



## OpenGovIntelligence

Fostering Innovation and Creativity in Europe through Public  
Administration Modernization towards Supplying and Exploiting  
Linked Open Statistical Data

---

### Deliverable 3.6

### Report on OpenGovIntelligence ICT tools

---

Leading partner:	NUIG
Participating partners:	CERTH, TUT, ProXML, SWIRRL,
Version-Status:	1.0
Dissemination level:	PU

## Deliverable factsheet

<b>Project Number:</b>	693849
<b>Project Acronym:</b>	OpenGovIntelligence
<b>Project Title:</b>	Fostering Innovation and Creativity in Europe through Public Administration Modernization towards Supplying and Exploiting Linked Open Statistical Data

<b>Deliverable title:</b>	Report on OpenGovIntelligence ICT tools
<b>Deliverable number:</b>	D3.6
<b>Official submission date:</b>	31 October 2018
<b>Actual submission date:</b>	31 October 2018

<b>Editor(s):</b>	Arkadiusz Stasiewicz (NUIG)
<b>Author(s):</b>	Dimitrios Zeginis (CERTH) Evangelos Kalampokis (CERTH) Konstantinos Tarabanis (CERTH) Efthimios Tambouris (CERTH) Arkadiusz Stasiewicz (NUIG) Mohamed Adel (NUIG) Agustín García Pereira (NUIG) Paul Hermans (ProXML) Bill Roberts (SWIRRL) Rick Moynihan (SWIRRL)
<b>Reviewer(s)</b>	Bill Roberts (SWIRRL)

<b>Abstract:</b>	This deliverable provides details on the software components delivered as a result of the OpenGovIntelligence project.
------------------	--

## Effort of Participating Partners

	<i>Name</i>	<i>Short Name</i>	<i>Role</i>	<i>Person Months</i>
1.	Centre for Research & Technology - Hellas	CERTH	Participant	1
2.	Delft University of Technology	TU Delft	None	0
3.	National University of Ireland, Galway	NUIG	Leader	2
4.	Tallinn University of Technology	TUT	None	0
5.	ProXML bvba	ProXML	Participant	0.25
6.	Swirrl IT Limited	SWIRRL	Participant	1
7.	Trafford council	TRAF	None	0
8.	Flemish Government	VLO	None	0
9.	Ministry of Administrative Reconstruction	MAREG	None	0
10.	Ministry of Economic Affairs and Communication	MKM	None	0
11.	Marine Institute	MI	None	0
12.	Public Institution Enterprise Lithuania	EL	None	0

## Revision History

<i>Version</i>	<i>Date</i>	<i>Revised by</i>	<i>Reason</i>
0.1	04-Oct-2018	A. Stasiewicz (NUIG)	Initial Draft
0.2	25-Oct-2018	A. Stasiewicz (NUIG)	General updates
0.3	26-Oct-2018	Lukasz Porwol (NUIG)	General updates
0.4	31-Oct-2018	A. Stasiewicz (NUIG)	Addressing internal review comments
1.0	31-Oct-2018	CERTH	Submission to EC

**Statement of originality:**

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

## Table of Contents

<b>DELIVERABLE FACTSHEET .....</b>	<b>2</b>
<b>EFFORT OF PARTICIPATING PARTNERS.....</b>	<b>3</b>
<b>REVISION HISTORY .....</b>	<b>4</b>
<b>TABLE OF CONTENTS .....</b>	<b>5</b>
<b>LIST OF FIGURES.....</b>	<b>6</b>
<b>LIST OF TABLES .....</b>	<b>7</b>
<b>LIST OF ABBREVIATIONS.....</b>	<b>8</b>
<b>EXECUTIVE SUMMARY .....</b>	<b>9</b>
<b>1 INTRODUCTION .....</b>	<b>10</b>
1.1 SCOPE.....	10
1.2 AUDIENCE.....	10
1.3 STRUCTURE.....	10
<b>2 OPENGOVINTELLIGENCE ICT TOOLS – OVERVIEW.....</b>	<b>11</b>
2.1 TOOLS DEVELOPED DURING THE FINAL DEVELOPMENT STAGE .....	11
<b>3 OPENGOVINTELLIGENCE ICT TOOLS - FINAL RELEASE .....</b>	<b>14</b>
3.1 DATA CUBE EXPLORER .....	14
3.2 DATA CUBE RDF VALIDATOR .....	15
3.3 DATA SCIENCE STUDIO SPARQL PLUGIN .....	16
3.4 CUBIQL - A GRAPHQL SERVICE FOR QUERYING LINKED DATA CUBES – UPDATE.....	17
3.5 TABLE2QB AND GRAFTER – UPDATE .....	21
3.6 DATA CUBE AGGREGATOR – UPDATE .....	21
3.7 SPARQL CONNECTOR FOR EXPLORATORY.....	21
<b>4 PILOT SPECIFIC TOOLS.....</b>	<b>22</b>
<b>5 CONCLUSION .....</b>	<b>23</b>
<b>REFERENCES .....</b>	<b>24</b>

## List of Figures

FIGURE 1. LOSD TOOL ECOSYSTEM .....	13
FIGURE 2. CUBIQL RESPONSE EXAMPLE.....	20

List of Tables

TABLE 2. OGI ICT TOOLS TECHNICAL STARTING POINT..... 12

## List of Abbreviations

The following table presents the acronyms used in the deliverable in alphabetical order.

<i>Abbreviation</i>	<i>Description</i>
API	Application Programming Interface
CMS	Content Management System
ICT	Information and Communication Technologies
LOSD	Linked Open Statistical Data
OLAP	OnLine Analytical Processing
RDF	Resource Description Framework
UI	User Interface
URI	Uniform Resource Identifier
WP	Work Package



## Executive Summary

This document is the deliverable “D3.6 - Report on OpenGovIntelligence ICT tools” (referred to as D3.6). It provides detailed information about the final result of the OpenGovIntelligence development in WP3: **latest development and major updates**. Complete list of the OpenGovIntelligence ICT tools is presented in deliverable “D3.5 OpenGovIntelligence ICT tools” (referred to as D3.5).

The goal of WP3 (“ICT tools development”) was to develop the OpenGovIntelligence ICT tools as a suite of open source and commercial tools. The final version of the developed ICT tools aims at supporting the OpenGovIntelligence framework created in WP2 (“Framework creation”) and enables:

- a) the creation of Linked Open Statistical Data (LOSD) from various sources,
- b) the expansion of LOSD with datasets from existing sources,
- c) the exploitation of LOSD for the co-production of public services.

With regards to the creation of LOSD, OpenGovIntelligence designed and developed ICT tools that assist the transformation of public sector data to standard machine readable forms (specifically to RDF Data Cubes) and the validation of the generated RDF. With regards to the exploitation of LOSD, OpenGovIntelligence designed and developed ICT tools that enable visualisation and analysis of statistical data.

While the first release of the OpenGovIntelligence ICT tools was focused on the tools supporting the creation and exploitation of LOSD, the final release focuses on exploitation of LOSD by providing new tools and improvements (functional and performance) to tools already developed and evaluated.

The final release of the OpenGovIntelligence ICT tools was guided by challenges and needs identified in WP1 (D1.1 *OpenGovIntelligence challenges and needs*), the OpenGovIntelligence framework (D2.2 *OpenGovIntelligence framework*) as well as the pilots operation and OpenGovIntelligence evaluation (D4.4 *Evaluation results - Second round*). The tools were evaluated during the pilot implementation stage and based on the outcomes, the existing tools were improved.

In the final year of the project, efforts were concentrated mainly on table2qb (creation of LOSD) and CubiqI (exploitation of LOSD). Both were used to support other tools in the area of expansion and exploitation. For example, OGI visualisation tools were integrated with CubiQL and support data generated using table2qb publishing pipelines.

Results presented in this document are the outputs of Task 3.2 OpenGovIntelligence ICT tools – second release. Despite the fact, that the development in the frame of the OpenGovIntelligence project has finished, further updates to the selected tools are expected.

## 1 Introduction

This section introduces the background of the work carried out in WP3 “ICT tools development”. Sub-section 1.1 presents the scope and the objectives of the current document, sub-section 1.2 describes the intended audience for this document, sub-section 1.3 outlines the structure of the document, while sub-section 1.4 addresses the comments received from the 1<sup>st</sup> Project Review.

### 1.1 Scope

This report documents the ICT tools developed during the WP3. To guide readers in understanding the context of these tools, Section 2 presents the overview of the OpenGovIntelligence tools. The final release provided tools that enable creation and exploitation of LOSD as well as the expansion of existing datasets.

### 1.2 Audience

The intended audience for this document is the OpenGovIntelligence consortium, in particular partner organisations responsible for the development of pilot trials, the European Commission (EC) and those who are interested in challenges and needs for opening-up and exploiting LOSD for the co-production of innovative data-driven services for governments.

### 1.3 Structure

The structure of the document is as follows:

- Section 2 presents the OpenGovIntelligence ICT tools overview
- Section 3 provides a detailed description of the individual OpenGovIntelligence ICT tools developed during the second development stage of OpenGovIntelligence project;
- Finally, Section 4 concludes the report and outlines the future development plan.

## 2 OpenGovIntelligence ICT tools – overview

This section aims to provide background information related to LOSD and OpenGovIntelligence tools.

The developed tools cover parts of the OpenGovIntelligence Architecture and are part of a related ecosystem of tools. The OpenGovIntelligence Architecture (presented in D3.2) enables stakeholders to collaborate towards the production of innovative data-driven public services by exploiting Linked Open Data technologies and statistical datasets. The initial version of the architecture was documented in detail in D3.2 and was tested and refined during the first round of pilot evaluations. The architecture is organised as follows:

- The architecture is divided into five layers: (i) Data Provision, (ii) Data Platform, (iii) Process Layer, (iv) Service Design, and (v) Service Provision.
- Key management responsibilities are shared across all layers.
- Each layer has a set of components that perform tasks specific to that layer.

The architecture informs the pilot implementations, as well as the other future implementations of the OGI software in other projects together with OpenGovIntelligence framework (documented in D2.2).

### 2.1 Tools developed during the final development stage

In the first implementation stage, the developed tools were mainly supporting data conversion and exploitation of the RDF Data Cubes. The second release of OpenGovIntelligence includes additional tools and supports a larger part of the OpenGovIntelligence framework. The final stage of the development addresses evaluation results and aimed at usability, performance and stability improvement.

While the development of majority of the tools were initialised during the OpenGovIntelligence project, some of them were using outputs of the DaPaaS<sup>1</sup> and OpenCube<sup>2</sup> projects as the technical starting point, and subsequently improved and extended within OpenGovIntelligence. Details are presented in Table 1. OGI ICT Tools and visualised in Figure 1. LOSD Tool Ecosystem. Also, Table 2 shows which tools are new components, started in year 2 and 3, and which tools were further developed on work started earlier.

---

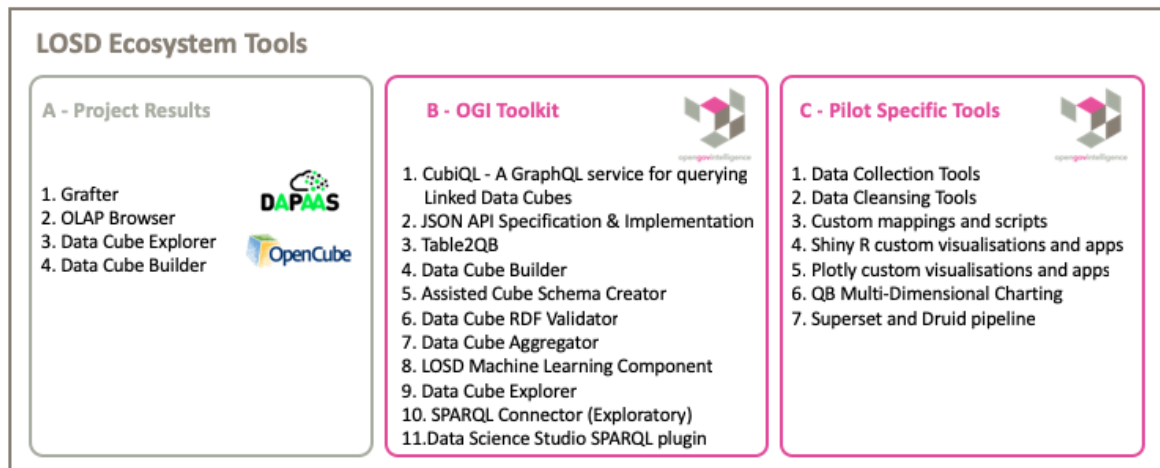
<sup>1</sup> <https://project.dapaas.eu>

<sup>2</sup> <http://opencube-project.eu>

Table 1. OGI ICT Tools

No.	Name	Development in Year 3	Foundation
1	CubiQL - A GraphQL service for querying Linked Data Cubes	Yes	OpenGovIntelligence
2	JSON API For Data Cube Specification & Implementation	No	OpenGovIntelligence
3	Table2QB And Grafter	Yes	DaPaaS
4	Data Cube Builder	No	OpenGovIntelligence
5	Assisted Cube Schema Creator	No	OpenGovIntelligence
6	Data Cube RDF Validator	Yes	OpenGovIntelligence
7	Data Cube Aggregator	Yes	OpenCube
8	LOSD Machine Learning Component	No	OpenGovIntelligence
9	Data Cube Explorer	Yes	OpenGovIntelligence
10	SPARQL Connector for Exploratory	Yes	OpenGovIntelligence
11	Data Science Studio SPARQL plugin	Yes	OpenGovIntelligence

Section 3 documents the tools, which **were developed or improved during the final development stage** as well as links by which they can be accessed. In general, tools developed during the project lifecycle can be accessed at the GitHub repository available at: <http://github.com/OpenGovIntelligence>.



**Figure 1. LOSD Tool Ecosystem**

**Note:** The OpenGovIntelligence project did not develop software tools for every component of the architecture or a single integrated system that matches this architecture as a whole. Implementations of the OpenGovIntelligence approach are a combination of a selection of tools and processes relevant to the use cases they address, following the principles set out in the architecture. Moreover, some parts of the architecture might require human actions or use appropriate pre-existing tools from outside of the OpenGovIntelligence project. Existing tools available for the purpose should be re-used where ever possible. For example, we have already identified a number of tools that can be re-used in the collaboration space. This list of tools is included in deliverable D2.1 – OpenGovIntelligence framework – first release.

### 3 OpenGovIntelligence ICT tools - final release

This section provides a detailed description of the individual OpenGovIntelligence ICT tools **developed during the final development stage** of the OpenGovIntelligence project and details of the **major updates to the tools developed in the first and second stages** of the OpenGovIntelligence project. Previously developed tools descriptions are available in deliverable “D3.2 – Report on OpenGovIntelligence ICT tools – first release” and “D3.4 – Report on OpenGovIntelligence ICT tools – second release”. The tools developed during the project lifecycle can be accessed at the GitHub repository available at: <http://github.com/OpenGovIntelligence>.

During the final stage of the development, the efforts were focused on the development of CubiQL - A GraphQL service for querying Linked Data Cubes, improvements to the table2qb tool for generating standards compliant LOSD, and general updates to the tools: performance optimisation, integration with CubiQL and improvements to the documentation.

Complete list of the OpenGovIntelligence ICT tools is presented in D3.5.

Moreover, latest description of the tools together with usage examples is available at:

<http://toolkit.opengovintelligence.eu>.

#### 3.1 Data Cube Explorer

Data Cube Explorer is a web-based tool that catalogues and presents details of available data cubes to the users, allowing them to understand and explore the contents of a collection of statistical datasets. It provides a search interface and enables the user to preview cube data using different visualisation tools: pivot table, OLAP Browser<sup>3</sup> and Cube Visualiser<sup>4</sup>.

##### 3.1.1 Functionality Description

The Data Cube Explorer enables the automatic exploration of the selected data endpoint. The user can search over the metadata or browse all the available datasets to finally preview the data using different visualisation interfaces:

##### Cube Visualizer

The Cube Visualizer creates and presents to the user graphical representations of an RDF data cube's one-dimensional slices.

##### OLAP Browser

---

<sup>3</sup> <https://github.com/OpenGovIntelligence/qb-olap-browser>

<sup>4</sup> <https://github.com/OpenGovIntelligence/CubeVisualizer>

The OLAP Browser enables the exploration of RDF data cubes by presenting each time a two-dimensional slice of the cube as a two-dimensional table and the execution of OLAP operations.

### **PivotTable**

The PivotTable presents details of selected data cube and enables the user to visualise the data using various visualisation methods. Moreover, it allows to calculate AVG (average), SUM, MIN, MAX and COUNT for selected dimensions.

### **3.1.2 Implementation Description**

The implementation of the Data Cube is based on the Django framework - a high-level Python Web framework. Data Cube Explorer uses CubiQL for querying Linked Data Cubes. The tool provides three main data visualisation interfaces: Cube Visualizer, OLAP Browser, PivotTable.

### **3.1.3 Availability**

Source code is available at GitHub:

<https://github.com/OpenGovIntelligence/data-cube-explorer>

### **3.1.4 License**

The software is available as open source under the Apache License (v2.0).

### **3.1.5 Pilots involved**

At this stage, the tools are not customised to be used in specific pilot use cases, thus the tool is universal and can be used across the OpenGovIntelligence Consortium.

## **3.2 Data Cube RDF Validator**

Data Cube RDF Validator is a collection of integrated test cases to confirm compatibility of the data (e.g. generated using table2qb) with CubiQL, a GraphQL service for querying Linked Data Cubes.

This tool runs a collection of test cases against a SPARQL endpoint. The endpoint can be either a HTTP(s) SPARQL endpoint or a file or directory on disk. Test cases can be specified as either a SPARQL query file, or a directory of such files. A standard collection of test cases has been prepared.

### **3.2.1 Functionality Description**

The repository contains versions of the well-formed cube validation queries defined in the RDF data cube specification. These are defined as SPARQL SELECT queries rather than the ASK queries defined in the specification to enable more detailed error reporting.

Test cases are expressed as either SPARQL ASK or SELECT queries. These queries are run against the target endpoint and the outcome of the test is based on the result of the query execution.

#### **ASK queries**

SPARQL ASK queries are considered to have failed if they evaluate to true so should be written to find invalid statements. This is consistent with the queries defined in the RDF data cube specification.

#### **SELECT queries**

SPARQL SELECT queries are considered to have failed if they return any matching solutions. Like ASK queries they should return bindings describing invalid resources.

### **3.2.2 Implementation Description**

The tool is implemented using the Clojure programming language.

#### **3.2.3 Availability**

Source code is available at GitHub:

**Available at:** <https://github.com/Swirrl/rdf-validator>

#### **3.2.4 License**

The software is distributed under the Eclipse Public License, either version 1.0 or any later version.

#### **3.2.5 Pilots involved**

The tools have been actively tested and used in the pilot use cases, thus the tool is generic and can be used across the OpenGovIntelligence Consortium.

## **3.3 Data Science Studio SPARQL plugin**

Data Science Studio of Dataiku Is a Collaborative Data Science tool for self-service analytics and machine learning. The 'Data Science Studio SPARQL plugin' developed during the project offers a connector to SPARQL endpoints as data source.

This plugin allows to add datasets from SPARQL SELECT queries to the DSS data science analytics flows. More info at: <https://www.dataiku.com/>

#### **3.3.1 Availability**

The plugin is available at: [https://github.com/OpenGovIntelligence/datasciencestudio\\_sparql\\_plugin](https://github.com/OpenGovIntelligence/datasciencestudio_sparql_plugin)



### 3.3.2 License

The software is available as open source under the Apache License (v2.0).

## 3.4 CubiQL - A GraphQL service for querying Linked Data Cubes – update

Following evaluation of initial implementations, it was decided to implement a second version of the JSON-qb API using the GraphQL query language for APIs, due its fast growing adoption in the web developer community and because its features are a good match for our requirements to search for and filter RDF Data Cubes.

The set of API methods has been re-specified using the GraphQL approach, and expanded as additional useful functions have been identified

The work on the API is available at <https://github.com/Swirrl/cubiql>

The functionality of CubiQL include:

- Cube search. It supports searching cubes based on their: 1) measures, 2) attributes, 3) dimensions, 4) dimension/attribute values and 5) Hierachies. The advanced QB search support both AND, OR operators. Details are presented on the following usage examples.
- The retrieval of cube metadata
- The slicing of cubes
- The aggregation of returned results
- The pagination or returned results

During year 2 and 3 of the project, the work was focused on new data access methods, performance optimisation, user documentation improvements, examples and validation tools.

Some example CubiQL calls are listed below:

Search cubes with measures median OR mean
<pre>{cubiql{   { datasets(measures: {     or: ["http://statistics.gov.scot/def/measure-properties/median"         "http://statistics.gov.scot/def/measure-properties/mean"]}) {     title     description     uri}}}</pre>
Search cubes that have geography at the level of countries OR council-areas
<pre>{cubiql{   {datasets(componentValue: {</pre>

```
or: [{component: "http://purl.org/linked-data/sdmx/2009/dimension#refArea",
      levels: ["http://statistics.gov.scot/def/foi/collection/countries",
               "http://statistics.gov.scot/def/foi/collection/council-areas"]}]]){
  title
  description
  uri}}}
```

#### Search cube with reference period values 2012-Q1 AND 2013-Q1

```
{cubiql{
  {datasets (componentValue:{
    and:[{component:"http://purl.org/linked-data/sdmx/2009/dimension#refPeriod"
          values:["http://reference.data.gov.uk/id/quarter/2012-Q1",
                  "http://reference.data.gov.uk/id/quarter/2013-Q1"]}]})) {
    title
    description
    uri}}}
```

#### Slising cube based on gender and population group

```
{cubiql{
  dataset_earnings {
    title
    description
    observations(dimensions:{gender:ALL
                             population_group:WORKPLACE_BASED }) {
      page {
        observation {
          gender
          measure_type
          population_group
        }
      }
    }
  }
}
```

#### Aggregation of the results

```
dataset_earnings {
  title
```

```
description
observations(dimensions: {gender: ALL,
                          population_group: WORKPLACE_BASED,
                          measure_type: MEDIAN}) {
  aggregations {
    max(measure: MEDIAN)
  }}}}
```

Moreover, section below presents tool usage and the configuration options.

Running CubiQL using the default configuration:

```
java -jar graphql-qb-0.5.0-standalone.jar --port 8085 --endpoint http://myendpoint.com
```

If CubiQL is running against a local directory containing the data the `--endpoint` parameter should specify the path to the directory:

```
java -jar graphql-qb-0.4.0-standalone.jar --port 8085 --endpoint ./ttl
```

CubiQL default configuration is as follow:

```
{
  :geo-dimension-uri nil
  :time-dimension-uri nil
  :codelist-source "component"
  :codelist-predicate "http://publishmydata.com/def/qb/codesUsed"
  :codelist-label-uri "http://www.w3.org/2000/01/rdf-schema#label"
  :dataset-label-uri "http://www.w3.org/2000/01/rdf-schema#label"
  :schema-label-language nil
  :max-observations-page-size 2000
}
```

If default configuration does not match the data, user can specify configuration file:

```
java -jar graphql-qb-0.4.0-standalone.jar --port 8085 --endpoint http://myendpoint.com/ --
configuration myconfig.edn
```

**Configuration parameters:**

*:geo-dimension-uri* - defines the geo dimension. The values of the geo dimension should have a label.

*:time-dimension-uri* - defines the time dimension. The values of the time dimensions should be defined by the reference.data.gov.uk e.g. <http://reference.data.gov.uk/id/year/2016>

*:codelist-source* - defines the source of the codelist that contains only the values used at the cube. The source can be: (i) "component" or (ii) "dimension". By default Table2qb uses the "component".

*:codelist-predicate* - defines the predicate that connects the *:codelist-source* with the codelist that contains only the values used at the cube. By default Table2qb uses: <http://publishmydata.com/def/qb/codesUsed>.

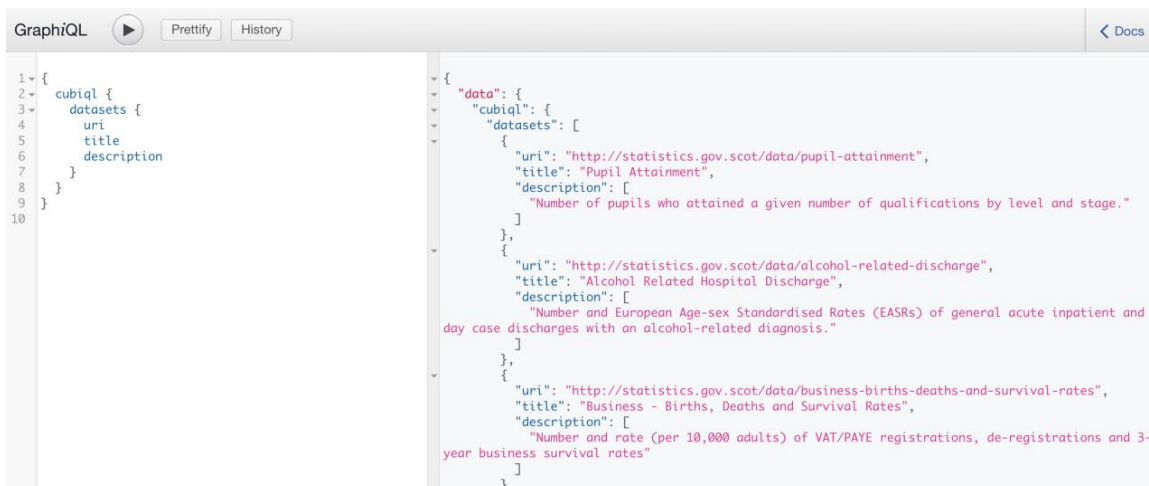
*:codelist-label-uri* - defines the label property that is used at the codelists. By default Table2qb uses: <http://www.w3.org/2000/01/rdf-schema#label>

*:dataset-label-uri* - defines the label property that is used at the dataset i.e. cube, DSD, components. By default Table2qb uses: <http://www.w3.org/2000/01/rdf-schema#label>

Datasets, dimensions, measures and codelist members should all have a label with a language tag matching the *:schema-label-language*. nil can be specified to use strings without an associated language tag.

*:max-observations-page-size* defines the maximum page size e.g. if there is a need to get all the observations with one query.

Example of CubiQL response is presented in the figure below.



```
1 {
2   cubiq {
3     datasets {
4       uri
5       title
6       description
7     }
8   }
9 }
10
```

```
{
  "data": {
    "cubiq": {
      "datasets": [
        {
          "uri": "http://statistics.gov.scot/data/pupil-attainment",
          "title": "Pupil Attainment",
          "description": [
            "Number of pupils who attained a given number of qualifications by level and stage."
          ]
        },
        {
          "uri": "http://statistics.gov.scot/data/alcohol-related-discharge",
          "title": "Alcohol Related Hospital Discharge",
          "description": [
            "Number and European Age-sex Standardised Rates (EASRs) of general acute inpatient and day case discharges with an alcohol-related diagnosis."
          ]
        },
        {
          "uri": "http://statistics.gov.scot/data/business-births-deaths-and-survival-rates",
          "title": "Business - Births, Deaths and Survival Rates",
          "description": [
            "Number and rate (per 10,000 adults) of VAT/PAYE registrations, de-registrations and 3-year business survival rates"
          ]
        }
      ]
    }
  }
}
```

Figure 2. CubiQL response example

### 3.5 Table2QB And Grafter – update

During year 2 and 3 of the project, further research has been carried out into the optimal design of the table2qb input format, to make it flexible enough for the range of use cases to be supported, but still easy for statisticians or analysts to produce RDF data, without needing technical knowledge of Linked Data.

- Further prototypes have been implemented and tested.
- Tutorials and examples of the publishing pipelines creating the output RDF compatible with CubiQL are provided.
- Improvements to the Grafter libraries have been implemented to improve performance and scalability when processing large data cubes.

The table2qb specification and implementation can be found at <https://github.com/Swirrl/table2qb>  
Grafter can be found at <https://github.com/Swirrl/grafter>

### 3.6 Data Cube Aggregator – update

The Data Cube Aggregator aggregates data across dimensions of a cube. The aggregate functions supported are the AVG, SUM, MIN, MAX and COUNT. This implementation aims in computing aggregations from raw RDF data. Thus, taking as input the raw RDF data it creates a cube that contains the corresponding aggregated observations.

Main update of the Aggregator in year 3 is the support of the "Measure Dimension" (i.e. qb:measureType) for expressing multiple measures. Moreover, the created aggregated cubes are compatible with CubiQL.

**Available at:** <https://github.com/OpenGovIntelligence/json-qb-api-implementation>

(integrated with JSON-QB API)

### 3.7 SPARQL connector for Exploratory<sup>5</sup>

SPARQL connector for Exploratory allows to connect the Exploratory Data Science tool to a SPARQL endpoint. This component is provided as R code together with JSON configuration file. It has become part of a largely deployed product. Details about the Exploratory extensions are available at: <https://docs.exploratory.io/import/extensions.html>

**Available at:** [https://github.com/OpenGovIntelligence/exploratory\\_sparql\\_plugin](https://github.com/OpenGovIntelligence/exploratory_sparql_plugin)

---

<sup>5</sup> <https://exploratory.io>

## 4 Pilot specific tools

In majority of the Pilots trials, there was a need to develop additional, Pilot Specific Tools in order to fulfil the use scenarios requirements.

In general, those tools covers parts of the OpenGovIntelligence Architecture and are part of related ecosystem of tools aligned with OpenGovIntelligence innovation framework.

However, due to nature of the requirements and in some cases its unique functions, it is not possible to clarify Pilot Specific Tools as a part of OGI Toolkit.

1. Data Collection Tools. This category covers tools designed to download the data coming from various sources: e.g. web portals. It also covers periodic data updates.
2. Data Cleansing Tools. Due to the nature of the data sources used in the pilot trials, there was a need to re-use or develop tools allowing data cleansing: e.g. conversion of address to geo location, add municipality based on the city name, remove duplicated entries and so on.
3. Custom mappings and scripts. In some cases data coming from various sources had to be mapped in integrated into single data source. Based on user inputs this category of tools allowed to generate RDF Data Cubes.
4. Shiny R custom visualisations and apps. In some cases pilots decided to use Shiny R in order to adapt visualisations and to rapidly prototype series of visualisation. In general those prototypes are available at: <https://github.com/OpenGovIntelligence>
5. Plotly custom visualisations and apps. Plotly (<https://plot.ly/>) was identified as easy to use visualisation framework covering e.g. Marine Institute Pilot Trial.
6. QB Multi-Dimensional Charting. This component is a multi dimensional charting dashboard written in JavaScript on top of a dc.js<sup>6</sup> library. It leverages d3 engine to render data driven charts in SVG format. The main objective of this tool is to provide an easy, yet powerful JavaScript dashboard which can be utilized to perform data visualization for cube data and analysis in the browser as well as on mobile devices. Available at: <https://github.com/OpenGovIntelligence/qb-multi-dimensional-charting>
7. Superset and Druid pipeline. This pipeline is using Apache Superset<sup>7</sup> and Apache Druid<sup>8</sup> in order to enable business intelligence in a web application.

Combination of the mentioned tools was used to query, and analyse large data streams.

---

<sup>6</sup> <https://dc-js.github.io/dc.js>

<sup>7</sup> <https://github.com/apache/incubator-superset>

<sup>8</sup> <http://druid.io/>

## 5 Conclusion

This deliverable provides description of the components delivered in the final phase of OpenGovIntelligence project (Month 33 of the project). Detailed information about usage of previously available tools are documented in Deliverable D3.2 – “Report on OpenGovIntelligence ICT tools – first release” and D3.4 – “Report on OpenGovIntelligence ICT tools – second release”.

All the tools available as the result of OpenGovIntelligence project were constantly updated based on the evaluation results: to improve usability and performance as well as to add functionality according to identified user needs.

The end result is a collection of software tools that supports the life cycle of managing, disseminating and exploiting statistical data in the context of co-created public services. All tools are available under an open source licence, enabling them to be used widely by public administrations and private companies across Europe. Where necessary they can be further developed, customised or integrated into other software systems.

Through a cycle of applying the tools in the context of the OpenGovIntelligence pilot projects, evaluating the results and identifying and implementing improvements, the tools have been shown to meet the needs of a range of practical applications. They constitute a significant advance in the capabilities of European public administrations for better-informed decision making and public service design.

## References

- [1] D1.1 OpenGovIntelligence challenges and needs
- [2] D2.1 OpenGovIntelligence framework - first release
- [3] D2.2 OpenGovIntelligence framework
- [4] D3.1 OpenGovIntelligence ICT tools - first release
- [5] D3.2 Report on OpenGovIntelligence ICT tools - first release
- [6] D3.3 OpenGovIntelligence ICT tools - second release
- [7] D3.4 Report on OpenGovIntelligence ICT tools - second release
- [8] D3.5 OpenGovIntelligence ICT tools
- [9] D4.1 Pilots and Evaluation plan
- [10] D4.4 Evaluation results - Second round