© The British Computer Society 2018. All rights reserved.
For permissions, please e-mail: journals.permissions@oup.com
doi:10.1093/comjnl/bxy064

Automating Data Mart Construction from Semi-structured Data Sources

Michael Scriney[1]*, Suzanne McCarthy[1], Andrew McCarren[1], Paolo Cappellari[2] and Mark Roantree[1]

[1]*Insight Centre for Data Analytics, School of Computing, Dublin City University, Glasnevin, Dublin 9, Ireland*
[2]*Information Systems and Informatics, City University of New York, Staten Island, New York, NY 10314, USA*
[*]*Corresponding author: michael.scriney@insight-centre.org*

The global food and agricultural industry has a total market value of USD 8 trillion in 2016, and decision makers in the Agri sector require appropriate tools and up-to-date information to make predictions across a range of products and areas. Traditionally, these requirements are met with information processed into a data warehouse and data marts constructed for analyses. Increasingly however, data are coming from outside the enterprise and often in unprocessed forms. As these sources are outside the control of companies, they are prone to change and new sources may appear. In these cases, the process of accommodating these sources can be costly and very time consuming. To automate this process, what is required is a sufficiently robust extract–transform–load process; external sources are mapped to some form of ontology, and an integration process to merge the specific data sources. In this paper, we present an approach to automating the integration of data sources in an Agri environment, where new sources are examined before an attempt to merge them with existing data marts. Our validation uses a case study of real world Agri data to demonstrate the robustness of our approach and the efficiency of materializing data marts.

Keywords: data model transformation; semi-structured data; ETL; data marts

Received 30 September 2017; revised 29 March 2018; editorial decision 20 May 2018
Handling editor: Peter Wood

1. INTRODUCTION

The Agri industry is increasingly making use of data mining and analytics for decision support, predictions and preparing reports. The sector regularly has requirements to analyse trends in pricing of a specific Agri product or monitor production of various products. Increasingly, this requires integrating data from web-based sources with organizations' own internal databases in order to view them in a single, homogeneous view for querying and analyses. The end goal is to provide Agri decision makers with a degree of certainty in their reports, rather than relying on instinct.

Apart from enterprise data, there is a significant volume of both online and third party datasets upon which to base predictions. However, correct usage can be difficult for several reasons: (i) the data are coming from multiple sources and there may be heterogeneity in the metadata structure or in the data itself; (ii) the data are often semi-structured, making it unsuitable for loading to a Data Warehouse without significant manual effort on the part of data engineers; and (iii) integrating datasets often requires a domain expert to provide the context and data understanding. Solutions involve processes for cleaning, restructuring and integrating data, which are generally both expensive and time consuming. One of the primary aims in this area of research is to reduce both cost and the time taken to make data available by automating these tasks as much as possible.

1.1. Background and motivation

Integrating semi-structured data sources is not new as research into integrating semi-structured data with enterprise data has been ongoing for almost 20 years, eg. [1] and integration of fully unstructured sources in [2]. However, there is very little research into auto-constructing the types of multidimensional schemas needed for online analytical processing (OLAP) and data mining [3]. Traditionally, the process for constructing a

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

warehouse (or multidimensional) schema involves matching business requirements with existing enterprise databases to specify the data marts which will generate the required datasets. An extract–transform–load (ETL) process [3] will be put in place to continuously harvest data from operational databases. In previous work, we used the same process to create a large Agri warehouse model [4] although we integrated non-enterprise sources to meet specific business needs. As part of this work, a multidimensional data model was specified to map online sources to the various data marts represented in the Agri warehouse.

The problem with online Agri (in addition to many other) data sources is that they evolve much more than in-house databases. Sources can disappear or change structure, new sources can become available and this can have a widespread effect as different user needs require the construction of separate data marts. Approaches to clustering online data such as [5] can be effective when query changes require a modification to the materialized data mart but cannot process or integrate new online sources. In effect, a warehouse setup that comprises online data sources requires a methodology for analysing unknown or altered data streams to detect facts and dimension hierarchies with an ability to create data marts from multiple data sources.

1.2. Approach and contribution

While integrating data from heterogeneous sources is often complex and domain-dependent, the data integration issues are well understood [6]. More recently, XML mapping technologies have been studied in terms of processing and transforming XML data [7] and XML sensor streams [8]. While the enhancement of data warehouses with web generated XML data was presented in [9], the schema design (facts and dimensions) was driven by enterprise databases. In this paper, we present a system which automatically creates multidimensional data marts from Agri data sources. Our approach provides a means of automatically detecting the attributes that make up a data mart from within each source stream, converting these sources to a pure multidimensional graph mode and then, using an ontology, attempts to construct the required data mart by integrating the source data. This provides a significant benefit to Agri knowledge workers as it greatly reduces the time to construct data marts and can also be used to evolve data marts where the structure of source data streams has been modified. Note that in this paper, we use the term *streams* to refer to the actual data sources. This is because the data sources are updated routinely and can be volatile, with no guarantee that data remain available or that historic data are preserved.

The research on deep learning Agri data presented in [10] required the *manual* construction of cubes (data marts) from web sources. Our previous work [11] introduced the *StarGraph* model which automatically captures the multidimensional concepts of *facts* and *dimensions* from the online data streams. To our knowledge, no other work attempts to build data marts from web sources using an *automated integration* approach. The contribution of the research presented in this paper is as follows:

- As the StarGraph has a 1–1 relationship with external sources, an integration strategy was required to construct data marts from multiple sources. In this work, we present the *ConstellationGraph* (or *Constellation* for short) which represents integrated StarGraphs.
- We also present a methodology to automatically integrate (as far as possible) StarGraphs to complete the process of forming an integrated data mart from online data sources.
- Finally, we present an evaluation process which takes a real-world Agri user requirement—a data mart for predicting pig prices—and constructs the data mart from data sources selected by our end-user partners. We run three exercises as part of the validation: firstly, the end user manually constructs the data mart; the second exercise sees the integration process take place without the use of an ontology; and finally, an ontology-based approach is used to drive the auto-construction of data marts which match those constructed by the end-user.

1.3. Paper structure

This paper is structured as follows: in Section 2, we provide a discussion on the state-of-the-art; in Section 3, we describe our StarGraph model and its functions, and the methodology for creating data marts from online sources; in Section 4, we present the metadata and ontology parts to our system; as our case study involves real end-user requirements from data streams that are currently in use. We present a brief overview of these data sources and a report on how they were integrated into our system in Section 5; in Section 6, we provide a case study evaluation using a user-defined requirement; and finally in Section 7, we present our conclusions.

2. RELATED RESEARCH

In [12], the authors present an ontology-based approach to constructing a data warehouse. Two different graphs are used: the datastore graph and the ontology graph. These graphs are linked by a series of formally defined mappings provided by a designer. This approach ensures semantic completeness between the datastore and the ontology. Similar to our StarGraph, the canonical model is a graph model. However, the StarGraph performs graph restructuring which attempts to define facts and dimensions *before* an ontology is required. Additionally, in [12], once a data source has been converted into a graph called a `datastore graph`, it must then be manually linked by a designer to an ontology while our approach provides a StarGraph with term and type information from an ontology in order to provide a level of automation to our integration process.

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

A system which uses a global ontology to facilitate the ETL process and user querying is presented in [13]. The authors demonstrate how the system can be used to enrich user queries by providing an *ad hoc* approximation search based on the global ontology. This *ad hoc* querying mechanism allows for in-depth analysis which was not previously defined by a query designer. However, there is no focus on the larger ETL architecture while we provide a full methodology for importing new data sources.

An ETL process for OLAP on Linked Data is presented in [14]. This work introduces HSPOOL, a framework which provides a user with the means to perform OLAP analysis on Linked Data by extracting facts, measures and dimension hierarchies. However, a suitable OLAP schema must be constructed first in order to determine hierarchies from an ontology, while our approach generates a target schema automatically based on the sources presented to the system. In addition, while the authors use Linked Data to construct the OLAP cube, we instead use a lightweight ontology to supplement the integration process for a ConstellationGraph.

In [15], the authors present a unified cube: a combination of data obtained from a data warehouse and open Linked Data. Similar to our approach, a common data model is used to represent all data, which is then grouped into the unified cube. However, the integration process for linking cubes is controlled by a query designer, while our approach is semi-automatic; constructing mappings and identifying facts, dimensions and measures automatically for a source and using a global lightweight ontology to facilitate semantic integration.

Similar to our work, an automatic ETL system is presented in [16], in which researchers use an ontology to facilitate semantic data integration. The authors use a combination of a thesaurus derived from a starting data warehouse schema, a lexicon (e.g. WordNET) and clustering to determine similarity between attributes. However, in our approach, we use abstract types defined by our metamodel to determine candidates for integration and propose a suitable integration strategy. In addition, the authors' approach assumes an existing data warehouse and schema, while our approach does not require this significant overhead.

The authors in [17] present a semi-automatic approach for combining business requirements and data sources to create a multidimensional schema and a corresponding conceptual ETL process. All data sources are captured in an Web Ontology Language (OWL) ontology with corresponding mappings while the business requirements are stored as a series of structured XML files. When the system is presented with a new business requirement, the requirement is first validated then compared to the ontology of data sources to generate a multidimensional schema and ETL process. Similar to our approach, an ontology is used to facilitate semantic integration. However, the authors require an ontology detailing the mappings for each data source, while our process *automatically generates* mappings during the StarGraph creation phase. In addition, the authors' approach requires an ontology per data source (local ontology), while our approach utilizes one lightweight ontology to facilitate integration across multiple sources.

In [18], the authors present a method for creating a data warehouse from heterogeneous data. Each data source is provided with a corresponding ontology, which are subsequently connected through a global domain ontology. The system uses structured requirements to extract suitable Data Warehouse schemas from the global ontology, based on requirements provided by a user. However, the ontologies for each data source identify the mappings, and instance data provided are stored in a relational format. Instead, our approach adopts a Data Lake approach, which stores instance data in its raw format and thus provides a means of quickly detecting changes to the structure of schemas. In addition, the mappings for a data source in our system are generated automatically and a local ontology for each data source is not required.

In the work presented in [19], the focus is on constructing a data warehouse based on user requirements. Initially, the system is presented with a domain ontology to derive facts. These facts are then presented to the user who selects the facts they wish to use. Similar to the authors' approach, we attempt to identify facts, dimensions and measures from a data source prior to user interaction. However, while the authors' approach requires a domain ontology in order to determine facts, dimensions and measures, our approach identifies these properties automatically.

An automatic ETL approach facilitated by domain-specific modelling is presented in [20]. In this work, researchers use their own language (DSL) to represent different stages of the ETL process. A user (typically a domain expert), using DSL, outlines the concepts and operations for the conceptual ETL process. This process is then automatically deployed as a domain-specific ETL process. The authors use domain modelling to encapsulate the data sources and their interactions while our approach utilizes an ontology to facilitate this semantic integration. Our approach extends this work by identifying facts, dimensions and measures automatically, requiring limited human intervention when integrating sources to a data mart.

The authors in [21] present an ETL process which utilizes Resource Description Framework (RDF)/OWL ontologies to create OLAP data cubes from heterogeneous data sources. Each data source is provided with a corresponding ontology which is used to create RDF representations of the source data. The data are then extracted using Resource Description Framework (RDF) queries obtained from a cube definition represented as OWL. The authors use RDF queries to extract the data from each source, whereas our approach uses the data source's native format (XPath, JSON etc.) to extract the data. Additionally, our system automatically generates these mappings without the need for a pre-defined ontology bound to a data source and our ontology is used purely for assisting only in some integration scenarios leading to faster development and deployment.

Summary. When comparing our research with the state-of-the-art, our approach can identify facts, dimensions and measures from *unseen* data sources, where mappings from target

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

to source are automatically generated; we do not require an existing warehouse as our system *automatically constructs* the data mart based on the content of source data; we require only *minimal* user interaction and lightweight ontology in order to construct the final data mart.

3. BUILDING INTEGRATED DATA MARTS

Our approach uses a *StarGraph* to model each data source and a *Constellation* to model the integrated data mart. We will use examples from the data sources described later in Section 5 to provide details of actual transformation and the generated mappings. For the data transformations, the methodology comprises four main processes to manage different aspects of the overall transformation.

- Stream introduction (Section 3.2). This process adds a data source to a Data Lake which effectively provides access to the data stream.
- StarGraph construction (Section 3.3). This process constructs a StarGraph from a single data source, through a series of transformations.
- Constellation construction (Section 3.4). This process creates a Constellation by integrating two or more StarGraphs.
- Materialize (Section 3.5). This process populates a data mart (StarGraph or Constellation).

We now provide a description of our system in terms of workflow and the transformations that are necessary to deliver an integrated data mart from non-enterprise data sources.

3.1. The StarGraph model

Our data model must be capable of representing multidimensional data, i.e. the facts, dimensions and relationships that conceptually represent a star schema. The *StarGraph* is a graph-based model with constructs to represent facts and dimensional hierarchies that are captured from semi-structured data sources. In the first part of this section, we focus on the properties of the StarGraph and, in the second part, we focus on its functionality. Let us start by formally defining the StarGraph and its parts.

DEFINITION 3.1 (StarGraph). *A StarGraph is a two-tuple* $\mathcal{S} = \langle V, E \rangle$, *where* V *is a set of vertices* (*nodes*) *and* E *a set of edges*, *connecting vertices*.

A StarGraph vertex (*or node*) $v \in V$ *is a four-tuple* $v = \langle n, c, s, t \rangle$, *where* n *is the name of the node*; c *indicates the type of node*; s *specifies the location of the data item in the schema and* t *is the datatype of the defined node. A StarGraph Edge* $e \in E$ *is a three-tuple* $e = \langle x, y, r \rangle$ *where* $x, y \in V$; r *is a type denoting the relationship between the nodes* x *and* y.

The s attribute for each StarGraph node can be XPath [22] (for XML/HTML data) or dot notation for JSON data.

For the c attribute, 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.
- `measure` indicates that this node is a measure.
- `subdimension` indicates the node is a subdimension (i.e. a dimension which holds a foreign key to another dimension which is of a higher granularity).

The type attribute t can be one of:

- `STRING`: any textual information.
- `OBJECT`: assigned to nodes which contain no information but instead link to other nodes. These nodes become Dimensions in the generated StarGraph.
- `NUMBER`: any *numerical* datatype. These nodes represent candidate measures.
- `DATE`: *date* datatypes.
- `GEO`: reserved for nodes containing geo-location data (latitudes and longitudes).

Between each StarGraph Edge e, the relationship attribute r can be: 1-1; 1-n; or m-n.

3.1.1. *StarGraph functions*

There are six main functions which define the behaviour of the StarGraph. These functions manage all of the operations involved in constructing the StarGraph, binding StarGraph elements to the original source data and enabling materialization of StarGraph instances. In later sections, we will show how a special form of the StarGraph can represent a data mart constructed from multiple sources. For this section, we will focus on the core aspects of StarGraph construction from a single data source. The six functions are listed below and discussed in detail in Section 3.3.

- `GraphAnnotate`: analysis and markup of the initial graph.
- `ClassifyNode`: node analysis and classification.
- `GraphEdit`: graph update operations.
- `ClassifyDim`: detect and classify dimensions, as part of graph analysis.
- `ClassifyMeasure`: detect and classify measures, as part of graph analysis.
- `Materialize`: populate or update the StarGraph.

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

3.2. Stream introduction

The stream introduction process (P1) is used to add new data streams with a *Stream Service process* to update a Data Lake [23] with new instances of each data source at their specified update interval. A user is necessary to provide the URL to the data stream, an update interval, and a schema of the stream data (if available). The streams used by our system have a low update frequency (e.g. weekly), and are available on the web. Unlike other streaming formats (e.g. sensors), these streams are not pushed to consumers, rather the consumer is required to have knowledge of the stream's location and update frequency and obtain a new instance of the stream per update.

We include in this definition formats such as statistical open data provided by governmental and private bodies which provide data in the form of reports. A Metabase (described later in Section 4) is used to capture both stream metadata and a tree representation of the schema, together with the source (URL) and update interval.

3.3. Stargraph construction

The process of constructing a StarGraph involves a number of transformational steps which initially create and then extend the StarGraph as it captures all of the multidimensional information contained in the data source.

3.3.1. *The* `GraphAnnotate` *function*

The role of `GraphAnnotate` is to construct a graph representation of that data source, annotated with metadata describing the source. Each node will also contain a query which is used to extract source data. Queries are generated by traversing the document or object structure (for XML/JSON) and returning the location of the node within the source. For XML or HTML data, XPath queries are used; for JSON data sources, dot notation is used; and CSV files, the node contains a reference to the column number.

Each edge is annotated with the cardinality of the relationship between nodes. Once the queries for each node have been constructed, datatypes are assigned to each node. These datatypes are determined using the same queries that extract the data. An example of the graph generated by the `aimis_1` dataset from the Pig Prediction Data Mart (Table 4) is shown in Fig. 1. The purpose of the box around the three blank nodes is to simplify the visualization, where each edge to the box represents an edge to *every* node inside the box.

3.3.2. *The* `ClassifyNode` *function*

After datatypes have been assigned to nodes, the next step is *node classification*. The role of `ClassifyNode` is to provide a classification to represent a node's *role* within the data source. This determines their position and *importance* within a data mart. Initially, the graph is scanned node by node, where a classification is assigned to every node (Definition 3.2). First, the node is examined to determine if it belongs to the `Container` class. Nodes which hold a one-to-one relationship to a `Dimension` node are classed as `Dimension-attributes`, nodes that hold a one-to-many relationship to `Dimension` nodes are classed as `Subdimension` nodes. All numeric values are classed as `candidate measures` and all other nodes are classed as `Dimension` nodes.

DEFINITION 3.2. (Node classification). *Node classification is a five-tuple* $NC = \langle D, DA, S, PM, C \rangle$ *where* D *is a Boolean indicating whether the node represents a Dimension*; DA *is a Boolean indicating whether the node represents an attribute of a dimension*; S *is a Boolean indicating that the node represents a Subdimension*; PM *is a Boolean indicating whether or not the node is a candidate measure and* C *is a Boolean indicating whether or not the node is a container element.*

3.3.3. *The* `GraphEdit` *function*

The `GraphEdit` function restructures the graph mainly through the removal of unnecessary nodes. The algorithm deletes nodes annotated as *containers* and re-links child edges from the container node to the containers' parent nodes. The `ClassifyNode` function is then rerun to ensure no new container nodes have been created during the restructuring process. This process repeats until no container elements are found in the graph. This process updates the mappings by updating the relationships for all nodes affected by the removal.

3.3.4. `ClassifyDim` *and* `ClassifyMeasure` *functions*

The next step identifies measures and dimensions in order to provide the StarGraph with its multidimensional properties. All items which have previously been classified as *candidate* measures are evaluated at this stage. In the event that they are determined *not* to be measures, they are reclassified as `dimension-attribute` nodes and become part of a dimension object.

When constructing the Fact element, the dimensions which describe the fact are those through which the measure is linked either directly or transitively. In the event that a dimension does not have a specified key, a primary-foreign relationship will be created. This relationship will be generated using attribute(s) captured on both sides of the relationship: if two dimension objects contain the same values for their attributes, they are considered to be of the same dimension.

Measures are grouped into the same fact based on their shared relationships. That is, if two nodes classified as measures contain edges to the same dimension nodes, they will be placed together within a single fact. This is determined by examining the edges connecting each measure to dimensions and grouping them into *dependency sets* (See Definition 3.3). Similar to functional dependencies, the dependency sets identify which

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

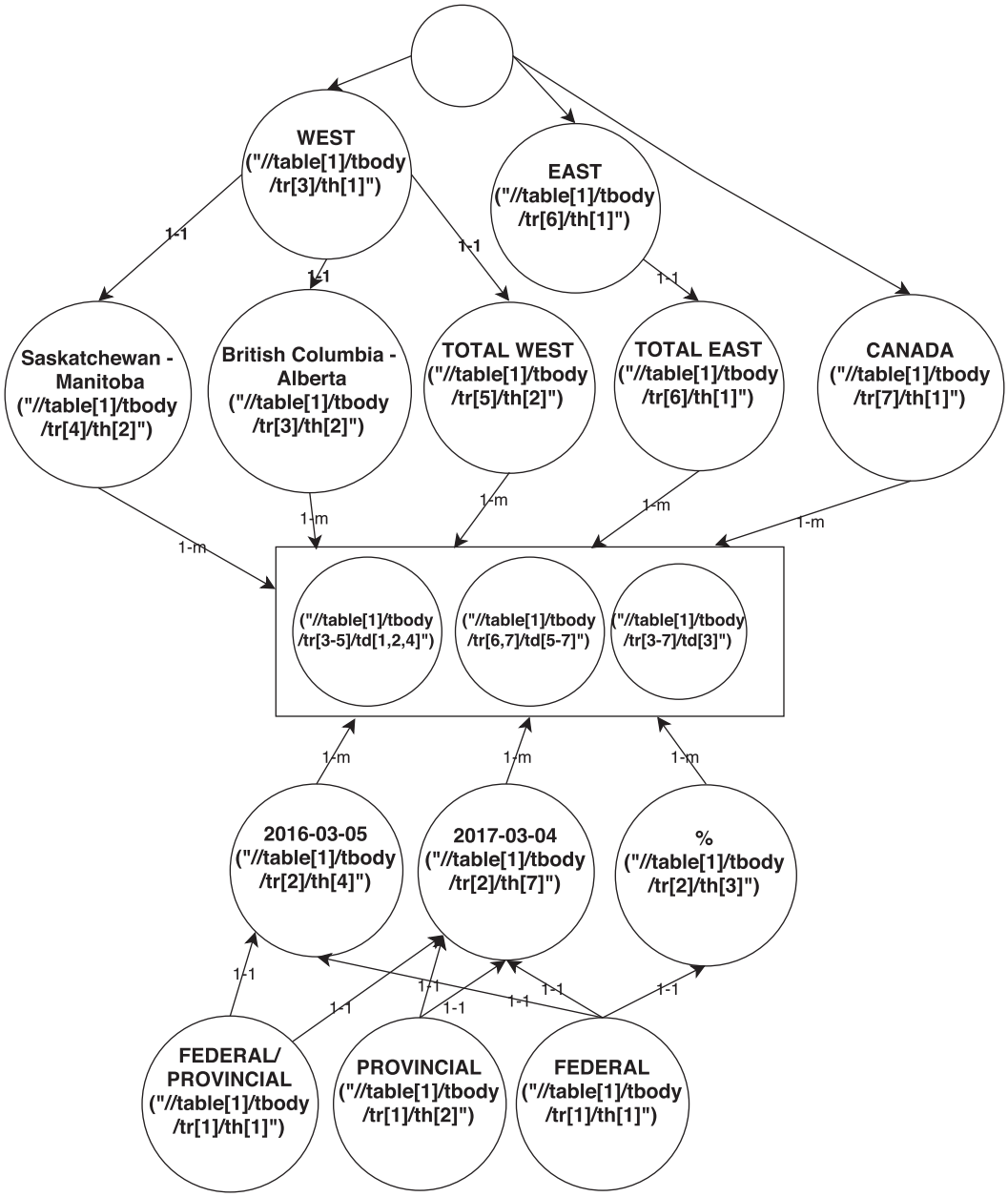

FIGURE 1. aim_1 dataset after GraphAnnotate.

dimensions are required for each measure. However, in addition to the structure of the graph, the cardinality of the relationships must also be considered in order to determine those measures which are part of the *same* Fact element.

DEFINITION 3.3. (Dependency set). *A dependency set* $DS = \langle (d_1, c_1), (d_2, c_2) \ldots (d_n, c_n) \rangle$ *is a set of tuples* $(d, c) \in DS$ *where* d *is the dimension to which a measure is connected and* c *is the cardinality of that connecting edge*.

Our method makes the important assumption that measures that have the *same* dependency set S are considered to be part of the *same* fact: they share the same dependencies. There may be multiple facts created from a single data source but these facts will share dimensions. Once all dependency sets

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

have been created, the final step is to construct the individual facts.

3.4. Constellation construction

Data sources are selected by the user with the system adopting a ladder strategy [24] for integration. In summary, the first two sources are integrated, the next source is then integrated into the previously integrated schema. This process continues until there are no more sources to integrate. This results in a process involving $n - 1$ integration steps where n is the number of sources to integrate. The five functions which perform the integration process are now described.

3.4.1. *The* `TermMap` *function*

The `TermMap` function resolves heterogeneities by annotating the graph with canonical terms for source attributes. A common usage for `TermMap` resides in the *Geo* dimension with the numerous ways different country names can be represented. For example, the terms '`IRL`' and '`Éire`' are not related using any standard string matching algorithm. However, both terms refer to the country Ireland.

DEFINITION 3.4 (TermMap). *A term map* $TM = \langle N, GC, CT \rangle$ *is a triple where* N *is the name of the data source node*, GC, *the Graph Context*, *is the identifier of the data source, and* CT *is the canonical term for the node*.

Definition 3.4 shows the attributes for each term map within the lightweight ontology. The context GC is required as different data sources may use similar terms within different contexts. For example, one source may use the term 'Ireland' to refer to the *sale* of pigs in Ireland, while another may use the same term to refer to the *production* of pigs in Ireland.

For the `aim_1` dataset, the application of the `TermMap` function removed ambiguities such as the lack of a name for the measures, and provided context in relation to terms such as 'East' and 'West', referring specifically to 'East-Canada' and 'West-Canada', respectively. This process extends the mappings for a StarGraph by adding a property `term` to each node denoting the canonical term obtained from the ontology.

3.4.2. *The* `TypeMap` *function*

There remains the issue of resolving concepts which should be treated as the same type, even though they appear different (e.g. 'Ireland', 'France' are two values of the same *Country* type). The type mapping assigns canonical *types* to each node by capturing the abstract concepts which link these nodes. There are five types specified as the minimum requirement for an automated integration as captured by the metamodel (Section 4.2).

For the Bord Bia dataset (see Section 5), the types `Date` and `Value` are appended to the graph. Similarly, for the `aim_1` StarGraph, there were three Metamodel attributes: `Date`, `Geo` and `Metric`. In addition, one type which was not part of the metamodel, but *was* a part of the ontology, `Region`, was annotated to the nodes `Federal`, `Provincial` and `Federal/Provincial`.

Similar to the `TermMap` function, this function extends the mappings by adding a property `onttype` (ontology type) denoting the type of the node as obtained from the ontology.

3.4.3. *The* `MetamodelCheck` *function*

The `MetamodelCheck` function ensures that a measure has sufficient information to complete the integration process. For every measure, the function confirms that the measure has connections to nodes with the types `Date`, `Geo`, `Item`, `Metric` and `Unit`. These types are required for each measure as multiple measures within a data source may represent different metrics. For example, one measure may provide a total count for a given date, while another may provide cumulative totals. However, without the necessary information `Item`, `Metric` and `Unit` for the measure, it is impossible to infer what the measure represents. The system examines each measure to determine if these attributes are present and, if not, prompts the user to supply the missing value.

For example, the `aim_1` dataset contains two measures: the total number of pigs slaughtered in a given week, and the percentage change of this number compared to the same week of the previous year. For this data source, the type `Item` refers to the high-level concept being analysed: here, `Item` is `Pigs`. `Metric` in this instance, is `Slaughter`. However, the type `Unit` is different for each measure, with the number of pigs slaughtered having a `Unit` value of `Total` and the percentage change measure having the `Unit` value `PCHANGE_WEEK_YEAR` referring to the percentage change of the value for the same week in the previous year.

Conversely, the `b_1` dataset contains a single measure, total pigs slaughtered per week. It has an `Item` value of `Pigs`, a `Metric` value of `Slaughter` and a `Unit` value of `Total`. By comparing both sets of metamodel attributes, the measure in `b_1` is semantically equivalent to the `Total` measure in `aim_1` and thus, the system can integrate both.

Where the metamodel properties are present, no changes are made to the underlying mappings. If the user supplies missing values, these are incorporated into the mappings. Figure 2 shows the StarGraph for the `aim_1` dataset after transformation. In comparison to Fig. 1, classifications are expressed using '⟨' and '⟩' brackets. Nodes expressed as ⟨`M`⟩ are classified as candidate measures; ⟨`DIM`⟩ refers to dimensions and ⟨`SUBDIM`⟩ refers to subdimensions.

3.4.4. *The* `GranularityCheck` *function*

The role of the `GranularityCheck` function is to examine two StarGraphs to determine if their dimensions have the same granularity. In the event that both graphs contain the same hierarchical dimension, they are checked to see if they both reside on the same *level* in the hierarchy. This function resolves conflicts which can arise when integrating data from, for example,

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

the date dimension which is hierarchical in nature. For example, where one data source is monthly and the second has a daily granularity, this triggers a `GRAIN_MISMATCH` event. The system requires the finer grained source to undergo a `rollup` operation so that both StarGraphs occupy the same level in the appropriate dimension hierarchy.

3.4.5. *The `DetermineStrategy` function*

The `DetermineStrategy` function analyses the metamodel attributes of data sources to build the integration strategy.

Step 1: Construct dependency sets. The first step constructs dependency sets for each measure in both data sources. Each set contains a measure and all metamodel attributes found for the measure. For the source `b_1`, there is only one dependency set shown in Example 1.

EXAMPLE 1 (Dependency set for `b_1`). ⟨`WEEK`, `GEO`, `PIGS`, `SLAUGHTER`, `COUNT`, `VALUE`⟩

As there are two measures in `aim_1`, there are two dependency sets, shown in Example 2. In the remainder of this discussion, we will refer to the set in `b_1` as `b_1_a` and the sets in `aim_1` as `aim_1_a` and `aim_1_b`.

EXAMPLE 2 (Dependency sets for `aim_1`). ⟨`WEEK`, `GEO`, `PIGS`, `SLAUGHTER`, `COUNT`, `VALUE`⟩, ⟨`WEEK`, `GEO`, `PIGS`, `SLAUGHTER`, `PERCENT_CHANGE`, `VALUE`⟩,

Step 2: Create combination table. This step determines how dependency sets may be integrated. There are three possible approaches: `row-append`, `column-append` and

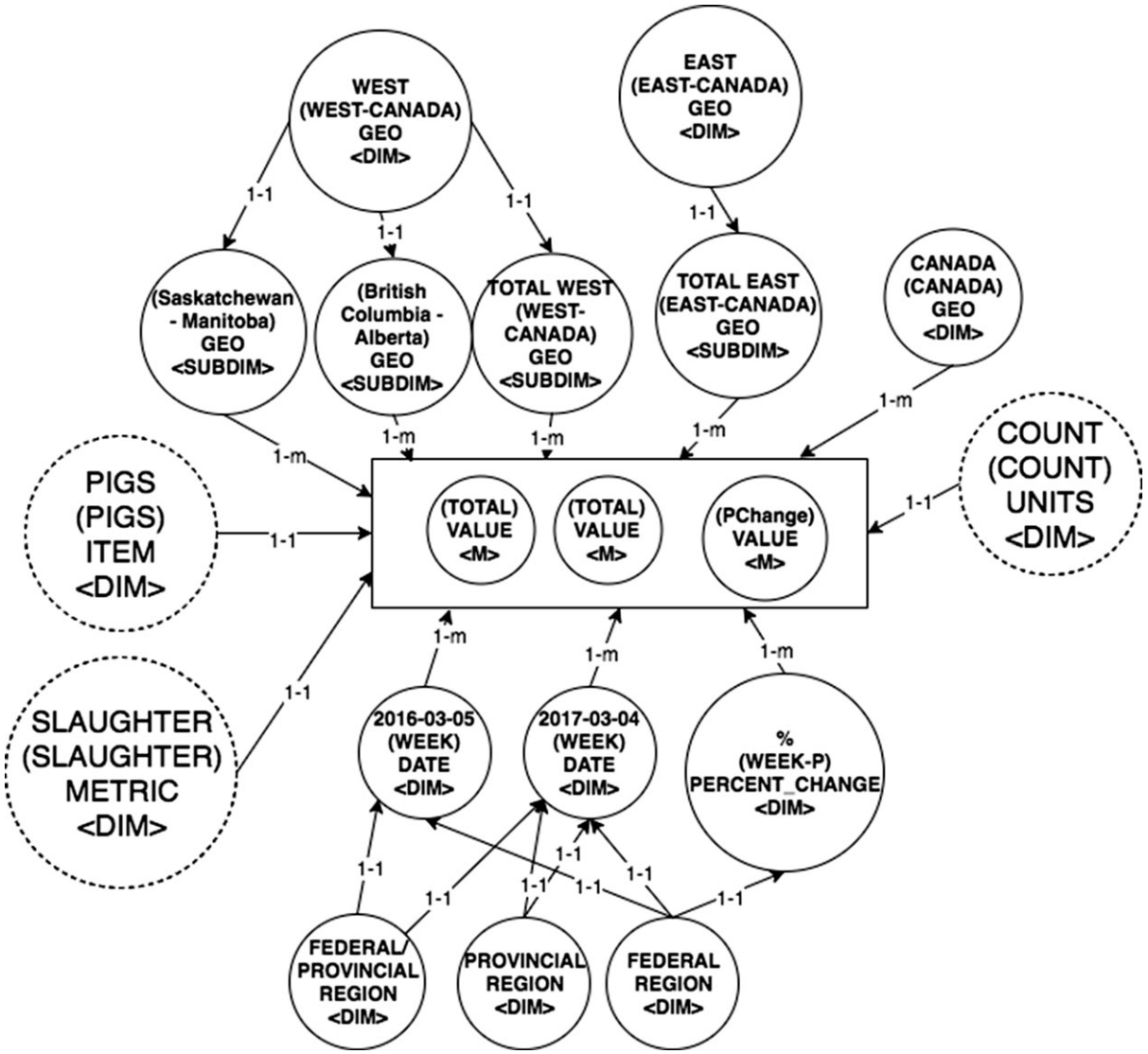

FIGURE 2. `aim_1` StarGraph after metamodel check.

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

no-integration. The output illustrates how each combination of dependency sets may be integrated.

The row-append method is used for sets which are identical, meaning that a direct integration can take place. The column-append method is used for sets which only differ based on their Metric and Unit values. In these instances, the facts are based around the same concept, but differ in the metrics and units used. The column-append strategy integrates data where two sources contain the same values for the metamodel values Date, Geo and Item. This strategy overcomes differences between units of measurement, and combines multiple measures into a single fact based on their similar metamodel attributes. Finally, no-integration means that dependency sets contain different Date, Geo or Item values, and integration cannot take place (Table 1).

Step 3: Create combination graph. The previous step shows how each pair of dependency sets may be integrated, but cannot manage integrations that should occur through transitive closure. A combination graph is an undirected graph, where each node in the graph represents a dependency set, and the edges between nodes are integration methods. Figure 3 shows the combination graph for the sources aim_1 and b_1. From the graph, we can see the nodes aim_1_a and b_1_a are linked through the row-append method and aim_1_b is linked to both using the column-append method.

Step 4: Collapse combination graph. The final step is to collapse the graph so that each remaining node relates to a single fact in the final Constellation. The function first examines all nodes which share a row-append edge and combines them. This new node inherits all of the edges from the collapsed node. This results in a graph similar to the one shown in Fig. 4, where the nodes aim_1_a and b_1_a have merged. This node holds two column-append edges to the node aim_1_b.

It is still possible to have a node which holds two edges to the collapsed node. This is resolved by selecting the highest possible edge from a hierarchy. Table 2 outlines the possible combinations of edges, and the resulting edge which is chosen. From this table, we can see that the column-append edge is selected, with the result shown in part *A* of Fig. 5. The process repeats until all row-append edges have been removed, and then proceeds for all column-append edges. At the end, there are two possibilities: a single unified node, which means that all sources can be combined into a single unified fact; or a series of merged nodes, which all hold no-integration links, as shown by the single node in part *B* of Fig. 5.

TABLE 1. Combine function results: aim_1 and b_1.

Set_1	Set_1	Method
b_1_a	aim_1_a	row-append
b_1_a	aim_1_b	column-append
aim_1_a	aim_1_b	column-append

This simple case study results in the set of integration steps shown in Table 3. Figure 6 presents the high-level Constellation created from the aim_1 and b_1 StarGraphs. The figure illustrates metamodel values (for Date, Geo, Item, Metric and Unit) extracted from each data source, and their interactions which constitute a *Fact*. The dotted nodes indicate different measures, and the dotted node Region is an optional dimension obtained from the aim_1 StarGraph.

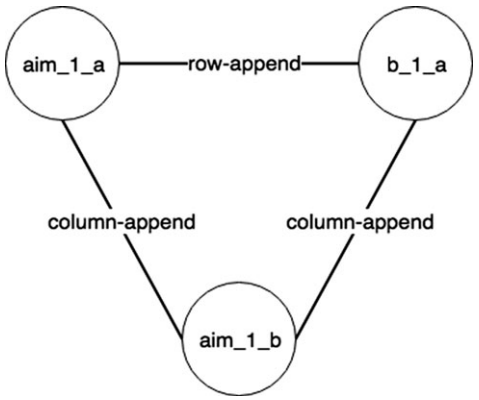

FIGURE 3. Combination graph for aim_1 and b_1.

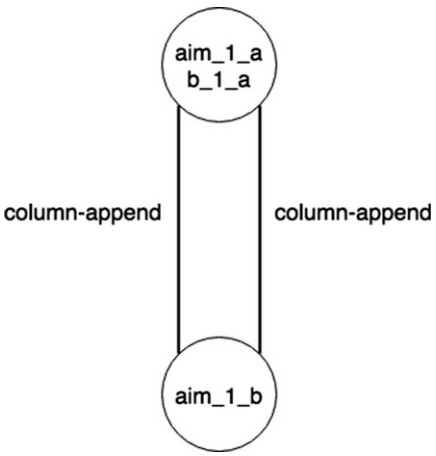

FIGURE 4. Merged row-append edges.

TABLE 2. Integration strategy based on edges.

Edge 1	Edge 2	Final strategy
Row-append	Row-append	Row-append
Row-append	Column-append	Row-append
Row-append	No-integration	Row-append
Column-append	Column-append	Column-append
Column-append	No-integration	Column-append
No-integration	No-integration	No-integration

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

3.5. Materialize

The `Materialize` logic depends on the data model and employs wrappers developed for JSON, XML, HTML and CSV [25]. A detailed description is outside the scope of this paper but the mappings shown in Example 3 outline how the wrappers extract the appropriate dimension and fact *containers*. This mappings example contains all metamodel and optional attributes for a Constellation under the `source_meta` property using the 'facts' property to list those sources that constitute a fact, together with their metamodel attributes.

EXAMPLE 3 (Final Constellation Mappings).

```
{
"name" : "aim\_1,bb\_1",
"source_meta" : {
    "aim\_1" : {
        "DATE" : [
            {
                name:"2016-03-05",
                type:"WEEK",
                src:"//table/tbody/tr[2]/th"
            },{
                 name:"2016-03-05",
                type:"WEEK",
                 src:"//table/tbody/tr[3]/th"
            } ],
        "GEO" : [
            {
                name:"CANADA",
                type"Geo",
                src:""//table/tbody/tr[6]/th""
            },...
        ],
        "ITEM" : [
        {
            name:"PIGS",
            type:"Item",
            src:"STATIC"
        } ],
        ...
    }
},
"facts" : [
    {
        "sources" : [ "aim\_1","b\_1"  ],
        "META_VAL" : [
            {
                "attr" : "DATE",
                "val" : "WEEK"
            }, {
                "attr" : "GEO",
            }, {
                "attr" : "METRIC",
                "val" : "COUNT"
            }, {
                "attr":"ITEM",
                "val":"PIG"
            },{
                "attr":"UNIT",
                "val":"Count"
            }
        ],
        "OPTIONAL":[
            {
                "attr":"Region"
            }  ]
    },...
    ]
}
```

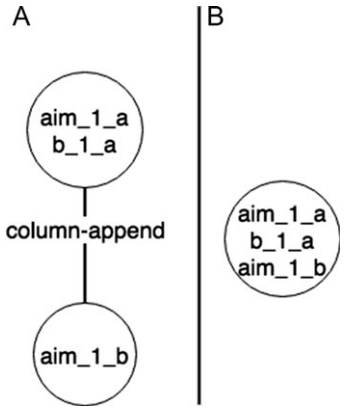

FIGURE 5. Merged column-append edges.

TABLE 3. Integration steps produced by the collapsed graph.

Step	Source_1	Source_2	Method	Join
1	{aim_1_a}	{b_1_a}	Row-append	All
2	{aim_1_a} {b_1_a}	{aim_1_b}	Column-append	DATE, GEO, ITEM

4. METADATA AND STORAGE

Three repositories are used in this system: the Data Lake, the Metabase and the Data Warehouse. The Data Lake is where raw data retrieved from the data sources are stored. The Metabase contains descriptive and administrative information regarding the data sources, the raw data, their mapping to the respective StarGraphs and Constellations, and the mappings to automate the data integration. The Data Warehouse holds the materialization of the data cubes resulting from our automated integration process. The rest of this section details these three repositories.

4.1. Data Lake

The Data Lake is the repository where raw data retrieved from the data sources are staged before being analysed and integrated. Raw data are retrieved (independently) from each known source at specified intervals, and 'appended' (i.e. added without transformation) to existing data already collected from the same source. Data are stored in its native format, e.g. JSON, XML or CSV. The Data Lake does not enforce a rigid structure on data pulled from the same data source: as the data structure from a source evolves, the Data Lake approach offers the flexibility to seamlessly accommodate any change without having to undergo a revision of our

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

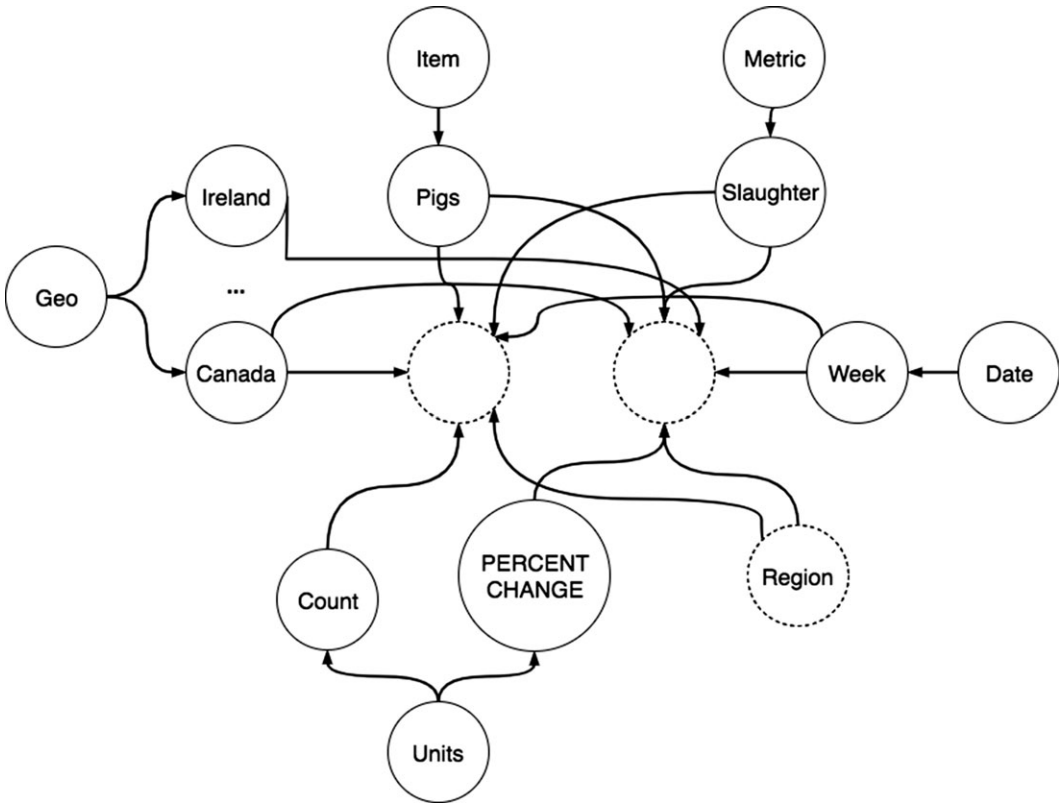

FIGURE 6. Constellation from `aim_1` and `b_1`.

repository. In fact, data belonging to the same source may be stored in multiple files, where each file may differ in both structure and format. Information about the origin, format, and location of data is captured as metadata and maintained in the Metabase as described in the next section.

4.2. The Metabase

The Metabase contains descriptive and administrative information about the input data, our integration graphs and processes, and data marts.

4.2.1. Data source and cube metadata

In our system, data sources and their instance data are described independently, so as to keep separate the origin of the data from the actual format that the raw data has when retrieved. Data sources and associated instances are described by the following two properties: *Data Source* and *Data Instance*.

DEFINITION 4.1 (Data Source). *A data source* $DS = \langle U, N, W, I \rangle$ *is a four-tuple where* U *is the unique identifier the system generates for each new data source*, N *is the mnemonic associated with the data source*, W *is the URL at which the raw data are available for retrieval from the remote data source*, *and* I *is the interval of time specifying the pace at which to pull data from the provided URL.*

The Data Source property characterizes the source of the data so that our system knows when and from where to retrieve (new) data. The Data Instance property, on the other hand, characterizes the actual data as it is stored in the Data Lake, so that the system knows how old the last batch of retrieved data is and in what format it should be accessed locally. The Data Instance property is defined as follows:

DEFINITION 4.2 (Data Instance). *A data instance* $DI = \langle I, U, D, T \rangle$ *is a four-tuple where* I *is the unique identifier the system generates each time new data are retrieved from a specific source*, U *is the identifier of the source from which data are retrieved*, D *is the UTC date-time of data retrieval and* T *is the format used to store the raw data into the data lake.*

Similar metadata are maintained for the data marts and cubes materialized in the Data Warehouse. Every time data from the input sources are integrated to update an existing cube, or to generate a new one, the system keeps track of the new data mart instance by the property *Cube Instance*, that is defined as follows:

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

DEFINITION 4.3 (Cube Instance). *A cube instance* $CI = \langle C, D, L \rangle$ *is a three-tuple where* C *is the unique identifier the system generates for each new data mart created,* D *is the UTC date-time of the cube creation/update, and* L *is number of times the cube has been loaded since its (first) creation. This is used in conjunction with* D *in order to determine which new instances from the data lake are required for population.*

4.2.2. *Integration ontology*

Mappings and processes required to populate the data mart, starting from the raw input data to an integrated data mart are stored in the Metabase. While StarGraph components are automatically extracted from the source data, the Constellation and its mappings are managed by a lightweight integration ontology, which is a separate component of the Metabase. This ontology is composed of two main parts: a series of canonical terms and a set of abstract types. A subset of these types constitute the metamodel which describes each StarGraph from a semantic viewpoint and assist in the integration process. Essentially, they provide integration points across data streams. While it would be considered a lightweight ontology, it consists of 29 distinct abstract types and 70,000 distinct terms and was built over a period of 18 months [4]. In addition, the ontology contains the rules which govern how data sources should be integrated, namely the logic for the `DetermineStrategy` function (Section 3).

Metamodel. The metamodel augments each attribute in the StarGraph with annotations that drive the integration process by helping to identify compatible attributes across different StarGraphs. We group these annotations under the property *Attribute Semantic*. When attributes from two (or more) StarGraphs are semantically compatible, the StarGraphs can be integrated to form a Constellation. Instances are semantically compatible if they both share the same types (as obtained from the ontology) and are of the same granularity. Instances of compatibility are described as mappings that are then executed to import actual data from the Data Lake into a data mart.

The attribute semantic property is described as follows.

DEFINITION 4.4 (Attribute semantic). *An attribute semantic* $AS = \langle D, G, I, M, U \rangle$ *is a five-tuple where* D *is a Boolean describing whether the attribute represents the Date dimension,* G *is a Boolean describing if the attribute is a Geo dimension,* I *indicates the name of the measure in case the attribute is a measure of interest,* M *describes the metric for the measure and* U *specifies the units for the metric.*

In the attribute semantic property, the set of elements $\{D, G\}$, and $\{I, M, U\}$ are mutually exclusive because an attribute can either be a quantitative measure or a dimension (Date or Geo). If the attribute is a measure, then all the elements $\{I, M, U\}$ must be specified in order to identify the name of the measure, the metric it is derived from, and the unit of reference for the attribute's values. These pieces of information are crucial to ascertain whether two dimensions or measures (e.g. metric) from different StarGraphs are semantically compatible, and whether some transformation is needed in order to achieve homogeneous data in the cube (e.g. units for the metric).

4.3. Data Warehouse

The Data Warehouse stores populated data marts created from either a StarGraph or Constellation. When data are required to populate the mart, the appropriate source data are obtained from the Data Lake and the facts, dimensions and measures required for a star schema are extracted using the mappings from the Metabase.

5. DATA SOURCE ANALYSIS

In this section, we present how our method fared with 120 unknown Agri data sources. We first briefly discuss the features and the type of data published in the considered data sources, then we analyse how these data sources are imported into our StarGraph and Constellation constructs. An examination of the results allowed us to create a set of classifiers on the benefit and usage of each data source.

5.1. Agri datasets

Here, we briefly describe the data sources captured in one of the sample Constellations and, using a small subset of the ontology, explain how a simple normalization process can create usable data marts. An approach advocated in [26] is to use lightweight dynamic semantics—which our system adopts—in order to import and integrate new source data. Table 4 in Section 3 provides details of how this process worked for the 13 Agri sources which comprise the data mart we describe here.

- **Agriculture and Agri-Food Canada** [27], in their Aimis webpage, publish weekly data reports on multiple measures from which we have extracted data on hog slaughters at packing plants in Canada (Aimis_1 and Aimis_2 in Table 4).
- **USDA** [28] provide the **Quickstats** API for easy access to several datasets such as production, price and slaughters. This data mart imports pig crop data from this source (usda in Table 4).
- **Bord Bia** [29] is the Irish food board, founded to promote sales of Irish food and horticulture as well as providing certification of Irish products. This data mart extracts pig prices (Bord Bia_1 in Table 4), dairy prices (also Bord Bia_1 in Table 4) and pig slaughters (Bord Bia_2 in Table 4).

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

TABLE 4. Sources used in the Pig Prediction Data Mart.

Name	ID	Format	Nodes	Edges	Dims	Measures	Materialize (ms)	Ins.
Aimis_1	aim_1	HTML	12	12	10	32	27	30
Aimis_2	aim_2	HTML	12	12	10	7	32	30
Bord Bia_1	b_1	HTML	2	2	1	1	70	Varies
Bord Bia_2	b_2	HTML	17	13	5	7	70	1–4
AHDB_1	p_1	HTML	88	88	11	8	122	8
AHDB_2	p_2	HTML	2	2	1	1	91	42
AHDB_3	p_3	HTML	4	2	3	1	150	156
AHDB_Bpex_1	bp_1	CSV	5	4	2	3	320	4
AHDB_Bpex_2	bp_2	CSV	5	4	2	3	320	3
cme_1	c_1	HTML	8	8	1	6	128	8
cme_2	c_2	HTML	8	8	1	6	131	12
imf	imf	XML	6	3	2	1	256	1122
usda	usda	CSV	39	38	33	6	166	84

- The **Agriculture and Horticulture Development Board** (AHDB)[30] is a statutory levy board who provide research and development programs and market information to farmers and businesses in the Agri sector. This data mart extracts annual per capita consumption (AHDB_1 in Table 4), annual pig slaughters (AHDB_2), pig prices (AHDB_3) and weekly pig slaughters (AHDB_Bpex_1 and AHDB_Bpex_2 in Table 4).
- **CME Group** [31] is a designated contract market who conduct and publish economic research in addition to publishing quotes for various commodities. Live pig data are used in this data mart (cme_1 and cme_2 in Table 4).
- **International Monetary Fund** [32] publishes daily exchange rates based on the unit of account SDR (Special Drawing Rights) (imf in Table 4).

5.2. StarGraph analysis

We attempted to import 120 unseen Agri data sources and to construct a StarGraph for each of them. However, it is important to note that, unlike traditional data streaming sources, where there is one StarGraph per source, multiple StarGraphs can be produced from a single data source. This could be the result (for example) of multiple `<table>` elements in the HTML document from where data are extracted. In this case, the 120 streaming sources provided 120 StarGraphs. However, not all sources are usable, or have dimensional data, or are in the format that we assume for data sources (e.g. HTML, CSV, etc.). By using 120 different data sources, there is enough empirical evidence for us to classify data sources in terms of their usability. The classifications are as follows:

- **Full**. This indicates the construction of a 'perfect' StarGraph, with all attributes of the source being correctly captured and stored.
- **Partial**. This indicates that, after StarGraph construction, not all metadata have been processed correctly but nevertheless it still functions as a working data mart.
- **Descriptive**. This indicates that a dimensional structure was found within the StarGraph, but no measure could be identified. However, the dimensional structure can be used in integration to enhance another StarGraph.
- **Missing**. These were constructed StarGraphs where dimensions were detected, but no facts could be identified and the dimensions found were not of sufficient depth to prove useful for integration.
- **Unusable**. In this case, the data source did not allow a StarGraph to be constructed.

From Table 5, we can see that of the 120 potential sources, our approach automatically imported and integrated 84 (70%) usable Agri sources of which 41 (34%) were imported with 100% accuracy. Each column illustrates how a classification was achieved: `Facts` indicates that one or more facts were identified; `Dimensions` indicates that one or more dimensions were found; `Holistic` confirms that no attributes were discarded during the creation of the StarGraph; `Integrated` indicates that the StarGraph can be used in a subsequent integration by providing dimension hierarchies and more details to a *usable* StarGraph; `Non-Spurious` indicates that data in the constructed StarGraph are valid and usable; and finally, `Count` denotes exactly the number of constructed StarGraphs for each classification. Note that the classification `Unusable` means that a StarGraph cannot be constructed while the `Non-Spurious` metric being false means that a StarGraph was constructed but the structure of

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

TABLE 5. Results of analysing 120 Agri data sources.

Classification	Facts	Dimensions	Holistic	Integrated	Non-Spurious	Count	Sources(%)
Full	✓	✓	✓	✓	✓	41	34
Partial	✓	✓	✓	✓	✗	29	24
Descriptive	✗	✓	✓	✓	✗	14	12
Missing	✗	✓	✗	✗	✓	2	1
Unusable	✗	✗	✗	✗	✗	34	29

the graph may contain redundancies or lack information required to construct a fact. In other words, the StarGraph has been created without a Fact component.

The sources which were classified as *Unusable* were mainly due to the format of the source data (e.g. a PDF) and for others, it was due to issues with the file structure (e.g. a file having a .CSV extension, but the content was not a CSV file). Some sources could be correctly parsed yet they still proved unusable. All of these sources were HTML documents which displayed data in a tabular fashion, but this was accomplished using CSS, while the underlying HTML failed to use tags in the correct fashion (e.g. using `<div>` elements to represent a table). Additionally, some sources which were classified as *Unusable* were, in fact, fully constructed data cubes. However, creating a StarGraph from fully constructed data cubes is not currently possible using our mapping process.

The two sources classified as *Missing* found dimensions but these proved to be unusable without context. In both instances, the only dimension found was a date dimension, and in the absence of a fact, measure or other dimensions to relate to, the constructed StarGraphs were of little use.

For those classified as *Descriptive*, no facts were found in the source data. However, a large degree of dimensions and hierarchies were discovered. These can be integrated with other StarGraphs in order that their dimensional data are reusable.

For those sources classified as *Partial*, StarGraphs were constructed but were unusable. An example of a *Partial* StarGraph is where a data source uses `<table>` elements to model both tabular data and to dictate layout of content on a webpage. When elements are misused in this manner, it is difficult for our system to determine whether or not the data located inside the `<table>` is dimension or fact values, or neither. In all of these cases, user interaction can be used to convert these StarGraphs to *Full* StarGraphs. In effect, this means that 58% of the unknown Agri sources are usable as data marts with a further 12% ready for integration into existing data marts.

6. EVALUATION AND DISCUSSION

For a robust evaluation of our methodology, we engaged with industry partners who specified a real-world business requirement requiring the construction of a multidimensional dataset. For the evaluation, we perform three types of integration: user defined, non-assisted, and ontology assisted. For the user-defined integration, the data mart was created manually by editing the structure of a StarGraph and forcing the generation of mappings. This provided our ground truth version for the data mart. Two of our industrial partners took part in the evaluation process. Their initial role, together with the researchers, was to build the manual data mart. However, they were also used to verify the correctness and completeness of the computer generated data marts using a series of SQL queries.

For the non-assisted integration, a data mart was constructed with a limited usage of our ontology to understand where issues with the *integration* of data lie. Here, the ontology was used to normalize terms before StarGraph construction but was not involved in the integration process. This was to evaluate the degree to which a fully automatic process could be used for integration and the degree to which an ontology is required. This is useful for domains for which no ontology is readily available.

For the ontology-assisted integration, we tested the automated construction of data marts but with a semi-automatic integration approach utilizing the ontology to facilitate semantic integration. This approach is fully automated for those streams classified as *Full* in our earlier analysis, but require some level of user intervention for streams classified as *Partial*. In these cases, the appropriate semantic attributes defined in our metamodel could not be detected.

The case study, to predict global pig prices, required the 13 data sources outlined in Table 4. `Name` is the name of the data source, `ID` is the experimental ID assigned to the source, `Format` is the source data model, `Nodes` refers to the number of nodes within a StarGraph constructed from this data source, `Edges` is the number of edges within a StarGraph constructed from this source, `Dims` is the number of dimensions identified in the data source, `Measures` is the number of measures identified, `Materialize` is the time taken to populate a data mart and `Ins` is the number of instances of facts obtained per data source. Of the 13 data sources specified by the end user, 8 sources were classified as `Full` while the remaining 5 were classified as `Partial`.

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

6.1. User-defined integration

Three measures were identified: the number of pigs slaughtered, the price of pigs and the future quotes. There were three main dimensions identified: `Date`, `Geo` and `Item`. The dimensions `Date` and `Geo` are dimensional hierarchies containing different levels of granularity. For the `Date` dimension, the data sources provided were either monthly or weekly. For the `Geo` dimension, the hierarchies were based on area. For example, the `USDA` source provided a breakdown of slaughtering by state, while the `Bord Bia` source lists the number of slaughters as a whole. Figure 7 presents a high-level overview of the Constellation, with details of dimensions and facts and the links between them. Internally, this structure is a Constellation but in order to convey the structure of this data mart, it is displayed as a traditional entity relationship diagram.

6.2. Non-assisted integration

In Table 6, integration is described by: *Source_1* and `Source_2` which indicate that sources used at different stages of the integration; *Join On* indicates the integration parameter; *Issues* highlights problems during the non-assisted integration; *OntologyAssist* indicates the rule to resolve the issue; and the final column, *User Supplied*, indicates the metamodel values supplied by the user. For example, `GEO` indicates that the user was prompted to provide the GEO attribute while `NONE` indicates that no user intervention was required. Where the value `Cons` is present in the `Source_1` column, it indicates the current version of the Constellation is used as the parameter for integration at that step.

In Step 1, the initial Constellation `Cons` is created from the datasets `aimis_1` and `aimis_2`. There was a high degree of integration at Step 1 because structurally `aim_1` and `aim_2` were very similar. However, granularity for integrated data proved a problem. Both of these sources contained a measure called % which denotes the percentage change between two dates. However, for one source the % meant the percentage change from the previous *week*, and for the second source, it meant the percentage change for a *year*. Despite these both being correctly identified as measures, the different granularities prevented the integration.

FIGURE 7. User-defined schema.

Step 2 integrated the `b_1` data into `Cons` with integration based on the `Date` dimension. As there was no matching node in the Constellation for the measure, it remains unconnected. However, this measure should have integrated with the previous two sources in addition to the `Date` dimension as they all relate to the sale and production of pigs. Without a form of abstraction (supplied by an ontology) to semantically link the two measures, they remain separate in the data mart produced by the non-assisted integration. Step 3 included the StarGraph created from `b_2` into the Constellation. Similar to `b_1`, this source was integrated based on the `Date` dimension as no suitable candidate was found for measure integration. In other words, the dimensional hierarchy was enriched but no new facts were added.

Step 4 integrated `p_1` into the Constellation. Once again, a matching `Date` candidate was found which saw a large reduction in the graph (as this source is largely time-series data). However, other dimensions which failed to integrate were country identifiers (e.g. 'Austria'). Again, an extension to the ontology to indicate a type hierarchy would see a large degree of integration produced (and subsequently a lower number of nodes and edges). Steps 5 and 6 integrate `p_2` and `p_3`. These sources were simple tables matching years to a measure. As such, the data was integrated based on the `Date` dimension and the measure was added to the fact table.

Step 7 integrated `bp_1`with the Graph partially integrated on the countries listed in `p_1`, as one dimension specified a country. As expected, without the ontology, which contains a full dimensional hierarchy, some countries failed to be integrated due to heterogeneous tags (e.g. 'Great Britain' and 'Northern Ireland' combined would be 'United Kingdom'). Step 8 integrated the `bp_2` data source, which was identical in structure to `bp_1`. As such, there was a 1-1 integration between this source and the Constellation with all measures merged with those in `bp_1`.

Step 9 integrated `c_1` with `Date` providing the only common attribute for integration. This is due to the fact that the data source `c_1` refers to future prices. However, a large number of measures were found within this data source, and as such have been added to the fact data joined on the `Date` dimension. Step 10 integrated `c_2` and, similar to Step 9, the structural similarity between `c_1` and `c_2` facilitated a 1-1 mapping between all nodes. Step 11 sought to integrate currency conversion data from the `imf` data source. The data were successfully integrated on the `Date` dimension, while

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

TABLE 6. Integration steps for creating constellation.

Step	Source_1	Source_2	Join on	Issues	OntologyAssist	User supplied
1	{aimis_1}	{aimis_2}	DATE, GEO	GRAIN_MISMATCH	GRAIN_CHECK	ITEM, METRIC, UNIT
2	Cons	{b_1}	DATE	MISSING_ATTR	METAMODEL_CHECK	GEO, ITEM, METRIC, UNIT
3	Cons	{b_2}	ALL	MISSING_ATTR	METAMODEL_CHECK	GEO, ITEM, METRIC, UNIT
4	Cons	{p_1}	DATE	TERM-TYPE_MISMATCH	TERM-TYPE_MAP	ITEM, METRIC
5	Cons	{p_2}	DATE	MISSING_ATTR	METAMODEL_CHECK	GEO, ITEM
6	Cons	{p_3}	DATE	MISSING_ATTR	METAMODEL_CHECK	GEO, METRIC
7	Cons	{bp_1}	DATE	TERM-TYPE_MISMATCH	TERM-TYPE_MAP	ITEM, UNIT
8	Cons	{bp_2}	ALL	TERM-TYPE_MISMATCH	TERM-TYPE_MAP	ITEM, UNIT
9	Cons	{c_1}	DATE	MISSING_ATTR	METAMODEL_CHECK	GEO, UNIT
10	Cons	{c_2}	ALL	MISSING_ATTR	METAMODEL_CHECK	GEO, UNIT
11	Cons	{imf}	DATE	TERM-TYPE_MISMATCH	TERM-TYPE_MAP	METRIC, UNIT
12	Cons	{usda}	DATE	TERM-TYPE_MISMATCH	TERM-TYPE_MAP	NONE

the new currencies occupied new dimensions and the rates were included as measures within the fact table. Finally, Step 12 integrated the usda data, once again using the Date dimension. The data was highly dimensional, adding in 30 previously unseen dimensions and 8 measures.

This process generated a large data mart containing 66 dimensions and 7 fact tables and thus, a final illustration cannot be displayed. There was a large degree of redundancy in this data mart as most items failed to integrate due to lacking semantics (e.g. country names, similar products).

6.3. Ontology-assisted integration

In this section, we describe how a close-to-automatic process for data mart construction was achieved. At various points in the process, it was necessary for the user to update the ontology (through a system prompt) so that integration could complete and to ensure a fully automated integration for the same sources in future data marts.

For the initial Constellation, both a Date and Geo dimensions were found for both data sources. Two measures were found for each source, but there was no defined Item, Metric or Unit attributes. As both of these sources were the same structurally and semantically, they were fully integrated. However, the process prompted the user for input in both cases.

The next source, b_1, was sparse, containing only two attributes: Date and a measure. In this instance, the integration processes stopped again to prompt a user for input for the dimensions Geo and the attributes Item, Metric and Unit, as the ontology again could not provide the precise level of detail. Once supplied, integration was completed using the Date and Geo dimensions and, along with the measure, by the Item dimension.

The next source, b_2, was missing the Geo dimension and Item, Metric and Unit attributes. Once provided, integration was completed using the Date and Geo dimensions and the aimis_1 and aimis_2 measures. The source p_1 was found to have Date and Geo dimensions but failed to provide the attributes Item and Metric. The Unit attribute was found (kg per head). The source p_2 had Date and Unit attributes but failed to provide Geo and Item. Once again, a user prompt indicated where the ontology required an extension.

The source p_3 provided a Date attribute and two Unit attributes. However, there was no Metric or Geo dimension. The source bp_1 provided all required attributes except the Item and Unit attributes. Once provided by a user, it was integrated into an existing Metric and Unit dimension. The source bp_2 was also missing the Item and Unit attributes and, after user prompting, integration with bp_1 was successful.

The source c_1 provided several new Metric attributes. However, a Unit was not listed and the Geo dimension was not found. c_2 was also missing Unit attributes and a Geo dimension. The final integration was different as the data source provided quotes about corn, and every Item thus far referred to pigs. Thus, the system integrated on the Date and Geo dimensions, which was correct.

The source imf found a series of Item attributes and a Date attribute. However, it failed to find a Metric or Unit attribute for each measure. Once again, this data was of an entirely new domain—currency conversion rates—and as such was integrated on the Date and Geo dimensions, after the ontology was updated. The final source usda contained all required attributes and was integrated automatically.

6.4. Analysis

The final Constellation contains four fact tables, separated by date granularity as shown in Fig. 8. Most dimensions contain little structure. For example, the Metric dimension contains an attribute name which stores the type of metric used (e.g.

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

price). This case study required the integration of 13 data sources and required the ontology 416 times. In terms of the user-defined approach, not only do they have to have an in-depth understanding of the data they are capturing, it was also necessary to manually design and create the data mart (e.g. construct the schema, add constraints, write queries to insert data). This raised the same issues as were seen during both non-assisted and ontology-assisted integration strategies, in terms of having to determine the correct grain in hierarchy dimensions and in detecting the correct attribute for integration.

Manual integration was estimated to take between 8 and 10 hour for what was a reasonably complex data mart (13 separate sources), while the automated approach required 118 ms. The data marts created by the two approaches are quite similar although, at a high level, the schemas in Figs 7 and 8 appear quite different. The difference is due to the compact data mart created by the manual process while the automated approach requires a ROLLUP to join facts, e.g. aggregate Month → Year.

The data mart obtained with an automated approach, shown in Fig. 8, is slightly more complex: it is composed of seven dimensions and four fact tables, although the table structure is simpler. The higher number of tables is the result of the inability of the automated approach to catch all those aspects that an expert eye would, thus leaving the automatically generated data mart with some redundancies.

In terms of data mart semantic correctness, it can be observed that the data mart obtained with the automated approach is a superset of the manually created one. Therefore, it can be said that the data mart generated with the automated approach can describe the same data as the manually created one and is, therefore, semantically sound. This was verified by the industry partner in the project. From the performance point of view, we have observed that populating the manually created data mart is faster, about 0.9 seconds, than populating the automatically generated one, about 1.1 seconds. This is caused by the higher number of tables generated by the automated approach, which ends up requiring more queries and data transfer.

It is worth highlighting the reasons why manual construction takes such a long time as these are generally the same issues that are resolved during ontology-assisted integration. This process also highlights how the ontology is used to aid integration and how mappings are updated when issues with the structure of data sources are uncovered.

Table 6 outlines the integration issues encountered by the non-assisted approach, and details how these issues were mitigated at each stage by the ontology-assisted method. Step 1 integrates two sources into a Constellation (Cons) and, from there, proceeds to integrate more sources into this Constellation. At a high level, there are three main issues. The first is GRAIN_MISMATCH, which occurs when two data source are of differing levels of granularity. For example, the Date dimension has one source that provides weekly data and the other providing monthly data. Similarly for the Geo dimension, some data sources indicate individual countries

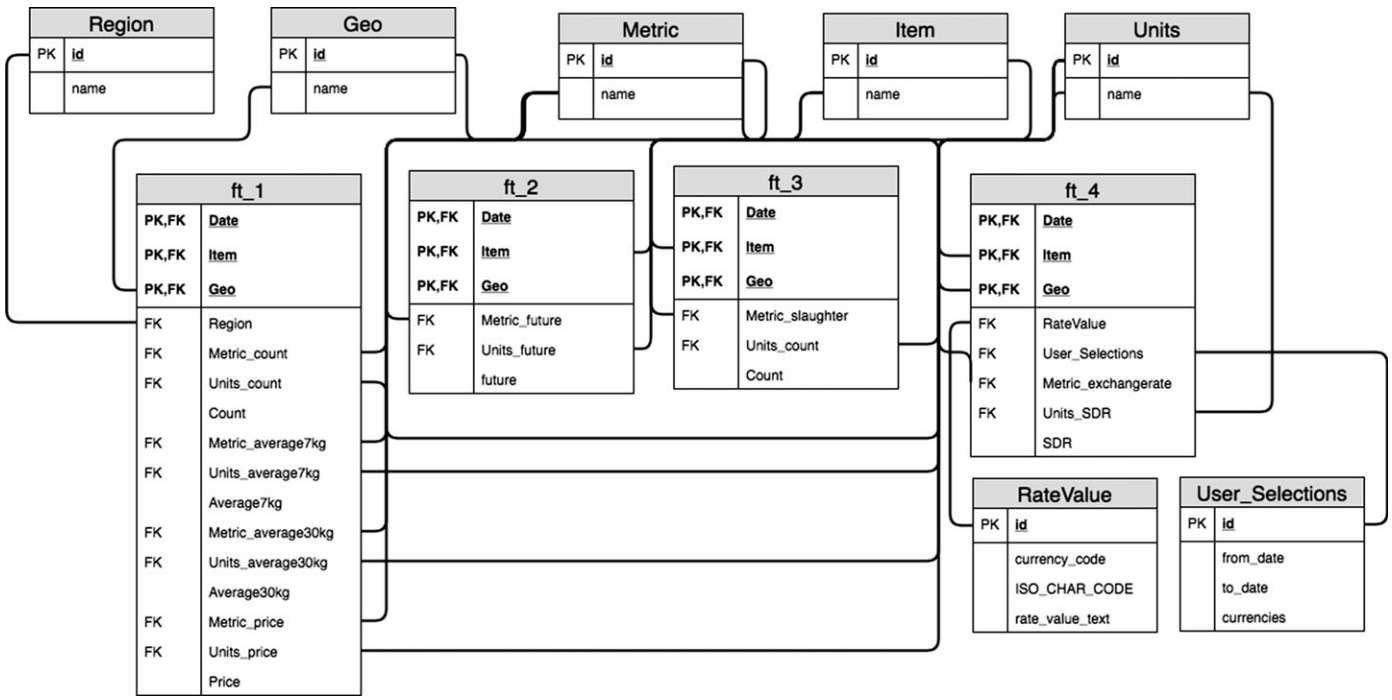

FIGURE 8. Ontology-assisted integration view.

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

while others provide statistics globally. The approach is to create a separate fact for each level of granularity. Later, a ROLLUP operation can be employed to join facts, i.e. aggregate a dimension to a higher level of granularity (e.g. Month → Year).

The second issue is labelled generically as `MISSING_ATTR`. This indicates that the data source did not contain enough information to correctly (semantically) integrate facts. In short, it means that the data source did not contain all of the attributes specified in the metamodel. Our approach is to highlight terms missing in the ontology. The most common reason is that data are sparse in terms of dimensions and attributes. When this occurs, it is impossible to infer these values.

The final issue is labelled as `TERM-TYPE_MISMATCH`, resulting from two issues. The first is the different terminologies used across data sources, particularly across the `Geo` dimension (for example 'US' *vs* 'America'). These issues are resolved by the ontology during the term mapping phase, where both possibilities are resolved into a single canonical term. The second problem originates from the lack of type information in both sources, and an issue arises when attempting to resolve different attributes and dimensions which are of the same abstract type. For example one source may have a dimension called 'France' and another 'Australia'. Without an ontology linking these two concepts under the common theme 'Country', they cannot be integrated.

For this case study, term and type mapping resolved integration problems for the `Geo` and `Item` dimensions, while the metamodel was used to enforce semantic integration by prompting a user for the `Metric` and `Unit` attributes.

6.5. Overall summary

The goal of our evaluation was to examine the differences between a manual integration approach and our automated approach; both to determine the value of our methodology and to identify potential improvements to our process. Table 7 outlines the results for all three approaches. Column `Name` describes the approach used: `user_defined` refers to the manual approach, `non_assisted` refers to non-ontology approach and `ontology_assisted` refers to our automatic approach with the ontology.

The columns `Sources` and `Instances` refer to the number of data sources involved in the integration, and the number of instances for each; `Metadata` refers to the number of attributes found in the multidimensional schema after integration; `Time` is the time to construct the final data mart; `Structure` relates to the usability of the data (yes/no); `Semantics` refers to the correctness of data (yes/no) and finally, the columns `Dimensions` and `Facts` relate to the number of dimensions and facts in the final data mart.

The column, `Time`, illustrates the savings in time using our approach. The manual approach required 8 hour approximately while the automated approach with the presence of the ontology was 11 minutes. In the earlier discussion on case studies, the automated time was reported as between 118 ms. However, here we include 1 minute for each user prompt and response (60 seconds was the longest time recorded). This clearly shows the benefit of the automated approach.

In terms of metadata, the manual, user-defined approach detected and removed more redundancies than the automated approach, and is due to the domain expert's knowledge of the dataset. While the semi-automatic non-assisted approach had similar results to the ontology-assisted approach, this is undoubtedly due to the non-assisted approach requesting information from the user as needed. This results in marginally slower times for batch updates as recorded in the case study discussion. The degree of redundant data determines the difference in data loading times and is our current area of research in terms of improving efficiency.

With regards to the size of the data marts generated, each approach generates a different number of (`Dimensions` and `Facts`) tables. The manual approach guarantees the best schema, at the cost of an extremely expensive upfront schema design. The automated approach is highly efficient: users are spared from the schema design, the schema is semantically sound and only slightly more complex than a manually designed one, and (most importantly) it can be generated in a matter of seconds instead of days. A limiting factor, perhaps, lies in the increased complexity of the schema generated. For example, a higher number of dimension and fact tables (and associated primary and foreign keys) leads to higher complexity for query developers and database administrators. Regarding the data mart created by the non-assisted approach, the number of constructed dimensions and facts highlights the need for an ontology to provide semantic mapping between attributes from heterogeneous sources.

However, the ontology-assisted approach needs two operations in order to achieve the same functionality: the removal

TABLE 7. Analysis of all case study approaches.

Name	Sources	Metadata	Instances	Time	Structure	Semantics	Dimensions	Facts
user_defined	13	24	262	8 hours	✓	✓	3	1
non_assisted	13	143	262	106 ms	✓	✗	66	7
ontology_assisted	13	24	262	11 m	✓	✓	7	4

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

of redundancies and a ROLLUP operation to check all facts to ensure they share the same level of granularity.

7. CONCLUSIONS

There are many websites generating information that covers a wide range of activities in the Agri sector. When properly processed, synchronized and aggregated, these sources can provide vital input for Agri decision makers. The main issues are that these data streams come and go, are prone to change, and can be costly to process for many Agri sector workers. In earlier work [11], we presented a method to convert smart city web streams into a StarGraph construct to enable the construction of a *single* source data mart from unseen source data. In this paper, we extend this work by integrating StarGraphs to create multi-source data marts which we call Constellations. For our evaluation, we used 120 unseen Agri data sources to first determine how many sources were usable by our system. This analysis showed that 70% of the sources were successfully transformed to our StarGraph model. We then worked with industry partners who provided the requirements for the case study and a list of data sources with which to create data marts. Our automated approach was shown to deliver considerable benefits, firstly by eliminating the need for manually constructing data marts; requiring a minimal effort by the data mart designer in terms of fully understanding the separate sources (user prompts for ontology updates refer only to specific parts of the schema); and finally, new information learnt about data sources is maintained in the ontology for future data mart construction.

FUNDING

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

REFERENCES

[1] Bergamaschi, S., Castano, S. and Vincini, M. (1999) Semantic integration of semistructured and structured data sources. *SIGMOD Rec.*, **28**, 54–59.

[2] Kittivoravitkul, S. and McBrien, P. (2005) Integrating unnormalised semi-structured data sources. In *Proc. Advanced Information Systems Engineering, 17th Int. Conf., CAiSE*, pp. 460–474. Springer-Verlag, Berlin.

[3] Inmon, W.H. (2002) *Building the Data Warehouse* (3rd edn). John Wiley & Sons, Inc, New York, NY, USA.

[4] McCarren, A., McCarthy, S., Sullivan, C.O. and Roantree, M. (2017) Anomaly detection in agri warehouse construction. In *Proc. Australasian Computer Science Week Multiconference, ACSW*, pp. 17:1–17:10. ACM, New York, USA.

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

[6] Batini, C., Lenzerini, M. and Navathe, S.B. (1986) A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv. (CSUR)*, **18**, 323–364.

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

[8] Roantree, M., Shi, J., Cappellari, P., O'Connor, M.F., Whelan, M. and Moyna, N. (2012) Data transformation and query management in personal health sensor networks. *J. Netw. Comput. Appl.*, **35**, 1191–1202.

[9] Martinez, J.M.P., Berlanga, R., Aramburu, M.J. and Pedersen, T.B. (2008) Integrating data warehouses with web data: a survey. *IEEE Trans. Knowl. Data Eng.*, **20**, 940–955.

[10] O'Donoghue, J., Roantree, M. and McCarren, A. (2017) Detecting feature interactions in agricultural trade data using a deep neural network. In *Big Data Analytics and Knowledge Discovery—19th Int. Conf., DaWaK*, pp. 449–458.

[11] Scriney, M., O'Connor, M.F. and Roantree, M. (2017) Generating cubes from smart city web data. In *Proc. Australasian Computer Science Week Multiconference, ACSW*, pp. 49:1–49:8. ACM, New York, USA.

[12] Skoutas, D. and Simitsis, A. (2007) Ontology-based conceptual design of etl processes for both structured and semi-structured data. *Int. J. Semantic Web Inf. Syst. (IJSWIS)*, **3**, 1–24.

[13] Priebe, T. and Pernul, G. (2003) Ontology-based integration of olap and information retrieval. In *Proc. 14th Int. Workshop on Database and Expert Systems Applications*, pp. 610–614. IEEE.

[14] Komamizu, T., Komamizu, T., Amagasa, T., Amagasa, T., Kitagawa, H. and Kitagawa, H. (2016) H-spool: a sparql-based etl framework for olap over linked data with dimension hierarchy extraction. *Int. J. Web Inf. Syst.*, **12**, 359–378.

[15] Ravat, F., Song, J. and Teste, O. (2016) Designing multidimensional cubes from warehoused data and linked open data. In *IEEE Tenth Int. Conf. on Research Challenges in Information Science (RCIS)*, pp. 1–12. IEEE.

[16] Bergamaschi, S., Guerra, F., Orsini, M., Sartori, C. and Vincini, M. (2011) A semantic approach to ETL technologies. *Data Knowl. Eng.*, **70**, 717–731.

[17] Romero, O., Simitsis, A. and Abelló, A. (2011) Gem: requirement-driven generation of etl and multidimensional conceptual designs. In *Int. Conf. Data Warehousing and Knowledge Discovery*, pp. 80–95. Springer, Berlin.

[18] Selma, K., IlyèS, B., Ladjel, B., Eric, S., Stéphane, J. and Michael, B. (2012) Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool. *Comput. Ind.*, **63**, 799–812.

[19] Romero, O. and Abelló, A. (2010) A framework for multidimensional design of data warehouses from ontologies. *Data Knowl. Eng.*, **69**, 1138–1157.

[20] Petrović, M., Vučković, M., Turajlić, N., Babarogić, S., Aničić, N. and Marjanović, Z. (2017) Automating etl processes using the domain-specific modeling approach. *Inf. Syst. e-Business Manage.*, **15**, 425–460.

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018

[21] Niinimaki, M. and Niemi, T. (2010) An etl process for olap using rdf/owl ontologies. *J. Data Semant. XIII*, **5530**, 97.

[22] Chamberlin, D., Simeon, J., Kay, M., Boag, S., Robie, J., Berglund, A. and Fernandez, M. (2010) XML path language (XPath) 2.0 (second edition). W3C recommendation. W3C. http://www.w3.org/TR/2010/REC-xpath20-20101214/.

[23] Quix, C. (2016) Data lakes: A solution or a new challenge for big data integration? *Proc. 5th Int. Conf. Data Management Technologies and Applications* 7. Springer, Berlin.

[24] Batini, C., Lenzerini, M. and Navathe, S.B. (1986) A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, **18**, 323–364.

[25] Scriney, M. (2018) Constructing Data Marts from Web Sources Using a Graph Common Model. PhD Thesis, Dublin City University.

[26] Barnaghi, P.M., Bermúdez-Edo, M. and Tönjes, R. (2015) Challenges for quality of data in smart cities. *J. Data Inf. Qual.*, **6**, 6:1–6:4.

[27] Agriculture and Agri-Food Canada http://www.agr.gc.ca/eng/home/?id=1395690825741 (accessed March 29, 2018).

[28] USDA https://quickstats.nass.usda.gov/ (accessed March 29, 2018).

[29] Bord Bia http://www.bordbia.ie/Pages/Default.aspx (accessed March 29, 2018).

[30] Agriculture and Horticulture Development Board http://pork.ahdb.org.uk/ (accessed March 29, 2018).

[31] CME Group https://www.cmegroup.com/ (accessed June 5, 2018).

[32] International Monetary Fund [Online]. http://www.imf.org/external/index.htm (accessed March 29, 2018).

Downloaded from https://academic.oup.com/comjnl/advance-article-abstract/doi/10.1093/comjnl/bxy064/5038272
by MIT Libraries user
on 19 June 2018