Generating Cubes from Smart City Web Data[*]

Michael Scriney[1]
michael.scriney@insight-centre.org

Martin F. O'Connor[2]
martin.oconnor@ittdublin.ie

Mark Roantree[1]
mark.roantree@dcu.ie

[1]Insight Centre for Data Analytics
School of Computing, Dublin City University
Glasnevin, Dublin 9, Ireland

[2]Department of Computing
Institute of Technology Tallaght
Dublin 24, Ireland

ABSTRACT

A smart city necessitates the incorporation of data sources from multiple providers and data formats. Similar to the Internet Of Things, this data is primarily obtained from web streams producing XML or JSON data. Various combinations of data obtained from different providers can be used to enhance the lives of citizens with respect to different characteristics such as transport, city planning, the environment and housing. However, data provided from these streams is not necessarily in a format suitable for analysis and OLAP queries, despite the fact that these streams often provide measures and some elements of dimensionality often found in OLAP queries. In this research, we present a StarGraph construct which is designed to import web generated data streams and automatically generate the cube format necessary for OLAP queries. Our validation shows how the data streams can be captured as StarGraphs and using a traffic data case study, demonstrates an efficiency for populating and updating the data cube.

CCS Concepts

•**Information systems** → **Data model extensions; Data warehouses; Data mining;** *Enterprise application integration tools;* •**Applied computing** → IT governance;

Keywords

Data Cubes, OLAP, Smart City

1. INTRODUCTION

The Web continues to provide new opportunities for sharing information from many varied applications and devices. A recent initiative has seen the emergence of the exchange and integration of knowledge from different government and infrastructural services. These projects have become known as Smart City applications [12]. A smart city utilizes the vast amount of data provided from a wide range of services to enhance the quality of life of its population. The data generated by a city can vary widely, from domains such as transportation, the environment and social data. A smart city encompasses many different data sources in order to enhance services such as city planning, policing and energy management.

Various combinations of the generated data can be used to address numerous problems in the city. For example, the combination of environmental data with transportation data may permit the examination of the carbon footprint of a city and possibly facilitate a means of addressing such problems. Each data source is produced independently and as such conforms to its own isolated schema and is published on the web generally in XML or JSON as a data stream. While the unification of data from heterogeneous sources is a complex task and domain-dependent, there are well understood approaches to tackling data integration issues [9]. More recently, XML mapping technologies have been studied in terms of processing and transforming XML data [20] [19] and XML sensor streams [18] with solutions to optimise XML views [13] and efficiently update XML [16]. However, none of this research looked at the generation of OLAP-ready cubes from the XML data sources. Enhancing data warehouses with web generated XML data was presented in [15] but again, the schema design (facts and dimensions) was driven by existing enterprise databases. In [8], the authors provide an efficient cube maintenance system based on relational data and this was extended for XML data in [17]. While this work can potentially integrate multiple XML cubes, it is based on a static cube and query set while our approach is to manage new and updating datasets coming from smart city sources.

1.1 Motivation and Case Study

One component of this research involves the processing of a data stream containing the travel times for all motorways (provided by Transport Infrastructure Ireland); extract the travel times between each pair of recorded locations (at a specified interval, currently every 5 minutes); and create data cubes to allow for queries across different dimensions and at different levels of granularity.

[*]This work is supported by Science Foundation Ireland under grant number SFI/12/RC/2289

ACSW '17, January 31-February 03, 2017, Geelong, Australia
© 2017 ACM. ISBN 978-1-4503-4768-6/17/01...$15.00
DOI: http://dx.doi.org/10.1145/3014812.3014863

Example 1.1.
```
{
"M7_eastBound": {
  "data": [
    {
      "status": "U",
      "distance": 1772,
      "from_name": "J9 Naas North",
      "current_travel_time": 59,
      "to_name": "J8 Johnstown",
      "free_flow_travel_time": 53
    }
]
}
```

The motivation for this work is best exemplified using these case studies which aims to provide smart routing for emergency responders. Due to the nature of their work, it is imperative that an emergency response vehicle reaches its destination in the fastest time possible. However, the time it takes to cross a road network from point A to point B can vary wildly due to the number of cars or incidents on the road network. The end result is an increase in travel (response) times and in instances where the delay may be significant, an alternate route is crucial.

In example 1.1, the first key in the JSON file `M7_eastBound` states the route taken. There are many objects listed under the *data* array but for this example only one is provided here for the sake of brevity. In this instance, it states that the time taken to travel between junctions 9 and 8 is 59 seconds which covers a distance of 1.7Km and the optimal time is 53 seconds. In this case, the aim is to determine the minimum travel times between two points using the *time of day* dimension. A combination of historical data coupled with the up to the minute data stream is utilized to determine the most efficient route for emergency responders. Details of the data stream and processing times are provided in our evaluation in Section 5.

1.2 Contribution

In this paper, we introduce the *StarGraph* as a multi-dimensional data model to represent web generated datasets. Our contribution lies in the methodology for transforming web sources (XML, JSON) into the StarGraph model. In effect, this delivers the automatic creation of any web source into a canonical multi-dimensional model which is then in a *cube ready* view definition. Our evaluation is twofold: firstly, we use multiple smart city sources to demonstrate the extent to which the StarGraph transformation algorithm absorbs the data stream; and secondly, we demonstrate the efficiency at creating and populating data cubes using our traffic monitoring case study.

1.3 Paper Structure

The paper is structured as follows: Section 2 outlines our related research; Section 3 provides a high-level overview of our system to construct a StarGraph from a smart city data source; Section 4 describes in depth the process to construct a StarGraph and its resulting fact table; Section 5 presents our evaluation of the construction of a smart city data cube; and finally, in Section 6 we provide our conclusions and outline our future work.

2. RELATED RESEARCH

At present most of the work on data integration for data warehousing focuses on the integration and analysis of XML-centric data.

In [10] the authors propose a means of storing XML data in an existing data warehouse using X-warehousing. They present a method which uses XML as the logical model of a data warehouse. In order to integrate XML documents into the data warehouse, the XML logical model and the XML document to be integrated are converted into attribute trees which are then pruned and grafted. However, this approach requires a user to specify a schema whereas our approach provides an automatic means of constructing a schema.

[14] proposes a system whereby dimensions can be derived from semi-structured data. The system takes in semi-structured data (e.g. a Twitter stream) and first maps it to a pre-defined relational model. The semi-structured data is then mined for additional data such as entities that are being mentioned in the text of a Tweet. However the authors do not present an automated system to initially discover dimensions in heterogeneous semi-structured data.

In [7] the authors present a new multidimensional model, the diamond model, to store semi-structured XML data in an OLAP system. The model is composed of three layers: the standard layer is composed of dimensions whose properties are obtained from the document structure; the semantic layer is the central dimension which adds semantics to the textual content derived from documents; and the document layer which is derived from documents with similar structures. Similar to our approach, the *fact* is derived from the dimensions extracted from the document. However, as the diamond model is created to analyse textual documents there is no established process for measure detection. They present a series of heuristic rules to determine dimensions, attributes and hierarchies based on functional dependencies between documents. We adopt a similar approach. However, we also employ a graph analysis methodology to discover and construct dimension hierarchies.

In [22], the authors present a system to design a data warehouse for semi-structured data. Similar to our approach a dependency graph is used to detail the individual elements. However a domain specific designer is needed in order to explicitly state the facts which are needed to construct the fact table unlike our automated approach.

3. AUTOMATING CUBE CONSTRUCTION

The process is composed of two main steps; StarGraph creation and database population.

Initially, a StarGraph is created from a single schema definition (XSD, JSON schema). Once the StarGraph is created it is analyzed and produces an empty star schema corresponding to the schema file provided to the system. Additionally, this process also produces a series of mapping rules which detail how the star schema should be populated with respect to data obtained from a data stream conforming to the provided input schema (XSD, JSON schema). Once the fact table is populated, data cubes can be created and analyzed.

3.1 Data Warehousing Core Concepts

We briefly review the core terminology so as to clarify what the terms mean and provide context for our work. A data warehouse is a database which manages a large amount of data. A data warehouse requires a database schema which details how the data is to be stored. One common data warehouse schema is a `star schema`. A star schema consists of a `Fact table` and a series of `Dimensions`. A `Fact table` represents a single 'fact', this can be an event or a transaction. A `Fact table` contains a `measure` (or series of measures), these are typically numerical data which constitutes the 'fact' (e.g. total cost of an order in a shop). Measures are supplemented by `Dimensions`, a Dimension contains data which supplements the fact (an example would be details about the shop in which an order was placed). Additionally, a measure cannot be a primary key of a dimension. A fact table contains foreign keys to a series of dimensions which provide additional information about the fact. Fact tables are used to construct `data cubes`. Data cubes [11] are a common means of aggregating data for further reporting and analysis. A data cube is created by calling some aggregation function (SUM,AVG etc..) on the measures residing in the fact table.

3.2 Transformation to StarGraph

The first step of the process is the transformation of a schema definition (obtained from the data provider) into our CDM (Common Data Model) called a StarGraph. A StarGraph is an enriched graph which represents the measures, dimensions and facts derived from the source schema. The StarGraph is then used to provide a series of mapping rules to convert data obtained from a stream into its constituent dimension, measures and facts and subsequently to construct a star schema. An overview of the process is shown in Fig. 1.

The framework takes input in the form of a schema definition (e.g. an XSD or a JSON Schema) which defines the data obtained from a real-time data stream.

3.3 Data Population and Cube Construction

The star schema generated from a StarGraph is used to construct an empty data warehouse with the dimensions, facts and measures. The mapping rules are used to populate the data warehouse with data obtained from a data stream. When data streams are introduced to the system, the mapping rules are analysed in order to extract the defined data and store them in the data warehouse. When the fact table is populated data cubes may be created and mined to analyse the smart city data. We have previously discussed the construction of data cubes for smart city data [21]. This paper focuses on the integration and generation of a data warehouse from which data cubes may be constructed.

3.4 Mapping Rules

The mapping rules produced by the StarGraph are dependent on the type of schema used to construct the StarGraph. For example, when an XSD is used to construct the StarGraph, the mapping rules generated are in the form of XPath expressions. Each expression is mapped to either an attribute of a dimension or a measure. When an XML data stream is introduced to the system, these XPath expressions are evaluated to extract the information necessary to populate the fact table.

Table 1 provides an example of the mapping rules produced by a StarGraph. The left hand `Source` column stores the XPath expressions used to extract the necessary data from the data stream. The right hand column specifies the location where the data should be stored in the star schema. For example, the attribute `id` for the node `child` corresponds to the column `id` in the table `Dimension1` .

Table 1: Example of the schema mappings produced by the StarGraph

Source	Target
`\node1\child\@id`	Dimension1.id
`\node1\description\text()`	Dimension1.description
`\node2\quantity\text()`	Fact_table.measure

4. WEB DATA TRANSFORMATION

In this section, we describe in detail the key components introduced in the previous section. We begin with a comprehensive description of the StarGraph and provide an overview of its creation process. We then discuss the identification of dimensions and measures. Finally, the construction of the empty fact table is presented.

4.1 StarGraph

Our graph is comprised of a set of nodes N and a set of edges E.

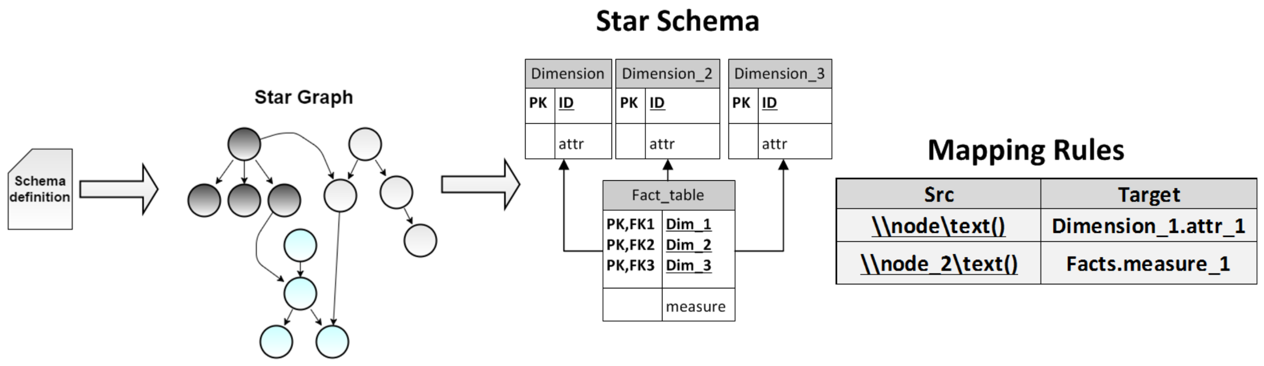
Each edge is a four-tuple $E = (X, Y, REL, I)$ where: $X, Y \in N$; REL is a type denoting the conceptual relationship which exists between the nodes X and Y. The possible values for REL are:

- 1-1, denoting a one-to-one relationship.
- 1-m, denoting a one-to-many relationship.
- m-m, denoting a many-to-many relationship.

The relationship type is derived by examining the schema structure and attributes. The type may be further constrained by specified indicators. For example, in an XSD data source, the relationship type may be constrained by the XML Schema indications such as `maxOccurs` and `minOccurs`. These schema indicators can be used to determine REL value (1-1, 1-m, m-m) of an Edge tuple. Lastly, the value I is a flag denoting whether or not the relationship is `implicit` or `explicit`. In brief, one-to-one relationships can be viewed as attributes (if implicit) and joins (if explicit), one-to-many relationships can be viewed as sub-dimensions (if implicit) and a one-to-many join (if explicit), many-to-many relationships can be viewed as a many to many join (if explicit).

An example of an `implicit` value is the hierarchical relationship that exists between two nodes in an XML document, this relationship is derived from the documents structure. An example of an `explicit` relationship would be a mapping using `key` and `keyref` elements in an XSD denoting a PK-FK (PrimaryKey-ForeignKey) relationship between two nodes which may have no relationship to each other inside the XML document structure. For example the tuple: $(Station, Name, 1-1, True)$ states that there is a relationship between the nodes `Station` and `Name`, they share a 1-1 relationship and that this relationship was derived from the document structure and as such `Name` can be viewed as an attribute of `Station`.

Conversely the tuple: $(DateTime, Scrapes, 1-m, False)$ denotes that there is an explicit 1-m relationship between `DateTime` and `Scrapes`, that there exists a one-to-many PK-FK relationship between these two values and as such,



Src	Target
\\node\text()	Dimension_1.attr_1
\\node_2\text()	Facts.measure_1

Figure 1: Construction of StarGraph from Schema Definition and the generation of Star Schema and Mapping Rules.

`DateTime` is a dimension with a defined primary key which must be referenced by any `Scrape`. However, in this instance, if the value I was set to true, it is regarded as an `implicit` one-to-many relationship and `Scrapes` would be viewed as a sub-dimension of `DateTime`

Each node $n \in N$ is a five-tuple node such that $n = (name, nodeType, source, dType, measure)$. *name* is the name of the node, *measure* is a flag indicating whether or not the node is a potential measure. *dType* is the datatype of the defined node. *source* is an indicator of where the particular data item is to be found in the schema. For example for an XML schema the source attribute is an XPath query detailing the location of the item and for JSON it uses dot notation. *nodeType* indicates the type of node, there are five possible types:

- `Dimension` marks a node which is the beginning of a dimension.
- `dimension_attribute` is a marker denoting that the node in question is an attribute of the parent dimension node.
- `container` indicates the node is an instance containing other nodes.
- `key-keyref` identifies the node type as a primary or foreign key relation.

The algorithm to assign the node types is detailed in Algorithm 1. The function call `isContainer` on line 3, detects whether or not a specific node is a container. The algorithm for detecting container elements is outlined in Algorithm 2.

Algorithm 2 Algorithm for Detecting Container Elements

```
 1: function isContainer(node)
 2:     if node.children.size!=1 then
 3:         return False
 4:     else if node.children.relType!=one-to-many then
 5:         return False
 6:     else if node.hasAttributes then
 7:         return False
 8:     else if graph.instance(node.children)>1 then
 9:         return False
10:     end if
11:     return True
12: end function
```

Algorithm 1 Algorithm for Assigning Node Types

```
 1: function setNodeType(node)
 2:     switch node do
 3:         case isContainer(node)
 4:             node.setType(Container)
 5:         case isKey(node)
 6:             node.setType(key-keyref)
 7:         case isDimension(node.parent)
 8:             if node.relType="one" then
 9:                 node.setType(Dimension-Attribute)
10:             else
11:                 node.setType(Dimension)
12:             end if
13:         case Default
14:             node.setType(Dimension)
15:     for child in node do
16:         setNodeType(child)
17:     end for
18: end function
```

The StarGraph is created by parsing an XSD or JSON schema. The root element of the XSD or JSON schema is removed and the remaining entities are stored as a set of disjoint subtrees. As the schema is parsed, each defined entity occupies a node in the subtree set and as such is assigned a `nodeType` attribute. At this stage all edges are deemed implicit as the relationships are derived from the structure of the schema. If the node is a top-level node it is deemed to be a `dimension`, a nodes which has a `one-to-one` mapping to a `dimension` node is deemed a `dimension_attribute`. In addition, any entities which have numerical data types are deemed to be potential measures.

Recall at this point we have a series of top level nodes as illustrated in Fig 2. In order to construct the StarGraph, it is necessary to identify existing relationships between nodes across disjoint subtrees. This is performed by resolving the primary-foreign key relationships previously identified during the initial processing of the schema. Once the container elements are removed we are left with a set of dimensions and their potential measures (Fig.3), from here we examine the `explicit` relationships defined in the schemas as primary and foreign key references. Each node which was marked as a `key-keyref` node is examined and linked to other nodes

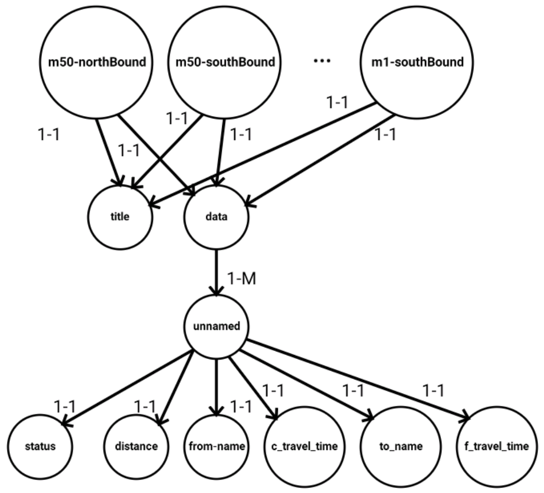


Figure 2: StarGraph with Containers.

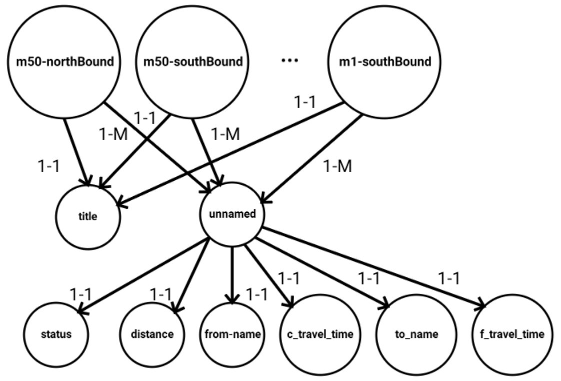


Figure 3: StarGraph with Containers Removed.

which share the same key. If the two nodes in question share a one-to-one relationship they are merged. Once the set of disjoint subtrees are linked across dimensions it becomes a StarGraph. As our working example does not contain any explicit primary-foreign key relationships we will illustrate this process with our bikes dataset. Fig. 4 shows a StarGraph created from our bikes dataset (XML). At this stage container elements are removed and we are left with a series of disjoint subtrees.

The finalised StarGraph with key-keyref edges added is illustrated in Fig. 5.

4.2 Dimension and Measure Identification

The next step of the process is the identification of facts and dimensions. All items marked as potential measures are evaluated. In the event that items are not deemed measures, they become dimension-attributes instead. Upon parsing the Schema definition, any numerical data types that are not defined as primary keys are considered to be measures.

Measure Detection. In our StarGraph (illustrated in Fig. 3 the identified measures are 'distance', 'current_travel_time' and 'free_flow_travel_time'. The graph is traversed to examine the number of dependencies for each

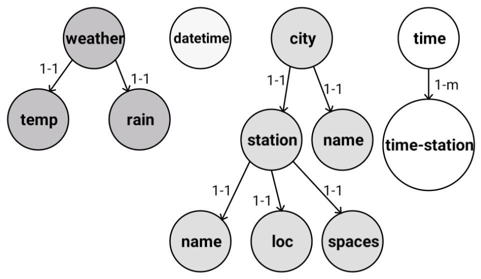


Figure 4: StarGraph created from bikes dataset with Containers Removed.

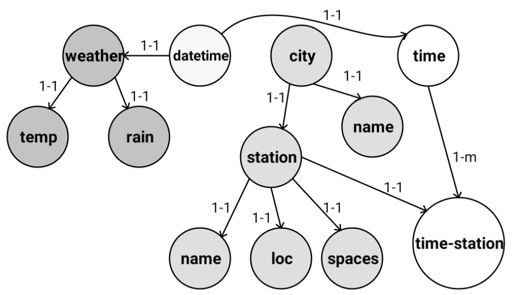


Figure 5: StarGraph with Explicit Links.

potential measure. Measures with the maximum number of dependencies are chosen as measures for the fact table, and the dimension tables are constructed. In a traditional fact table a measure item has the highest number of dependencies because it relies on the foreign keys of each dimension and subsequently the attributes of each dimension. In this instance all three potential measures share the same number of dependencies and as such will be grouped inside the same fact table.

4.3 Fact table Generation

Once all measures and dimensions are identified, a fact table schema is created. If a dimension has a defined primary key, it is used as a foreign key in the fact table. In the event an identified dimension does not contain a primary key, one will be generated for it. The src attribute for each node is extracted to provide a series of mapping rules which are used to extract the information from a data stream and populate the fact table. The final step in the creation of the fact table is the addition of a datetime dimension automatically created by the StarGraph. This addition is used to provide a timestamp for all facts which are introduced to the system. As a data cubes is being populated from a live streaming source, in addition to the facts and dimensions derived from the mapping rules generated by the StarGraph, the datetime dimension is automatically populated to denote the time at which the data was extracted from the streaming source.

5. EVALUATION

Our evaluation is twofold: firstly we discuss the creation of a StarGraph for each of our five data sources with respect to the facts, dimensions and measures derived; secondly we examine the performance of our system in populating a datacube created from our motorway traffic data.

5.1 Datasets

BikesData.

Dublin Bikes is a bike rental system located in Dublin city. Users may 'rent' a bike from one of its many stations located around the city for a maximum of 24 hours. In addition, they provide users of the system with a real-time service detailing the number of bikes available at each station. The data is published in XML [4] and enables users to plan where they may pick up or drop off a bike. The data published by this real-time service can be used to model the usages of this system throughout the city, additionally it can be used to predict the number of available bikes per station, and can also be used to recommend which stations require more bikes.

Ambient sound.

There are a number of sensors located throughout Dublin city which measure the ambient sound at 5 minute intervals [3]. The data can be used to estimate the density of areas with regards to population and transportation. It can be analysed to identify congested areas of the city as well as examine the environmental effects noise pollution has on the surrounding environment.

Motorway travel times.

The data is obtained from a json file which is provided by a near real-time data stream which is updated every 5 minutes [5]. Each file contains a list of each motorway coupled with a direction e.g. "M50 Southbound" and each one of these directions contains a list of the travel times between different junctions on that motorway at the time of scraping. An example is shown in Example 1.1.

For the distance between two junctions there are three measures; the distance between the two junctions in metres, the current time in seconds to get from junction A to B, and the optimal time taken to travel between the two junctions.

Car Park Data.

This data provides a near real time indication of the number of available parking spaces available in various car parks throughout Dublin city [6]. The data is published in XML. This data can be used by individuals to find a parking space in the city centre at any given time.

Air Quality Index.

This data provides an indication of the air quality at selected sites located across Ireland [2]. It provides a region of Ireland coupled with an air quality index as defined by the EPA [1].

5.2 StarGraph creation

The purpose of this section is to examine StarGraphs created from different datasets of different types (XML, JSON). Each StarGraph is examined with respect to the number of nodes and edges created in the graph, the number of dimensions, attributes and measures created in the fact table and finally the time taken to create the StarGraph. Table 2 outlines the performance of StarGraph creation for each schema.

The times taken for each StarGraph to be created were between 260ms for the CarPark dataset and 504ms for the Ambient Sound and Air Quality datasets. Overall the StarGraphs constructed from XML data completed faster than those derived from a JSON schema with both XML datasets completing in 260ms and all JSON datasets taking longer than 500ms to construct. This is most likely due to different parsers being required to examine and traverse an XSD and a JSON schema.

The complexity of the provided schema undoubtedly increases the time taken to construct a StarGraph. Ignoring the additional time taken to parse a more complex schema, the result would be a StarGraph with additional nodes and edges. This would undoubtedly increase the time taken to generate a fact table due to the time taken to traverse the graph to detect and remove containers and generate dimensions.

Motorway.

For the motorway dataset, 10 dimensions were created from the supplied data stream. At the highest level, the process extracted *motorway name* and *direction* (e.g. `m50_southbound`). Both dimensions linked to an object which contained two attributes: *title* which was the title of the object and *data* which was an array of JSON objects. As all of the top-level dimensions held a *1-1* relationship with the title attribute, it was classified as an attribute of each dimension. The `data` object was identified as a container as it was an array of smaller objects and as such, was removed. The `unnamed` object which was contained in the array was created as a **shared sub-dimension** as it held a *1-m* relationship with the top-level dimensions. The `unnamed` dimension contained three potential measures: `distance`, `current_travel_time` and `free_flow_travel_time`. As all of these measures shared the same number of dependencies, the algorithm includes them in the same fact table. The constructed fact table can be seen in Fig. 6. The StarGraph was created in 523ms.

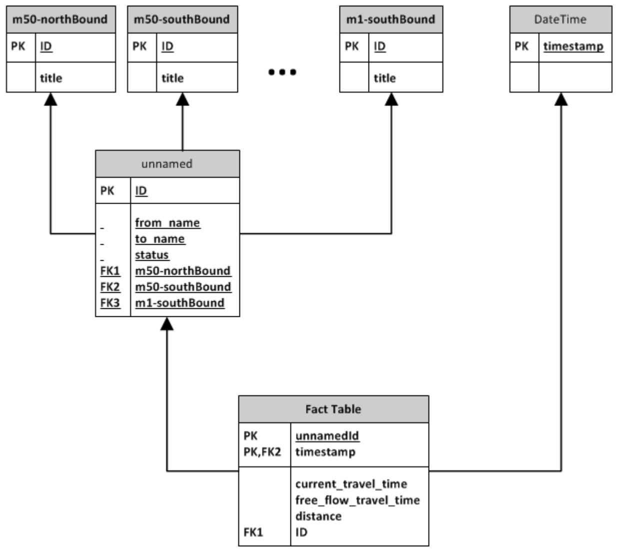


Figure 6: Fact table generated from the motorway dataset.

Table 2: StarGraph Construction Times for Different Datasets.

Dataset	Format	Schema size	Nodes	Edges	Dimensions	Attributes	Measures	Unused	Time
Bikesdata	XML	3.7KB	18	14	4	3	1	4	266ms
Ambient S	JSON	0.4KB	3	2	2	0	1	3	504ms
Motorway	JSON	1.2KB	17	26	10	4	3	2	523ms
CarPark	XML	2.3KB	7	6	5	1	1	1	260ms
Air Quality	JSON	0.4KB	5	2	3	2	0	1	504ms

Table 3: Mapping rules generated by BikesData

Source	Destination
`/city/@name`	City.name
`/station/@name`	Station.name
`/weather/temp`	Weather.temp

Table 4: Mapping rules generated by ambient sound dataset

Source	Destination
`date`	Date.val
`time`	Time.val
`aleq`	Fact_table.aleq

BikesData.

There were 4 dimensions identified in the BikesData dataset: datetime, city, station and weather. A dimension hierarchy was created between *city* and *station* as *city* and *station* hold a *1-to-many* relationship. Four container elements were identified and removed from the graph: weather-inf, datetimes, stations and data. The identified measure was `[time-station/@spaces]`. A sample of the mapping rules constructed can be seen in Table 3. The StarGraph was created in 405ms .

Ambient Sound.

The AmbientSound dataset provides two dimensions: `date` and `time`. A sample of the mapping rules for the AmbientSound dataset is presented in Table 4. Two dimensions were created one for `date` and one for `time` with one measure called `values`. There were three unused elements in this schema, each one being the container array element for each dimension and measure. The StarGraph was created in around 504ms.

Car Park.

For the Car Park dataset 5 dimensions were found. Of these one was a subdimension which contained an attribute called `name` which was the name of the carpark. One measure called `spaces` was found which was the number of available spaces at a carpark. The schema took the quickest time to complete in 260ms. This is due to both the fact that the schema was not very complex and was in XML format.

Air Quality.

For the Air Quality dataset 3 dimensions were found and one container element was found and removed. No measure was found for the Air Quality dataset due to the absence of numerical values found in the schema. Nevertheless a fact table was constructed with three dimensions. The StarGraph was constructed in 504ms.

5.3 StarGraph Cube Creation

Using the StarGraph created from the motorway schema, a corresponding link to a web service was provided to the system which provides data corresponding to the specified schema at 5 minute intervals. Each response from the web service was 9.4KB in size. Containing 10 dimensions, 4 attributes, 3 measures and 2 unused elements (Table 2). This data was collected at 5-minute intervals for one week resulting in a data-mart containing 12Mb of data and 120,000 individual facts. As the data is read from the webservice [5] it is first examined with respect to the mapping rules generated from the StarGraph. These are then used to extract the data and store them in a database (MongoDB) and create datacubes. The generated fact table schema can be seen in Fig. 6. A sample of the data provided by the web service can be seen in Example 1.1. For each response, the time taken to extract the data and store it in a database and update the fact table was around 1.5 seconds. However there are a number of factors which affect the speed of this operation, the first being the number of mappings generated by the StarGraph (these would undoubtedly increase with the complexity of a provided schema), secondly the number of items returned per-request from the web service would impact the time taken as a combination of a larger response, and a longer extraction process would increase the time taken to collect, extract and store the data.

6. CONCLUSIONS

In this paper, we introduced the concept of a StarGraph as a canonical data model for individual schema definitions. This facilitated the automatic generation of a star schema representation which is used to construct an empty data warehouse, and the automatic generation of a series of mapping rules specifying how to populate the data warehouse from a variety of data streams. Additionally, we demonstrated the process of StarGraph construction, fact table generation and the formation of mapping rules. Using the fact table and mapping rules derived from the StarGraph we created a data cube from a web service using the `Motorway` dataset.

Our future work involves the creation of multiple StarGraphs from 27 agricultural sources. From here these StarGraphs will be integrated to produce an agri-data warehouse and a series of integrated mapping rules which can be used to populate the data warehouse from multiple sources.

7. REFERENCES

[1] Air Quality Index [Online]. http://www.epa.ie/air/quality/index/. Accessed: 2016-08-04.

[2] Air Quality [Online]. http://www.dublindashboard.ie/pages/DublinEnvironment. Accessed: 2016-08-03.

[3] Ambient Sound [Online]. http://www.dublindashboard.ie/pages/DublinEnvironment. Accessed: 2016-08-03.

[4] Dublin Bikes API [Online] (originally available in XML, now only available in JSON). http://api.citybik.es/v2/networks/dublinbikes. Accessed: 2016-08-03.

[5] Mortorway Travel Times [Online]. https://dataproxy.mtcc.ie/v1.5/api/traveltimes. Accessed : 2016-08-04.

[6] Real Time Car Park Data [Online]. http://www.dublincity.ie/dublintraffic/cpdata.xml. Accessed: 2016-08-04.

[7] M. Azabou, K. Khrouf, J. Feki, C. Soule-Dupuy, and N. Vallès. Diamond multidimensional model and aggregation operators for document olap. In *Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on*, pages 363–373. IEEE, 2015.

[8] X. Baril and Z. Bellahsene. Selection of materialized views: A cost-based approach. In *International Conference on Advanced Information Systems Engineering*, pages 665–680. Springer, 2003.

[9] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM computing surveys (CSUR)*, 18(4):323–364, 1986.

[10] O. Boussaid, R. B. Messaoud, R. Choquet, and S. Anthoard. X-warehousing: an xml-based approach for warehousing complex data. In *Advances in Databases and Information Systems*, pages 39–54. Springer, 2006.

[11] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery*, 1(1):29–53, 1997.

[12] J. M. Hernández-Muñoz, J. B. Vercher, L. Muñoz, J. A. Galache, M. Presser, L. A. H. Gómez, and J. Pettersson. *Smart cities at the forefront of the future internet*. Springer, 2011.

[13] J. Liu, M. Roantree, and Z. Bellahsene. A schemaguide for accelerating the view adaptation process. In *International Conference on Conceptual Modeling*, pages 160–173. Springer, 2010.

[14] S. Mansmann, N. U. Rehman, A. Weiler, and M. H. Scholl. Discovering OLAP dimensions in semi-structured data. *Information Systems*, 44:120–133, 2014.

[15] J. M. P. Martinez, R. Berlanga, M. J. Aramburu, and T. B. Pedersen. Integrating data warehouses with web data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 20(7):940–955, 2008.

[16] M. F. O'Connor and M. Roantree. SCOOTER: A Compact and Scalable Dynamic Labeling Scheme for XML Updates. In *DEXA (1)*, pages 26–40, 2012.

[17] M. Roantree and J. Liu. A heuristic approach to selecting views for materialization. *Softw., Pract. Exper.*, 44(10):1157–1179, 2014.

[18] M. Roantree, D. McCann, and N. Moyna. Integrating sensor streams in phealth networks. In *14th International Conference on Parallel and Distributed Systems, ICPADS 2008, Melbourne, Victoria, Australia, December 8-10, 2008*, pages 320–327, 2008.

[19] M. Roantree, J. Shi, P. Cappellari, M. F. O'Connor, M. Whelan, and N. Moyna. Data Transformation and Query Management in Personal Health Sensor Networks. *J. Network and Computer Applications*, 35(4):1191–1202, 2012.

[20] M. Roth, M. A. Hernández, P. Coulthard, L. Yan, L. Popa, H.-T. Ho, and C. Salter. Xml mapping technology: Making connections in an xml-centric world. *IBM Systems Journal*, 45(2):389–409, 2006.

[21] M. Scriney and M. Roantree. Efficient cube construction for smart city data. In *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops 2016, Bordeaux, France, March 15, 2016.*, 2016.

[22] B. Vrdoljak, M. Banek, and S. Rizzi. Designing web warehouses from xml schemas. In *Data Warehousing and Knowledge Discovery*, pages 89–98. Springer, 2003.