an hour cube: *`:Aggregate-hour`* and for a day cube: *`:Aggregate-day`* and so on.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

:OBS-21u6c412 a do:Observation,
  events:PowerConsumptionEvent;
  edwh:consumerDepartment "facilities"^^xsd:string;
  edwh:consumerType dr:Room01;
  edwh:day "15"^^xsd:long;
  edwh:month "5"^^xsd:long;
  edwh:year "2013"^^xsd:long;
  edwh:powerUsage "34132.6"^^xsd:long.
```

Listing 6: Cube Observation

A cube is composed of several observations, as depicted in Figure 13b, and an example of one such observation of a cube is shown in Listing 6. The example represents a cube generated using SUM(`powerUsage`) function, grouped by `ConsumerDepartment` and `ConsumerType` and aggregated over a period of day.

Once cubes are generated, they are stored in EDWH (specifically in a named-graph) and are published to the linked building data cloud [15]. Recall that, in the work presented in this article, a triple store is used for storing linked data. As it is obvious that the generated cubes are in linked data format, the triple store provides all the necessary support for exposing linked data as open linked data and querying the stored linked data using SPARQL. Also recall that, the named-graph is specified during the registration process of an event. The named-graph symbolizes the concept of dataset. Therefore, in addition to linked sensor data dataset, which was collected and published (as discussed in Section, 3), the EDWH now stores another dataset - which contains the generated data cubes. Both datasets (containing linked sensor data and data cubes) facilitate the idea of general data warehouse and OLAP system within the data warehouse. Moreover, since each generated cube has its own aggregate-type, this information is used to maintain and differentiate between different types of cubes in the named-graph (containing data cubes) of EDWH.

5.2. DERI Linked Building Data Cloud

The linked building data cloud for Digital Enterprise Research Institute (DERI) building has been implemented using the Linked dataspace for Energy Intelligence (LEI) [14] developed by the Green and Sustainable IT group at DERI. The Building Energy Explorer dashboard is a central user interface that makes extensive use of the merged data within the linked building data cloud.

The dashboard visualizes relevant linked building data together with energy consumption sensor data and presents it in an actionable manner that requires minimal effort for users to leverage the knowledge within energy-related decision-making. The objective of the dashboard is to help users identify energy leaks and increases energy usage awareness within the DERI building by linking actual energy usage data to the entity(ies) responsible for the energy usage.

The main screen of the dashboard is presented in Figure 14, within box (1) data from the rooms, people, and group vocabulary can be seen; it is used to add context to the energy consumption readings. In (2) historical usage along with real-time instant measures from the energy sensors are shown, along with a breakout for consumption type (lights, heat, sockets). The interface also displays the output of the Energy Situation Awareness Service (ESAS) via a widget in (3). ESAS is a situation awareness service that assists user to interpret and understand energy data to make decisions. The widget shown in Figure 14 (3), performs energy situation assessment by comparing the accumulative consumption with historical usage data and usage targets to detect high usage situations. In the widget,

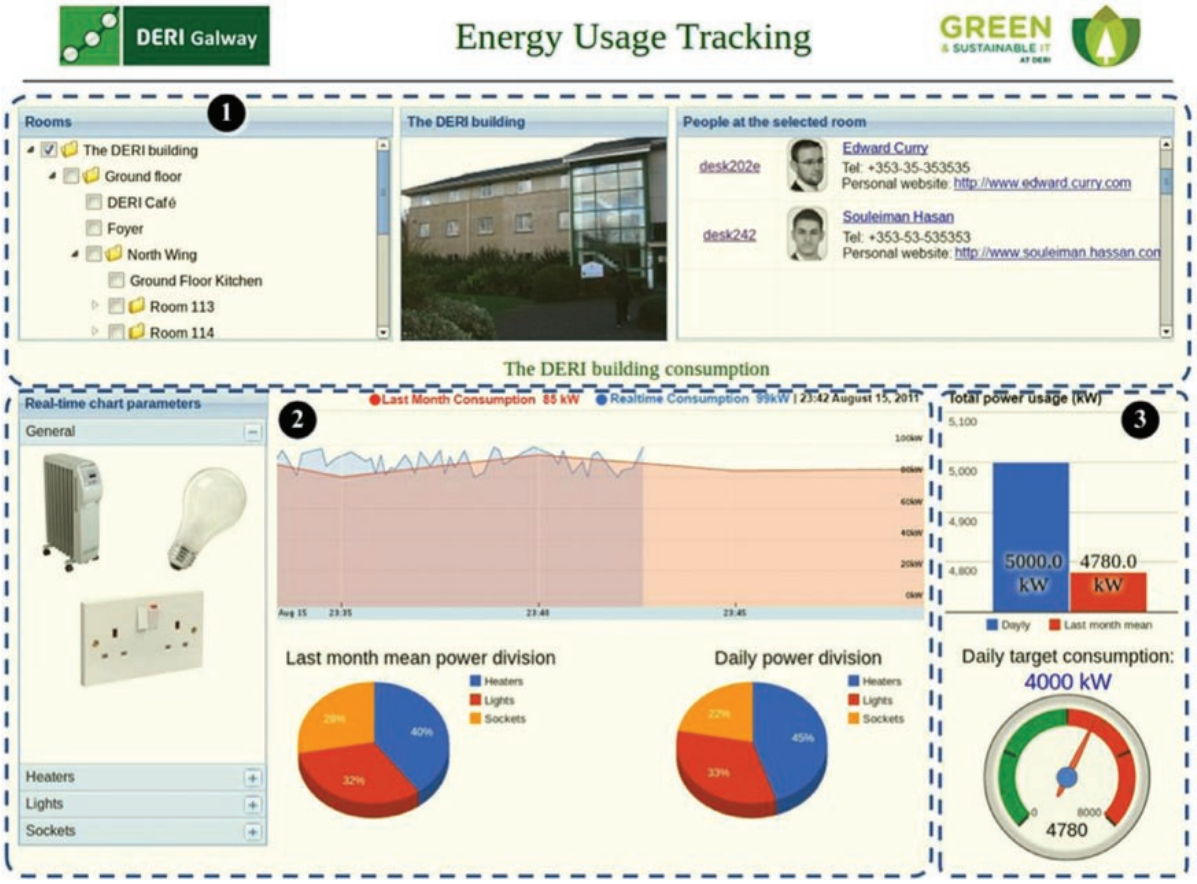


Fig. 14. Building energy explorer dashboard [14]

two bars are used to show the daily accumulated energy usage in comparison with the monthly average to highlight any deviations in the consumption pattern.

6. Evaluation

6.1. Experimental Setup

In order to conduct the evaluation of our proposed approach, we used dedicated server running 64-bit Windows7 OS, with 4GB of RAM and an Intel Core i5 (2.53GHz) CPU. We use in this evaluation a set of sensor observations collected during an entire day. Our evaluation consists of analyzing various quantitative dimension: storage size, query execution time, execution time for storing and generating data cubes, and accuracy of generated cubes.

Table 1
Cube Storage Size and Count for One Day

		Raw	Quarter	Hour	Day
No. of Cubes	**Avg**	—	96	24	1
	Sum	—	96	24	1
	Count	—	96	24	1
	Sub-Total	—	288	72	3
	Total Cubes	**363** Per Day			
Storage	**Per Cube**	—	175-180 KB	25-30 KB	60-66 KB
	Per Day	45-50 MB	49-51 MB	1.7-2.2 MB	180-200 KB
	Total Size	**96-104 MB** Per Day			

Table 2 Linked Data Stats for One Day

Parameter	Count
Total Number of Triples	235536
Distinct Subjects	16628
Distinct Objects	16478

Table 3 Number of resource with respect to Subject

Subject	Resources
`http://..../events#PowerConsumptionEvent`	19723
`http://energy.../ontology#Observation`	19602
`http://energy.../ontology#Cube`	121
`http://energy.../edwh#Aggregate-quarter`	96
`http://energy.../edwh#Aggregate-hour`	24
`http://energy.../edwh#Aggregate-day`	1

6.2. Results

For the first part of the evaluation, refer to Table 1. This table contains the results gathered while performing the size evaluation for the event-data and data cubes. The first half of Table 1 contains information about the type of aggregate function (Average, Sum, Count) used for generation of cubes along-with the number of cubes generated using the aggregate function. It is visible from the table that a total of 363 cubes are generated per day for the three aggregate functions. The second half of Table 1, gives information on the storage size required to store the generated cubes. Each entry in Table 1 for storage, represents the size of RDF data serialized in N3 notation and encoded as UTF-8 with respect to the different types of cubes generated using three dimensions. Since 'Time' dimension is common in all cubes, it is therefore considered implicit in all experiments.

One notable thing is difference in 'Per Cube' storage size requirement for different type of cubes (Quarter, Hour, Day). Primarily, the difference in storage size among same cube types, for instance (175-180 KB for Quarter Cube) is because of different number of sensor readings recorded during each quarter (recall, the sensor readings are recorded at a frequency of 70 and 90 events per minute). The difference in 'Per Cube' storage for different types of cubes (Quarter, Hour, Day) is because the corresponding 'Time' dimension is dropped (eliminated or deleted), values are aggregated and a cube is created. For instance, when *Quarter* cube is generated, the 'second (Time)' dimension is dropped and while generating *Hour* cube, the 'minute (Time)' dimension is eliminated.

It can be further noticed that, in case of *Day* cube, the 'Per Cube' storage size should be less than the *Hour* cube (seeing that the 'hour (Time)' dimension is eliminated). However, when *Day* cubes are generated, the number of observations in *Day* cube are higher than the number of observations in *Quarter* or *Hour* cubes. For instance, consider two different rooms within same building, say *A* and *B* (with 'location' being a dimension), consume power at different times in a day. When generating quarter or hour cubes for these rooms, every non-zero reading will generate additional observation in the cube, similarly, if the readings are zero, no observations will be added in the cube. However, while aggregating data for an entire day, each observation (recorded at anytime during the day) is included in the cube, thus, resulting in more number of observations than *Quarter* or *Hour* cubes.

It is important to notice that the size of cubes is dependent on different factors, for example: number of dimensions, size of incoming event, frequency of events etc. Taking the example of the *PowerConsumptionEvent* illustrated previously, each event-data that records power consumption, encoded as $UTF-8$ has a size between 0.093 and 0.098 KB. As, the power consumption events are gathered by listener and client components at a frequency between 70 and 90 events/minute, it is understood that the number of dimensions selected for each event (during registration) will have an impact on over-all size of generated cubes. To do so, the same event (`PowerConsumptionEvent`) was registered multiple times with different number of dimensions. A chart showing the impact of changing number of di-

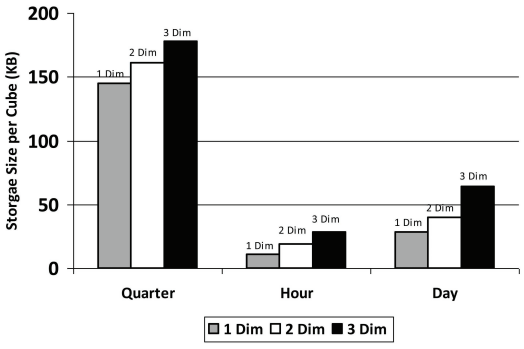


Fig. 15. Size Chart for different No. of dimensions

mensions for `PowerConsumptionEvent` event is shown in Figure 15. It can be easily assessed from Figure 15 that, the storage size is directly proportional to the number of dimensions in all three cases (Quarter, Hour, Day). That is, with an increase in number of dimensions, the overall storage size required to store a cube also increase.

Since it is understood that triple stores deal with linked data, therefore, in addition to the results regarding storage size and the number of generated cubes, the linked data statistics were also exported using OpenLink Virtuoso Conductor. The linked data statistics are given in Table 2, where the total number of triples and number of distinct subject and distinct objects can be seen. The statistics presented in Table 2 are exported for data cubes generated for one day using three dimensions and only 'Sum' aggregate function. Table 3 provides information on the number of resources for each RDF:Type. For example, a total of 96 resources of RDF:Type: `Aggregate-quarter` are generated in one day.

For the second part of the evaluation, the required Query Execution Time (QET) is observed. A set of most commonly used queries in our scenario were defined and executed on different types of cubes stored in the EDWH as well as on raw observations - stored in the staging area. The type of queries that were executed are shown in Table 4. The comparison of the recorded values for QET are shown in Table 5.

Looking at Table 5, it can be observed that, the QET improves for all of the queries tabled in Table 4. The improvement in QET is referred to as the minimum time required to execute and retrieve the desired results. While comparing the QET of queries executed on Raw event-data to Quarter cube, a significant reduction in QET can be seen. It can also be observed that the improvement in QET increases with further aggregation, that is, the QET further reduces on Hour cubes and similarly, the QET is minimum in case of Hour cubes.

Table 5
Query Execution Time Comparison

Query	Query Execution Time (in milliseconds)			
	Raw	Quarter	Hour	Day
Q1	5747	1938	—	—
Q2	5381	1121	541	—
Q3	4449	1003	535	380
Q4	5939	774	—	—
Q5	6316	795	541	—
Q6	6770	754	521	301
Q7	4330	351	256	181
Q8	5121	397	298	237

In order to further evaluate our system, a set of similar data cubes (Quarter, Hour, Day) were generated using state-of-the-art RDF-Data Cube Vocabulary (RDF-DCV). A comparison of the QET execution time between the approach proposed in this article (EDWH) and *RDF-DCV* was carried out. The results of this comparison can be seen in Figures 16 - 21.

In Figure 16, the QET results for both *EDWH* and *RDF-DCV* for quarter cube are shown. While, in Figure 17, a comparison between both approaches (EDWH and RDF-DCV) is shown. In Figure 17, the comparison is computed by taking the results of *RDF-DCV* at the reference line and calculating the percent increase or decrease in QET. The percent increase signifies that the QET of *RDF-DCV* is less than QET of *EDWH*, and percent decrease signifies that *EDWH* has better QET than *RDF-DCV*. From Figure 17, it can be seen that QET for *EDWH* is less than QET of *RDF-DCV* for five queries (Q1, Q2, Q6, Q7, Q8) and QET

Table 4
Set of Queries

Q1	Give me per minute powerUsage details for all consumers, consumerTypes and consumerDepartments.
Q2	Give me per hour powerUsage details for all consumers, consumerTypes and consumerDepartments.
Q3	Give me per day powerUsage details for all consumers, consumerTypes and consumerDepartments.
Q4	Give me per minute powerUsage details for consumer=Hadoop009 and all consumerTypes and consumerDepartments.
Q5	Give me per hour powerUsage details for consumerType=Laptop and all consumers and consumerDepartments.
Q6	Give me per day powerUsage details for consumerDepartment=facilities and all consumers and consumerTypes.
Q7	Give me per day powerUsage details for consumerDepartment=facilities, consumerType=Laptop and all consumers.
Q8	Give me per day powerUsage details for consumerDepartment=facilities, consumerType=Laptop, and all consumers where powerUsage is between 1000 and 2000 Watts.

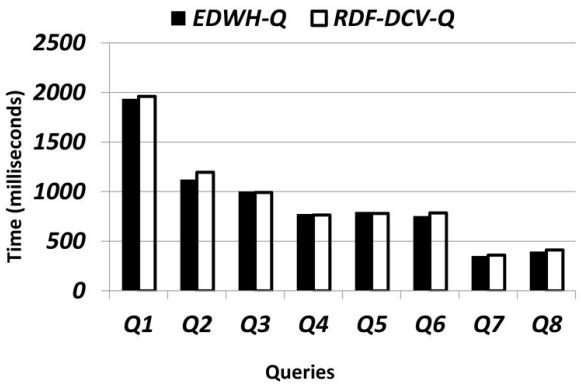


Fig. 16. EDWH and RDF-DCV (Quarter Cube)

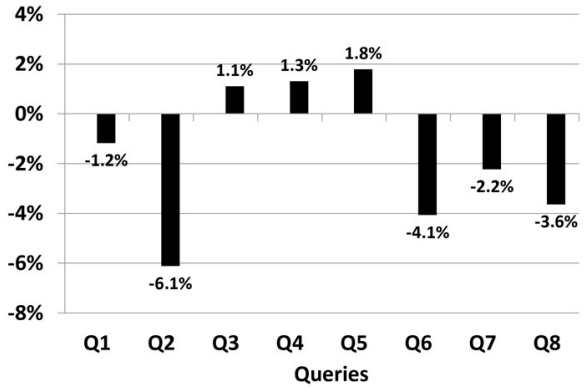


Fig. 17. EDWH vs RDF-DCV (Quarter Cube)

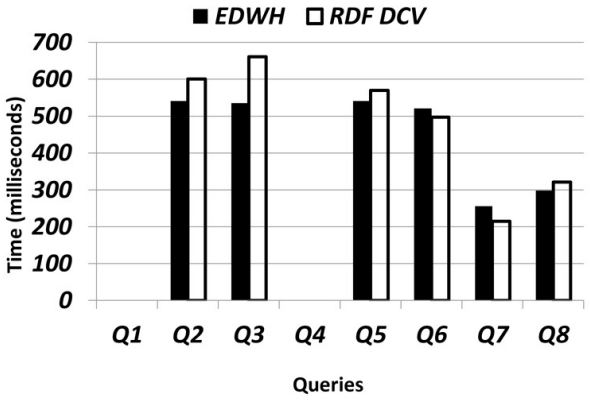


Fig. 18. EDWH and RDF-DCV (Hour Cube)

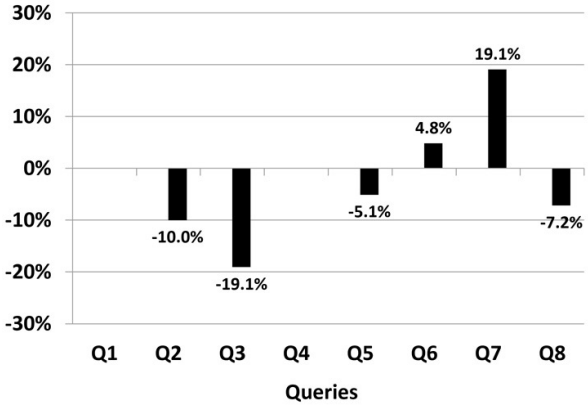


Fig. 19. EDWH vs RDF-DCV (Hour Cube)

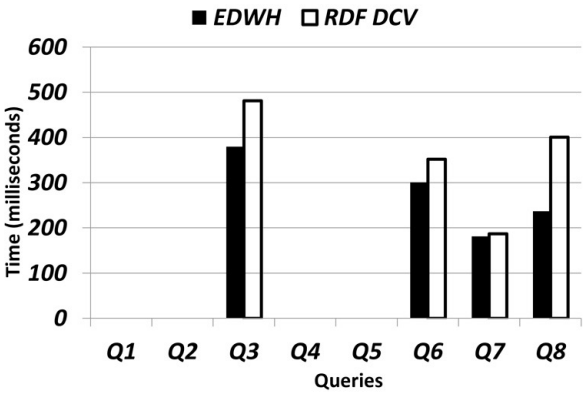


Fig. 20. EDWH and RDF-DCV (Day Cube)

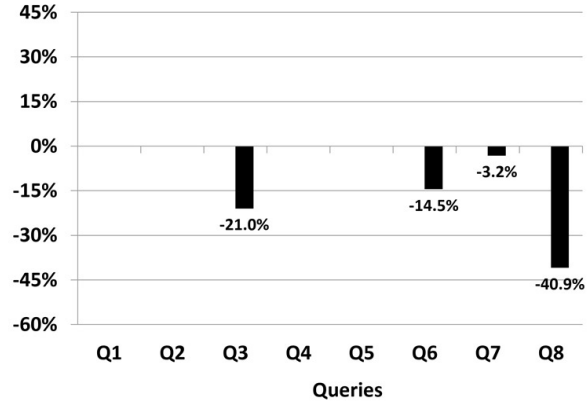


Fig. 21. EDWH vs RDF-DCV (Day Cube)

for *RDF-DCV* is better than *EDWH* for three queries (Q3, Q4, Q5).

Similarly, the results of QET for hour cube, in case of both approaches are shown in Figure 18 and a comparison is shown in Figure 19. It can be observed from Figure 19 that *EDWH* approach gains better results for four queries (Q2, Q3, Q5, Q8) and *RDF-DCV* has better results for two queries (Q6, Q7), when executed

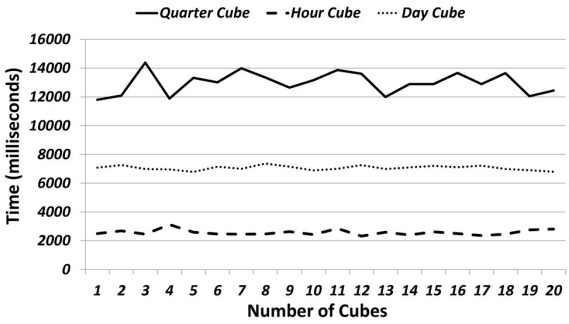


Fig. 22. Performance Chart

on hour cubes - generated using both approaches. And looking at the QET results for day cube in Figure 20 and comparison in Figure 21, it is clearly visible that QET for *EDWH* approach has better results than *RDF-DCV* for all the queries (Q3, Q6, Q7, Q8), when executed on day cubes - generated using both approaches.

For the third part of the evaluation, the performance of the system is evaluated and compared with RDF-DCV approach. In this evaluation, we use the term performance referring to the time the system takes to process one particular type of cube, that is, the time the system (EDWH) approach takes to retrieve data, generate a cube and store in EDWH. It is important to note here that the configuration information of a registered event is critical in the process of generating any particular type of cube for such event, because the data retrieval, the generation of a cube, and the storage in the EDWH is dependent on it.

The performance values were computed for a sample of 20 cubes on each Quarter, Hour and Day types of cubes as shown in Figure 22. Since we are proposing a solution that generates data cubes on-the-fly, the time the system takes to process any particular type of cube has to be reduced as much as possible for providing an efficient system. As we can see in Figure 22, the

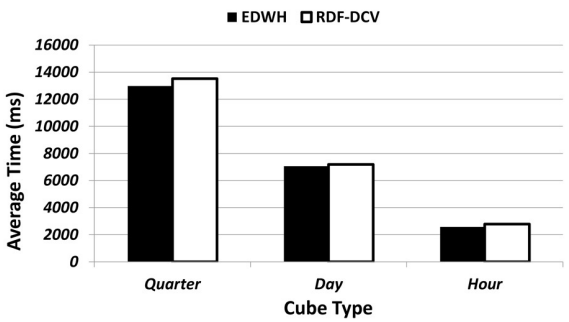


Fig. 23. Performance Comparison (EDWH vs RDF-DCV)

time taken to process a quarter cube is between 12 and 15 seconds, less than 3 seconds for an hour cube, and between around 7 seconds for a day cube. The variations in cube generation time is due to variation in the data size and the frequency of incoming events.

A comparison between the approach presented in this article (EDWH) and the cubes generated using *RDF-DCV* is also performed. For RDF-DCV, again, a sample of 20 cubes for each Quarter, Hour and Day were generated using RDF Data Cube Vocabulary (RDF-DCV). First, an average time for generating cubes (Quarter, Hour, Day) using RDF-DCV was computed and then an average time for generating cubes (Quarter, Hour, Day) using EDWH approach was computed. A comparison between average times (in milliseconds) to generate Quarter, Hour and Day cubes using both approaches is given in Figure 23. It can be observed from the Figure 23, that the average time taken by EDWH approach is less as compared to RDF-DCV for all three types of cubes.

For the fourth and final part of the evaluation, the accuracy of the generated cubes is evaluated. Initially, two different approaches for conducting this evaluation were considered: 1. Randomly select cubes of each type (Quarter, Hour, Day) and validate against the data in staging area, and 2. Randomly select cubes of each type and validate against the real values, that is, the comparison of cube values against the values recorded by meter (energy, heat, power). As the cubes are generated using the data present in staging area, the chance of an error in this case can be as low as almost zero, therefore, the second approach was selected. The error value was computed using a simplified formula given as:

– $Error = |RealVal - AchievedVal|$

Where $RealVal$ is the value recorded by the meter and $AchievedVal$ is the value of the generated cube.

A set of 20 randomly selected cubes were selected, including all Quarter, Hour, and Day cubes and an error value was computed using the formula given above. The error results obtained are given in Figure 24. It is obvious from Figure 24, that the error value in day cube is much higher than hour and quarter cube. It can also be observed that the error value of hour cube is higher than the quarter cube. This variation in error values can be due to different factors, among which, the loss of precision can be one. As the values are aggregated at higher time-frame (Hour, Day, Month, Year), the error values as compared to the lower time-frame increases. However, it can also be observed that

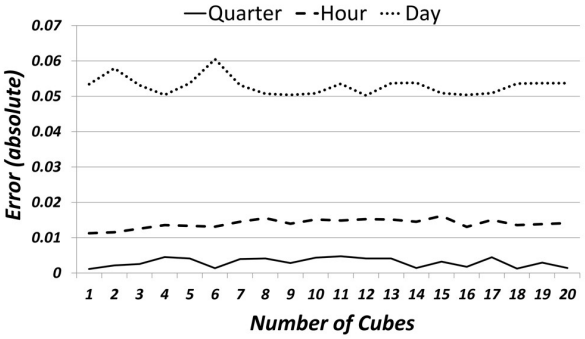


Fig. 24. Error Chart

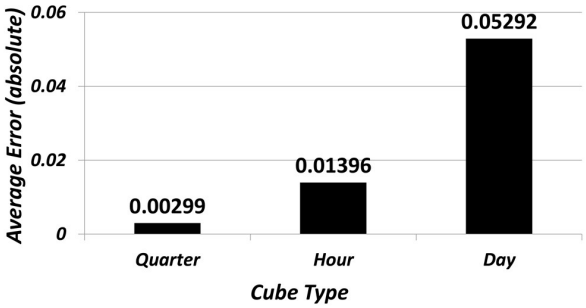


Fig. 25. Average Error

the average error values, given in Figure 25 for different cubes (Quarter, Hour, Day) are almost negligible in the use-case presented in this study.

7. Conclusion

The bottom line, for using SSN and WoT in Smart Environment settlement, in the approach presented in this article, it can be concluded that, 1) sensor data was successfully collected, transformed, encoded as RDF (linked data), enriched with meta-data and, published to LOD Cloud and, 2) multidimensional data cubes using EDWH ontology were chronologically and on-the-fly generated and, published to the LOD Cloud.

Specifically, while looking at the results, it can be safely concluded that on many occasions, for different results, the EDWH approach performs better than state-of-the-art RDF-DCV for most evaluation cases. It is obvious from the size evaluation that the cubes (Quarter, Hour, Day) generated for an overall timespan of one day have an affordable storage requirement. However, this can be further improved. Even while looking at the impact of increasing the number of dimensions, the storage size requirement is nominal.

Results collected during the QET evaluation conclude that, the results achieved by EDWH approach mostly has better results than RDF-DCV results with larger improvement margins. Specially, in case of QET for day cubes, the results obtained are 100 percent improved. After looking at performance charts, it can be concluded that the EDWH approach provides a faster way of generating data cubes on-the-fly in a real-time sensor network as opposed to RDF-DCV. However, the execution time for EDWH approach can be further improved by identifying different overheads and latency leaks which might be effecting the overall execution time.

Assessing the accuracy evaluation, it can be observed that the overall error values are minimum and are almost negligible, specifically, it is obvious from the chart showing the average error values. However, the negligence of these error values is dependent on the domain or application where the proposed methodology is realized. In the use-case presented here, it can be neglected, but in case of Healthcare and Life Sciences domain, a minor error in the values can be devastating.

Moreover, after testing our system with different event-types (for example, `HeatUsageEvent`, `WeatherEvent`), it can be said that the EDWH approach can serve as a generic approach towards generation of multidimensional cubes from linked sensor data in real-time SSN and WoT in Smart Environments.

7.1. Related Work

A plethora of works have looked at transforming data into RDF or generating multidimensional data cubes, but we have limited to report the related work which is closely related to the approach presented in this article. Focusing on publishing data while respecting linked data principles, [27] proposes an approach that produces RDF-based climate data. In their work, the authors define a new ontology for describing a specific data set containing temperature data. In this work, RDF Data Cube vocabulary and Semantic Sensor Network Ontology have been extensively used for generating and publishing aggregated data. However, contrary to our contribution, the authors propose a fixed set of dimensions as they are dealing with a domain specific data. In our case we grant users the right to select any number of dimensions and measures that suit their need.

It is also worth mentioning that our approach is used to generate on-the-fly data cubes from heterogeneous

sensors, while in [27], the authors discuss converting a 100 year homogenized daily temperature dataset into linked sensor data cube.

In [17], the authors discuss the need to semantically describe sensor capabilities for an optimized indexing of sensors in a particular environment. This contribution focuses exclusively on describing sensors using a custom developed ontology, however they fail in using any of the W3C standards such as the Semantic Sensor Network Ontology [10].

The proposed work in [25] introduces an approach to interact with an OLAP system using Microsoft's Kinect. In this work, AnduIN's event processing ability is used to detect gestures. Once gestures are detected, a query is formed and executed on a multidimensional data cube to define new and complex gestures using a star schema rather than RDF.

While we propose in our work to process event-data, transform it into RDF and then generate multidimensional cubes, in [28], the authors discuss a novel E-Cube model which combines techniques of complex event processing and online analytical processing for multidimensional event pattern analysis at different levels of abstraction.

In the paper [2], the authors discuss a system that uses Electronic Health Records (EHR) aggregated from different data sources for advancements on medical research and practice. Using this approach, the authors generate data cubes and store them in RDF format to support data analysis from a single place. In comparison to our work, the approach given in [2] uses a batch mechanism as opposed to real-time transformation of events and generation of data cubes.

While [18] proposes a sensor data management platform similar to ours, the main focus was rather on the predictive analytics part using both sensor and open weather data. Even though the data management part proposed here is quite similar to ours, the authors do not provide a solution for optimizing data storage such as the aggregations adopted in our work.

7.2. Prospective Work

During implementation, it was observed that even though triple stores have been favorite for persisting and managing RDF data [43], there are other NoSQL stores that can provide a better query performance and storage mechanism for RDF data. Therefore, researching techniques that support use of other types of databases and not just triple store can be a viable future work.

Event enrichment is an important activity in the approach presented in this article, therefore, investigating native approaches to event enrichment [23] and approximate semantic event-processing [22] techniques and determining how this would affect on-the-fly cube generation can also lead to a significant contributive work. Further improvements in the ontology, implementation and the methodology to generate data cubes on-the-fly in minimum time possible can also be a good research prospect. Apart from all this, the source code and packaged implementation of the work presented in this article will be made publicly available to research community.

Furthermore, as discussed in early stages of this article, the LOD Cloud comprises of billions of facts covering hundreds of datasets. In accordance with the Linked Data principles, these datasets are connected by a variety of typed links, forming an interlinked "Web of Data". The growing diversity of the Web of Data makes it more and more challenging for publishers to find relevant datasets that could be linked to, particularly in specialist domain-specific settings. Therefore, a set of baseline methods to automatically identify a list of public SPARQL endpoints whose content is deemed relevant to a local dataset based on queries generated from a local set of domain-specific keywords can be investigated. For this purpose, work on a method which can be used to extract necessary keywords from the local dataset (in this case, generated linked sensor data and data cubes) and another method that can utilize the extracted keywords to form simple queries to probe LOD datasets for relevance can also be a strong candidate for future work. The term simple queries used here implies queries which do not take a high amount of execution time and which are standard SPARQL queries.

Several query federation engines have recently been proposed for accessing public Linked Open Data sources. However, in some particular domain, resources can be sensitive and access can be tightly controlled by stakeholders: privacy is a major concern when federating such datasets. We are also working on SAFE: a query federation engine that enables decentralized, policy-aware access to sensitive statistical information represented as distributed RDF Data Cubes.

In our data management part, we are creating sensor observations in RDF without putting any design effort in defining newly generated URIs. A potential future work consists of applying important design patterns for generating "good" URIs [1]

References

[1] S. Abbas and A. K. Ojo. Applying design patterns in URI strategies - naming in linked geospatial data infrastructure. In *47th Hawaii International Conference on System Sciences, HICSS 2014, Waikoloa, HI, USA, January 6-9, 2014*, pages 2094–2103. IEEE, 2014.

[2] A. Antoniades, C. Georgousopoulos, N. Forgo, A. Aristodimou, F. Tozzi, P. Hasapis, K. Perakis, T. Bouras, D. Alexandrou, E. Kamateri, et al. Linked2safety: A secure linked data medical information space for semantically-interconnecting ehrs advancing patients' safety in medical research. In *Bioinformatics & Bioengineering (BIBE), 2012 IEEE 12th International Conference on Bioinformatics & Bioengineering*, pages 517–522. IEEE, 2012.

[3] G. Antoniou and F. Van Harmelen. *A semantic web primer*. MIT press, 2004.

[4] T. Berners-Lee, J. Hendler, O. Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[5] C. Bizer, R. Cyganiak, T. Heath, et al. How to publish linked data on the web. 2007.

[6] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22, 2009.

[7] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web (ldow2008). In *Proceedings of the 17th international conference on World Wide Web*, pages 1265–1266. ACM, 2008.

[8] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(4):247–267, 2005.

[9] M. Compton, P. Barnaghi, L. Bermudez, R. Garcia-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al. The ssn ontology of the w3c semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32, 2012.

[10] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al. The ssn ontology of the w3c semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2012.

[11] D. Cook and S. Das. *Smart environments: Technology, protocols and applications*, volume 43. John Wiley & Sons, 2004.

[12] D. N. Crowley, E. Curry, and J. G. Breslin. Leveraging social media and iot to bootstrap smart environments. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 379–399. Springer, 2014.

[13] E. Curry. System of Systems Information Interoperability using a Linked Dataspace. In *IEEE 7th International Conference on System of Systems Engineering (SOSE 2012)*, pages 101–106, 2012.

[14] E. Curry, S. Hasan, and S. O'Riáin. Enterprise Energy Management using a Linked Dataspace for Energy Intelligence. In *The Second IFIP Conference on Sustainable Internet and ICT for Sustainability (SustainIT 2012)*, Pisa, Italy, 2012. IEEE.

[15] E. Curry, J. O'Donnell, E. Corry, S. Hasan, M. Keane, and S. O'Riain. Linking building data in the cloud: Integrating cross-domain building data using linked data. *Advanced Engineering Informatics*, 27(2):206–219, 2013.

[16] R. Cyganiak, D. Reynolds, and J. Tennison. The rdf data cube vocabulary, w3c working draft 05 april 2012. *World Wide Web Consortium*, 2012.

[17] W. Derguech, S. Bhiri, S. Hasan, and E. Curry. Using formal concept analysis for organizing and discovering sensor capabilities. *The Computer Journal*, 2014.

[18] W. Derguech, E. Bruke, and E. Curry. An autonomic approach to real-time predictive analytics using open data and internet of things. In *2014 IEEE 11th International Conference on Ubiquitous Intelligence and Computing and 2011 IEEE 11th International Conference on Autonomic and Trusted Computing, UIC/ATC 2014*, 2014.

[19] D. Guinard, V. Trifa, F. Mattern, and E. Wilde. From the internet of things to the web of things: Resource-oriented architecture and best practices. In *Architecting the Internet of Things*, pages 97–129. Springer, 2011.

[20] D. Guinard, V. Trifa, and E. Wilde. A resource oriented architecture for the web of things. In *Internet of Things (IOT), 2010*, pages 1–8. IEEE, 2010.

[21] S. Hasan, E. Curry, M. Banduk, and S. O'Riain. Toward situation awareness for the semantic sensor web: Complex event processing with dynamic linked data enrichment. *SEMANTIC SENSOR NETWORKS*, page 60, 2011.

[22] S. Hasan, S. O'Riain, and E. Curry. Approximate semantic matching of heterogeneous events. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pages 252–263. ACM, 2012.

[23] S. Hasan, S. O'Riain, and E. Curry. Towards unified and native enrichment in event processing systems. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, DEBS '13, pages 171–182, New York, NY, USA, 2013. ACM.

[24] T. Heath and C. Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[25] S. Hirte, A. Seifert, S. Baumann, D. Klan, and K. Sattler. Data3–a kinect interface for olap using complex event processing. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on Data Engineering*, pages 1297–1300. IEEE, 2012.

[26] Y. Khan, M. Saleem, A. Iqbal, M. Mehdi, A. Hogan, P. Hasapis, A.-C. N. Ngomo, S. Decker, and R. Sahay. Safe: Policy aware sparql query federation over rdf data cubes.

[27] L. Lefort, J. Bobruk, A. Haller, K. Taylor, and A. Woolf. A linked sensor data cube for a 100 year homogenised daily temperature dataset. In *5th International Workshop on Semantic Sensor Networks (SSN-2012), CEUR-Proceedings*, volume 904, 2012.

[28] M. Liu, E. Rundensteiner, K. Greenfield, C. Gupta, S. Wang, I. Ari, and A. Mehta. E-cube: Multi-dimensional event sequence processing using concept and pattern hierarchies. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on Data Engineering*, pages 1097–1100. IEEE, 2010.

[29] D. C. Luckham. *The power of events*, volume 204. Addison-Wesley Reading, 2002.

[30] D. C. Luckham. *Event Processing for Business: Organizing the Real-time Enterprise*. Wiley, 2011.

[31] M. Mehdi, A. Iqbal, A. Hasnain, Y. Khan, S. Decker, and R. Sahay. Utilizing Domain-Specific Keywords for Discovering Public SPARQL Endpoints: A Life-Sciences Use-Case. In *ACM SAC (SWA track)*, 2014 (to appear).

[32] M. Mehdi, A. Iqbal, A. Hogan, A. Hasnain, Y. Khan, S. Decker, and R. Sahay. Discovering domain-specific public SPARQL

endpoints: a life-sciences use-case. In *18th International Database Engineering & Applications Symposium, IDEAS 2014, Porto, Portugal, July 7-9, 2014*, pages 39–45, 2014.

[33] M. Mehdi, R. Sahay, W. Derguech, and E. Curry. On-the-fly generation of multidimensional data cubes for web of things. In *Proceedings of the 17th International Database Engineering & Applications Symposium*, IDEAS '13, pages 28–37, New York, NY, USA, 2013. ACM.

[34] A.-C. N. Ngomo and S. Auer. LIMES – a time-efficient approach for large-scale link discovery on the Web of Data. In *IJCAI*, pages 2312–2317, 2011.

[35] S. Palmer. The semantic web: An introduction. 2007.

[36] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf. w3c recommendation, january 2008, 2008.

[37] L. Sanchez, J. A. Galache, V. Gutiérrez, J. M. Hernández, J. Bernat, A. Gluhak, and T. García. Smartsantander: The meeting point between future internet research and experimentation and the smart cities. In *Future Network & Mobile Summit (FutureNetw), 2011*, pages 1–8. IEEE, 2011.

[38] M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. In *The Semantic Web–ISWC 2014*, pages 245–260. Springer, 2014.

[39] F. Scioscia. Web of things.

[40] Z. Shelby, B. Frank, and D. Sturek. Constrained application protocol. *CoRE IETF WG, draft. shelby-corecoap-00*, 2010.

[41] A. Sheth, C. Henson, and S. S. Sahoo. Semantic sensor web. *Internet Computing, IEEE*, 12(4):78–83, 2008.

[42] K. Su, J. Li, and H. Fu. Smart city and the applications. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 1028–1031. IEEE, 2011.

[43] J. Team. Jena semantic web framework api.

[44] M. van Vliet, P. Ligthart, H. de Man, and G. Ligtenberg. Complex event processing. 2007.

[45] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk - a link discovery framework for the Web of Data. In *Linked Data On the Web (LDOW) Workshop*. CEUR, 2009.