# ODIN: A Dataspace Management System

Sergi Nadal, Kashif Rabbani, Oscar Romero, and Shumet Tadesse

Universitat Politècnica de Catalunya (BarcelonaTech), Barcelona, Spain

**Abstract.** ODIN is a system that supports the incremental pay-as-you-go integration of data sources into *dataspaces* and provides user-friendly querying mechanisms on top of them. We describe its main characteristics and underlying assumptions, including the user interactions required. ODIN's novelty lies in a largely automated bottom-up approach (i.e., driven by the sources at hand) that includes the user in the loop for disambiguation purposes. The on-site demonstration will feature an ongoing project with the World Health Organization (WHO).
**Online demo and videos**: `www.essi.upc.edu/dtim/odin/`

## 1 Introduction

A prominent approach to virtual data integration is that of exposing an **ontology**, which conceptualizes the domain of interest, to offer a uniform query interface over the **sources**. Queries over the ontology are rewritten over the sources via schema **mappings**. The maintenance of such constructs (i.e., evolving the ontology, or adding new sources and mappings) is well-known to be an arduous and manually-intensive task that hinders the ability of such systems to flexibly adapt and provide right-time integration. This limitation has been coined as the data variety challenge, which refers to the complexity of providing on-demand integration over a vast and evolving set of data sources.

Dataspaces represent a major step towards tackling the variety challenge. With the vision of reducing the usual upfront and maintenance costs, dataspaces claim for the adoption of a flexible and dynamic pay-as-you-go approach where different integration tasks are automated [1]. Supporting the end-to-end lifecycle of dataspaces is a technically challenging task. The state of the art on automatic construction of an ontology from the data sources (and their respective mappings), commonly known as *bootstrapping*, is BOOTOX [2]. Targeted to ontology-based data integration, BOOTOX generates OWL 2 QL ontologies from relational databases, together with R2RML mappings to the sources. Yet, this approach falls short in settings where managing data variety is a key requirement. On the one hand, the extraction is restricted to relational databases and misses widely used semi-structured data formats such as CSV, JSON or XML. On the other hand, such mappings conform to the *global-as-view* (GaV)


This work was partly supported by the GENESIS project, funded by the Spanish Ministerio de Ciencia e Innovación under grant TIN2016-79269-R.

family, which characterize the ontological classes and properties in terms of SQL queries, which are not well suited for highly dynamic and evolving settings.

Odin (short for *On-demand Data INtegration*), a dataspace management system grounded in knowledge graphs, was conceived to overcome the aforementioned challenges. Fig. 1 depicts how Odin supports the dataspaces complete lifecycle. Odin automatically extracts the schemata from structured (e.g., relational) and semi-structured (e.g., JSON) data sources and translates them into a canonical data model, namely RDFS. To this end, a set of production rules parse their metadata and automatically generate RDFS-compliant *source graphs*. Next, the source graphs are aligned while considering the user feedback throughout this process. As result, Odin generates *provenance graphs* (PG) tracing the results of the previous stages. A PG is a target-agnostic metadata construct (i.e., not tailored for a specific tool) about the integration of a particular set of data sources. PG captures the results of bootstraping the sources and aligning their schemata, and guarantees we can generate target-specific metadata from them[1]. Thus, PGs are used to generate the specific constructs of a given integration tool. In this demo, ODIN generates the constructs required by [3]. Precisely, *conjunctive query (CQ)-oriented graphs*, which expose the sources schemata in first-normal form, which are then linked via *local-as-view* (LaV) schema mappings (represented as graphs) to the *global graph*. LaV mappings characterize the sources in terms of a query over the ontology, which make them inherently more suitable in data variety settings. This entails a more complex query answering process, which boils down to the problem of answering queries using views. In our demo, however, we will show the feasibility of our approach in real cases.

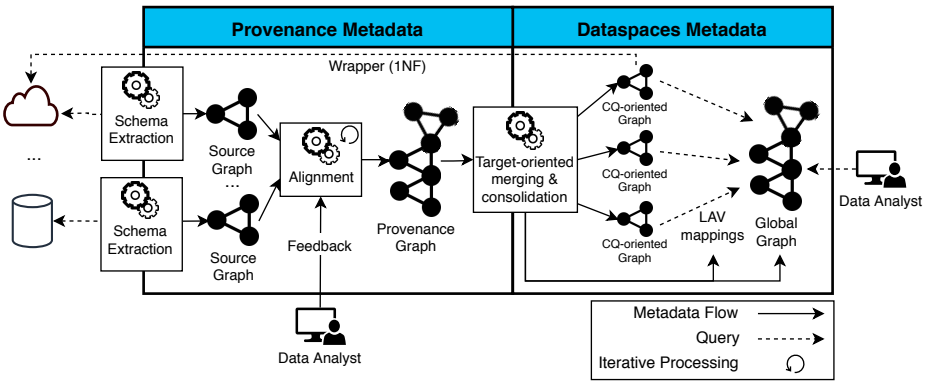


Fig. 1: High-level overview of the functionalities offered by Odin

## 2 Main Features

In this section, we detail how Odin deals with the process of generating the PGs and from there the specific constructs of a given tool.

[1] Although the focus of this paper will be on answering queries, during the demo we will highlight the ability to generate the required metadata for other ontological reasoning services (e.g., DL-Lite for satisfiability checking) from a PG.

**Generation of RDFS schemata from sources.** ODIN adopts a meta-modeling approach to bootstrap disparate sources in order to create an RDFS representation of their schemata. For each source data model, ODIN defines an equivalent first order logic representation of its meta-model. Given a source model (e.g., relational or JSON), a set of pre-defined production rules (i.e., *tuple-generating dependencies* defined at the meta-model level) generate an equivalent RDFS model[2]. Source graphs are the result of this bootstrapping phase.

**User-driven source graph alignment.** From source graphs, ODIN incrementally generates the PG, where it annotates source graph alignments in the form of taxonomies. To discover alignments, ODIN uses an enhanced version of LogMap[3], which considers Wordnet synonyms. Candidate alignments are ranked, and ODIN prompts the user to accept or reject them. Further, since aligning two ontologies is a hard task, ODIN also provides an intuitive interface to manually assert alignments.

**Querying the sources via the ontology.** This final step consists in generating the required metadata constructs to pose and resolve queries over the dataspace. To this end, from PGs, ODIN automatically generates the global graph (i.e., a merged view of the aligned source graphs) and CQ-oriented graphs, which expose a first-normal form structure of the sources. To guarantee the incremental evolution of the system, ODIN also generates LaV mappings from CQ-oriented graphs to the global graph. Since PGs were created in a bottom-up approach, we are able to automate the definition of all required constructs. Consequently, given that ODIN explicitly models the schema that sources expose, LaV mappings are exact and they are not required to deal with incompleteness on the sources. Finally, ODIN provides a user-friendly interface to pose conjunctive queries (CQs) on the global graph, that are automatically translated to SPARQL. A rewriting algorithm interprets such query and generates the certain answers under the *closed-world assumption* in terms of unions of CQs [3]. The demo will show that such constructs are automatically generated in linear time (w.r.t. the size of PG).

## 3 Demo

We will present the functionalities of ODIN via the WHO Information System to Control and Eliminate Neglected Tropical Diseases (WISCENTD)[4]. The goal of WISCENTD is to provide support in the collection, integration and analysis of data coming from different monitoring systems surveilling different aspects of neglected tropical diseases (NTDs). Data related to NTDs are largely fragmented and their integration is mandatory to shed light on NTDs around the world. The demo will simulate the day-by-day of a WHO data analyst and how ODIN is used to first collect and integrate different sources of relevance for a certain NTD, and later cross-query them. We will use relevant datasets, such as UN Data (open-data JSON datasets) about health economics indicators and

[2] `http://essi.upc.edu/dtim/ardi`
[3] `https://github.com/ernestojimenezruiz/logmap-matcher`
[4] `https://www.who.int/neglected_diseases/disease_management/wiscentds/en`

migrant information per country[5], data about diagnosis and treatment per country periodically extracted from WIDP[6] (that hosts a relational database), data about drug distribution periodically extracted from WIMEDS[7] as CSVs, etc. We will first showcase how the data analyst, just interacting with ODIN's interface, is able to integrate and query such sources in a friendly manner. ODIN allows the interested users to browse the metadata generated throughout the whole process: (i) source bootstrapping, (ii) their alignment to construct the PG (Fig. 2), and (iii) the automatic creation of the constructs for query answering. The audience will be encouraged to participate including new sources in an incremental manner, query the global graph, or even apply ODIN to other domains.

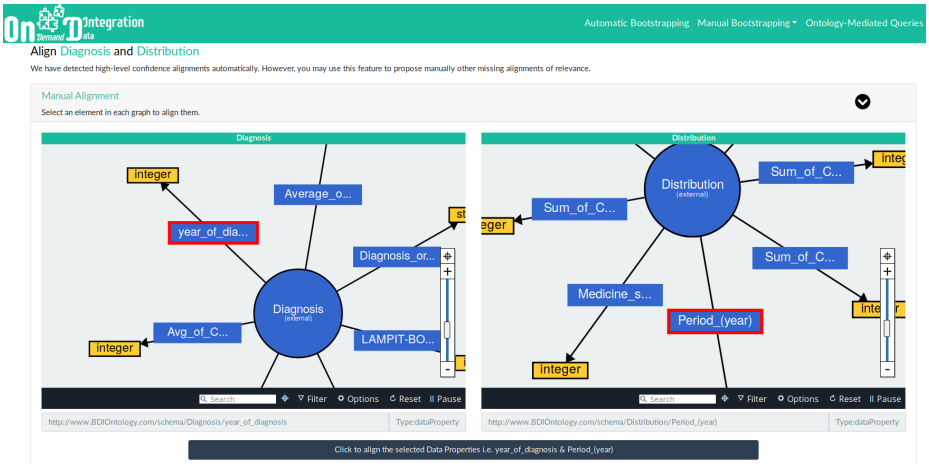


Fig. 2: Interactive alignment process

**Implementation details.** ODIN follows a service-oriented architecture, which enables extensibility and separation of concerns. The frontend is implemented in *Javascript* and resides in a *Node.JS* webserver. ODIN uses WebVOWL to visualize and interact with graphs. The backend, is implemented as a set of REST APIs defined using *Jersey* for *Java*. To deal with RDF graphs, this component makes heavy use of *Jena* and its persistance engine *Jena TDB*.

## References

1. Franklin, M.J., Halevy, A.Y., Maier, D.: From databases to dataspaces: a new abstraction for information management. SIGMOD Record **34**(4), 27–33 (2005)
2. Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M.G., Thorstensen, E., Mora, J.: BootOX: Practical Mapping of RDBs to OWL 2. In: ISWC 2015. pp. 113–132 (2015)
3. Nadal, S., Romero, O., Abelló, A., Vassiliadis, P., Vansummeren, S.: An integration-oriented ontology to govern evolution in big data ecosystems. Inf. Syst. **79**, 3–19 (2019)

[5] `http://data.un.org`
[6] `http://bit.ly/whowidp`
[7] `http://bit.ly/whowimeds`