OLAP4LD – A Framework for Building Analysis Applications over Governmental Statistics

Benedikt Kämpgen and Andreas Harth

Institute AIFB, Karlsruhe Institute of Technology, Karlsruhe, Germany
benedikt.kaempgen@kit.edu, harth@kit.edu

Abstract. Although useful governmental statistics have been published as Linked Data, there are query processing and data pre-processing challenges to allow citizens exploring such multidimensional datasets in pivot tables. In this demo paper we present OLAP4LD, a framework for developers of applications over Linked Data sources reusing the RDF Data Cube Vocabulary. Our demonstration will let visiting developers and dataset publishers explore European statistics with the Linked Data Cubes Explorer (LDCX), will explain how LDCX makes use of OLAP4LD, and will show common dataset modelling errors.

1 Introduction

According to the G8 Open Data Charter and Technical Annex[1] governmental statistics provide data of high value for improving transparency and encouraging innovative re-use of data. In frontends such as Microsoft Excel, pivot tables have proved intuitive to build and easy to understand for exploring statistics. If published using Linked Data, statistics become easier to integrate with other data, e.g., the GDP of a country in one and the population in another dataset with linked country identifiers allow to compute the GDP per capita.

We have published more than 5,000 datasets from Eurostat[2] as Linked Data, yet, there are challenges to allow citizens to explore such datasets in pivot tables: Eurostat datasets exhibit varying number of dimensions, e.g., geo, time, gender and age for the population dataset, which makes visualisations, such as in two-dimensional line diagrams, and comparisons difficult. How to translate analytical operations to queries over Linked Data sources? How to reduce the dimensionality of datasets? Eurostat statistics are originally published using SDMX and re-published as Linked Data in a Google App Engine; other statistics such as from the World Bank are published differently. How to ensure that necessary data is extracted from distributed sources and properly modelled? How to pre-process and integrate heterogeneously modelled datasets?

Current systems [1, 5, 2, 4] do not help developers to build pivot analysis applications over general datasets. This demo paper provides the following contributions:

[1] https://www.gov.uk/government/publications/open-data-charter/g8-open-data-charter-and-technical-annex

[2] http://estatwrap.ontologycentral.com/table_of_contents.html

(1) In Section 2, we present OLAP4LD, a development framework for applications over Linked Data sources reusing the RDF Data Cube Vocabulary.

(2) In Section 3, we present the Linked Data Cubes Explorer (LDCX) that is based on OLAP4LD and allows the exploration of governmental statistics.

After describing related work in Section 4, we conclude in Section 5.

2 OLAP4LD Integration and Analysis Framework

OLAP4LD[3] is an Open Source Java framework for building analysis applications over statistics published as Linked Data. As illustrated at the top of Figure 1, OLAP4LD implements olap4j, a standard interface between OLAP frontends and backends. Application developers can make use of a common abstraction of datasets as data cubes, the quasi-standard analytical query language MDX, and existing olap4j clients such a Saiku and JPivot. As illustrated at the bottom of the architecture, OLAP4LD provides access to multidimensional datasets published as Linked Data reusing the W3C-standardised RDF Data Cube Vocabulary (QB). QB allows to represent general datasets such as statistics or from sensors and is widely-adopted.

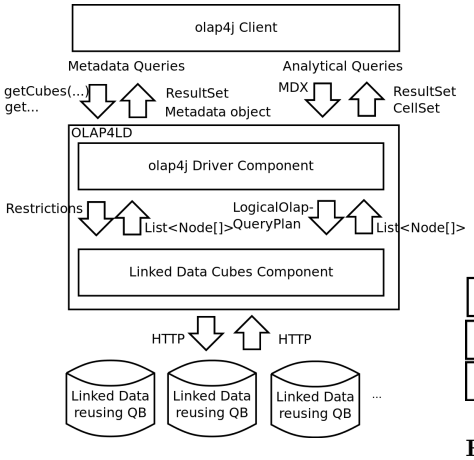

Fig. 1. Components of OLAP4LD.

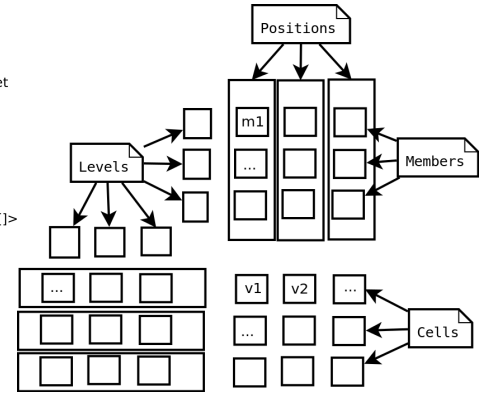

Fig. 2. Pivot table schema from a typical MDX query.

OLAP4LD roughly consists of two components. The *olap4j Driver Component* translates queries from an olap4j client to queries more suitable for processing over Linked Data sources in the *Linked Data Cubes Component*. Vice-versa, the olap4j component translates results from Linked Data sources to representations understandable by the client.

Metadata queries are methods such as `getCubes(...)` and `getMeasures(...)` that return multidimensional elements, i.e., data cubes containing facts with members for a pre-defined set of dimensions (independent variables) that determine the value of one or more measures (dependent variables). In Linked Data,

[3] `http://linked-data-cubes.org/index.php/Olap4ld`

multidimensional elements are identified and described using sets of RDF terms; OLAP4LD represents RDF terms using `Node`s from the *NxParser* library. The schema of multidimensional elements from Linked Data, i.e., the name and types of columns of `List<Node[]>` is adopted from the olap4j specification.

Analytical (MDX) queries return data from a data cube to be displayed in a two-dimensional pivot table as illustrated in Figure 2: one or more queried data cubes, lists of member combinations (positions) from a fixed set of levels for rows and columns, and member combinations from a fixed set of levels as filter conditions. For instance, from a Eurostat dataset "Employment Rate", we may query for a pivot table containing sex as dimension on columns (e.g., position `F`), time and place as dimensions on rows (e.g., position `2005`, `AT`). Results of analytical queries fill the cells of the pivot table, e.g., that 64.9% of women in Austria in 2005 have been employed.

OLAP4LD defines an analytical query as a nested set of common operators from an OLAP algebra: For every queried data cube, `Base-Cube` loads the relevant data defined by the dataset URI; for the chosen measures or if no measure is chosen the first measure, `Projection` removes not selected measures; for every possible member combination on axes, `Dice` removes filtered dimension members; `Slice` removes every dimension not mentioned in either column or row axis (i.e., aggregates over with aggregation function of measures); and for any higher level selected on columns or rows axes, `Roll-Up` aggregates dimensions to higher levels. Finally, all resulting data cubes are joined via `Drill-Across`. Note, the drill-across operator requires as input equally-structured data cubes. For more information about the definition of analytical queries, see the documentation and previous work [3].

In the *Linked Data Cubes Component*, for metadata queries, any instance of `qb:DataSet` is mapped to a data cube. Similarly, other resources represented in QB are mapped to multidimensional elements. For analytical queries, OLAP operators are executed over instances of `qb:DataSet`.

Queries can be executed in different ways, e.g., reusing a common OLAP engine over relations or in-memory, and directly with an RDF store; aggregated values from the data cube may be computed on demand or views selected and maintained; similarly, data pre-processing and integration can be done differently, e.g., a database may be pre-filled with all relevant data in advance or populated dynamically; also, there are various ways with which information can be provided, e.g., packed in data dumps or queryable from several SPARQL endpoints. For executing metadata and analytical queries, an OLAP4LD application has to implement a *Linked Data Cubes Engine*. Developers can reuse existing engines and concentrate on the challenges of query execution and integration over Linked Data sources.

3 OLAP4LD for the Linked Data Cubes Explorer

The Linked Data Cubes Explorer (LDCX)[4] is based on OLAP4LD and allows citizens to explore governmental statistics. In our demonstration, we will let vis-

[4] `http://ldcx.linked-data-cubes.org/projects/ldcx/`

itors try the three-step user interface of LDCX: 1) a user selects one or more comma-separated URIs of *qb:DataSet*s. With "Explore Dataset…", metadata queries are issued to populate the user interface; 2) the user selects measures to be displayed in the pivot table cells; 3) the user selects dimensions to add member combinations to rows and columns of the pivot table and clicks "Update Table…". Note LDCX automatically queries every dimension on the most granular level since multi-level hierarchies are rarely used and users can still slice dimensions to view datasets on a higher aggregation level. In our demonstration we will show how changes in modelling are propagated to LDCX by live modifying a published QB dataset. Also, we show common modelling errors in existing QB datasets such as missing dimension `rdfs:range` or `qb:CodeList` and observations not adhering to data structure definitions.

LDCX implements an *EmbeddedSesameEngine* as Linked Data Cubes Engine that evaluates metadata queries using SPARQL templates filled with $Node$ parameters. For each multidimensional element, there are several SPARQL templates for different ways of modelling, e.g., measures can define their own aggregation functions or AVG and $COUNT$ are used by default. To evaluate analytical queries, the logical query plan is translated to a physical query plan; for each separate drill-across sub-query plan, we execute our *OLAP-to-SPARQL* algorithm [3] and join the results. Before executing a metadata or analytical query with SPARQL, the EmbeddedSesameEngine automatically loads necessary data into an embedded Sesame RDF store. EmbeddedSesameEngine first resolves all queried dataset URIs, then in turn asks SPARQL queries to its store for additional URIs to resolve and load; EmbeddedSesameEngine resolves all instances of concepts defined in the QB specification in the order they can be reached from the dataset URI, from `qb:DataStructureDefinition`s over `qb:ComponentProperty` to single `qb:Concept`s. Since there is no standard way to publish QB observations, the engine assumes that the observations are represented as blank nodes and stored at the location of the dataset URI. Such "directed crawling" of the data cube has the advantage that necessary data is found quickly and not all information has to be given in one location, but can be distributed and reused, e.g., the range for the *ical:dtstart* dimension is provided by its URI. EmbeddedSesameEngine ensures that the entire QB dataset is loaded and well-formed according to the QB specification; SPARQL queries materialise implicit information and check integrity constraints.

For an online questionnaire not considering drill-across, 8 of 20 asked business engineering students at KIT used LDCX in 11 tasks, e.g., to find "the average GDP for Germany" in example and real datasets and rated the system according to 13 statements with average 2.5 from 1 (strongly agree) to 10 (strongly disagree), e.g., regarding usability. LDCX seems usable and robust; improvements are possible regarding slow performance, counter-intuitive error messages and cumbersome selection of datasets. For a workload of 5 drill-down and 5 slice queries over datasets with 10 to 1000 observations we observed that elapsed query time was mainly spent for loading the datasets and much less for query plan generation and execution.

4 Related Work

The most common format to share datasets is XML. Other representations such as the Google *Dataset Publishing Language*, *SDMX* and *XBRL* require specific tools or focus on specific domains and provide few possibilities to link, and less-widely adopted mechanisms to access data over the Web.

Applications are available that, similar to LDCX, try to hide most RDF-specificities from the user to analyse a QB dataset. In the *stats.270a.info* analysis platform [1] users can select two datasets from a fixed set for integration on the time and location dimension and for finding correlations in a scatter plot. McCusker et al. [4] present *qb.js* to analyse the effect of tobacco policies on consumption. Though presenting useful systems to analyse QB datasets in specific data integration scenarios, it is unclear how well such approaches can be applied to more general use cases. Other systems provide more general analyses: Salas et al. [5] present *CubeViz* that offers faceted-browsing and visualizations of QB datasets. Hoefler [2] present the *CODE Visual Analytics Wizard* that automatically suggests appropriate chart types for QB datasets. LDCX automatically loads and checks the modelling of datasets and allows exploration of general datasets in pivot tables. Although the interface of LDCX is not as nice as of other systems, we argue that OLAP4LD reduces the costs of building analysis applications since UIs and backends are separated and can be reused.

5 Conclusions

In this demo paper, we have presented OLAP4LD, a framework for building analysis applications with Linked Data reusing the RDF Data Cube Vocabulary. In a demonstration of OLAP4LD we allow visitors to validate and explore governmental statistics with the Linked Data Cubes Explorer (LDCX).

References

1. Capadisli, S., Auer, S., Riedl, R.: Linked statistical data analysis. In: Semantic Web Challenge 2013 (2013)
2. Hoefler, P.: Linked Data Interfaces for Non-expert Users. In: The Semantic Web: Semantics and Big Data. Springer Berlin Heidelberg (2013)
3. Kämpgen, B., Harth, A.: No Size Fits All - Running the Star Schema Benchmark with SPARQL and RDF Aggregate Views. In: ESWC (2013)
4. McCusker, J.P., McGuinness, D.L., Lee, J., Thomas, C., Courtney, P., Tatalovich, Z., Contractor, N., Morgan, G., Shaikh, A.: Towards Next Generation Health Data Exploration: A Data Cube-Based Investigation into Population Statistics for Tobacco. In: Proceedings of the 2013 46th Hawaii International Conference on System Sciences (2013)
5. Salas, P.E.R., Martin, M., Mota, F.M.D., Auer, S., Breitman, K., Casanova, M.A.: Publishing Statistical Data on the Web. In: Proceedings of the 2012 IEEE Sixth International Conference on Semantic Computing (2012)