Ollscoil na Gaillimhe
University of Galway

Provided by the author(s) and University of Galway in accordance with publisher policies. Please cite the published version when available.

Title	On-the-fly generation of multidimensional data cubes for web of things
Author(s)	Sahay, Ratnesh; Derguech, Wassim; Curry, Edward
Publication Date	2013
Publication Information	Muntazir Mehdi and Ratnesh Sahay and Wassim Derguech and Edward Curry (2013) On-the-fly generation of multidimensional data cubes for web of things IDEAS
Item record	http://hdl.handle.net/10379/4141

Downloaded 2024-03-20T11:22:36Z

Some rights reserved. For more information, please see the item record link above.

On-The-Fly Generation of Multidimensional Data Cubes for Web of Things

Muntazir Mehdi[+*]
m_mehdi10@cs.uni-kl.de

Ratnesh Sahay[+]
ratnesh.sahay@deri.org

Wassim Derguech[+]
wassim.derguech@deri.org

Edward Curry[+]
ed.curry@deri.org

[+]Department of Computer Science
Technical University Kaiserslautern
Germany

[*]Digital Enterprise Research Institute (DERI)
National University of Ireland, Galway (NUIG)
Ireland

ABSTRACT

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. This article specifically addresses the problem of adapting data model specific or context-specific properties in automatic generation of multidimensional data cubes. The idea is to generate data cubes on-the-fly from syntactic sensor data to sustain decision making, event processing and to publish this data as Linked Open Data.

Keywords

Multi-dimensional Data Cubes, Web of Things (WoT), Linked Sensor Data, Linked Open Data (LOD).

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 [14,15]. 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 [15]. 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 [21]. We observe a huge emergence of complex events based data within a smart environment like Smart Building, Smart Enterprises and Smart Cities. The backbone of these smart environments are sensors that produce huge amounts of data.

Smart environments are thus complex systems demanding an approach to manage and visualize data of their operations. Additionally, complex systems need to support real-time and effective decision-making [2]. The recent initiative of "Web of Things" 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 over the "Web of Things" 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.

In this paper, we propose a solution that generates multidimensional data cubes from incoming event (sensor) data. These cubes are then stored in a database and are published to the linked data cloud, therefore, supporting decision making and event processing on huge amounts of multidimensional event data. The approach is validated within a real world smart building. We use event-data that is generated from sensors deployed in the building to record parameters like (light, temperature, heat, power, etc.). This event-data is then enriched with necessary metadata using the W3C Semantic Sensor Network Ontology [7]. All necessary metadata for enriching the events is stored in a triple store and is retrieved via a SPARQL [17] end-point. We defined an ontology using general and basic concepts of W3C RDF Data Cube Vocabulary [5] for generating multidimensional data cubes. Therefore, the main motivation behind our work is to generate multidimensional data cubes on-the-fly for WoT or Semantic Sensor Networks to support decision making in smart environments.

The remainder of the paper is structured as follows: Section 2 describes our motivations while describing our use case scenario and introduces some of the required concepts used in this paper. Section 3 describes how event-data is captured, enriched, transformed into Resource Description Framework (RDF) [16] and published. Section 4 details how do we generate the multidimensional data cubes. Section 5 reports on the evaluations that we conducted over our developed system. Before concluding in Section 7, we oppose our contribution to other related works in Section 6.

2. MOTIVATION AND BACKGROUND

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 case can be a smart building within a energy management application to control supply and demand of energy. In this context we propose an approach that supports energy related decision making on a real world use case realized in the Digital Enterprise Research Institute (DERI).

The building has been retrofitted with energy sensors to monitor the consumption of power 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 [4] with a sensor network-based situation awareness scenario presented in [7].

An Event processing technique can be applied to process this real time sensor information to support the energy management applications [3]. However, it has been observed that when it comes to event processing, there is a limited support to decision making, data mining and knowledge discovery [13]. 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 generated by these sensors would increase accuracy of results.

In order to build an efficient energy management systems in such context, two main challenges need to be handled: *heterogeneous* and *big* data management.

2.1 Heterogeneous data management

Sharing information and data itself 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 or event data into RDF and publishing it into linked open data would support applications relying on combined data.

The approach we provide not only transforms sensor data into RDF, but also it generates contextual multidimensional data cubes and publishes them to the linked open data cloud.

Linked data is 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 [10].

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. The LOD cloud shown in Figure 1, is used by research and scientific communities in different domains [2].

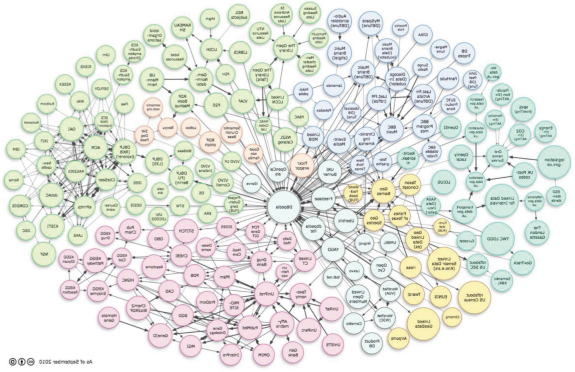

Figure 1: The Linked open data cloud [2]

2.2 Big data management

Since event processing involves providing an immediate response over a huge amount of data, a system that provides fast response to complex data retrieval queries is needed. To overcome this obstacle, we propose in this paper a system that uses a data-warehouse for managing multidimensional data cubes.

Contrary to a general database management system, data-warehouses are large stores of information containing historical data; they provide a multidimensional view of the data they contain. The main purpose behind keeping historical data is to support decision making, knowledge discovery, identification of hidden patterns and data mining.

The multidimensional shape of data inside data-warehouse is also referred to as cubes. A cube structures information into dimensions and facts. The dimension characterizes and represents a context for the analysis of facts (e.g. location, type, time) and a fact contains interesting measures (e.g. power usage, electricity usage) [5].

The vision that we adopt in our work is similar to the one proposed by the Web of Things (WoT). The realization of WoT requires 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 our use case we are limited to sensor data deployed within the DERI building. However, our approach can be easily extended to consider other smart devices.

The power of WoT comes from light-weight HTTP servers embedded within devices to enable the use of HTTP as an application protocol rather than a transport protocol [6]. In our scenario we use CoAP (Constrained Application Protocol) which is built on top of HTTP. CoAP is a Web transfer protocol that provides a request/response model for interaction between endpoints [18].

3. PROCESSING META-DATA FOR CUBE GENERATION

This sections describes the process that we follow to prepare sensor data for the creation of multidimensional cubes.

3.1 Capturing and Publishing Sensor Data

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 2. Components of this process are briefly described in the remainder of this sub-section.

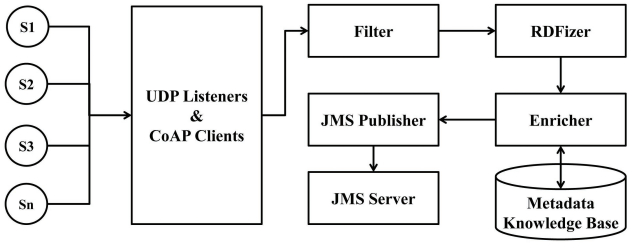

Figure 2: Process for Capturing and Publishing Sensor Data

3.1.1 Sensors, Listeners and Clients

S1, S2, S3 ... Sn, depicted as circles in figure 2 represent the sensors deployed within the building. The sensed data streams are retrieved by the listeners and clients. In this paper, we will use the terms "sensed data stream", and "event-data" interchangeably. Most of these sensors generate event-data at a frequency of between $1-2$ events per second. In order to capture the information from sensors, we have developed UDP (Universal Datagram Protocol) listeners and CoAP clients.

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* or *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

3.1.2 Filter: Cleaning Event Data

The event-data collected by UDP listeners or CoAP clients is forwarded to Filter component. Sometimes incoming event-data contains some irrelevant or unnecessary information in the form of noise, for example: empty values, repeating characters, unknown characters, etc. The filter component uses string processing techniques (string or character elimination or replacement) for cleaning this event-data and filtering irrelevant information. For example, filtering the previously introduced raw event-data of Listing 1 consists of: concatenating SIC and 'Channel', conversion of Date and 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

3.1.3 The RDFizer

The RDFizer component is responsible for converting and representing the event-data into RDF data. The RDF data model requires data to be structured in the form of triples where, a triple is a set of Subject, Predicate and Object {S, P, O}. Furthermore, the RDF data model requires that each subject and predicate must 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. Listing 3 shows the event-data shown in listing 2 converted to RDF and represented in N3. In this RDF graph, the main subject is *sensorReading*, with predicates *sensorIdentifier, sensorType, sensorReadingType, timestamp, unit* and *value*.

```
@prefix r1: <http://energy.deri.ie/../sensorReading#>.
@prefix rdf: <http://www.w3.org/../22−rdf−syntax−ns#>.
@prefix xsd: <http://www.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.2., type of the consumer e.g., database server, Web server or 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 event Enricher is the most important and critical component in the process of making data useful for generating data cubes. 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. This enrichment is done using concepts in W3C Semantic Sensor Network Ontology.

Some of major types of events that we deal with in our scenario are ElectricityUsageEvent (i.e., an event recording values of electricity usage), HeatUsageEvent (i.e., an event recording values of heat usage), WeatherEvent (i.e., an event recording values of temperature), PowerConsumptionEvent (i.e., an event containing information on recorded and sensed values of power consumption) and DeviceStateChangeEvent (i.e., 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 1, instance of one type of event after being enriched with meta-data information about consumer, consumerLocation, consumerDepartment and consumerType has RDF:Type *events: PowerConsumptionEvent*. Listing 4 shows an instance of such generated event data, serialized in N3.

```
@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 JMS Server and JMS Publisher

Apache ActiveMQ [19] is one of the most powerful and commonly used open-source message oriented middleware to support JMS. We use ActiveMQ as a JMS server to publish RDF event-data. The JMS Publisher component deals with publishing event data on JMS Server. In our scenario, we use publish/subscribe protocol to deal with event-data. The event-data, which has been filtered, converted into RDF and enriched with meta-data, is published on ActiveMQ JMS topics. These JMS messages are then broadcasted to components that subscribe to those topics.

3.2 EDWH Ontology Overview

Due to lack of vocabularies that bridge W3C Semantic Sensor Network Ontology and W3C Data Cube Vocabulary together [12], we define an ontology that uses concepts of W3C RDF Data cube vocabulary to generate Data Cubes. Mainly due to the servicing characteristics of this ontology, we call it EDWH (Event Data Ware-House) Ontology. The Class hierarchy of the ontology is shown in figure 3.

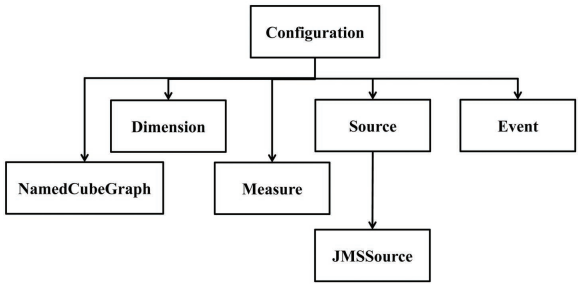

Figure 3: Class Hierarchy of EDWH Ontology

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.

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) [5]. In our case, an observation symbolises 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 [5] (e.g., powerUsage).

The third child class of Configuration is *Event*, which is responsible to capture 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.

NamedCubeGraph 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 of source of event data. Source is a subclass of configuration and is a super class of *JMSSource* class. JMSSource class is actually a specialized class of Source class to serve specific use of JMS Server.

3.3 Capturing Metadata

As mentioned before, sensors deployed in the DERI building sense different types of values, e.g., heat, temperature, power consumption, etc. Each of the sensed data values, also known as event-data, represents a different type of event. In the proposed approach, to generate multidimensional data cubes of any particular event type, an event registration is required. Specifically, the registration of an event in the system symbolises capturing the necessary information 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 4. 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 next step the user specifies the URL of the JMS server, the topic that is broadcasting events, the username and the password for accessing the server. Since there can be different types of events broadcasted on the same topic on the same server, the next thing is to select the event type for which data cubes must be generated.

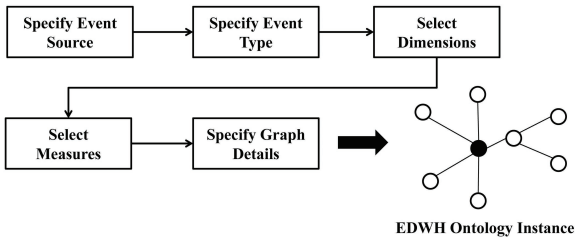

Figure 4: Event Registration Process

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 form of RDF 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 set of dimensions from set P. After selecting dimensions, the user selects a set of measures. The set of dimension 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 the dimensions and measures, the user also has to specify intendedObjectURI and desiredObjectURI properties of each dimension or measure. An example selection of dimensions and measures for PowerConsumptionEvent can be seen in figures 5 and 6. Both intendedObjectURI and desiredObjectURI are data properties of dimension and measure classes of the ontology mentioned in subsection 3.2. And both of these data properties contain values of the predicates. In the example shown in figure 5, the user has selected *consumer, consumerType* and *consumerDepartment* as 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 5 and 6 URIs are represented using circles and literal values are depicted by rectangles. In figure 6, 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 user to specify graph details. This graph detail is used to store generated data cubes. While inserting values for graph details, the user specifies name, detail and comments. The name of the graph is used to create a named-graph in triple store. After getting all necessary information from the user for registering an event is completed, the system will generate an instance of EDWH

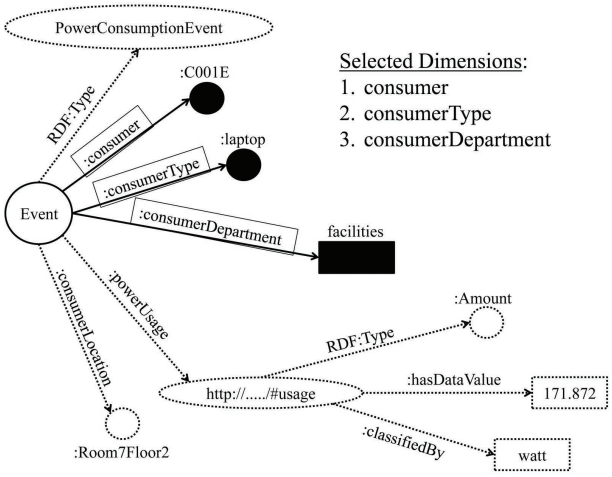

Figure 5: Example of selected dimensions

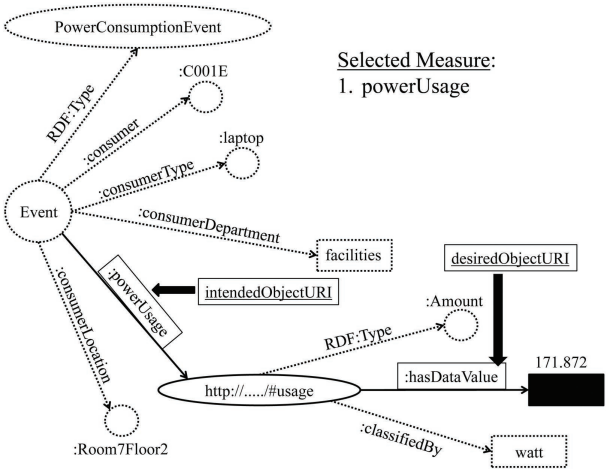

Figure 6: Example of selected measures

ontology. A complete instance of the ontology contains the required information on the selected dimensions, measures, graph detail and source of incoming events.

Once generated, EDWH ontology instance is stored in triple store. Triple store containing information of registered events is called "Registered Events Knowledge Base (REKB)". The REKB is used by EDWH to retrieve registered events and their corresponding configurations. Creation and storage (in triple store) of multidimensional data cubes is responsibility of an EDWH (Event Data WareHouse) agent. An EDWH agent is a software agent that is introduced in section 4.

4. CUBES GENERATION

The generation of multidimensional cubes is the core element of our work. Multidimensional data cubes provide support to decision making tools and applications to run complex queries on historical data. In our system implementation, we have defined an agent called the EDWH agent. 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 depicted in figure 7.

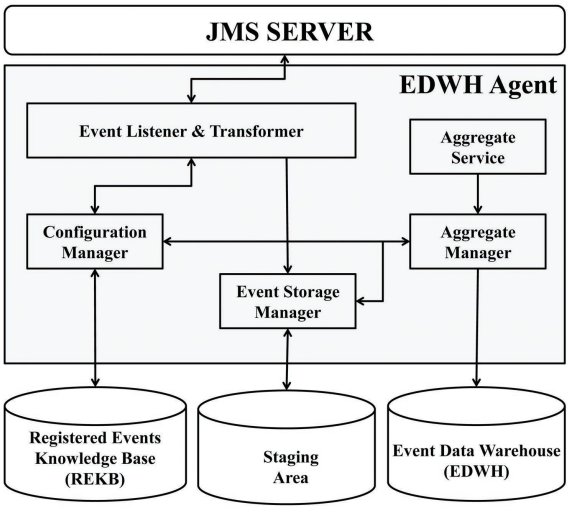

Figure 7: EDWH Agent structure

In order to generate aggregates on data and create multi-dimensional 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 all registered events from Registered Events Knowledge Base (REKB) and their corresponding configurations. 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 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 are also considered. The transformation of an event is done by adding only selected dimensions and measures values; 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' and 'PowerConsumptionEvent'.

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

:Observation−20c4c408 a dr:Observation;
  a 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 operations on the staging area. These operations are: insert, update and retrieve. 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 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. The "Aggregate Service" component triggers cube creation routine in "Aggregates Manager". The service executes each 15 minutes (quarter), hour, day and month. And 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 as EDWH and staging area. The hour cubes are generated from data of quarter cubes. A day cube is generated from hour cubes 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 8 and its corresponding RDF graph is shown in figure 9.

The example cube that is shown in figure 8 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 9, the main subject of the RDF graph is "Cube" with RDF:Type *:Cube* and RDF:Type *:PowerUsageEvent*. A cube is composed of several observations, as depicted in figure 9, 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.

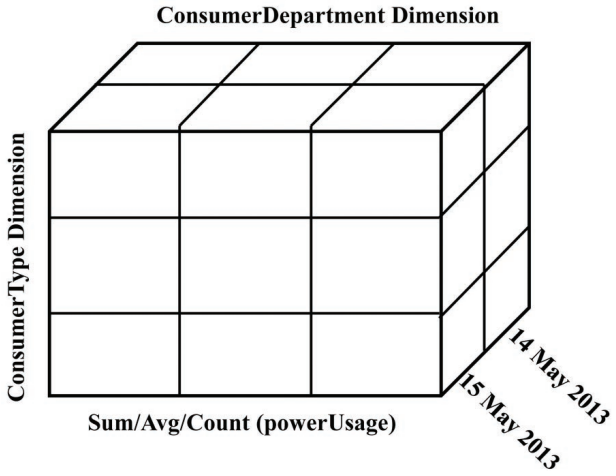

Figure 8: An example multidimensional cube

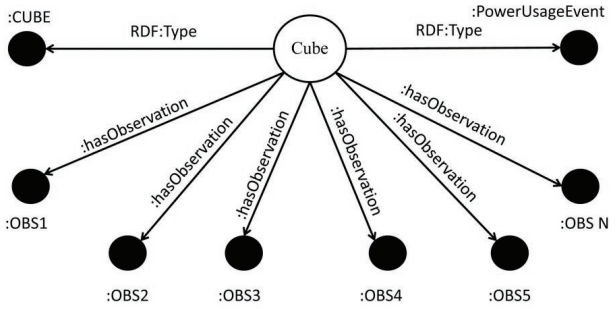

Figure 9: An example RDF graph of cube

```
:20c4c408 a cube, do: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

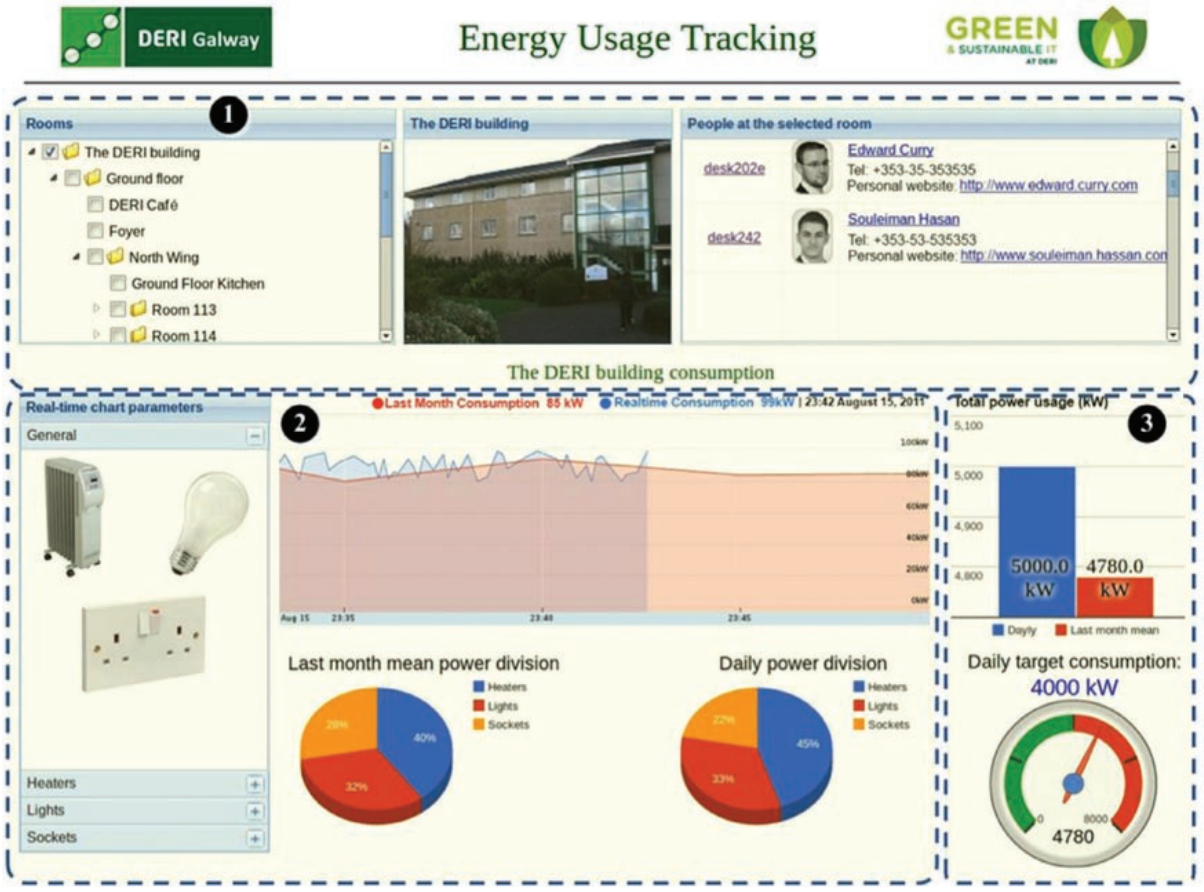

Figure 10: Building energy explorer dashboard [3]

Once cubes are generated, they are stored in EDWH and are published to the linked building data cloud [4]. The linked building data cloud for Digital Enterprise Research Institute (DERI) building has been implemented using the Linked dataspace for Energy Intelligence (LEI) [3] 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 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 10, 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 10 (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, 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.

5. EVALUATION

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 for this evaluation a data collection of events observed over an entire day. We conducted two quantitative evaluations related to the required storage size and execution time for storing event data and generating data cubes.

For the first part of the evaluation, we refer to Table 1. This table contains the results that we gathered while performing the size evaluation for the event data and data cubes. Each entry in Table 1 represents the size of RDF data serialized in N3 notation and encoded as UTF-8 with respect to the different types of cubes generated. 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 of between 0.093 and 0.098 KB. As the power consumption events are gath-

Table 1: Cube Storage Size and Count

		Raw Observation	Quarter Cube	Hour Cube	Day Cube
Number of Cubes	**Avg**	—	4/hour & 96/day	24	1
	Sum	—	4/hour & 96/day	24	1
	Count	—	4/hour & 96/day	24	1
	Sub-Total Per Day	—	12/hour & 288/day	72	3
	Total Cubes	**363** Per Day			
Storage Size	**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			

ered by listener and client component at a frequency of between 70 and 90 events/minute we wanted to determine the impact of the number of dimensions created. To do so, we registered the same event (PowerConsumptionEvent) with different number of dimensions. A chart showing the impact of changing number of dimensions for PowerConsumptionEvent event is shown in figure 11.

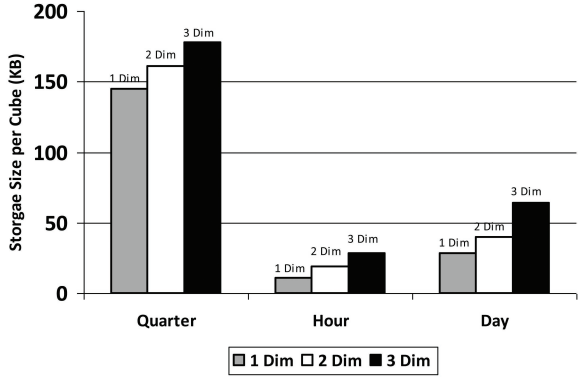

Figure 11: Size Chart for different No. of dimensions

For the second part of the evaluation, we observed the required query execution time (QET) and the performance of our system during the generation and storage of data cubes. Indeed, we defined a set of most commonly used queries in our scenario and executed them on different types of cubes stored in the EDWH and raw observations stored in the staging area. The type of queries that we executed are shown in table 2. The comparison of the recorded values for QET are shown in table 3.

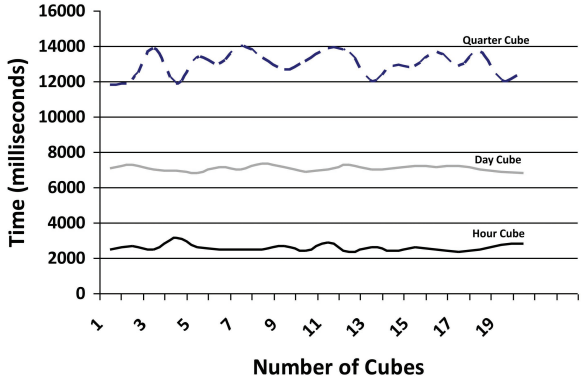

Figure 12: Performance Chart

In this evaluation, we use the term performance referring to the time the system takes to process one particular type

Table 2: 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.

Table 3: 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

of cube, i.e. time it 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 12. 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 12, the time taken to process a quarter cube is between 12 and 14 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.

6. RELATED WORK

Focusing on publishing data while respecting Linked Data principles, [12] propose an approach that produces RDF-based climate data. Here the authors define a new ontology for describing a specefic data set containing temperture 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, authors propose a fixed set of dimensions as they are dealing with a domain specific data. Recall, in our case we grant the user the right to select any number of dimensions and measures that suit his need.

It is also worth mentioning that our approach is used to generate on-the-fly data cubes from heterogenous sensors, but in [12], authors discuss converting a 100 year homogenised daily temperature dataset into linked sensor data cube.

The proposed work in [11] 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, trasform it into RDF and then generate multi-dimensional cubes, in [13], 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 [1], the authors discuss a system that uses Electronic Health Records (EHR) aggregated from different data sources for advancements on medical research and practice. In this approach, 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 [1] uses a batch mechanism as opposed to real-time transformation of events and generation of data cubes.

7. CONCLUSIONS AND PERSPECTIVES

With the approach presented in this paper, we were able to enrich events with necessary meta-data, and process enriched events to generate on-the-fly data cubes.

After looking at performance chart depicted in figure 12, it is safe to conclude that our approach provides a good way of generating data cubes on-the-fly in a real-time sensor network. Apart from this, after testing our system with different types of events, we believe that our approach can serve as a generic approach towards generation of multidimensional cubes from linked sensor data in real-time sensor network environment.

During implementation, we observed that even though triple stores have been favorite for persisting and managing RDF data [20], there are other NoSQL stores that can provide a better query performance and storage mechanism for RDF data. Therefore, we are researching on techniques that support other types of databases and not just triple store. Event enrichment is an important activity in our approach, therefore, we are also investigating native approaches to event enrichment [9] and approximate semantic event-processing [8] techniques and determining how this would effect on-the-fly cube generation. We are also working on improving the ontology, implementation and our methodology to generate data cubes on-the-fly in minimum time possible. Apart from all this, we are also working on making our source code and packaged implementation publicly available to research community.

Acknowledgment

This work has been funded by Science Foundation Ireland under grant number SFI/08/CE/I1380 (Lion-2). The authors would like to thank Mr. Souleiman Hasan and Mr. Yongrui Qin for their technical help and support during the implementation of the system. The authors would also like to thank Prof. Dr. Stefan Dessloch and Mr. Weiping Qu for their additional motivation and supervision during the conceptualisation and the implementation of the system.

8. REFERENCES

[1] 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*, pages 517–522. IEEE, 2012.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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*, pages 1297–1300. IEEE, 2012.

[12] 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.

[13] 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*, pages 1097–1100. IEEE, 2010.

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

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

[16] F. Manola, E. Miller, and B. McBride. Rdf primer. *W3C recommendation*, 10:1–107, 2004.

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

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

[19] B. Snyder, D. Bosnanac, and R. Davies. *ActiveMQ in action*. Manning, 2011.

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

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