Educational Open Government Data: from requirements to end users

Rudolf Eckelberg, Vytor Bezerra Calixto, Marina Hoshiba Pimentel, Marcos Didonet Del Fabro, Marcos Sunyé, Leticia M. Peres, Eduardo Todt, Thiago Alves, Adriana Dragone, and Gabriela Schneider

C3SL and NuPE Labs
Federal University of Paraná, Curitiba, Brazil
{rce16,vsbc14,marina,marcos.ddf,sunye,lmperes,todt}@inf.ufpr.br,
thiagoalves@ufpr.br,adrianadragone@yahoo.com.br,gabis0905@gmail.com

Abstract. The large availability of open government data raises enormous opportunities for open big data analytics. However, providing an end-to-end framework able to handle tasks from data extraction and processing to a web interface involves many challenges. One critical factor is the existence of many players with different knowledge, who need to interact, such as application domain experts, database designers, and web developers. This represents a knowledge gap that is difficult to overcome. In this paper, we present a case study for big data analytics over Brazilian educational data, with more than 1 billion records. We show how we organized the data analytics phase, starting from the analytics requirements, data evolution, development and deployment in a public interface.

Keywords: Open Government Data; Analytics API; data evolution

1 Introduction

The large availability of Open Governmental Data raises enormous opportunities for open big data analytics. Opportunities are always followed by challenges and when one handle Big Data, difficulties lie in data capture, storage, searching, sharing, analysis, and visualization [2]. Providing useful Open Data would need a complete team, from the domain expert up to the web developer, so often it cannot be handled only by a data scientist for example. In addition, processing large amounts of data is not always possible in a desktop computer and/or using a spreadsheet-based applications. Furthermore, there are many trade-offs to be made, meaning that it is not always possible to choose the best state-of-the-art approach for different technical reasons.

In this context, we present a case study of Open Data Web framework in the educational domain. The end-user requirement is to have a Web interface with a set of educational metrics, extracted from Open Educational data produced by the Brazilian Government. These metrics need to be extracted from a set of public datasets composed by more than 1 billion records, which are made available

on a yearly basis. We formed a small research and development team, composed by domain specialists, database and application developers, who are all students or professors in our University. We show how we organized the data analytics phase, from the analytics requirements, data evolution, development, to deployment in a public web interface and the five main challenges we have faced are: **Challenge 1**: how to find the most effective specification format to be exchanged by the domain experts with the database/application developers. **Challenge 2**: the open data was made available on a yearly basis by the Brazilian government and we had no control over the provided data sources. This means that we had to handle schema and data evolution. **Challenge 3**: having time constraints in mind, how to choose the most appropriate data model, with less impact on query development and still obtaining performance. **Challenge 4**: how to couple the query development with a REST API, since the main goal was to make the data easily available. **Challenge 5**: how to develop an attractive front-end on a fast way.

All the above mentioned challenges are complex by itself, but we had to tackle them in a way to have less impact between each data analytics phase and we may found distinct existing approaches and frameworks. For instance, we could use open source datawarehouse tools, such as *Pentaho* and *Talend*. They help to implement traditional data warehouse (DW) [5] approaches, however, they provide much more capabilities than we needed. In addition, it would be necessary to stick to their implementation format. We intended to have a simpler structure, without the burden of many intermediary tables, limiting the use of staging areas and data marts. Other newer tools, such as *Apache Drill* [4] enable data transformations, but are still quite complex to extract the data. Dozens of approaches of schema matching[6] and schema evolution would help on tracking the changes on the data over the years. In our case, we adopted a simplified solution keeping track of 1-to-1 relationship.

There are some web portals and research on this domain, though most of them focus on how to calculate specific metrics, without presenting open questions about Open Government Data Web engineering issues. The survey carried out by [7] shows that the informatics community in education in Brazil began to be interested in this area from 2013, so there are still not many works reported. [3] proposed an application that processes open government data related to social, economic and educational indicators of the 5,565 Brazilian municipalities and estimates the demand for unfilled vacancies in Brazilian public higher education. The proposed model works over a specific metric. In the work from [1], the authors shows the CultivEduca platform. This work also targets a single metric. These initiatives show the need for such solutions, though they often present ad-hoc solutions which are difficult to generalize.

We describe the requirements with details, and the choices we made to achieve a final web based application with educational open data, called LDE (*Laboratório da Dados Educacionais* - Educational Data Laboratory). The web tool can be accessed at `http://dadoseducacionais.c3sl.ufpr.br/`.

This article is organized as follows. First, we present the main requirements of our big data analytics life cycle. In the sequence, we describe our solutions for the main challenges risen. Then, we present the lessons learned from the development process and research done, followed by the conclusions.

2 Application requirements

The platform of educational data was built to serve diverse segments of society. Educational metrics support the planning of educational provision, assessment of access and guarantee of the right to education, research, policy formulation and generation of information to the general public. The project is carried out through joint efforts between the Nucleus of Educational Policies (NuPE) and the Center for Scientific Computing and Free Software (C3SL), both linked to the Federal University of Parana. The NuPE team has the knowledge about the educational data used in the project and is responsible for specifying the metrics available. The C3SL team is responsible for (i) developing the systems, composing the API development team and (ii) loading educational data and making them available on database management systems, composing the database maintenance team.

The goal of the application is to provide a web interface with a set of indicators/metrics over educational resources in Brazil. The need for new indicators is frequent, and the update of the older ones are on an yearly basis. The indicator's definition must be edited in a website easily accessible and with the possibility of combining different dimensions. In addition, the results should be returned in a reasonable time.

2.1 Data characteristics

The input Open Data have the following characteristics: for each year, there are at least four main sets of raw data released from an institute subordinated to the Ministry of Education [1]: Students, Teachers, Schools and Sessions, having respectively 50 million, 2 million, 200 thousand and 2 million records per year. There are other smaller indicators that are released on a similar basis, such as monthly family income (absolute and per capita), and demographic data. Educational census combines information which are specific to classes, schools, teaching staff and enrollments, but also administrative information and school resources. The sum of all data has more than 1 billion records so far.

The analyzed input data groups about 100 different fields each. All of this information must be available for all levels of dimensions - example: how many students are enrolled on federal schools, or how many biology teachers work on a specific state. The name of the columns, as well as its value, may evolve from year to year, so we have an important schema and data evolution issue. Along with

[1] The raw data are produced by INEP - Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira - http://portal.inep.gov.br/microdados

modeling changes, existing columns change from one census year to the next, so, to preserve consistency, pairing of the added values must be ensured through data transformations. These characteristics, giant amount of data and fields, as well as access and evolution capabilities are important application requirements.

We needed to carefully choose the data modeling to be applied. The balance between choosing the most appropriate normalization (if any), and also being capable of rapidly releasing new indicators, considering the data evolution, is very important. For such architectures, it is more convenient to be able to generate the complete database from raw original data when necessary. For this reason, the application must have its own backend tools to create and maintain database structure, as well as treating and inserting data.

2.2 The API and front end

After the digestion of the input data, they need to be queried and the results made available in a web interface, on tabular formats. For each provided metric, the API needs to support any combination of the available filters, and then to create the appropriate query to the main storage. The results need to be easily processed, in order to be formatted back into a graphical and user friendly interface. While the specification of this requirement seems to be simple, the outputs need to be synchronized with any changes on the database, thus adding complexity to the process.

3 Open Data engineering of educational metrics

In this section we present the general architecture of our Open Data and web engineering approach, designed in order to tackle to the previously explained application requirements.

3.1 Metrics specification form

The domain experts provide a complete form describing the required metrics containing the following specification: metrics name (e.g., enrollment), dimensions (e.g., state or region), filters, periodicity and how it is calculated, free text explaining the metric and transformation functions. The transformation functions specify if an input data is transformed into a different value, i.e., transforming a set of states into regions, considering this data was not available in the input source. [2]

3.2 Database creation and maintenance

The input raw data are extracted and injected into a column store, and it need to be revisited every time a new release is made available. The database components are the following:

[2] An example specification form (in Portuguese) can be found at: `https://dadoseducacionais.c3sl.ufpr.br/static/904db38dc8c8a3e0d79da5568b7c485f.pdf`

Database modelling: the choice of the database modeling and its underlying DMS (Database management system) has one important limitation: for every new year, large CSVs (Comma Separated Values) for all involved sources are available and need to be processed. The information is generally not normalized and we have no control over the input fields names and values. This is common in available open data, since it is simpler to provide one single file per kind of data. We choose to keep an de-normalized view, to minimize the difference between the input data and the created schema, and moreover it facilitates the interaction with the domain experts. We have chosen to use MonetDb column-store database [3], after having executed a comparison benchmark[4] with the PostgreSQL[5] database. This technology choice, and all the following, was restricted only to open source solutions.

Data extraction and evolution A mapping file was created to relate fields' meanings and names with their respective years, along with the expressions that convert them to their final values. While we could use schema matching solutions to create the initial mapping, we opted a simpler solution, searching only for similar names with a string similarity metric and setting up 1-to-1 relationships. Any more complex relationship needs to be manually specified. The resulting files are used as input to create underlying database structures, migrate them and, along with the original source raw data, populate the database with data entries. Though it is possible to migrate schemas on most widely used databases, many subsequent data structure transformations can lead to inconsistencies and performance drops over long time spans.

We have developed a tool that manages the input mapping and that operates over the input data. The chosen technologies in the context are Python 3 along with SQLAlchemy. The tool has two main classes: **Mapping**, which sets ups the relationships of columns over the years, as well as the required data transformations and **Processing**, which applies the mapping and create the desired structures. In order to make it easily accessed by developers, we provided a CLI (Command Line Interface) to directly call the data management operations, which were: create - creation of a table along with table mapping for possible migrations, given columns information and relations; *insert* - bulk insertion of one or more data files on a given table, considering pairing conditions; *drop* - drops a table and its mapping information; *update* - updates data on specific columns to match a data file - useful for new transformed variables or update to new revised raw data from the original source; *remap* - update of table structures to match mapping with new required structure.

3.3 API-driven queries

Given the established database, a set of queries is necessary to access the data and they would be accessed by a Web API[6]. The REST API is implemented

[3] https://www.monetdb.org/
[4] https://gitlab.c3sl.ufpr.br/simcaq/bd_bench/
[5] https://www.postgresql.org/
[6] The API documentation is available at: `https://simcaq.c3sl.ufpr.br/doc/`

using *Node.js* and it access the data in a consistent way and it provides enough flexibility so that different web clients with different requirements can get the data. Metrics are aggregations done by the API in the database data, such as sums, counts and percentages, and it is the core that provides the information the client needs. A metric can be filtered by parameters defined as dimensions. The API is used to generate the SQL queries, which are driven by the API, i.e., the user interface submits an URL and it is translated into an SQL query. The API queries have the following format:

```
http://SITE_NAME/api/METRIC_NAME?dims=DIMENSION1,DIMENSION_N
           &filter=FILTER_NAME1:FILTER_VALUE1,
           filter=FILTER_NAME\_N:FILTER_VALUE_N
           &format=CSV OR JSON
```

The API requests works as follows: a request is made to a metric route (students enrollment, number of schools, number of classrooms, etc), with details about required dimensions and filters. Three execution examples are below. The first one contains one dimension and more than one filter. The second one contains one dimension and returns a given format. The last one contains an initial year, stating that the historical information starts in year 2014.

```
https://simcaq.c3sl.ufpr.br/api/v1/population?filter=city_size:1,region:1&dims=state
https://simcaq.c3sl.ufpr.br/api/v1/enrollment?dims=adm_dependency?format=csv
https://simcaq.c3sl.ufpr.br/api/v1/teacher?filter=min_year:2014
```

3.4 Web Interface

Finally, the web interface is the final goal to present the processed open data. The API returns the queried result, along with the dimensions and filtered as specified. A client only needs to know how to create this requests to obtain any information available. The possibilities of combining all the dimensions and metrics makes it unfeasible to define specialized charts, thus we have provided a table-based interface, with 2 different options: the first one is a simple table presenting data of a given metric and also for a set of years, for instance, the number of enrollments from 2013 to 2017. The second one is a dynamic table, where the different dimensions are generated and plotted in the left side, as shown in Figure 1: it shows the number of enrollments by administrative dependencies: Federal, State, County or Private.

3.5 Discussion and Lessons learned

We have learned several lessons on developing this project, going from human to technological ones. We can assure that developing a complete Open Government Data initiative is a very complex task, which would be hard to be provided by isolated experts in the field or data scientists. The main difficulty is not to provide a single metric or indicator, which is doable with lesser efforts, but to keep a continuous pace on delivering new indicators with very large amounts of data, and updating them every time a new data release is available.

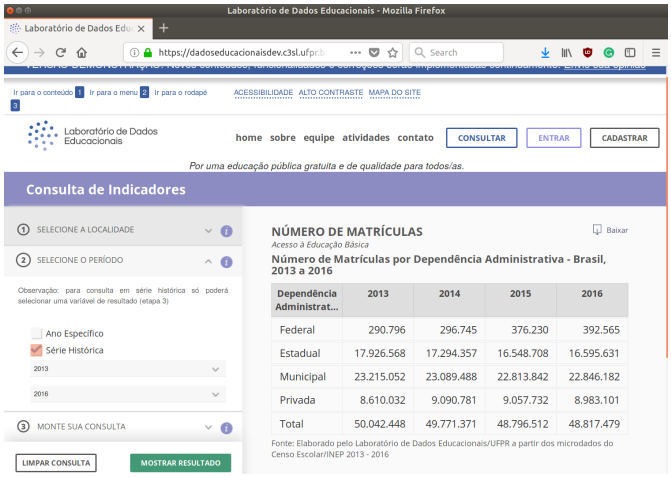

Fig. 1. Screenshot of the enrollments metrics

Concerning the metrics' specification, we have created a simple format with less than ten fields, to keep only essential information. We think this was appropriate for the developers, but also for further validation of the results by external users. It was not necessary to create complete use cases, since the metrics have a more focused scope.

Second, the dependency from the input sources diminishes the freedom of data modeling. Receiving a new data source, without any guarantee that its format and specification is stable, adds the challenge of schema and data matching for every new release. For this, we did not create automatic solutions, but a rather simple mapping format that could be easily validated by the experts. In other words, keeping the mapping in CSV files was very useful, since they could be seen in spreadsheet editors, without the need to install specific tools. The mappings and its corresponding scripts were used to generate the datasource from the input raw data. It was very important to automatize the process, since it was possible to regenerate the entire data in a rather straightforward way, so the mappings were the central artifacts in the specification.

This evolution also added constraints in the data transformations. We decided not to normalize the data, in order to keep a modeling relatively close to the original source. This avoids having new complex transformations to be updated for every new release. In addition, the queries had lesser joins, thus they were executed rather quickly. Keeping all the artifacts synchronized was of vital importance. We could have chosen an open source ETL tool to help on the extraction process. However, they are very complex and they offer much more capabilities than we needed, since the scope is smaller. In addition, they have specific formats, and it is difficult to migrate from one to another, if needed. For these reasons we opted for creating our own specific scripts, which has proven to be effective. Choosing a column store, MonetDb, proved to be effective, since we

could execute the queries directly on the input tables, without the need to create intermediate tables with pre-calculated values. The automatic query generation from the API was very important, so we could have different combinations of dimensions and filters. Since the application is only for specific queries, we did not have to support full SQL generation, but 4 main groups of expressions. The final generated interface, based on table, enabled to have a simple correspondence with the output from the API, and still leveraging important information.

4 Conclusions

In this paper we presented our end-to-end case study of a web application engineering for providing Open Government Data for final users. We have seen that providing useful Open Data for the final users is a challenging task, involving teams with distinct expertise and many difficult computational problems. There are several challenges, including communication, database modelling and evolution and practical API design, which are all interconnected. We have described the main challenges and the adopted solutions, which are often trade-offs and also suggested some innovations from existing approaches. We have observed that such approach is largely viable, and it has already been deployed on a real scenario with success.

Acknowledgments

We thank the C3SL and NuPE team that contributed on developing the solution. This work was partially funded by *Sistema de Monitoramento de Políticas de Promoção da Igualdade Racial (SNPPIR)* and *Simulador de Custo-Aluno-Qualidade (SASE/MEC)*.

References

1. Marie Jane Soares Carvalho, Breno Neves, and Rafaela Melo. Plataforma cultiveduca. In *Anais dos Workshops do SBIE*, volume 5, page 134, 2016.
2. CL Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314–347, 2014.
3. Pedro Calais Guerra, Rodrigo Yuji Mizobe Nakamura, Eduardo Raul Hruschka, et al. Estimativa de demanda potencial de matrículas em ensino superior usando dados públicos e múltiplos modelos de regressão. In *2nd KDMile*. SBC, 2014.
4. Michael Hausenblas and Jacques Nadeau. Apache drill: interactive ad-hoc analysis at scale. *Big Data*, 1(2):100–104, 2013.
5. Ralph Kimball and Margy Ross. *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons, 2011.
6. Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, Dec 2001.
7. Philippe Santos, Rafael Ferreira, and Péricles Miranda. Dados abertos educacionais: Uma revisão da literatura brasileira. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 28, page 11, 2017.