Building an Open Data Visualisation Web App using a Data Server: The Styrian Diversity Visualisation Project

Keith Andrews
Institute for Information Systems and Computer Media,
Graz University of Technology,
Inffeldgasse 16c,
8010 Graz, Austria
kandrews@iicm.edu

Thomas Traunmüller
Institute for Information Systems and Computer Media,
Graz University of Technology,
Inffeldgasse 16c,
8010 Graz, Austria
ttraun@iicm.edu

Thomas Wolkinger
Institute of Journalism and Public Relations,
FH Joanneum,
Alte Poststraße 152,
8020 Graz, Austria
thomas.wolkinger@fh-joanneum.at

Robert Gutounig
Institute of Journalism and Public Relations,
FH Joanneum,
Alte Poststraße 152,
8020 Graz, Austria
robert.gutounig@fh-joanneum.at

Julian Ausserhofer
Institute of Journalism and Public Relations,
FH Joanneum,
Alte Poststraße 152,
8020 Graz, Austria
julian.ausserhofer@fh-joanneum.at

ABSTRACT

Statistical open data is usually provided only in the form of spreadsheets or CSV files, which can sometimes be very large. The writer of an open data app is confronted with two choices: restrict themselves to managable bite-sized chunks of data, which can be consumed (read, parsed, and held in memory) in one go, or install and maintain their own data server which the app can query on demand.

The Styrian Diversity Visualisation project was conceived to visualise the diversity of inhabitants of the Austrian Province of Styria (Land Steiermark) using open data served from a data server (triple store). The corresponding web app queries the data server at run rime with a SPARQL query to obtain exactly the data required at that particular time, greatly simplifying its internal logic. There is no need to parse and store entire data sets in memory.

The data server is an instance of a Virtuoso Open Source server. The web app (client) is written in HTML5 and uses the leafletjs JavaScript library to provide mobile-friendly interactive maps. The user interface was designed as a set of three stories, each guiding users through a scenario with accompanying interactive visualisations based on corresponding open data sets.

CCS Concepts

•**Information systems** → *Graph-based database models; Mashups; Resource Description Framework (RDF);* •**Human-centered computing** → *Visualization; Visualization techniques; Geographic visualization;*
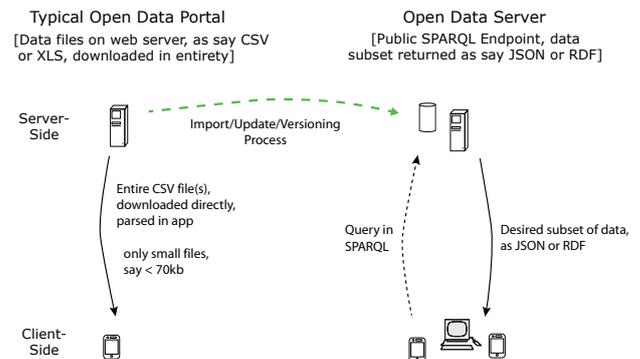
Figure 1: On the left, a typical open data portal, serving data files in CSV format. Apps can only manage bite-sized chunks at a time, since they have to be parsed and held in memory. On the right, a data server accepts a query from an app in SPARQL and returns only the requested subset of data on demand.

Keywords

Open data, data server, triple store, SPARQL, web app, HTML5, responsive, geovis, leaflet-js, information visualisation

1. INTRODUCTION

Thanks to various initiatives around the globe, open data [11] is becoming increasingly more prevalent. Local, regional, and national governements [7, 10, 17], institutions [6], and even private organisations [9] are publishing open datasets. Particularly for statistical data, most open datasets are made available as spreadsheet files (csv, xls) hosted on a data portal. This is illustrated on the left side of Figure 1.

On occasion, open data providers have been known to ask "Why do so few people download our open data?" or "Why are there so

few apps?" The former can be explained by the multiplier effect of a single download being propagated on, for example a dataset integrated into OpenStreetMap, which although being of benefit to thousands of users would only show up as a single download. The latter has perhaps more to do with the way open data is provided: only the very smallest datasets can be loaded directly from the open data portal as say CSV into the app, be parsed, and then held in memory by the app. When an app can fetch data on demand from a server over an API, the app can remain lean and slim, concentrating on its inherent purpose. Indeed, in the rare cases where open data is made available through a query interface (API) on a server, for example for local transport data [20] and map-based data [13], there are significant numbers of apps which make use of that data.

The emerging technologies of the semantic web (linked open data [3], triple stores [15], and the SPARQL query language [19]), provide a standardised infrastructure to host statistical open data on a server, which can be accessed on demand as necessary. The research goals of this project are threefold:

- To gather first-hand knowledge and experience of refining, publishing, and maintaining statistical open data on a data server (triple server with SPARQL endpoint) and to document and publish this for the community.

- To build a technology demonstrator webapp, in the context of a real project, using Linked Open Data and SPARQL to dynamically fetch data on demand, which can be used as a template for future webapps.

- To create a convincing webapp demonstrating the diversity of the inhabitants of the Province of Styria and thus promote integration and understanding.

Ultimately, it is hoped that this project will be a first step towards local government in the future providing statistical open data not only as spreadsheet files, but also hosting such data on their own open data servers, public SPARQL endpoints with guaranteed availability and performance.

2. LINKED OPEN DATA AND DATA SERVERS

Linked Open Data (LOD) refers to a set of five best practices for publishing and connecting structured open data on the web as defined by Tim-Berners Lee in 2010 [3]. Data servers, or triple stores [15], are databases which store data as RDF triples [18], and are queried using SPARQL queries [19]. These technologies form the basis for the semantic web or *data web* [16].

Numerous implementations of triple stores are available, both open source and commercial. For smaller projects, Fuseki [2] is a good choice. For larger projects, Virtuoso Open Source Edition [12] is often used. Figure 1 illustrates the difference between using data files and a data server for a LOD project. With data files (on the left hand side), an app can only download byte-size pieces of data, which are managable and can be held in memory inside the app. With a data server (on the right hand side), an app can remain lean, with no need for code to parse a CSV file and no need to hold entire data sets in memory, but simply issuing a SPARQL query whenever data is needed and *only* for that data which is needed. The desired subset of data is typically delivered to the app in JSON or RDF format.

Of course, the extra functionality and flexibility of a data server does not come for free. The open dataset must be transformed into (RDF) triples and imported into the data server. Procedures must be in place to consistently name data fields and provide for versioning.
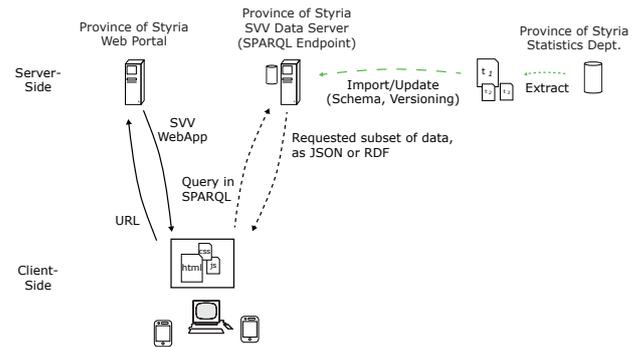

Figure 2: The client-server architecture of the SVV webapp. The user enters a URL into their web browser to request webapp, which is hosted on the Province of Styria's web server. The webapp is downloaded into the web browser and runs there. The webapp then requests any data it needs on demand using a SPARQL query from the Province of Styria's SVV data server.

Figure 3: The initial display of the SVV app on a wider screen.

Running a data server also requires regular backups, maintenance procedures, security updates, and monitoring, all of which require significant commitment on the part of potential hosters.

3. THE STYRIAN DIVERSITY VISUALISATION (SVV) PROJECT

The Styrian Diversity Visualisation (in German "Steirische Vielfalt Visualisiert" or SVV) project comprises a webapp and a data server, as shown in Figure 2. The SVV webapp is hosted on the official web server of the Province of Styria (Land Steiermark). The user enters the URL of the webapp into their web browser, and it is downloaded into the browser to run. As and when the webapp needs data, it requests it on demand by sending a SPARQL query to the Province of Styria's SVV data server. The data requested (and only that subset) is then returned to the webapp, which displays it in an appropriate visualisation.

The project began with user research and a thorough content audit. In parallel, international best practice examples of data visualisation projects in similar areas were looked at, in order to generate ideas for user scenarios. This in turn led to a content strategy and interface specification document.

After several rounds of discussion with the project partners, three scenarios began to crystallise. All three scenarios have in common
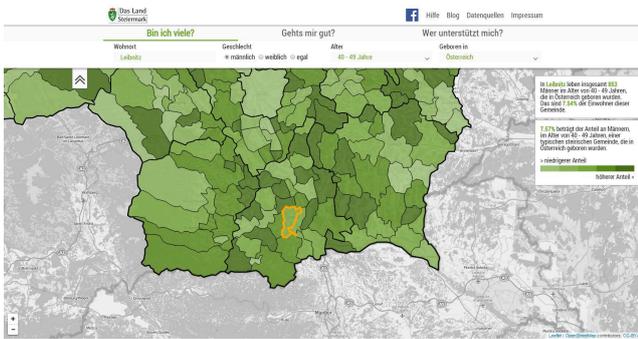
Figure 4: As the user reveals more about themselves, the SVV app filters the data and regenerates the choropleth map accordingly.

the idea of progressive disclosure: that the user should disclose a piece of information (say, their place of residence or age range), then in return should be rewarded with a visualisation of corresponding statistics. The three scenarios are:

1. *"Bin ich viele?" ("How Many am I?")*: The user is prompted to enter their place of residence, gender, age range, and place of birth, in that order. In return, a choropleth map of Styria is successively updated to reflect the distribution of other people in the same demographic.

2. *"Geht's mir gut?" ("How am I Doing?")*: In a similar vein to Scenario 1, the user is prompted to enter their place of residence, their employment status, and their gender. In return again, corresponding statistics are successively updated in a choropleth map of Styria.

3. *"Wer unterstützt mich?" ("Who Can Support Me?")*: In this scenario, the user is asked to disclose their place of residence and which kind of support they are interested in. The map of Styria is updated with markers showing the locations of individual integration partners.

The initial screen of the SVV webapp is shown in Figure 3. Towards the top of the page, Scenario 1 "Bin ich viele?" ("How Many am I?") is selected by default. In the input panel, the four input fields are successively activated. Upon each input, the choropleth map updates to reflect the current state of the user's input. Figure 4 shows the display after three of the four inputs have been entered. Supplementary information is displayed in the information boxes to the right hand side. If the user mouses over (or taps) on the map, another information box opens to display detailed information for that specific district or community.

The webapp is written in HTML5 with CSS3 and Javascript as a responsive webapp. Responsive web design (RWD) [8] is the new approach to building web sites and web apps which respond to the characteristics of the display device. The same web page adapts its display to fit the available screen width: on narrow screens, some blocks of content are relocated, menus are collapsed and placed behind a button, font sizes and spacing are reduced, and so forth. The web page asks whether the display device supports touch interaction, for example, and if so, loads appropriate code to support touch events. Figure 5 shows the webapp in a narrow window. The menus have been collapsed, only one of the four input fields is visible at any one time, and the information boxes have been moved to the bottom of the screen.

On the data server side, an instance of Virtuoso Open Source Edition (VOS) runs on a Linux server with 16 gb of RAM and 4
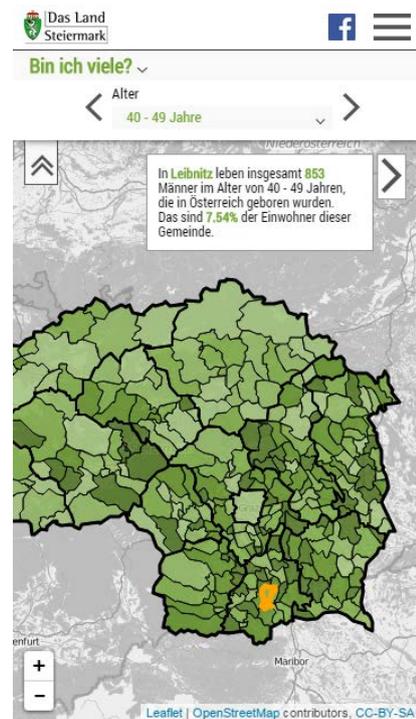

Figure 5: The SVV app is responsive. On a narrower screen, the design adapts to having less space available. Touch interaction is enabled on supported devices.

cores. Individual data sets are prepared and then converted into LOD format (triples) using LODRefine [21]. They can then be uploaded onto the VOS data server. The VOS server is configured to use its standard port 8890 and http as the SPARQL endpoint to which client apps can send SPARQL queries. The first time a new dataset is converted, a script can be written to semi-automate the process for later versions of the same data, say for data issued yearly. In the test phase, both the web app and the data server were hosted on Linux servers at Graz University of Technology. For the publicly available app, they will be hosted on servers belonging to the Province of Styria.

A test app was written to emulate the data requests of the real app, as shown in Figure 6. By manipulating the user interface on the left, a SPARQL query is formulated on the right. The query can then be submitted to the data server and the data records returned are displayed in the bottom box.

The choropleth map visualisation in the web app is realised using the Leaflet open source Javascript library [1]. Leaflet, in turn, fetches map tiles from a map server. Currently, the SVV web app uses map tiles from the OpenStreetMap tile server.

4. RELATED WORK

The "Qui Sommes Nous?" ("Who Are We?") visualisation by the city of Rennes [5, 14] has similar goals to the Styrian Diversity Visualisation project: to promote open data, "to improve the understanding of census data," and to "spark a debate around the sense of belonging to the urban community.". It also has a similar user-driven approach, where the user is first asked to disclose something about themselves (gender then age) before seeing a personalised perspective of the data. However, the Rennes visualisation uses quite different technology. Data are loaded as multiple individual

SELECTION

Gemeinde: Gemeinde eingeben

Geschlecht: ○ männlich ● weiblich ○ egal

Alter: ————●———————— 30 - 39 Jahre

Geburtsland: Geburtsland eingeben

SPARQL-QUERY

```
prefix communes: <http://data.steiermark.at/communes#>
prefix countries: <http://data.steiermark.at/countries#>
prefix country_born: <http://data.steiermark.at/people/country_born#>
SELECT ?commune_name ?commune_code (sum(?commune_count) as ?count)
WHERE {
  GRAPH <http://data.steiermark.at/2014-01-01/people/country_born> {
    ?dataset country_born:communeCode ?commune_code .
    ?dataset country_born:sex 2 .
    ?dataset country_born:ageGroup 4 .
    ?dataset country_born:countryCode ?country_code .
    ?dataset country_born:count ?commune_count .
  }
  GRAPH <http://data.steiermark.at/2014-01-01/communes> {
    ?none2 communes:communeCode ?commune_code .
    ?none2 communes:communeName ?commune_name .
  }
}
ORDER BY asc(?commune_name)
```

submit

RESULTS | 210.0 ms   switch view

Gemeindename	Gemeindekennzahl	Einwohner
Pusterwald	62021	20
Hohentauern	62010	20
Wildalpen	61251	23
Radmer	61112	27
Sankt Kathrein am Hauenstein	61744	31
Wald am Schoberpaß	61119	32

Figure 6: This test app emulates the data requests of the real app. By manipulating the user interface on the left, a SPARQL query is formulated on the right. The query can then be submitted to the data server. The set of data records returned are displayed in the bottom box.

CSV files, which are then parsed and held in memory in the web browser. Graphical output is performed using the D3.js [4] library, which essentially means dynamically injecting SVG nodes into the DOM.

5. CONCLUDING REMARKS

The Styrian Diversity Visualisation (SVV) project was conceived to visualise the diversity of inhabitants of the Province of Styria (Land Steiermark) using open data served from a data server. The knowledge and experience gathered will hopefully benefit future open data webapp projects in Styria and beyond.

The webapp uses only open data, which is freely available via the Province of Styria open data portal. The webapp code developed during the SVV project will be released as open source and the project documentation will be made publicly available. The code libraries in the webapp used are all open source, as is the Virtuoso Open Source Edition data server [12]. The project is due to launch to the public in autumn 2015.

At some point in the future, it is hoped that the Province of Styria and many other open data providers will provide their statistical open data not only as data files on open data portals, but also on publicly accessible data servers with SPARQL endpoints.

6. ACKNOWLEDGEMENTS

References

[1] Vladimir Agafonkin. *Leaflet*. 24th Jun 2015. http://leafletjs.com/.

[2] Apache. *Fuseki2*. Apache Jena Project. 24th Jun 2015. http://jena.apache.org/documentation/fuseki2/.

[3] Tim Berners-Lee. *Linked Data*. 2010. http://w3.org/DesignIssues/LinkedData.

[4] Mike Bostock. *D3: Data-Driven Documents*. May 2015. http://d3js.org/.

[5] Dataveyes. *A Data-Driven Snapshot of the Population of Rennes*. blog post. Jun 2014. http://dataveyes.com/#!/en/case-studies/rennes-metropole.

[6] Eurostat. *Eurostat*. European Commission. 24th Jun 2015. http://ec.europa.eu/eurostat/.

[7] Land Steiermark. *Open Government Data, Land Steiermark*. 24th Jun 2015. http://data.steiermark.at/.

[8] Ethan Marcotte. *Responsive Web Design*. 2nd edition. A Book Apart, 2nd Dec 2014. 153 pages. ISBN 1937557189. http://abookapart.com/products/responsive-web-design.

[9] ODP. *Open Data Portal Austria*. Wikimedia Austria. 24th Jun 2015. http://opendataportal.at/.

[10] OGD. *Open Government Data Austria*. Bundeskanzleramt. 24th Jun 2015. http://data.gv.at/.

[11] OKFN. *Open Data Handbook*. Open Knowledge. 24th Jun 2015. http://opendatahandbook.org/.

[12] OpenLink. *Virtuoso Open Source Edition*. OpenLink Software. 24th Jun 2015. http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/.

[13] OSM. *OpenStreetMap Overpass API*. 24th Jun 2015. http://wiki.openstreetmap.org/wiki/Overpass_API.

[14] Rennes. *Qui Sommes Nous?* Rennes Métropole. 23rd Jun 2015. http://dataviz.rennesmetropole.fr/quisommesnous/.

[15] Juan Sequeda. *Introduction to: Triplestores*. dataversity. 31st Jan 2013. http://dataversity.net/introduction-to-triplestores/.

[16] Nigel Shadbolt, Wendy Hall and Tim Berners-Lee. "The Semantic Web Revisited". In: *IEEE Intelligent Systems* 3.21 (Jan 2006), pages 96–101. ISSN 1541-1672. doi:10.1109/MIS.2006.62.

[17] Stadt Graz. *Open Government Data Graz*. 24th Jun 2015. http://data.graz.gv.at/.

[18] W3C. *Resource Description Framework (RDF)*. W3C RDF Working Group. 25th Feb 2014. http://w3.org/RDF/.

[19] W3C. *SPARQL 1.1 Overview*. W3C Recommendation. 21st Mar 2013. http://w3.org/TR/sparql11-overview/.

[20] Wiener Linien. *Wiener Linien Open Data*. 24th Jun 2015. http://wienerlinien.at/eportal2/ep/channelView.do/pageTypeId/66528/channelId/-48664.

[21] Zemanta. *LODRefine: LOD-enabled Version of OpenRefine*. 5th Oct 2013. http://sourceforge.net/projects/lodrefine/.