THE SEMANTIC DATA CUBE SYSTEM PLATO AND ITS APPLICATIONS*

Dimitris Bilidas[1], Anastasios Mantas[1], Filippos Yfantis[1], George Stamoulis[1], Manolis Koubarakis[1], José María Tárraga Habas[2], Eva Sevillano Marco[2], Fabien Castel[3], Camille Laine[3]

[1]Dept. of Informatics and Telecommunications,
National and Kapodistrian University of Athens, Greece
[2]Image Processing Laboratory, Universitat de València, Spain
[3]Murmuration SAS, France

ABSTRACT

We present Plato, the first semantic data cube implementation that utilizes ontology-based data access technologies. Plato is demonstrated in two use cases of the Horizon 2020 project DeepCube, introducing a semantic approach that allows combining information from data cubes and other sources to tackle climate induced migration in Africa and a sustainable tourism service based on Copernicus data.

Index Terms— Semantic data cubes, ontologies, ontology based data access, earth observation

1. INTRODUCTION

A data cube is a multidimensional array of values, inherently serving as a fundamental data structure for the storage of Earth observation (EO) data and other multidimensional data for analysis purposes. Various data cube infrastructures have arisen with a primary focus on EO data, including the Open Data Cube infrastructure in Australia, the Euro Data Cube, and the Earth System Data Cube. These infrastructures come equipped with libraries and APIs, such as xarray and YAXArrays, meticulously designed to facilitate the efficient storage and querying of multidimensional data.

Plato is a pioneering semantic data cube system utilizing geospatial ontology-based data access (OBDA) technologies. Using the OBDA approach, we design an ontology that captures the geospatial knowledge about entity classes and properties within a specific application domain, and introduce mappings to the underlying data cubes and sources. This task proved to be very challenging, due to the *impedance mismatch* between the concepts of an ontology language (directed graphs of classes, instances, properties and values) and the concepts of data cubes (multidimensional arrays of values).

The system Plato is demonstrated in two real world scenarios: (i) climate induced migration in Africa and (ii) sustainable tourism. Both applications utilize data cubes to store various sources of information and implement machine learning models to produce displacement causal graphs and an interactive sustainable tourism dashboard respectively.

2. SEMANTIC DATA CUBE SYSTEMS

The concept of *semantic EO data cubes* (or *semantic data cubes* for simplicity) was first presented by Augustin et al. in [1]. The term *semantic* was used to distinguish them from regular EO data cubes that contain numbers without high-level meaning for the user (e.g., reflectance values). In semantic data cubes, these values are intricately connected to *symbolic high-level concepts*, allowing users to not only gain insights into the specified concepts but also establish associations with the original values. Beyond imparting knowledge through interpretations, a semantic data cube has the potential to streamline the integration of external knowledge (datasets) and enable the linking of such information with the original values, fostering a comprehensive combined analysis. Utilization of such systems becomes evident in numerous scenarios involving geospatial data. For instance, demographic data released by a governmental organization can be leveraged to pinpoint major cities situated within a specified distance from regions designated as pine forests.

In their work [2], Sudmanns et al. demonstrate that this combined analysis is possible, through an infrastructure which supports users to transform geodata into information. Their system, called *Sen2Cube.at* [3], utilizes computer vision (CV) to automate semantic enrichment on a big EO data scale (all Sentinel-2 MSI images covering Austria; however, the approach is transferable to other geographical regions and sensors), while an interactive web-based graphical user interface (GUI) allows users to create, save and share queries in a knowledgebase, without the need of technical expertise. More specifically, a user first defines an area and a timeframe

*THIS WORK WAS SUPPORTED BY THE HORIZON 2020 PROJECT DEEPCUBE (GA NO. 101004188), THE HORIZON EUROPE PROJECT STELAR (GA NO. 101070122) AND THE ESA PROJECT DA4DTE (SUBCONTRACT 202320239).

of interest for the generic factbase, where multiple data cubes can be accessed. Then, by combining spectral categories, continuous variables and additional geographic information, they can build a semantic model in the knowledgebase (the model is translated into a query against the factbase using an inference engine [4]). These spectral categories (e.g., types of vegetation, landforms, water depth, etc.) are generated using the Satellite Image Automatic Mapper (*SIAM*) [5] software, which is responsible for the base-level semantic enrichment by performing categorization of optical multi-spectral EO imagery from multiple sensors in an automated manner. This methodology is also present in a more recent publication [6] by Sudmanns et al., where the greenness of Austria is measured by combining three information layers (i.e., density, change, and lifespan of vegetation) in one semantic model.

Following a different approach, the system Plato [7] goes beyond the work of Augustin et al. by designing a semantic data cube system focusing on techniques from the area of *geospatial ontology-based data access* and utilizing optimizations on the data access side. Performance evaluation of the system [7] consists of experiments with different query types over data cubes varying in size and complexity, along with vector datasets and the results of our optimization techniques. The five different data cubes we used contain time, latitude and longitude as the primary dimensions, along with several data variables. Alongside those, we used vector data concerning fire prediction data for Greece (point geometries), Natura-protected areas for Europe (multipolygon geometries), and administrative data for Brazil (polygon geometries). We show that utilizing a cache table allows us to overcome the overhead introduced by FDWs when retrieving requested data from a foreign table. This is more apparent in queries that concern a range of dates, where the cache implementation shows the best results. Regarding joins between raster and vector data, we show the benefits of our join optimizations instead of letting PostGIS handle the joins by making the necessary pixel-to-point transformations. Finally, caching data for specific observations and timeframes, when combined with Raptor Join, provide the best speedup (for data of substantial size).

3. THE SYSTEM PLATO

The architecture of Plato is shown in Figure 1. The two main components of Plato are the OBDA system Ontop [8] and the PostGIS backend. During initialization of the Ontop engine, an ontology in the OWL2 ontology language is defined, alongside a specified set of mappings.

The PostGIS backend contains virtual tables used to communicate with data cubes (stored locally or remotely). This communication is achieved through Python scripts utilizing the Xarray [9] library and the Multicorn package [10], to implement Foreign Data Wrappers (FDW). Cache tables of raster data and efficient joins of raster and vector data are also implemented at the PostGIS level. These techniques optimize the handling of large volumes of data through caching, as well as joining of raster and vector data with Raptor Join.

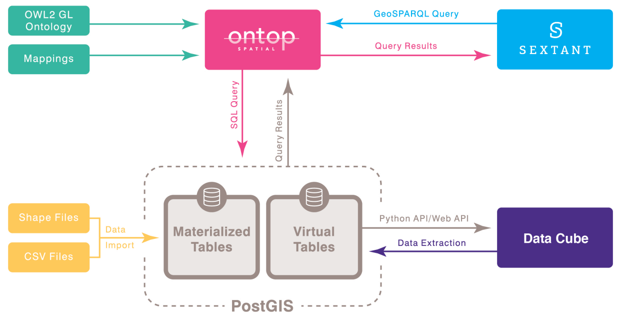


Fig. 1. The architecture of Plato

Caching raster data in PostGIS through modifications to the Ontop plugin, allows us to efficiently query large volumes of data cubes by materializing and readily storing various portions of them. To achieve this, during the translation of queries from GeoSPARQL to SQL, Plato identifies chunks of raster data that require access and transformation into geometries, saving them in an intermediate cache table within the database. To verify the presence of the requested data in the cache table, we implemented data structures as indices within the Ontop plugin. Currently, a fully implemented hash table handles the time dimension of the datasets, while similar functionality for latitude and longitude dimensions is in progress using R-trees. If the requested data is identified in the cache table, the query translation process into SQL is modified to access the cache rather than a FDW virtual table.

Following extensive testing of various GeoSPARQL queries on large data cubes, it became evident that accessing significant portions of data cubes and transforming each pixel into a vector point created a bottleneck in our system. The Raptor Join method, as outlined in [11], addresses this issue by selectively reading parts of the raster that overlap with a set of vector geometries. Notably, this method eliminates the need for conversions between raster and vector forms to execute a join. Implemented in Plato as a FDW, the Raptor Join method computes the result of a spatial operation as output. The necessary inputs include a set of vector geometries, an EO variable name (raster), an aggregate function name (e.g., sum, max, count, etc.), and a specific time frame. By introducing properties that represent these parameters to a designated ontology, a single mapping suffices to connect Ontop with the FDW operator.

In Plato, data cubes are stored either in a `.zarr` directory format or as `.nc` (netCDF) files. Despite employing compressed formats, numerous data cubes prove to be excessively large for unpacking and materialization within a PostgreSQL database. To address this challenge, we employ FDWs and the Xarray package, enabling us to conveniently handle labeled multi-dimensional arrays. Employing FDWs and Xarray may prove to be resource-intensive, both in terms of time and memory, and is not a practical approach for handling large data cubes. Consequently, we opted to explore parallelization modules in Python. Since FDW applications are not IO-bound, multi-threading did not yield significant benefits. On the contrary, multiprocessing led to substantial speed improvements by leveraging data chunking and dispatching reading tasks to multiple spawned processes whenever feasible.

In order to facilitate testing and a successful deployment of the entire pipeline, we have developed a dockerfile to build an image that installs all necessary components (PostgreSQL, Python3, Multicorn, Xarray, Zarr) and exposes a port to access the database within the created container.

4. APPLICATIONS

The system Plato was developed and tested for three use cases in the context of the H2020 project DeepCube: (i) climate induced migration in Africa, (ii) sustainable tourism and (iii) fire risk management. In [12] we presented a pipeline that utilizes EO data to produce fire risk maps for the Mediterranean region. In this paper we present Plato in two different application domains, showing the versatility of the system.

The approach we follow for each application involves domain experts in order to understand the input sources and design an appropriate ontology to capture the knowledge. The ontology terms are then used to produce the appropriate mappings that our system requires to translate our SPARQL queries to SQL. This process allows us to enhance our original input sources with semantic interpretations, interlink them and provide a way to query them using terms from the ontology.

Based on the requirements of each application, we can then formulate appropriate GeoSPARQL queries and present the results to our end-users. Plato allows us to express the following classes of queries:

1. Queries on satellite image attributes (EO data).
2. Semantic queries on the low-level content (raw values of data cube variables).
3. Semantic queries on the high-level content (values concerning classes and their properties).
4. Any of the above query classes together with a spatial and temporal extent.
5. Any of the above query classes together with a reference to an external data source.

In current data cubes, queries of Classes 1 through 4 are able to be answered in some manner. In Plato, implemented in DeepCube, all of the above classes of queries are possible, with an emphasis on Class 5. Using the OWL 2 Web Ontology Language, an ontology has been created for each of the use cases, based on their EO and non-EO datasets. For the EO datasets the notion of observation from the RDF Data Cube Vocabulary was adopted, that has been a W3C recommendation since 2014. Each observation has an acquisition time, coordinates and attributes. For the non-EO datasets, different classes with object and data properties to represent the various attributes were created.

4.1. Climate induced migration in Africa

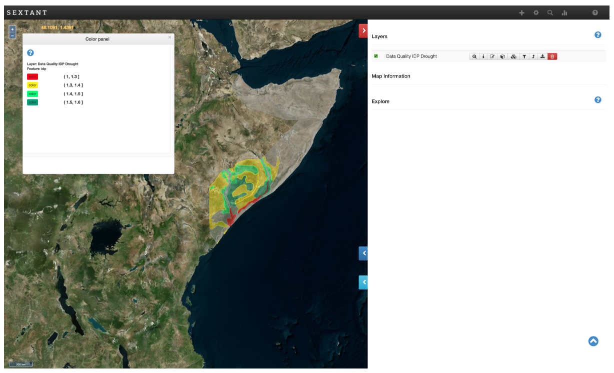
This use case focuses on how the biosphere and anthroposphere are affected by extreme weather events, such as heatwaves, droughts, and floods, as well as changing climatic circumstances. The climate issue is both caused and impacted by humans, and mitigation and adaptation strategies greatly depend on a knowledge of both effects. The overarching goal of the use case is to model, anticipate, and understand climate-induced migration flows in Africa from reliable data. We aim to offer insights into drought-induced displacement trends by utilizing causal time series analysis to identify displacement triggers and quantify their causal relationships and time lags. Additionally, the use case products aim to anticipate and follow-up displacement and provide narratives for these displacements. To achieve this, we need to combine different data sources in order to (i) identify the main environmental and socioeconomic drivers of human mobility and develop models able to reproduce and forecast migration flows, (ii) apply causal discovery methods to gain a deeper understanding of the characteristics of the climate-induced migration flows, and (iii) establish the causal relationships of environmental and socioeconomic drivers with human mobility in sub-Saharan Africa.

To this purpose, we compiled and structured a data cube integrating socioeconomic, environmental and climatic variables and the longest drought displacement time-series existing (covering the 2006-2022 period in Somalia). Through them, causal graphs are derived, applying causal inference at district level. The data we utilized include socioeconomic indexes, displacement data, ERA5 land observations and precipitation from the Climate Hazards Group InfraRed Precipitation with Station.

With the flexibility of FDWs, concepts like the aforementioned variables can be expressed in our ontology [13] as data properties of broader classes, which categorize diverse user requirements. An example query regarding the Causality class is the following:

```
SELECT ?district ?causal_variable ?causal_link ?mean ?total_mean
    ?geometry
WHERE {
?cs a geo:Causality ;
    uc2:hasDistrictName ?district ;
    uc2:causalVariable ?causal_variable ;
    uc2:causalLink ?causal_link ;
    uc2:hasMean ?mean ;
    uc2:hasTotalMean ?total_mean ;
    uc2:hasAcquisitionDate ?date ;
    uc2:hasDistrictGeometry ?geometry .
FILTER(?causal_variable = "IDP Drought")
FILTER(?date > "2010–01–01T00:00:00" && ?date < "
     2013–01–01T00:00:00")
}
```

In this query we request the districts of Somalia that have a causal link between *drought* and any other index available. Alongside those, two averages of the drought variable for each district are calculated and returned: ?total_mean, which spans the entire 2006-2022 period, and ?mean, which spans a user-defined 3-year interval (start of 2010 - end of 2012).



Layer: Data Quality IDP Drought
Feature: idp

🟥	(1, 1.3]
🟨	(1.3, 1.4]
🟩	(1.4, 1.5]
🟩	(1.5, 1.6]

Data Quality IDP Drought

Fig. 2. Data quality layer in Sextant, for the Internally Displaced Persons (IDP) Drought index. The color map shows districts in Somalia that pass the quality check threshold, along with their quality value.

In this use case, Plato is used to allow integration of EO (raster) and non-EO (vector) datasets in a unified manner and formulate GeoSPARQL queries that are used to produce thematic maps with the visualization tool Sextant, as narratives for displacements. There are three types of semantic queries that were implemented based on the use case needs: (i) data quality queries to check the quality of variable's values for each district, (ii) early warning queries over the different variables for a specific district and (iii) queries over the causal graphs.

4.2. Copernicus services for sustainable tourism

The main objective of this application is to produce a pricing tool for hotels and package tours, which is independent of the current major booking platforms and which incorporates an environmental and sustainable tourism dimension. Our motivation is to reduce the impact of tourism on planet Earth by implementing a business system based on supply, demand, but also environmental impact. The objectives of this use case are: (i) to characterize the present tourism environmental footprint, tourism demand and tourism offer on the areas of interest, (ii) to develop an engine evaluating the environmental footprint of tourism on a given destination and at a given period, (iii) to develop a pricing engine evaluating downward or upward trend to be applied to a tourism package depending on its destinations and travel periods.

The data cubes used to produce the services, combine air quality information from the Copernicus Atmosphere Service (CAMS) along with weather conditions from the ERA5-Land hourly dataset. We also rely on tourism visit data provided by the Orange FluxVision service and tourism pricing information. FluxVision data is obtained by processing a mix of mobile phone network events and socio-demographic customer data, allowing after data adjustment processes to provide a statistical estimate of a number of people present in an area. The tourism pricing information are extracted from the Amadeus API[1].

For this application, we use Plato to integrate air quality data with tourism data. In order to access the different input sources, we designed an ontology [13] to capture the domain, that allows us to pose GeoSPARQL queries that combine all our sources to analyse the data. A query showcasing links between air quality and tourism frequentation is presented below:

```
SELECT ?area ?avg_no2 ?total_excursionists ?wktAOI
WHERE {
?tf a uc5:Tourism ;
    uc5:hasArea ?area ;
    uc5:hasDate ?date ;
    uc5:hasExcursionists ?total_excursionists .
?aoi a uc5:AOI ;
    uc5:hasName ?area ;
    geo:asWKT ?wktAOI .
?join a uc5:Raptor ;
    uc5:hasResult ?avg_no2 ;
    uc5:hasVariable "cams_no2_conc"^^xsd:string ;
    uc5:hasAggregateFunction "avg"^^xsd:string ;
    uc5:hasAcquisitionDate ?date ;
    uc5:hasGeometry ?wktAOI .
FILTER(?date = "2022–01–01T00:00:00")
}
```

This query utilizes the Raptor Join technique in order to calculate the average nitrogen dioxide value for several areas of France, in a particular day. For the same spatiotemporal input, the total number of day-trippers is also requested. We could alternatively ask for different types of visitors, and/or different CAMS variables. The returned results can be viewed as a thematic map with the areas of interest using Sextant, to provide a visual interpretation for our end-users.

[1]https://developers.amadeus.com/

5. REFERENCES

[1] H. Augustin, M. Sudmanns, D. Tiede, S. Lang, and A. Baraldi, "Semantic Earth observation data cubes," *Data*, vol. 4, no. 3, 2019.

[2] Martin Sudmanns, Hannah Augustin, Lucas van der Meer, Andrea Baraldi, and Dirk Tiede, "The austrian semantic eo data cube infrastructure," *Remote Sensing*, vol. 13, no. 23, 2021.

[3] M. Sudmanns M. Belgiu D. Tiede, A. Baraldi and S. Lang, "Sen2cube.at: Semantic earth observation data cube analysis," 2021.

[4] L. van der Meer, M. Sudmanns, H. Augustin, A. Baraldi, and D. Tiede, "Semantic querying in earth observation data cubes," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLVIII-4/W1-2022, pp. 503–510, 2022.

[5] A. Baraldi, "Satellite image automatic mapper - SIAM™," 2001.

[6] European Commission, Joint Research Centre, P Soille, S Lumnitz, and S Albani, *Proceedings of the 2023 conference on Big Data from Space (BiDS'23) – From foresight to impact – 6-9 November 2023, Austrian Center, Vienna*, Publications Office of the European Union, 2023.

[7] Dimitris Bilidas, Anastasios Mantas, Filippos Yfantis, George Stamoulis, and Manolis Koubarakis, "Plato: A semantic data cube implementation using ontology-based data access technologies," in *BiDS*, 2023.

[8] G. Xiao et al., "The virtual knowledge graph system ontop," in *ISWC*, 2020.

[9] S. Hoyer and J. Hamman, "xarray: N-D labeled arrays and datasets in Python," *Open Research Software*, vol. 5, no. 1, 2017.

[10] R. Dunklau and F. Mounier, "Multicorn - PostgreSQL extension," 2015.

[11] S. Singla, A. Eldawy, T. Diao, A. Mukhopadhyay, and E. Scudiero, "The Raptor Join operator for processing big raster + vector data," in *ACM SIGSPATIAL*, 2021.

[12] Dimitris Bilidas, Anastasios Mantas, Filippos Yfantis, George Stamoulis, Manolis Koubarakis, Spyros Kondylatos, Ioannis Prapas, and Ioannis Papoutsis, "Fire risk management using data cubes, machine learning and OBDA systems," in *ACM SIGSPATIAL*, 2023.

[13] George Stamoulis, "Deepcube: Ontologies for semantic data cubes," Feb. 2023.