Linked Open Statistical Data API: requirements and design criteria

xxx ddd and yyy sss

No Institute Given

Abstract.

1 Introduction

Recently, many governments, organisations and companies are opening up their data for others to reuse through *Open Data* portals [6]. These data can be exploited to create added value services, which can increase transparency, contribute to economic growth and provide social value to citizens [4].

A major part of open data concerns statistics (e.g. economical and social indicators) [2]. These data are are often organised in a multidimensional way, where a measured fact is described based on a number of dimensions. In this case, statistical data are compared to data cubes. Thus, we onwards refer to statistical multidimensional data as *data cubes* or just *cubes*.

Linked data has been introduced as a promising paradigm for opening up data because it facilitates data integration on the Web [1]. Concerning statistical data, the RDF data cube (QB) vocabulary enables modelling data cubes as linked data [3]. In this way it facilitates their integration. Data provided using the QB vocabulary can be accessed using the existing machinery of Linked Data. However, skills and tooling for use of Linked Data (e.g. RDF, SPARQL) are less widespread than some other web technologies (e.g. REST, JSON). For example, there are many visualization libraries that consume data in JSON format (e.g D3.js, charts.js), while there are just a few that consume RDF. That's one of the reasons that there are not so many application for linked open statistical data [REF???].

Moreover, many portals that use the QB vocabulary often adopt different publishing practices [5], thus hampering their interoperability. As a result it is not easy to create generic software tools that operate across linked open statistical datasets. Usually, case specific software are created which assume that linked statistical data are published in a specific way.

In this paper we describe the requirements and design criteria of an API that standardizes the interaction (i.e. input and output) with Linked Open Statistical Data in a way that facilitated the development of generic software. In this way we keep the advantages of linked data (e.g. data integration) but hide all the complexity by supporting developers to use linked statistical data stored in the form of an RDF Data Cube, while assuming minimal knowledge of Linked Data technologies. Moreover, the API offers a uniform way to access the underlying data hiding any data discrepancies, thus enabling the development of generic software tools that operate across datasets.

2 Methodology

In order to achieve the objectives of the paper we adopt the following methodology:

– Study the related work. We focus on existing APIs that standardize the interaction with multidimensional data, as well as on existing data formats that can be exploited to represent the result (output) of the API.
– Collect user requirements. We organized a workshop to collect the needs and requirements of developers that currently develop applications for Linked Open Statistical data, using the “traditional” tooling (e.g. RDF, SPARQL, QB vocabulary).

Currently, there exist some API that handle data cubes and facilitate the performing of online analytical processing (OLAP) operations. OLAP enable users to analyze multidimensional data interactively from multiple perspectives. An API that handles data cubes is developed by Oracle [7]. The API allow users to select, aggregate, calculate, and perform other analytical tasks on data stored at Oracle data warehouse. It also enables an application to explore the metadata of a cube.

Olap4j[1] is another API for accessing multi-dimensional data. It is an open source Java API compatible with many OLAP server (e.g. Mondrian, Palo and SAP BW) which supports MDX queries. Multidimensional Expressions (MDX) is a query language for OLAP. Olap4j enables also the browsing of metadata including the cubes, dimensions, hierarchies and members in an OLAP schema.

There exist also other APIs for different programming languages and data stores offering similar functionality. For example the Data Brewery[2] offers a set of Python frameworks and tools for data processing and analysis which includes an API that covers the cubes logical model metadata and aggregation browsing functionality. Apache Lens[3] is a unified analytics platform that aims providing a single view of data across multiple data stores. Lens offers also an API to handle data cubes.

All the above APIs handle multidimensional data, but they are based on traditional databases and data-warehouwse. None of them handles LOSD.

Regarding the output of an API, JSON is an already successful and simple format. Some JSON extension formats have been proposed to model statistical data and linked data. Specifically, JSON-stat is a simple lightweight JSON format for statistical data and metadata exchange. While JSON-LD is a lightweight Linked Data format. It a way to help JSON data interoperate at Web-scale.

Related work:

[1] http://www.olap4j.org
[2] http://databrewery.org/
[3] lens.apache.org

– OLAP APIs interaction with multidimensional data (input): Oracle OLAP API [1], Olap4j [2], ++
– Standardization of outcome: Json-stat, Json-ld, ++

Discussion with developers: Workshop, +++

3 Solution overview

A large part of statistical data is isolated, meaning that exists in different portals as different datasets. The technology of Linked data and RDF Data Cube vocabulary solves the problem of distributed data sources through their integration, creating interoperable linked statistical data portals. These portals are used for the design and creation of a Linked Open Statistical Data API.

The architecture of the API is developed in JSON format offering benefits which are not exist until now. The implementation of JSON-API abolishes the need to implement different data access layers for each tool is created. In the traditional architecture data access layers had to be coded separately leading to additional costs. The JSON-API can be installed on top of any RDF repository and offers basic and advanced operations on RDF Data cubes.

Moreover, using JSON for accessing RDF Stores is an easier way to build software applications. Developers need to know and use neither RDF Data Cube vocabulary nor expert programming skills at LOSD. They have full access at LOSD just using JSON, a much more common and understandable programming language. Through SPARQL queries, JSON-API has access at data cubes returning the asked data in the re-used format of JSON. By this way, all LOSD-related programming and its complexity is now hidden.

As figure 1 shows, JSON-API helps developers create custom applications according to their needs. JSON language is commonly re-used by many libraries offering data representation and visualization. +++

4 Requirements and design criteria

– need to know what datasets are available
– need to know about structure to subset the observations
– in order not to return everything, need to subset
– don't necessarily need a n-array/ tabular response - array of observations is sufficient. can always get back to the table
– Filtering
– Multilinguality
– Ordering & paging
– merging, aggregations
– json-ld representation is sufficient for query and response format
– ++

API functionality:

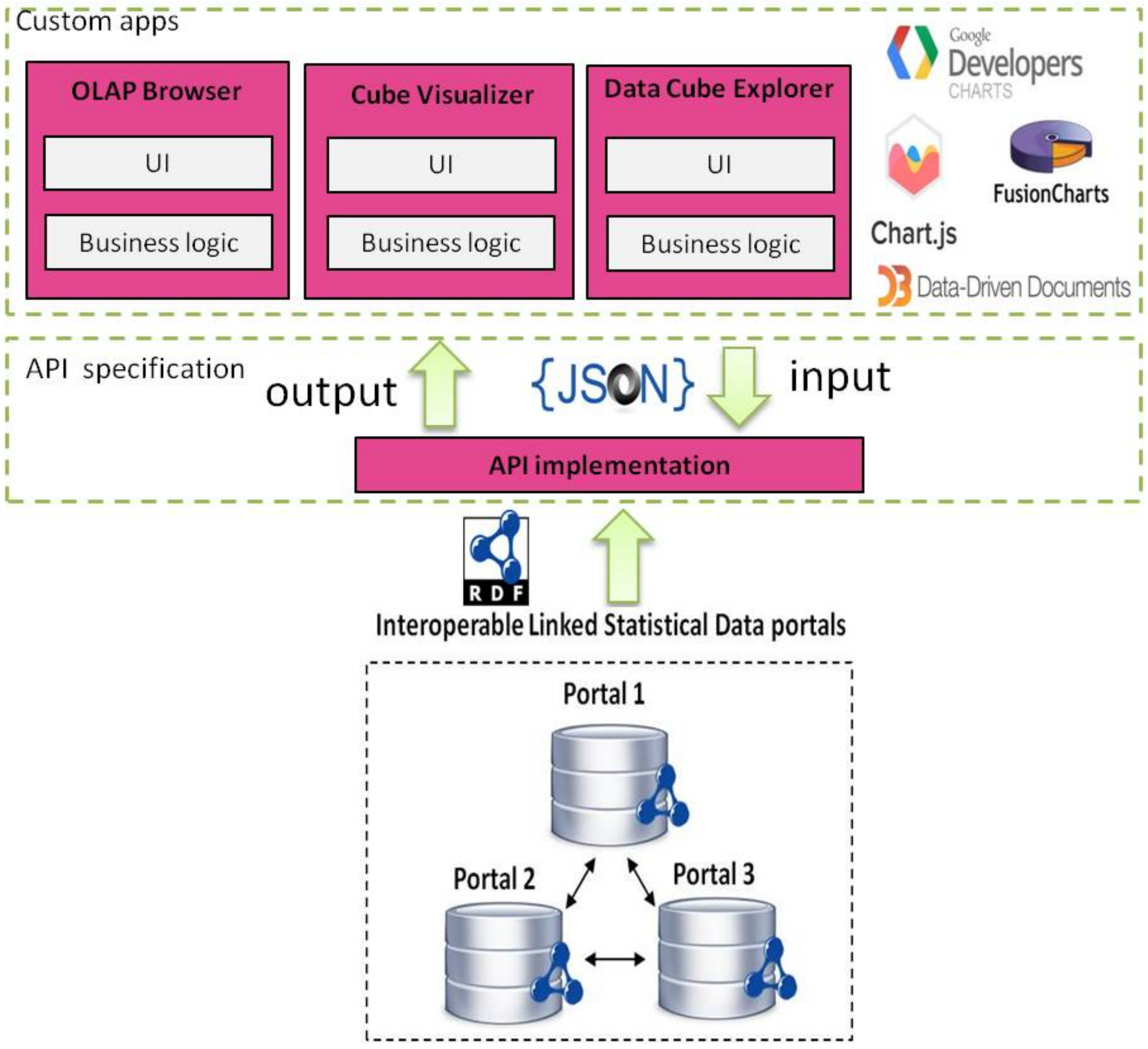

Fig. 1. Solution overview

- GET dataset-metadata
- GET dimensions
- GET attributes
- GET measures
- GET dimension-values
- GET attribute-values
- GET dimension-levels
- GET slice
- GET table
- GET cubes
- GET aggregationSetcubes
- GET create-aggregations
- GET cubeOfAggregationSet

possible example for slice/ observation-selection query:

```
{
  "jqql:dataset": "scot:home-care-clients",
  "jqql:filter": {
"dimension:gender": "gender:male",
   "dimension:age": { "jqql:greater-than": 50 }
  },
  "jqql:order": {
    "dimension:refPeriod": { "jqql:order-predicate": "ui:sortPriority", "jqql:direction": "
  },
  "jqql:page": {
    "jqql:limit": 10,
    "jqql:offset": 0
  }
}
```

output:

```
{ "observations": [
{ "Average Cost": "1182",
      "Date": "1-1-2013",
  "Day": "Tuesday",
  "Number of crashes": "5",
  "Time": "No available time",
      "Total Cost": "5908",
  "@id": http://id.mkm.ee/observation/1" },
{ "Average Cost": "400",
  "Date": "1-1-2013",
  "Day": "Tuesday",
  "Number of crashes": "1",
  "Time": "24:00",
    "Total Cost": "400",
  "@id": "http://id.mkm.ee/observation/2" }
]}
```

5 Implementation

6 Conclusion

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
2. Capadisli, S., Auer, S., Ngonga Ngomo, A.C.: Linked sdmx data. Semantic Web 6(2), 105–112 (2015)
3. Cyganiak, R., Reynolds, D.: The RDF data cube vocabulary: W3C recommendation. Tech. rep., W3C (January 2014)
4. Janssen, M., Charalabidis, Y., Zuiderwijk, A.: Benefits, adoption barriers and myths of open data and open government. Information Systems Management 29(4), 258–268 (2012), http://dx.doi.org/10.1080/10580530.2012.716740
5. Kalampokis, E., Roberts, B., Karamanou, A., Tambouris, E., Tarabanis, K.: Challenges on developing tools for exploiting linked open data cubes. In: Proceedings of the 3rd International Workshop on Semantic Statistics (SemStats2015) within the 14th International Semantic Web Conference (ISWC2015). vol. 1551. CEUR-WS (2015)
6. Kalampokis, E., Tambouris, E., Tarabanis, K.: A classification scheme for open government data: towards linking decentralised data. Int. J. Web Eng. Technol. 6(3), 266–285 (Jun 2011), http://dx.doi.org/10.1504/IJWET.2011.040725
7. Oracle: Oracle olap developer's guide to the olap api, 10g release 2 (10.2) (2006)