Web Semantics: Science, Services and Agents on the World Wide Web 65 (2020) 100596

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

GTFS-Madrid-Bench: A benchmark for virtual knowledge graph access in the transport domain

David Chaves-Fraga *, Freddy Priyatna, Andrea Cimmino, Jhon Toledo, Edna Ruckhaus, Oscar Corcho

Ontology Engineering Group, Universidad Politécnica de Madrid, Boadilla del Monte, Spain

ARTICLE INFO

Article history:
Received 4 October 2019
Received in revised form 6 July 2020
Accepted 28 July 2020
Available online 8 August 2020

Keywords:
Virtual knowledge graph
Benchmark
Query translation
Data integration
GTFS

ABSTRACT

A large number of datasets are being made available on the Web using a variety of formats and according to diverse data models. Ontology Based Data Integration (OBDI) has been traditionally proposed as a mechanism to facilitate access to such heterogeneous datasets, providing a unified view over their data by means of ontologies. Recently, the term "Virtual Knowledge Graph Access" has begun to be used to refer to the mechanisms that provide query-based access to knowledge graphs virtually generated from heterogeneous data sources. Several OBDI engines exist in the state of the art, with overlapping capabilities but also clear differences among them (in terms of the data formats that they can deal with, mapping languages that they support, query expressivity that they allow, etc.). These engines have been evaluated with different testbeds and benchmarks. However, their heterogeneity has made it difficult to come up with a common comprehensive benchmark that allows for comparisons among them to facilitate their selection by practitioners, and more importantly, for their continuous improvement by the teams that maintain them. In this paper we present GTFS-Madrid-Bench, a benchmark to evaluate OBDI engines that can be used for the provision of access mechanisms to virtual knowledge graphs. Our proposal introduces several scenarios that aim at measuring the query capabilities, performance and scalability of all these engines, considering their heterogeneity. The data sources used in our benchmark are derived from the GTFS data files of the subway network of Madrid. They have been transformed into several formats (CSV, JSON, SQL and XML) and scaled up. The query set aims at addressing a representative number of SPARQL 1.1 features while covering usual queries that data consumers may be interested in.

1. Introduction

Over the last few years, a growing number of datasets have been made available in various open data portals. For example, at the time of writing, the European Data Portal[1] aggregates approximately 500 K datasets from EU countries in a diversity of domains. In this context, RDF has been proposed as a standard format for data interchange on the Web, and RDF Schema and OWL ontologies have begun to appear so as to provide shared models in some domains. However, the amount of non-RDF data (e.g., CSV, JSON, XML) that are published in these open data portals continues to dominate the scene (see Table 1), and interoperability issues hinder their (re)use and consumption.

Data integration is not a new problem, it was already identified and addressed several decades ago with an emphasis on data in relational databases, but it is exacerbated by the availability of such open data on the Web. Different techniques and tools have been used to address this problem. In our work, we focus on those approaches based on ontologies. In Ontology Based Data Access (OBDA) [1] data consumers issue queries over a dataset according to a common unified view (an ontology). The information needed to reformulate the queries is usually available in the form of declarative mappings. In Ontology Based Data Integration (OBDI) [1], these techniques are expanded to address heterogeneous datasets, whose data need to be integrated to provide answers to these queries. In both ontology-based approaches, two different alternatives exist to enable data access: (1) those where data are materialized taking into account the mappings and the ontologies (for example, data is transformed into RDF and loaded into a triple store, so that it can be queried using SPARQL), and (2) those where the transformation is done on the queries, which can then be evaluated on the original data sources. This last alternative is the one considered for our work, because it removes the need for materialization, something especially useful for very dynamic data sources [2]. We refer to it as "virtualized knowledge graph access".

* Corresponding author.
E-mail address: dchaves@fi.upm.es (D. Chaves-Fraga).

[1] https://www.europeandataportal.eu/catalogue-statistics/Evolution.

https://doi.org/10.1016/j.websem.2020.100596

Table 1
Most commonly used formats (and percentage over the total number of datasets) to publish data in mature EU open data portals.[2]

Data portal	1st format	2nd format	3rd format
Spain	CSV (50%)	XLS (35%)	JSON (33%)
Norway	CSV (77%)	GEOJSON (17%)	JSON (14%)
Italy	CSV (76%)	JSON (35%)	XML (25%)
Croatia	XLS (63%)	CSV (40%)	HTML (33%)
Luxembourg	ZIP (25%)	CSV (24%)	PDF (18%)
Ireland	JSON (49%)	CSV (39%)	TXT (22%)

To facilitate data exploitation in this context, application developers need to understand the strengths and weaknesses of existing data integration tools. Additionally, tool developers may want to know if their engines cover the requirements of real-use-case scenarios. In both cases the challenge is to develop a benchmark that covers the requirements for virtual knowledge graph access, and to ensure that is extensible and sustainable over time. In general, it is necessary to have an overview of state of the art engines that are tailored to different source formats, accepting as input those mappings that are represented in a variety of declarative languages.

Several benchmarks already exist in the state of the art of OBDA [3,4], as well as in SPARQL query federation [5–7]. The OBDA BSBM benchmark [3] is focused on comparing the performance of SPARQL-to-SQL query translation versus the performance of native RDF Stores, and only considers OBDA engines that access relational data stores. The NPD benchmark [4] specifically analyzes OBDA requirements related to datasets, query sets, mapping rules and query languages. In the area of federated SPARQL engines, existing benchmarks [5–7] are tailored to the context of SPARQL endpoint federation in an homogeneous format. As a result, none of these benchmarks address the requirement of virtualized access of multiple datasets available in heterogeneous formats. Additionally, OBDI engines have been evaluated in an ad-hoc manner [8,9] and to the best of our knowledge, no benchmarks have been developed to evaluate OBDI proposals in a systematic manner.

We have identified several challenges for the development of a benchmark for virtual knowledge graph access that can be grouped into data, queries and mappings dimensions. The data challenges refer to having multiple data sources in an assortment of formats, based on real-world data and that can scale to large sizes. The queries challenges point to SPARQL queries where different sources can be identified, where relations among sources (according to the specific data model) are exploited, and where necessary features of SPARQL are included to represent real-life use cases. Finally, the main mappings challenge is to include the relevant parameters that affect the generation of the knowledge graph [10] and give support to a set of mapping languages.

In this paper we describe a virtual knowledge graph access benchmark, GTFS-Madrid-Bench, that serves several purposes: (i) to evaluate and compare the performance of a mix of OBDA engines that access several (homogeneous) sources in the same format, but where the mapping language used by each engine is specific to the data format considered; (ii) to evaluate OBDI engines when data are centralized in a single location; and (iii) to evaluate the strengths and weaknesses of both, OBDA and OBDI engines. The general case of GTFS-Madrid-Bench is the comparison of the performance of OBDA and OBDI engines. The proposed benchmark is composed of the following elements:

- Several collections of sources in different formats (e.g. CSV, JSON, SQL, XML), which derive from the GTFS. [3] The General Transit Feed Specification (GTFS) is a de-facto standard developed by Google for the description of public transport planning, routes and fares, among others. In recent years its popularity has increased thanks to its simplicity and the fact that it has not only been adopted by Google Maps, but also by other route planning systems such as Open Trip Planner or navitia.io. feed from the city of Madrid metro. These collections are scaled up so as to allow scalability testing.
- A set of mappings represented in the family of declarative languages that address different source formats (RML, R2RML, xR2RML, ontop OBDA mappings) that map the GTFS-based data sources into the Linked GTFS ontology.[4]
- A set of 18 SPARQL queries of varied complexity.
- A set of well-established measurements [4,11] that can be taken during the different phases of the OBDI workflow [4, 12], such as query rewriting, query translation, query execution and query aggregation time.

GTFS-Madrid-Bench offers a fair environment for the comparison of different OBDA and OBDI engines, regardless of the mapping language they have implemented, as long as the new mappings follow the same restrictions and specifications defined in the benchmark. Thus, newly released tools may be evaluated with the benchmark. Additionally, although we have generated our datasets from the GTFS feed of the city of Madrid metro system, any other city's GTFS feed may be used as data in the benchmark. We provide a data generator to scale up the original data in terms of size, and distribute the datasets over different formats (e.g. JSON, XML, CSV, RDB). We demonstrate the use of GTFS-Madrid-Bench with five open-source engines: Morph-RDB,[5] Ontop,[6] Ontario,[7] Morph-CSV.[8] and Morph-xR2RML[9]

In summary, the main contributions of this work are:

1. C1: The proposal of a comprehensive and representative benchmark that includes a set of data sources, queries and mappings to be able to evaluate and comparing multiple OBDA and OBDI engines for virtual knowledge graph access.
2. C2: The extension of existing OBDA benchmark requirements to take into account (i) metrics that are commonly used in federated query-processing benchmarks; and (ii) steps defined in the new generation of OBDA and OBDI engines [2].
3. C3: A data generation process where single and mixed data formats are scaled-up based on the features of the original data model, integrating state of the art data-generator proposals for benchmark OBDA engines [13].
4. C4: Evaluation of the proposed benchmark over five different engines, discussion of the obtained results, and identification of the current limitations in the state of the art and future lines of work.

The rest of this paper is structured as follows: Section 2 introduces several notions and definitions relevant for the proposed work; Section 3 presents GTFS-Madrid-Bench and its main features, i.e. queries, datasets, mappings and metrics; Section 4 reports our experiment on evaluating five open source engines over

[2] Statistics obtained in January 2019 (note that one dataset can be made available in multiple formats).

[3] https://developers.google.com/transit/gtfs/.
[4] https://github.com/OpenTransport/linked-gtfs.
[5] https://github.com/oeg-upm/morph-rdb.
[6] https://github.com/ontop/ontop.
[7] https://github.com/SDM-TIB/Ontario.
[8] https://github.com/oeg-upm/morph-csv.
[9] https://github.com/frmichel/morph-xr2rml.

GTFS-Madrid-Bench; we discuss our findings in Section 5; Section 6 reports the related work in OBDA and SPARQL federation benchmarks, OBDA/OBDI approaches and mapping languages; and finally, Section 7 recaps our findings and conclusions.

2. Preliminaries

In this section, we introduce the main concepts and definitions that are later used to explain our work. Besides this, well-known concepts from the literature such as SPARQL queries and result sets [14], or ontologies [15], will be used throughout the paper.

Sources & dataset: we define a source as a tuple $\gamma = (\varphi, \Sigma, f)$, where φ is the data of any entity from our domain, Σ is the model of the data, e.g. the columns of a CSV or the schema of a database table for SQL, and f is a specific data format such as CSV, JSON, XML, or SQL, among others. We define a dataset as a set of *Sources*, i.e., $\mathcal{D} = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$.

Example 1. We define the following dataset $\mathcal{D}_1 = \{(Routes, \Sigma_1, SQL), (Stops, \Sigma_2, JSON)\}$ that involves the data of the metro routes (13 instances) and metro stops (1262 instances) in SQL and JSON formats, respectively. Both sources rely on different schemata Σ_1 and Σ_2, the first specifies the columns of a table, and the second the JSON keys.

Dataset generator: we define a dataset generator as a function δ that takes as input a tuple $(\mathcal{D}, s)$, where $\mathcal{D}$ is a dataset and s is a non-negative number that specifies a scale factor. The output of δ is a dataset $\mathcal{D}'$ containing enlarged versions, according to s, of the data (φ) within the sources of $\mathcal{D}$.

Example 2. Assuming $\mathcal{D}_1$ from *Example* 1 and a scale factor s of 2.5, a dataset generator may produce the following $\mathcal{D}' = \{(Routes{-}2.5, \Sigma_1, SQL), (Stops{-}2.5, \Sigma_2, JSON)\}$. The schemata and the formats are the same, but the data of *Routes*2.5 and *Stops*2.5 has been scaled up from their versions in $\mathcal{D}_1$, containing 189 and 3536 instances respectively.

Mapping: a mapping m is a set of rules that specify the relationship between an ontology and the model of one or more sources. A mapping rule relates the elements within the schema of a source, with elements from an ontology, including constants. In other words, a mapping rule r contains the correspondences between an element e within a schema of a source Σ and an element e_* of an ontology Σ_*. The ontology is known as a unified view, since it is the output of translating heterogeneous sources into the same model.

Example 3. Given the Linked GTFS ontology and a CSV file with the columns "id" and "route", a mapping may state that each row generates a subject that includes the value of the column "id', the predicate *foaf:name*, and its object with the corresponding value in the column "route".

Experiment configuration: we define an experiment configuration c as $(\mathcal{D}, q, M)$ where $\mathcal{D}$ is a dataset, q is an SPARQL query and M is a set of mappings.

Example 4. We can specify the following experiment configuration $(\mathcal{D}_1, q1, \{shapes, trips\})$, where $\mathcal{D}_1$ is the dataset specified in *Example* 1, q_1 is the SPARQL query reported in Table 5, and M is the set of mappings $\{shapes, trips\}$ reported in Table 4.

Processor: Given an experiment configuration c and an ontology Σ_*, a processor represents a software component that encodes the function ϕ that takes as input a pair (c, Σ_*), and outputs a SPARQL result set R [14].

Internally, the processor translates the SPARQL query q into one or more queries expressed in different languages, depending on the formats within the dataset of c, using the mappings M. Then, the processor distributes and evaluates the queries and gathers the results. Consequently, a unified result set is provided as output. This task is known as **Virtual Knowledge Graph Access**. We distinguish two kinds of processors: OBDA and OBDI. The former are able to handle only experiment configurations where all the data sources have the same data format, while the latter are able to handle any experiment configuration.

3. The GTFS-Madrid-Bench

The GTFS-Madrid Benchmark consists of an ontology, an initial dataset of the metro system of Madrid following the GTFS model, a set of mappings in several specifications, a set of queries according to the ontology that cover relevant features of the SPARQL query language, a data generator based on a state of the art proposal [13], and a set of relevant metrics. In the following sections we describe in detail the resources of our virtual knowledge graph access benchmark. They are aligned with an extension of the requirements detailed in [4] (focused on benchmarks for OBDA) that we tailor to our context (Table 2). All the resources described in this section are available online.[10]

3.1. The linked GTFS ontology

GTFS is a *de-facto standard* developed by Google for the description of public transport schedules, routes, fares, etc. The specification defines the headers of 13 types of CSV files and a set of rules. Each file, as well as their headers, can be mandatory or optional and they have relations among them.

The Linked GTFS vocabulary[11] can be seen as an ontology that represents the entities, properties and relationships described in the GTFS specification. The GTFS-Madrid-Bench mappings have been aligned to a subset of this vocabulary, since the subway feed provides only the mandatory CSV files from the GTFS specification. Its conceptual model is shown in Fig. 1, and a description of its classes is given in Table 3. The ontology usually defines one class for each of the sources in the GTFS specification with the corresponding data and object properties, but there are some additions. The `gtfs:Service` class represents information on the dates when a service (represented in GTFS in the files calendar and calendar_dates) is available for one or more routes; the ontology also adds the `gtfs:ServiceRule` class, together with its two subclasses (`gtfs:CalendarRule` and `gtfs:CalendarDateRule`), to represent the service rules specified in the calendar and calendar_dates files. Finally, the class `gtfs:WheelchairBoardingStatus` and its three possible values (instances) have also been added to represent the corresponding field definitions in stops and trips.

In general, all of the ontology classes have been populated except for `gtfs:FareClass` and `gtfs:FareRule`, because the Madrid GTFS data does not contain information on these two entities. The `gtfs:RouteType` class is not considered, because the data covers only the Metro system.

10 https://github.com/oeg-upm/gtfs-bench.

11 https://github.com/OpenTransport/linked-gtfs.

Table 2
Virtual Knowledge Graph Access Benchmark Requirements.

Variable	Requirement
Ontology	The **ontology** should include classes with data and object properties
Dataset	The **virtual instance** should maintain the constraints defined in the original dataset
Dataset	The **virtual instance** should be based on real world data
Dataset	The **virtual instance** should be distributed in different data formats
Mappings	The **mappings** should be able to indicate the format of the source
Mappings	The **mappings** should be expressed using well known mapping languages
Queries	The **query set** should be based on actual user queries
Queries	The **query set** should be complex enough with relations among same but also different data sources
Metrics	The **metrics** should provide relevant general information but also specific measures for each defined phase

Table 3
LinkedGTFS classes and their descriptions.

Class	Description
Agency	Agency that operates a certain transport mode
Stop	Physical location where a vehicle stops or leaves. Multiple routes may use the same stop.A stop may be wheelchair-accessible.
Route	Collection of one or more trips.Usually two trips in each direction.
Trips	A trip in a certain direction passes by several stops.A trip is associated with a shape.
StopTimes	An ordered sequence of stops.Includes their arrival and departure times.
Service	Set of dates when a service is available.A Service follows a rule that may have exceptions.
ServiceRule	May be a calendar rule or a calendar date rule.
CalendarRule	For a certain period, weekdays where active.
CalendarDateRule	Date to add or delete a service.
Shape	A polygon associated to a trip.
Frequency	Frequency of a trip.
WheelchairBoardingStatus	Indicates whether wheelchair boarding is possible.Available for a trip or a stop.

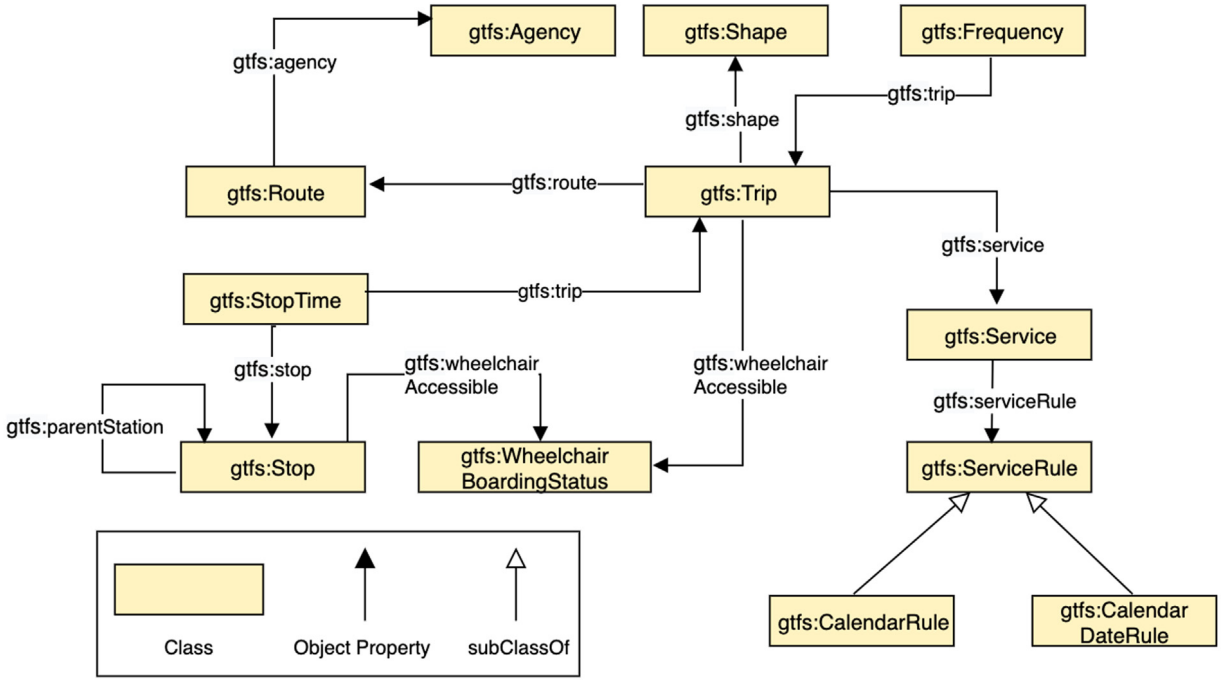

Fig. 1. LinkedGTFS Ontology. Subset of the LinkedGTFS ontology used in the GTFS-Madrid-Bench for virtual knowledge graph access. There are eleven object property relations among the classes, and two subClassOf relations.

3.2. Dataset generation

Dataset generation for a virtual knowledge graph access benchmark should be focused on the two main variables that allow testing the capabilities of the engines: (i) data size, and (ii) formats in which data can be expressed. In the context of data generation for OBDA, VIG [13] proposes the use of R2RML mappings for an efficient scale-up of the size of an RDB dataset instance. In this case, only one data format (SQL) is involved in the process.

We use GTFS as the original data source for several reasons: First, GTFS has been the *de-facto* standard for publishing transport data on the web; it also comes with a clear specification, making it easy to understand. Second, the GTFS model comprises several entities that are related through a variety of relationships. In addition it includes different data types such as strings, integers, and booleans. Finally, many cities have adopted the GTFS data model and have published their GTFS data online. Although in our benchmark we propose the use of the GTFS Madrid subway data, GTFS data from a different city could be used as the original data source.

The GTFS-Madrid-Bench proposes an extended workflow which uses VIG as the data generator engine for the generation of the datasets, and takes into account multiple data formats (see an

example in Fig. 2). We describe the detailed steps of the proposed data generation workflow, together with some examples:

(**1**) **Data preparation.** The original data source, GTFS, is in CSV format. VIG requires an instance of an RDB and an R2RML mapping for scaling up the data source. We use Morph-CSV [16], which takes as inputs a set of spreadsheets in the form of CSV files, their corresponding annotations using CSVW [17], and an RML mapping [18]. It automatically produces the corresponding schema of an RDB (identifying typical constraints such as datatypes, PK/FK, indexes and NULLs) and an R2RML mapping document, which are the inputs for VIG.
For the Madrid-GTFS-Bench, we use as input an open dataset $GTFS^{csv}_{mad}$ = ($GTFS_{mad}$, GTFS, CSV). $GTFS_{mad}$ is the set of data sources of the subway network of Madrid that has been provided by its transport authority according to the schema GTFS as described in its specification.[12] This dataset is composed of a set of CSV files containing data of Agency, Route, Shape, Frequency, Trip, StopTime, Stop, Calendar and CalendarRule. This input is not modified during process, which means that the generated datasets are defined by the same schema, and all of the generated data is obtained from this initial dataset. We manually create the corresponding RML mapping rules and CSVW metadata annotations and, using the Morph-CSV engine, we automatically generate the corresponding RDB instance GTFS-SQL-1=($GTFS^{sql}_{mad}$, (1) dataset with the integrity and domain constraints of the source model, and the R2RML mapping rules.

(**2**) **Data creation.** VIG [13] takes into account the ontology and the set of R2RML mappings to generate each dataset. This engine also receives as input a scale value *s* that indicates that the size of each table of the database increases *s* times. The output of VIG is a set of CSV files, one file for each table of the RDB. In this step the dataset GTFS-CSV-s=($GTFS^{csv}_{mad}$, (s) is generated, where *s* is the selected scale value.

(**3**) **Data distribution.** Finally, each dataset generated using VIG is distributed in several formats. We use open source tools to perform this step such as csv2json, from Python CSVKit,[13] and di-csv2xml,[14] depending on the data formats (JSON and XML). We divide the distribution into two categories in order to cover both OBDA and OBDI approaches:
In the first category, focused on providing support to OBDA techniques, the sources of each dataset are transformed into a single format (e.g. CSV files are transformed into JSON files). The dataset is transformed to the corresponding one in JSON, XML, SQL and MongoDB, obtaining the following datasets: GTFS-F-s=($GTFS^{F}_{mad}$, (s) where *s* is the scale value and F ∈ {JSON, XML, SQL, MongoDB}.
In the second category, focused on OBDI approaches, the sources of each dataset are transformed from the CSV files into multiple formats (e.g. CALENDAR is a JSON document, AGENCY is an XML file, etc.). The benchmark provides a configurable generator to obtain the desirable dataset. More in detail, the user may select the sources associated to each format, and then the tool generates the corresponding dataset and set of mapping rules. With this approach, the data distribution in the benchmark has the flexibility that allows the study of the impact of different parameters that affect the virtual knowledge graph access engines.

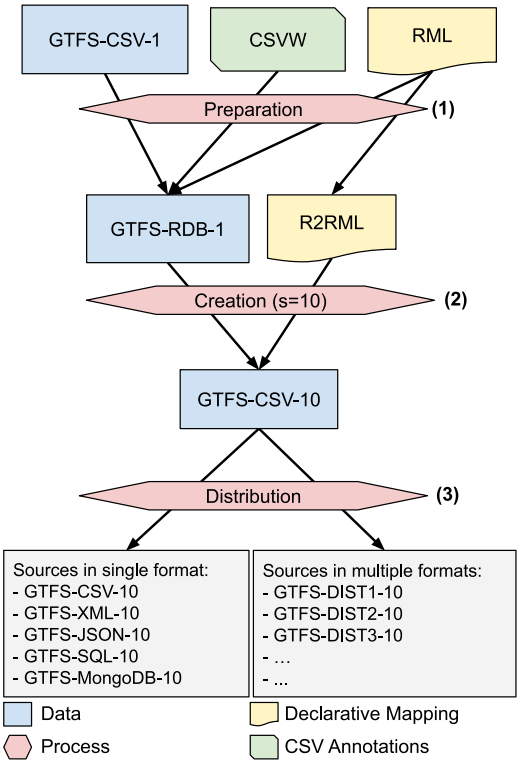

Fig. 2. GTFS-Madrid-Bench Generation Workflow with scale value 10. From the original 10 CSV files of Madrid Metro GTFS (1) we use Morph-CSV to generate the corresponding RDB instance and an R2RML mapping that are the required inputs for (2) scaling up the data using VIG and (3) distributing the generated dataset to different formats.

For example, a parameter that can be studied is the join selectivity. The value of this parameter between shapes and trips is different than between routes and agencies, and depending on the format of each source, the total query execution time of a processor may be impacted. Other parameters such as the number of joins among sources in same/different formats and the impact of the data size in different sources can also be studied.

We want to be able to compare the results obtained by processors, with the results obtained by the materialized graph in RDF. For this purpose, we take the output of VIG (e.g GTFS-CSV-5, GTFS-CSV-10) and we run a knowledge graph creation process using the SDM-RDFizer[15] engine, which generates the materialized KG in RDF using RML mapping rules. We selected this tool because it passed all the RML Test Cases [19] for CSV files,[16] hence we assume that the generation is correct and that it provides a set of techniques to optimize the generation of RDF at scale.

3.3. Mappings

Mappings play one of the most important roles in the benchmark since they are the main element used for the query translation process. In the state of the art there are multiple engines and tools that use different mapping languages. We select a set of the most relevant declarative mapping languages in the state of the art and we generate the corresponding mapping rules. In more detail, the GTFS-Madrid-Bench provides:

- One R2RML mapping document for accessing SQL datasets.

12 https://developers.google.com/transit/gtfs/.
13 https://csvkit.readthedocs.io/en/1.0.3/scripts/csvjson.html.
14 https://github.com/blue-yonder/di-csv2xml.
15 https://github.com/SDM-TIB/SDM-RDFizer.
16 http://rml.io/implementation-report/.

Table 4
Mapping features of GTFS. Each TriplesMap of the GTFS mapping file and its corresponding features: the related source, number of Classes, PredicateObjectMaps, Predicates, Objects and RefObjectMaps (joins).

TriplesMap	Source	Classes	#PredicateObjectMap	#Predicates	#Objects	#RefObjectMap
shapes	shapes	gtfs:Shape	4	4	4	0
trips	trips	gtfs:Trip	8	8	5	4
calendar_rules	calendar	gtfs:CalendarRule	9	9	9	0
calendar_date_rules	calendar_dates	gtfs:CalendarDateRule	2	2	2	0
stops	stops	gtfs:Stop	12	12	11	1
stoptimes	stop_times	gtfs:StopTime	9	9	7	2
routes	routes	gtfs:Route	8	8	7	1
agency	agency	gtfs:Agency	6	6	6	0
frequencies	frequencies	gtfs:Frequency	5	5	4	1
feed	feed_info	gtfs:Feed	6	6	6	0
service1	calendar	gtfs:Service	1	1	0	1
service2	calendar_dates	gtfs:Service	1	1	0	1
Total	10	11	71	71	60	11

- One xR2RML mapping document for accessing MongoDB datasets.
- Seven RML mapping documents[17] for accessing CSV, JSON, XML, SQL, MongoDB.
- One CSVW metadata file to provide annotations for the CSV datasets.
- An RML-Generator for obtaining the corresponding mappings of datasets with sources in several formats.

Conceptually, all the mappings represent the same relations among the concepts of the ontology and the concepts of the GTFS model, but each one has been developed according to a specification that handles the characteristics of each data format. The mappings are composed by a set of rules representing the relation of one element in the ontology with the corresponding schema element from a source. An overview of the rules within the mappings developed for this benchmark is shown in Table 4. These mapping rules are very relevant since they contain many parameters that impact on the performance of virtual knowledge graph access tools [10].

More in detail, each source of the GTFS feed has one associated TriplesMap, with a rule to associate the generated entities to the class defined in the ontology, and a set of rules for the object and data properties. Additionally, there is a (virtual) entity, Service, in the data model, with no corresponding source, which implies the definition of a set of mapping rules to generate the instances of the corresponding class (`gtfs:Service`). Following the GTFS specification, the identifier of Service can be found either in calendar or calendar_dates sources. This means that to be aligned with standard declarative mapping specifications (e.g. RML and R2RML only allow one source per TriplesMap), the mapping document needs to define two TriplesMap, one for calendar (service1) and another for calendar_dates (service2). This also implies that the trips TriplesMap has one predicate (`gtfs:service`) with two associated refObjectMaps, where the parentTriplesMap are service1 and service2; this allows generating all `gtfs:Service` defined in the original data source. Because the instances of `gtfs:WheelchairBoardingStatus` class are only objects in `gtfs:Trips` and `gtfs:Stops` triples, they are generated using the template property in trips and stops TriplesMap. In summary, the mapping contains rules to generate instances of 12 classes, 71 PredicateObjectMaps and Predicates, 60 Objects and 11 RefObjectMaps, covering the main features defined in state-of-the-art mapping specifications for OBDA/OBDI. All of the GTFS mappings are detailed in Appendix C using YARRRML [20].

3.4. Queries

Table 5 presents all the variables considered for the 18 queries in our benchmark. We have developed queries that are based on the Linked GTFS ontology, and are aligned with user stories in Madrid's transport domain, together with different combinations of values for the variables. It should be mentioned that the queries cover all of the data sources that were generated by the Madrid's transport authority as GTFS data from the metro system. These include agencies, routes, stops, trips, frequencies, shapes, calendar, and calendar dates. Although in the benchmark we have defined mappings to translate queries into the underlying query language of the source, these are independent from the queries. We have used these mappings to generate the materialized knowledge graph in the data generation step.

We have defined two sets of 18 queries with identical templates but with differences in the constants that appear in subjects or objects of bounded triple patterns: (1) Baseline queries with constants that belong to Madrid's GTFS Linked Data, and (2) VIG queries with constants that belong to the datasets generated by the tool; these queries are executed in the evaluation described in Section 4.

3.5. Metrics

In this section we define the metrics that are used to evaluate the performance of Virtual Knowledge Graph access engines. The metrics consider the workflow followed by Virtual Knowledge Graph systems, and for each of the steps identified in the workflow we introduce a set of metrics to be measured and reported.

The workflow extends the OBDA phases identified by Mora and Corcho [11], and Lanti et al. [4]. In addition, it includes some of the steps that are defined by proposals that federate queries [21]. General metrics to be captured are **overall execution time**, **completeness of answers** and **initial delay**. Other metrics may be considered when the engine generates answers following a continuous behavior [22], such as **dief@k** or **dief@t** proposed in [23]. Additionally, for each phase of a workflow, a virtual knowledge access engine may capture specific metrics that allow the identification of bottlenecks in the implementations. This relevant set of metrics for each phase are: (i) **loading time**

[17] Provided in RML and YARRRML serializations.

Table 5
GTFS-Madrid-Bench Queries. Rows correspond to the query identifier, its text in natural language and properties. Columns correspond to the query variables that influence most of the benchmark metrics. Each cell contains the value of the variable in the query. Additionally, the column *Mapping features* describes the rules from the general mapping involved in the query where TM means TriplesMap, PSOM means Predicate Simple Object Map (reference, template or constant) and PROM means Predicate Reference Map (join conditions).

Query	Description	#Triple Patterns	#Sources	OPTIONAL	Aggregation	Otherfeatures	FILTER		#Star-shaped groups		Mapping features
							equal to	relational	w/oconstants	w/constants	
q1	All shapes	4	1						1	0	1TM, 4PSOM
q2	All stops where the latitude is larger than a specific value	5	1	✓				✓	0	1	1TM, 5PSOM
q3	Accessibility information of all stations	5	1	✓			✓		0	1	1TM, 6PSOM
q4	All agencies and their routes	9	2	✓					2	0	2TM, 8PSOM, 1PROM
q5	Services that have been added after a specific date	5	2					✓	1	1	3TM, 5PSOM, 2PROM
q6	Number of routes covered by a specific agency	3	2		✓		✓		0	2	2TM, 1PSOM, 1PROM
q7	All wheelchair-accessible stops in a specific route	15	4	✓		DISTINCT	✓		1	3	6TM, 11PSOM, 5PROM
q8	Routes and their related trips, services, stops and stop times	14	5	✓					5	0	8TM, 10PSOM, 7PROM
q9	Trips and associated shapes where latitude is larger than a specific value	7	2	✓				✓	1	1	5TM, 4PSOM, 4PROM
q10	Number of trips that have a duration over a number of minutes	4	2		✓	DISTINCT		✓	1	1	2TM, 3PSOM, 1PROM
q11	Trips that are available on a certain date	12	3			NOT EXISTS		✓	3	2	5TM, 5PSOM, 4PROM
q12	Number of stops that are wheelchair-accessible grouped by route	10	4		✓	GROUP BY			3	1	5TM, 7PSOM, 3PROM
q13	The accesses of all stations	6	1	✓					0	1	1TM, 3PSOM, 1PROM
q14	All stops times and their related routes and stops ordered by their sequence, in a specific direction and service	8	3	✓		ORDER BY			3	0	2TM, 5PSOM, 3PROM
q15	For all properties, triples that contain a specific word in the object placeholder	3	1				✓		0	1	1TM, 15PSOM, 1PROM
q16	For all routes, all calendar changes in a specific month	8	3					✓	2	1	6TM, 6PSOM, 5PROM
q17	Trips with their start and end time of the frequencies and associated routes	9	3						3	0	3TM, 7PSOM, 2PROM
q18	All routes that have trips on Sunday	8	5			UNION			4	1	6TM, 6PSOM, 5PROM

Table 6
Relation between each relevant metric for the Madrid-GTFS-Bench and the dimensions that can impact over that metric. In the Dimension column, Q means query, M mappings and D data.

Metric	Type or phase	Dimension
General Metrics		
Total execution time	General	D, Q, M
# answers	General	D, Q, M
Initial delay	General	D, Q, M
Dief@k	C. Behavior	D, Q, M
Dief@t	C. Behavior	D, Q, M
Specific Metrics (Phases)		
Loading time	Starting	Q, M
Mapping trans. time	Starting	M
# requests	Distribution	Q
Source selec. time	Distribution	Q, M
Query gen. time	Distribution	Q
Query rewrit. time	Rewriting	Q
Query trans. time	Translation	Q, M
Query exec. time	Execution	Q, D
Query aggreg. time	Finishing	D

during the starting phase when the ontology, mappings and query are loaded; (ii) **total number of requests** and **source selection time** during the source selection phase (the engine identifies the sources that can be used to answer the query); (iii) **query generation time**, when the set of sub-queries to be evaluated over each data source is created, and the query plan is generated; (iv) **mapping translation time**, when the engine must translate a provided mapping into another one in a different language, maintaining a set of properties between them [2]; (v) **query rewriting time**, when the generated sub-queries are rewritten to other queries, taking into account potential inferences from the ontology and information in the mapping [24]; (vi) **query translation time**, when the engine, taking the mapping into account, translates each sub-query to another one in the query language, supported by the underlying data sources such as SPARQL-to-SQL [25]; (vii) **query execution time**, when the translated queries are evaluated against the underlying data sources and the results are translated to RDF or as SPARQL bindings using the rules provided in the mappings; and (viii) **query aggregation time**, when the results obtained for each sub-query are aggregated, including the removal of duplicates and the linking of resources. Variables that have an impact on the metrics have been grouped into three dimensions: Query, Data, and Mappings. The relation between each metric considered and the dimensions that can impact over that metric is shown in Table 6.

Query. The Query dimension variables refer to the structure of the queries, e.g. #triple patterns, #sources, and #star-shaped groups. A Star-shaped group is a group of triple patterns that are "joined" over the same subject or object variable [26]. The most common case in real-world scenarios are subject star-shaped groups that represent properties of one source. The benchmark considers an increasing number of triple patterns, from 3 to 15, also, the number of sources vary from 1 to 5. In particular we have several queries on 1 source with a varying number of triple patterns, and queries that have a large number of triple patterns combined with 4 and 5 sources. With respect to these two variables, our aim is to balance real-life use cases, where several properties in the specification need to be combined and retrieved, and query complexity. Furthermore, a large number of sources or triple patterns combined with a large number of non-instantiated star-shaped groups should impact overall execution time and also specifically impact query generation, query rewriting, query translation, and query execution times.

In general, queries in GTFS-Bench-Madrid combine those that contain single star-shaped groups ($q1$, $q2$, $q3$, $q15$) with those that contain chains of star-shaped groups, that is, where the object of a pattern in a group is the subject in the next group (with joins across different sources): $q4$, $q5$, $q6$, $q7$, $q9$, $q10$, $q11$, $q12$, $q16$, $q17$, $q18$. According to the ontology structure shown in Fig. 1, `gtfs:StopTime` relates to stops and trips and may lead to hybrid shapes such as $q8$ and $q14$. There is also the case of query $q13$, which refers to one source, and contains a self-join that relates an access to a station to its "parent" station.

Besides, as mentioned in [7], query plans generated by query evaluation systems during the subquery generation phase may be affected by the structural properties of a query. If the sources in the dataset are all represented in the same format (OBDA), then query plans will be generated by the underlying engine (either an RDB engine or a NoSQL engine), and execution time will be affected by the number of joins within star-shaped groups and among these groups. When the sources of the dataset are not in the same format (OBDI), the engine has to create the query plan. The performance will be affected by the plan proposed by the OBDI engine. Different combinations of these variables are considered in GTFS-Madrid-Bench queries: on the one hand we have a large number of triple patterns, sources and star-shaped groups in $q7$ and $q8$, and on the other hand queries like $q18$ combine a large number of sources and star-shaped groups with a medium-sized query (8 triple patterns).

Complexity of SPARQL queries is presented in [27], considering the SPARQL fragment with only AND and FILTER operators. Complexity is linear on the product of the dataset size and the size of the query (# triple patterns), and evaluation is NP-complete for queries constructed with AND, FILTER and UNION operators. Several queries in GTFS-Madrid-Bench have FILTER clauses and, specifically, $q18$ contains a UNION of two triple patterns.

The evaluation problem becomes harder when the OPTIONAL operator is added [27]. The work described in [28] presents optimization techniques applied in an OBDA setting specifically for queries that have to deal with OPTIONAL triple patterns, claiming that the underlying database systems do not optimize adequately these class of queries. Similar problems may be expected for querying CSV, XML and JSON data sources. We have designed eight queries that use OPTIONAL graph patterns (according to the corresponding non-mandatory attributes in the specification).

Constants in triple patterns together with FILTER with equality operators increase the selectivity of queries and are likely to reduce the cost of evaluating the query. According to [7], instantiated triple patterns have an important impact on the potential number of join intermediate results that may be generated throughout query execution. However, using a FILTER relational operator specially in the case of open ranges, e.g. a FILTER with a $>$ operator, may generate a large number of answers. We have considered several combinations of number of star-shaped groups with and without constants; $q8$ has no constants, whereas in $q4$ both star-shaped groups in the query have bindings. An example of an intermediate case occurs in $q12$ with 1 out of 4 instantiated star-shaped groups.

Three queries contain the aggregated COUNT function, and one of these queries contains additionally the GROUP BY modifier. Other queries use language features like DISTINCT and ORDER, that will impact on the query execution time metric because all of them require an ordering of the tuples/entries of the underlying sources. We cover the impact of these variables in $q7$ and $q10$ with DISTINCT, and $q12$ and $q14$ with GROUP BY and ORDER BY respectively. Finally, having unbounded predicates in a query ($q15$) increases its complexity, because the search space during query evaluation may be large.

The work in [29] studies the impact of negation in the computational complexity of SPARQL queries, it distinguishes four types of negation: negation of filter constraints, negation as failure,

negation by MINUS, and negation by NOT EXISTS. The use of NOT EXISTS introduces similar issues to sub-query evaluation because of the presence of correlated variables and the use of a nested iteration method to evaluate queries that contain this type of negation. Hence, $q11$ contains negation with NOT EXISTS.

Mappings. Features of mappings are relevant because they may impact the performance of the engines. Previous work by [10] evaluates different mapping variables that impact in the construction of a knowledge graph. Similarly, we consider that the following mapping variables influence overall query execution time and, specifically, query translation and query rewriting times. Regarding structure, we have considered the variables #Classes, #PredicateObjectMaps, #Predicates, #Objects, and #RefObjectMap that are presented in Table 4. Another variable is relation type; the mappings of the Madrid-GTFS-Bench include 1–1, 1–N, N–1 and N–M relation types. In general, mappings for sources that represent N–M relationships (e.g. stop_times) are more complex and thus time consuming for query execution. Additionally, the variable `rr:termtype` of the `rr:objectMap` may also have an effect because the cost of generating a constant, a reference or a template is not the same.

Dataset. Variables in this dimension include dataset size and the formats of its sources. As already mentioned in Section 3.2, datasets with different scale factors are generated in GTFS-Madrid-Bench. Size has an impact on the overall execution time, on the initial delay and, specifically, on query execution time because of the larger number of intermediate results. It also influences query aggregation time because in the benchmark, queries against larger datasets generate a larger number of answers.

The format variable may take a single value for datasets in only one format (RDB, CSV, XML, MongoDB, JSON) or multiple formats (configurable by the user) This variable has an impact on the overall execution time, specifically on the query translation and query execution times, as well as on the number of answers because different formats have different access methods and different underlying query languages.

The work in [7] presents partitioning and data distribution in this dimension. In GTFS-Bench-Madrid there are fixed values for these variables: the partitioning is vertical and datasets and databases are loaded in local machines.

4. Evaluation

In this section we describe the evaluation performed using our benchmark. We first describe the selected OBDA and OBDI engines involved in the evaluation, we describe the evaluation methodology and infrastructure, based on the use of docker images to ensure the reproducibility of the experiments and, finally, we provide the obtained results. All the resources used in this evaluation, such as queries, data, mappings, running scripts, results and docker images for engines and databases are publicly available online.[18]

4.1. Tools

We selected the most relevant open source OBDA and OBDI engines in the state of the art:

Ontario. Ontario [8][19] is an OBDI engine that is based on the concept of RDF molecule templates (RDF-MT) [30]. Ontario exploits the information provided by the mapping rules for creating the corresponding RDF-MT over the data sources. After the source selection and sub-query generation processes, Ontario translates the SPARQL query into the corresponding query language of the original data source. It supports the following formats: RDF, MySQL, CSV, TSV, JSON, XML, MongoDB and Neo4j.

Ontop. Ontop [31][20] is an OBDA system that includes both materialization and virtualization techniques. Ontop translates R2RML mappings into its own mapping language, called "OBDA mappings". These mappings, and a SPARQL query if available, are transformed into datalog rules, allowing semantic optimization techniques to be applied, and generating efficient SQL queries (e.g., self-join elimination). It only supports the SQL format.

Morph-RDB. Morph-RDB [32][21] is an R2RML engine that also includes materialization and virtualization techniques. The formalization of its query translation technique is based on the R2RML-based extension of SPARQL-to-SQL query translation algorithm proposed by Chebotko et al. [25], originally designed to work with RDB-backed triples store. Similar to Ontop, several optimization techniques are also incorporated in order to generate more efficient SQL queries. It supports SQL and CSV files.

Morph-xR2RML. Morph-xR2RML [33][22] uses the xR2RML mappings to support the generation of RDF lists, and to query data stored in NoSQL databases such as MongoDB.

Morph-CSV. Morph-CSV [16][23] exploits the information of CSVW annotations and RML mappings to enforce implicit constraints over tabular data. It can be integrated on top of any existing SPARQL-to-SQL engine in order to enhance query completeness and performance.

We also intended to include other OBDI engines such as Squerall [9] or Polyweb [34]. In both cases, either the code is not available as open source or it was not feasible to run the engine due to lack of documentation or mapping or SPARQL operators features coverage (e.g., Squerall does not support POM with join conditions or SPARQL queries with OPTIONAL). Issues have been reported in their corresponding repositories, with the intention of alerting the authors and maintainers about the current limitations.

Table 7
Experiment configuration example set. List of experimental configurations and processors for q4. D is a dataset where s is the scaling factor (i.e., 1, 5, 10, 50, 100, 500), M is the set of mappings, q is the SPARQL query, ϕ is a processor. q is a SPARQL query defined in the Appendix B.

Query q	Dataset D	TriplesMap M	Processor ϕ
q4	$GTFS^{csv}_{mad}$-s	$\{routes,agency\}_{RML}$	Morph-CSV
		$\{routes,agency\}_{R2RML}$	Morph-RDB
		$\{routes,agency\}_{RML}$	Ontario
	$GTFS^{sql}_{mad}$-s	$\{routes,agency\}_{R2RML}$	Morph-RDB
		$\{routes,agency\}_{RML}$	Ontario
		$\{routes,agency\}_{OBDA}$	Ontop
	$GTFS^{mongodb}_{mad}$-s	$\{routes,agency\}_{xR2RML}$	Morph-xR2RML
	$GTFS^{xml}_{mad}$-s	$\{routes,agency\}_{RML}$	Ontario
	$GTFS^{json}_{mad}$-s	$\{routes,agency\}_{RML}$	Ontario
	$GTFS^{minextj}_{mad}$-s	$\{routes,agency\}_{RML}$	Ontario
	$GTFS^{maxextj}_{mad}$-s	$\{routes,agency\}_{RML}$	Ontario

4.2. Setup

In this section we describe how we use our benchmark to evaluate several processors/engines that have been described in Section 4.1.

We have several experiment configurations for evaluating the selected processors. As an example, the experiment configurations for query $q4$ can be seen in Table 7. These experiment

18 https://github.com/oeg-upm/gtfs-bench.
19 https://github.com/SDM-TIB/Ontario.
20 https://github.com/ontop/ontop.
21 https://github.com/oeg-upm/morph-rdb.
22 https://github.com/frmichel/morph-xr2rml.
23 https://github.com/oeg-upm/morph-csv.

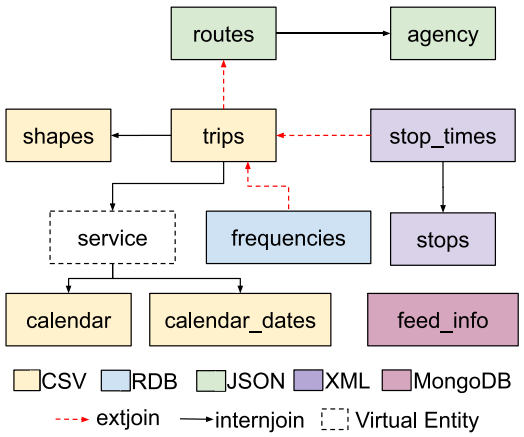

Fig. 3. Example of MINEXTJ dataset. $GTFS_{mad}^{minextj}$ dataset distributes the formats over the data sources ensuring that at least there is one source per each format and the joins among different formats are minimized. *extjoin* means that there is a relation between sources in different formats and *internjoin* means that the joins are between sources in the same format.

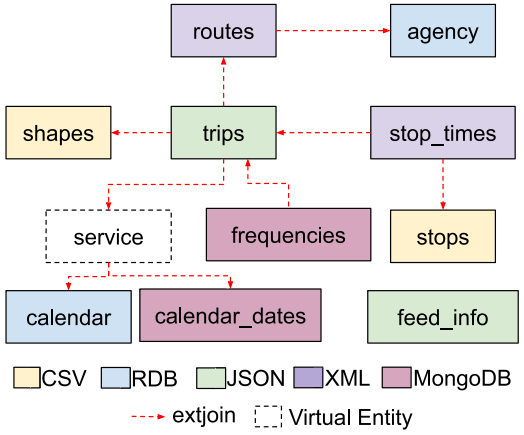

Fig. 4. Example of MAXEXTJ dataset. $GTFS_{mad}^{maxextj}$ dataset distributes the formats over the data sources ensuring that at least there is one source per each format and the joins among different formats are maximized. *extjoin* means that there is a relation between sources in different formats.

configurations have a fixed set of mappings with routes and agencies. The processor used to evaluate this query depends on the dataset, for example, Ontario in the case of the JSON dataset or Morph-RDB and Ontop for SQL.

All the experiment configurations are loaded into a machine with the following characteristics: 2 GHz CPU with 15 cores, 32 RAM, 200 GB HDD with Ubuntu 18.04 as its operating system. The machine contains a docker image for each of the processors: Morph-RDB v3.12.5, Ontop v3.0.0, Morph-CSV v0.1, Ontario v0.3, Morph-xR2RML-1.1-RC2. All the engines are configured with the recommended settings provided in the corresponding online repository.

In terms of data size, we decided to evaluate the engines over the scale values (5, 10, 50, 100 and 500). After some preliminary tests, we observed that these values provide a good overview of the current state of the engines in terms of query evaluation performance. For each SQL dataset size, we create two docker images where the data is loaded, one as an instance of the MySQL Database Server v5.5 and another as an instance of the MySQL Community Server v8.0. Similarly, for each MongoDB dataset size, we create a docker image of an instance of the MongoDB Community Server v3.4 where the dataset is loaded. The rest of the datasets, which correspond to raw data (CSV, XML and JSON), are loaded into the machine and are accessible to all the processors.

To test OBDI engines and to demonstrate the capabilities of the benchmark resources covering multiple scenarios, we chose to analyze the impact of the number of joins among different formats. The main reason to test this parameters is because Ontario is focused on improving the performance of these kind of queries. The dataset were created taking into account the selected formats (JSON, CSV, XML, SQL and MongoDB) and varying the number of relations (joins) among different formats. More specifically, the dataset distributions are the following:

- MINEXTJ dataset: The number of joins among sources in different formats is minimized but ensuring that all of the formats are covered. The aim of this configuration is to study the behavior of the engines when they have to deal with different data sources but where most of the joins are done between sources in the same format. Hence they may delegate their treatment to the underlying data source manager (e.g. MySQL in RDB) and apply common optimization techniques in query translation approaches [32]. To meet this requirement and, having 5 possible formats for the data sources, the proposed groups for this dataset are: trips, shapes, calendar and calendar_dates sources in one group, routes and agency in another, frequencies in the third group, stop and stop_times in the fourth one and feed_info in the last one. This composition generates the $GTFS_{mad}^{MIN-EXTJ}$ dataset. We show the used dataset in the evaluation in Fig. 3.
- MAXEXTJ Dataset: The number of joins among sources in different formats is maximized and the five formats are covered. In this distribution, all the possible joins are among sources in different formats. This means that the OBDI engine may be enforced to perform the joins after the execution of the translated queries over the original data sources. In the same manner as the minimized dataset, the groups of sources are: shapes and stops in one group, trips and feed_info in another, calendar and agency in the third group, routes and stop_times in the fourth and calendar_dates and frequencies in the last one. This composition generates the $GTFS_{mad}^{MAX-EXTJ}$ dataset. We show the used dataset in the evaluation in Fig. 4.

In the case of Morph-RDB, we use it together with the docker images containing the instances of the MySQL Community Server v5.5, according to the corresponding documentation. As for Morph-xR2RML, we use it together with the docker images containing the instances of MongoDB server version v3.4. For these experimental configurations and processors, we evaluate all the 18 queries both in warm and in cold mode. Each query is run five times. In warm mode we want to analyze how the cache mechanism may affect the performance. In order to do so, we first evaluate the query, discard its result and then run the query again five times; we then compute the average query execution time. On the contrary, in cold mode, we want to study the performance of the processors without the effect of the cache. In order to do so, we run the query five times and we always restart the database server after each run, so as to clean all the caches.

Additionally, we use Ontario and Ontop with the docker images containing the instances of MySQL server v8.0, the latest version at the time of writing. Note that the use of the cache is not supported anymore in MySQL v8.0, so that we only evaluate our queries in cold mode. We perform the rest of the experiment configurations with Ontario against the CSV and JSON datasets, and Morph-CSV against CSV datasets.

4.3. Results

In this section we report the results obtained through our experimental configurations. Table 8 presents the results obtained

Table 8
Overall execution time (in seconds) of benchmark queries in experiment configurations with original size datasets. W means that the engine obtained a different number of results in comparison to the baseline. E means that the processor is not able to execute the query. TO means that the processor is not able to evaluate the query within the timeout duration (3600 s).

Dataset	Processor		Query																	
	Cache	Name	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-SQL-1	Warm	Morph-RDB	5.85	2.07	E	1.82	W	1.86	1.97	E	26.02	1.80	E	1.81	2.06	W	1.89	E	2.11	E
		Ontario	18.02	E	TO	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
	Cold	Morph-RDB	7.14	2.65	E	2.42	W	2.36	2.43	E	28.65	2.38	E	2.41	2.69	W	2.58	E	2.68	E
		Ontop	8.37	5.04	5.18	E	W	E	W	E	16.56	E	E	E	5.06	W	5.10	W	5.00	E
GTFS-MongoDB-1	Warm	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	28.67	W	W	6.52	W
	Cold	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	28.17	W	W	6.96	W
GTFS-CSV-1	Cold	Morph-RDB	6.94	3.04	E	2.78	E	2.78	TO	E	TO	2.97	E	6.23	3.97	E	E	E	3.14	E
		Morph-CSV	15.11	10.88	E	10.72	E	9.95	10.84	E	40.90	10.70	E	11.60	11.82	E	E	E	11.48	W
		Ontario	W	E	17.34	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-XML-1	Cold	Ontario	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
GTFS-JSON-1	Cold	Ontario	18.04	E	17.14	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MINEXTJ-1	Cold	Ontario	W	E	E	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MAXEXTJ-1	Cold	Ontario	W	E	17.34	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E

for all the datasets and all the processors with scale 1 and a timeout of 3600 s (1 h). The rest of the Tables 9–13 report the results for the other scale values (5, 10, 50, 100 and 500), with the same timeout. When an engine reports an error (e.g. a SPARQL query parsing error, memory overhead, etc.), we represent it with an E in the table. When the engine does not report any error, but the number of results obtained differs with respect to the baseline (RDF materialized graph), we represent the cell with a W. We do not report the total execution time of those queries because, in general, these cases report 0 results in the execution but without error, so the time is not relevant. The tables comparing the number of results obtained by the baseline and the evaluated engines is reported in Appendix A.

In terms of the comparison among different data formats, we can observe that CSV and SQL data formats are the ones best supported by the available engines. In these cases, most of the engines are able to answer a significant number of queries. As to the effect of cache, as expected, evaluation in the warm mode needed less time, yet, the difference is insignificant due to the relatively small size of the datasets.

We can also see that in general, it takes more time to evaluate queries over CSV datasets than over SQL datasets. This is expected because available engines need to first load the CSV dataset in a SQL database server in order to be able to query the dataset.

This is not the case of other data formats such as JSON and MongoDB, where the engines are only able to answer one or two queries. This is even worse in the case of the XML format, where the only engine that supports it is not able to answer any query. Similarly, in the distributed format, the only query that can be answered by the OBDI engine is a query that is evaluated against a JSON dataset.

This trend holds in the other scale factors up to 100. In the scale factor 500, only those engines that use SQL datasets are able to answer queries.

Analyzing the results in general, the errors obtained in the execution of the queries (E in the tables) over the tested engines may be due to two main reasons: (i) the engine does not support a SPARQL operator in the original query (ii) the engine is not able to manage large (intermediate) results, for example, maintaining them in memory. Additionally, the differences obtained in terms of query completeness (W in the tables) may be because: (i) the engine supports the SPARQL operator, but it does not translate it correctly to an operator of the underlying database, hence, the query is executed but the number of results obtained are different; (ii) the interpretation of the mapping rules is not performing correctly, hence, the semantics of the original query is not preserved in the translated query.

5. Discussion

In this section, we provide a general analysis of the design, implementation and execution of Madrid-GTFS-Bench. It should be pointed out that our aim is to have a proposal that follows the benchmark requirements and gives a general overview of the results and problems we observed during the development of the GTFS-Madrid-Bench. We do not intend to rank the performance of the evaluated engines, but rather to identify current limitations in the state of the art in terms of the capabilities of the engines, so as to provide useful information to the developers of each engine as well as to general practitioners. We also describe the process of creating a benchmark for virtual knowledge graph access, and depict the problems and limitations of the tools employed for creating its resources. This analysis can be used to solve open issues, propose improvements and identify future work in the field.

In terms of the capabilities of OBDA/OBDI engines, the main issue we observe is that many of them do not support some of the commonly used SPARQL operators, such as UNION, ORDER BY and NOT EXISTS. The engines that cover a wider range of SPARQL operators are the ones that execute a SPARQL-to-SQL query translation, due to the fact that this technique has been widely studied in the state of the art [32,35]. The engines that perform query translation over raw data (e.g. CSV, JSON) or over a NoSQL database (MongoDB in this case), produce a lot of errors in the query translation and evaluation processes. For example, in the case of Ontario, the engine is more focused on the generation of an efficient query plan (i.e., distributing star-shaped groups (SSG) taking into account the molecule templates), than performing a correct translation and execution of each SSG over the raw data. The engine does not give support to most of the SPARQL operators, and that is the main reason why it is not able to answer most of the queries. The same happens in terms of query evaluation time; some SPARQL-to-SQL approaches include several optimization techniques [28] so that they can evaluate the translated queries efficiently, while the other translation techniques that target non SQL query languages are not as efficient. These observations point out the need for a deeper analysis of the techniques that perform efficient query translation from SPARQL to non SQL query languages and raw data (CSV, JSON, XML). Our main conclusions regarding the obtained results are:

- Only the SPARQL-to-SQL engines provide an acceptable support for SPARQL operators, although there are still some operators that are not included (e.g., FILTER NOT EXIST in Morph-RDB).

Table 9
Overall execution time (in seconds) of benchmark queries in experiment configurations with size 5 datasets. W means that the engine obtained a different number of results in comparison to the baseline. E means that the processor is not able to execute the query. TO means that the processor is not able to evaluate the query within the timeout duration (3600 s).

Dataset	Processor		Query																	
	Cache	Name	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-SQL-5	Warm	Morph-RDB	12.65	2.47	E	1.89	2.06	1.78	1.93	E	E	1.74	E	1.88	2.14	4.58	2.88	E	2.61	E
	Cold	Ontario	117.00	E	TO	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
		Morph-RDB	15.14	3.24	E	2.40	2.71	2.34	2.62	E	E	2.41	E	2.70	2.82	5.59	3.89	E	3.39	E
		Ontop	13.87	5.40	5.31	E	W	E	W	E	W	E	E	E	5.24	6.61	W	W	5.37	E
GTFS-MongoDB-5	Warm	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	TO	W	W	TO	W
	Cold	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	TO	W	W	TO	W
GTFS-CSV-5	Cold	Morph-RDB	14.42	4.38	E	3.81	E	3.64	TO	E	TO	6.57	E	TO	12.45	E	E	E	9.25	E
		Morph-CSV	43.41	W	E	33.51	E	34.44	W	E	TO	33.86	E	36.08	34.90	E	E	E	35.26	E
		Ontario	W	E	18.34	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-XML-5	Cold	Ontario	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
GTFS-JSON-5	Cold	Ontario	W	E	15.66	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MINEXTJ-5	Cold	Ontario	W	E	E	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MAXEXTJ-5	Cold	Ontario	W	E	18.34	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E

Table 10
Overall execution time (in seconds) of benchmark queries in experiment configurations with size 10 datasets. W means that the engine obtained a different number of results in comparison to the baseline. E means that the processor is not able to execute the query. TO means that the processor is not able to evaluate the query within the timeout duration (3600 s).

Dataset	Processor		Query																	
	Cache	Name	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-SQL-10	Warm	Morph-RDB	23.78	2.88	E	1.93	W	1.75	1.97	E	E	1.85	E	1.94	2.46	6.61	3.46	E	3.07	E
	Cold	Ontario	415.60	E	TO	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
		Morph-RDB	27.25	3.72	E	2.54	W	2.36	2.55	E	E	2.38	E	2.50	3.22	8.16	4.48	E	3.77	E
		Ontop	24.05	5.56	5.57	E	W	E	W	E	W	E	E	E	5.29	7.58	W	W	5.62	E
GTFS-MongoDB-10	Warm	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	TO	W	W	TO	W
	Cold	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	TO	W	W	TO	W
GTFS-CSV-10	Cold	Morph-RDB	25.90	6.06	E	5.20	E	4.89	TO	E	TO	16.06	E	TO	38.15	E	E	E	38.90	E
		Morph-CSV	97.00	W	E	69.39	E	68.78	W	E	TO	69.28	E	71.01	68.79	E	E	E	72.29	W
		Ontario	W	E	19.51	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-XML-10	Cold	Ontario	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
GTFS-JSON-10	Cold	Ontario	W	E	17.21	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MINEXTJ-10	Cold	Ontario	W	E	E	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MAXEXTJ-10	Cold	Ontario	W	E	19.51	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E

Table 11
Overall execution time (in seconds) of benchmark queries in experiment configurations with size 50 datasets. W means that the engine obtained a different number of results in comparison to the baseline. E means that the processor is not able to execute the query. TO means that the processor is not able to evaluate the query within the timeout duration (3600 s).

Dataset	Processor		Query																	
	Cache	Name	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-SQL-50	Warm	Morph-RDB	108.42	4.91	E	2.08	W	1.75	1.97	E	E	1.89	E	2.29	3.69	22.55	8.27	E	5.56	E
	Cold	Ontario	TO	E	TO	E	E	E	E	W	E	E	E	E	E	W	E	E	W	E
		Morph-RDB	121.31	6.01	E	2.68	W	2.31	2.63	E	E	2.59	E	2.91	4.54	27.02	10.00	E	6.89	E
		Ontop	119.89	6.92	6.61	E	W	E	W	E	W	E	E	E	6.05	15.69	W	W	7.31	E
GTFS-MongoDB-50	Warm	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	TO	W	W	TO	W
	Cold	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	TO	W	W	TO	W
GTFS-CSV-50	Cold	Morph-RDB	128.40	22.17	E	19.85	E	19.60	TO	E	TO	351.23	E	TO	1,039.29	E	E	E	TO	E
		Morph-CSV	575.15	449.54	E	442.60	E	436.06	W	E	TO	444.84	E	443.12	447.74	E	E	E	443.47	W
		Ontario	W	E	35.16	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-XML-50	Cold	Ontario	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
GTFS-JSON-50	Cold	Ontario	W	E	23.74	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MINEXTJ-50	Cold	Ontario	W	E	E	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MAXEXTJ-50	Cold	Ontario	W	E	35.16	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E

- OBDA/OBDI proposals beyond relational databases are not mature enough, and more research is needed in order to, for example, provide wider support of SPARQL operators or generate efficient query plans that take into account parameters such as data format or join selectivity.
- The problem of translating SPARQL queries for querying raw data (CSV, JSON, XML) should not be understood as a technical case of SPARQL-to-SQL where the management of the data is delegated to RDB wrappers such as Presto, Spark and Apache Drill. Techniques and optimizations, and the analysis of features uniquely associated to these data sources have to be proposed.
- The distribution of SPARQL queries over heterogeneous sources exploiting mapping rules, and their translation and execution over different query languages, are the two main points for developing robust OBDI engines. Although the adaptation of current techniques proposed by federated SPARQL engines to OBDI has been successfully proved in [8, 9], they do not support the majority of the SPARQL operators and they do not correctly execute the queries when the data

source is beyond RDB instances. New investigation should be performed to address these issues.

Additionally, in our evaluation we only have the possibility of obtaining the total execution time of each engine. Other metrics are proposed in the benchmark, such as initial delay, loading time or query translation time. However, they are only available in some of the engines. We point out the importance of providing all these metrics to identify possible bottlenecks in the evaluation process.

We have also found possible improvements in terms of data and mapping generation in the process of creating the resources in this benchmark. In the data generation process, one of the main improvements may be the incorporation of semantics. For example, in our benchmark we have a file that represents the calendar of the trips, which has a start and an end date. The data generator should validate that the start date must be earlier than the end date, so that queries can be created to exploit this constraint. Another example that would improve with the inclusion of semantics is the scaling of dataset sources that are related and may be "joined". Currently, even if each dataset is scaled, the number of tuples per join-attribute value does not change. Ideally, this should be scaled only in certain cases. Additionally, the inclusion of a set of constraints or validation rules may improve this process (e.g. define a range of possible values for a column).

With respect to the mapping generation process, we find two main issues. First, as mappings need to relate the ontology with the data source, the raw data need some changes in a preprocessing step in order to be aligned with the features of the ontology (e.g classes or properties). There are some proposals to include these transformation functions in mappings, such as the Function Ontology [36] or R2RML-F [37], but at the moment of writing only Squerall and Morph-CSV are able to parse RML mappings with functions(RML+FnO) Finally, we have to create manually the mapping documents required to test the engines. Following the proposal in [2], an improvement will be to be able to define the mappings conceptually, independently of the language, and then, have techniques to translate them to a specific language. With this approach we will ensure the correctness of the mappings.

5.1. Sustainability and extensibility

The Madrid-GTFS-Bench is supported by a set of robust resources in order to ensure its sustainability. The benchmark can be adapted to any other virtual knowledge graph access engine that uses other mapping rules languages, or to other non-declarative proposals. The developers or users only have to create the mapping documents according to that specification. Additionally, virtual knowledge graph access engines that work with other graph query languages (e.g., Morph-GraphQL [38]) can take advantage of our benchmark.

A set of improvements for the data generation that we have identified are based on VIG, a robust and efficient engine for the generation of scalable datasets. Additionally, all the generated resources are available online,[24] and their deployment (engines and databases) is done using docker images to ensure the reproducibility of the obtained results. Finally, because we define the dimensions of mappings and datasets taking into account the relevant parameters in the process of constructing knowledge graphs [10], this benchmark can be also used to test the engines called *rdfizers*, such as RMLMapper,[25] RocketRML[26] or SDM-RDFizer,[27] since at this moment there is no proposal to evaluate the performance and completeness of these engines.

The possibility to extend this benchmark is also one of the main points that differentiates this proposal to previous ones. First, multiple benefits are obtained from relying on an open data model from the transport domain, such as linking this data with other data from the city, and also having other GTFS transport systems feeds (e.g., metro and train datasets). In addition to queries that take into account the specific characteristics of the selected datasets, it is also possible to incorporate more complex mapping rules with extended features such as specific transformation functions [36], something difficult to address by previous proposals, as their data models are usually relational database oriented [3,4]. The incorporation of these features will ensure that we cover new characteristics of the new generation of virtual knowledge graph access engines without having to create a benchmark from scratch.

6. Related work

In this section we provide an overview of two groups of benchmark proposals: benchmarks for federated SPARQL engines, and benchmarks for SPARQL OBDA engines. Additionally we present a general description of existing OBDA and OBDI approaches, and the different proposals of mapping languages that are aimed at establishing transformation rules between an ontology and different data representations.

6.1. Federated SPARQL benchmarks

In the context of federated SPARQL engines, many benchmarks have been proposed to evaluate engines that distribute a SPARQL query over several RDF-based endpoints, applying techniques of query distribution when all of the sources are RDF datasets. Benchmarks that have been proposed are the Fedbench suite [5] and LSLOD [6].

Fedbench [5] provides a framework to evaluate the efficiency and effectiveness of federated SPARQL query strategies over three different datasets: cross-domain, life science, and SP^{2}Bench which is set in the DBLP context. Evaluation is carried out, and comparisons are done on various setups: centralized, local SPARQL endpoints, and federations of endpoints. Another evaluation setup is the linked data scenario, where sources are retrieved through URI lookup. For the cross domain and life-science queries, the metrics are total execution time and number of requests to endpoints. The metric for the evaluation of SP^{2}Bench is total execution time, and the Linked Data setting considers execution time and number of dereferenced sources. The focus of Fedbench is on the evaluation of different scenarios of RDF data federation. The purpose of GTFS-Madrid-Bench is to evaluate scenarios where there may be an assortment of data in different formats, not only RDF, that is, a centralized heterogeneous setup (currently denoted as a Data Lake) where the highly dynamic nature of these data require virtualized access. While Fedbench considers both real-world and artificial data federations, GTFS-Madrid-Bench is based on the use of the Madrid subway Linked GTFS feed as a starting point for the generation of scaled instances in the different formats. Both Fedbench and GTFS-Madrid-Bench follow a query design approach in combining SPARQL language coverage and real-world user requirements, where one of the aspects considered is the number of sources that need to be accessed in order to answer the queries.

The work in [7] remarks that there are two groups of variables (*dependent* and *independent*) in federated benchmark definitions

[24] https://github.com/oeg-upm/gtfs-bench/.
[25] https://github.com/RMLio/rmlmapper-java.
[26] https://github.com/semantifyit/RocketRML/.
[27] https://github.com/SDM-TIB/SDM-RDFizer.

Table 12
Overall execution time (in seconds) of benchmark queries in experiment configurations with size 100 datasets. W means that the engine obtained a different number of results in comparison to the baseline. E means that the processor is not able to execute the query. TO means that the processor is not able to evaluate the query within the timeout duration (3600 s).

Dataset	Processor		Query																	
	Cache	Name	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-SQL-100	Warm	Morph-RDB	221.11	7.48	E	2.30	W	1.75	1.96	E	E	1.99	E	2.65	4.68	42.44	15.51	E	8.54	E
	Cold	Ontario	TO	E	TO	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
		Morph-RDB	245.98	8.83	E	3.05	W	2.33	2.52	E	E	2.63	E	3.38	5.76	50.99	19.45	E	10.38	E
		Ontop	1,477.38	8.87	8.25	E	W	E	W	E	W	E	E	E	6.80	27.18	W	W	9.20	E
GTFS-MongoDB-100	Warm	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	TO	W	W	TO	W
	Cold	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	TO	W	W	TO	W
GTFS-CSV-100	Cold	Morph-RDB	E	43.59	E	38.52	E	38.43	TO	E	TO	1582.52	E	TO	TO	E	E	E	TO	E
		Morph-CSV	1,254.19	W	E	958.43	E	933.69	W	E	TO	957.95	E	951.53	952.93	E	E	E	947.82	W
		Ontario	W	E	85.59	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-XML-100	Cold	Ontario	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
GTFS-JSON-100	Cold	Ontario	W	E	33.56	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MINEXTJ-100	Cold	Ontario	W	E	E	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MAXEXTJ-100	Cold	Ontario	W	E	85.59	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E

Table 13
Overall execution time (in seconds) of benchmark queries in experiment configurations with size 500 datasets. W means that the engine obtained a different number of results in comparison to the baseline. E means that the processor is not able to execute the query. TO means that the processor is not able to evaluate the query within the timeout duration (3600 s).

Dataset	Processor		Query																	
	Cache	Name	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-SQL-500	Warm	Morph-RDB	TO	29.85	E	3.39	W	1.81	1.96	E	E	3.19	E	6.34	13.60	220.35	93.72	E	33.64	E
	Cold	Ontario	TO	E	TO	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
		Morph-RDB	TO	32.71	E	3.92	W	2.09	2.30	E	E	3.62	E	6.95	14.69	218.00	99.00	E	35.77	E
		Ontop	W	20.93	17.17	E	W	E	W	E	W	E	E	E	10.82	114.59	W	W	23.95	E
GTFS-MongoDB-500	Warm	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	TO	W	W	TO	W	W	TO	W
	Cold	Morph-xR2RML	W	W	W	W	W	W	W	W	W	W	W	W	W	TO	W	W	TO	W
GTFS-CSV-500	Cold	Morph-RDB	E	TO	E	TO	E	TO	TO	E	TO	TO	E	TO	TO	E	E	E	TO	E
		Morph-CSV	TO	TO	E	TO	E	TO	TO	E	TO	TO	E	TO	TO	E	E	E	TO	TO
		Ontario	W	E	E	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-XML-500	Cold	Ontario	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
GTFS-JSON-500	Cold	Ontario	W	E	E	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MINEXTJ-500	Cold	Ontario	W	E	E	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E
GTFS-MAXEXTJ-500	Cold	Ontario	W	E	E	E	E	E	E	W	E	E	E	E	E	W	E	E	E	E

that have not been considered, but which may have an impact on the measurement of engines performance. The independent variables are those that may be specified to ensure the reproducibility of the proposed scenarios; they are grouped into four dimensions: query, data, platform and endpoint. The dependent variables are those that will be measured during the evaluation: endpoint selection time, execution time (divided into (i) time for first answer, (ii) time for the distributed reception of query answers, and (iii) total execution time), and answer completeness. The impact of the dependent variables on the evaluation metrics is analyzed, and then different configurations of variables are applied to the evaluation of FedBench queries. The authors observe that the performance of the federated query engines is indeed affected by the group of independent variables. Similarly, GTFS-Madrid-Bench has also considered in its design and experimental setup some of these query, data and platform independent variables, taking into consideration their impact on the execution time. Some of these are query plan shape, # basic triple patterns, # of instantiations, usage of query language expressivity, dataset size and cache on/off. However, some of the variables are not relevant to GTFS-Madrid-Bench setup, such as those related to the Endpoint dimension and some of the Data dimension variables, such as data endpoint distribution.

LSLOD [6] provides queries and real-world data from the life-science domain. It is very specific to the context of SPARQL endpoint federation. Several query characteristics are considered in LSLOD such as number of basic graph patterns, number of triple patterns, number of join vertices, and use of different SPARQL clauses. GTFS-Madrid-Bench does not aimed at evaluating federated SPARQL queries, therefore using LSLOD was not an option. Unlike GTFS-Madrid-Bench, LSLOD does not include some SPARQL operators that may impact on the behavior of OBDA/I engines.

6.2. OBDA Benchmarks

Several benchmarks have been developed to measure the performance of SPARQL to SQL query translation of OBDA engine techniques. The main two proposals in this field are the Berlin SPARQL Benchmark (BSBM) [3] and the Norwegian Petroleum Directorate Benchmark (NPD) [4]. The BSBM benchmark sets its context in the e-commerce domain, and provides a configurable data generator and a set of SPARQL queries together with their equivalent SQL queries. This benchmark has been used to compare the query performance of native RDF stores with the performance of OBDA engines that execute virtualized SPARQL access against relational databases.

Similar to BSBM, GTFS-Madrid-Bench uses a data generator to scale up the size of the dataset used during experimentation. Unlike BSBM, GTFS-Madrid-Bench is based on real data, using Madrid subway network data to be more specific, and this data is aligned with an established data model. Furthermore, BSBM does not measure specific requirements of OBDA systems, as it has been developed with the aim of comparing OBDA engines with native RDF triple stores. Also, it considers only OBDA engines that access relational data sources. GTFS-Bench-Madrid considers data sources in multiple formats, thus it is tailored to evaluate and compare an assortment of OBDA engines and OBDI engines as well. Specific OBDA requirements have been analyzed by the

authors of the NPD benchmark in the setting of a real-world scenario from the oil industry. The nine proposed requirements are related to the datasets, query sets, mappings and query languages. The benchmark includes a data generator, VIG [13], to generate scaled RDB instances that obtain a number of expected triples from a SPARQL query, using as inputs an ontology, an R2RML mapping document, and the schema of the RDB together with its corresponding instance. In our work we extend the workflow defined in NPD from OBDA to OBDI and hence define the requirements of an OBDI benchmark. Additionally, we extend VIG in order to transform the RDB scaled instances to the different data source representations handled by the engines.

Simple queries defined in LSLOD [6] have been used in order to evaluate the performance of the Ontario [8] engine. In this work, all of the original RDF datasets were translated into RDB tables. The idea was to evaluate an heterogeneous setup consisting of RDF and RDB sources, focused on the source selection problem, and the generation of the corresponding optimized query plan. GTFS-Madrid-Bench extends the ideas from this evaluation with the aim of involving a wider assortment of formats that are usually available nowadays in the Web. Also, our proposal differs from LSLOD in the sense that it is supported by an extensible open data model in a smart city context. Additionally, the simple queries from LSLOD are focused on the evaluation of the distribution of star-shaped groups that do not exploit some of the features of the SPARQL language, such as FILTER, ORDER BY, GROUP BY and NOT EXISTS, which are relevant in the context of real-life use cases.

6.3. Virtual OBDA and OBDI approaches

There are multiple proposals focused on the translation of SPARQL queries to query data in their original format. The concept of OBDA and OBDI is defined in [1], and the first proposal for translating SPARQL-to-SQL is defined in [25]. Based on this idea and with the launch of R2RML [39] as a W3C recommendation, multiple works are proposed in the optimization of this process that take into account this mapping specification, such as Morph-RDB [32], Ontop [35] or Ultrawrap [40]. Additionally, there are specific studies on how SPARQL operators affect the translation of the query to SQL [28]. Beyond relational databases, Morph-xR2RML [33] formally defines the translation from SPARQL to NoSQL databases. Finally, Morph-CSV [16] is a proposal to enhance the SPARQL-to-SQL process when the data source is a set of tabular data (i.e. CSV files). It exploits information from tabular metadata and mapping rules to explicitly enforce implicit constraints of the original datasets.

There are two main proposals of OBDI engines: Ontario and Squerall. Ontario [8] is based on the concept of RDF molecule templates [30] which aims to perform efficient source selection in a data lake composed of heterogeneous data sources in their original format. It creates a set of star-shaped sub-queries that match the RDF Molecule Templates (RDF-MT), and applies optimization techniques to define the query plan that will be executed. Similarly, Squerall [9] is a system that implements OBDI for heterogeneous data sources. It takes input data and mappings, and offers a middleware that is able to aggregate the intermediate results in a distributed manner. Although the aforementioned systems are able to evaluate queries against raw tabular data and exploit some information encoded in the query, they do not exploit the constraints declared in annotations or mapping rules to enhance this process. Polyweb [34] is another proposal that is able to translate and distribute queries using RML mappings over relational databases and CSV files.

6.4. Mapping languages

Different mapping languages have been proposed for defining transformation rules between ontology representation languages and data sources in different formats; these include SQL and NoSQL databases, as well as data in plain text such as CSV, XML and JSON. The RDB2RDF W3C Working Group published two recommendations for transforming the content of relational databases into RDF: Direct Mapping [41] and R2RML [39]. The Direct Mapping approach specifies a set of transformation rules that requires no intervention from users. R2RML allows specifying transformation rules, such as how URIs should be generated, in which columns of the database are used for the transformation to RDF triples that represent tuples in the original tables, and so on. After the recommendation was released, new needs and requirements arose in relation to supporting other formats beyond relational databases, and this resulted in the creation of new mapping languages such as RML [18] which considers data sources in CSV, JSON and XML formats, xR2RML [33] for MongoDB databases, KR2RML [42] that considers nested data, CSVW [17] to annotate CSV files on the Web, and D2RML [43] for XML, JSON and REST/SPARQL endpoints, among others. These are declarative proposals that define mapping rules, which have some important features, such as the improving of the maintainability, readability or understandability of the data integration process. Non-declarative mapping languages have also been proposed, for example SPARQL-Generate [44] extends the SPARQL 1.1 by taking as input an RDF dataset and a set of documents in multiple formats, and generating RDF through the SPARQL CONSTRUCT clause.

Current OBDA benchmarks use mappings defined in a single language according to the underlying format of the engines that are evaluated. The Ontario [8] OBDI engine has conducted their evaluation with an extension of the LSLOD benchmark that defines RML mappings. The Squerall [9] engine has picked the BSBM benchmark and has defined mappings in an extended version of the RML language that uses functions defined with entities from the Function Ontology (FnO).[28] The Polyweb tool [34] uses mappings defined RML and R2RML mappings; however the system is not publicly available and thus cannot be reused or extended. GTFS-Madrid-Bench uses mappings defined mappings in all of the state of the art mapping languages: RML, R2RML, xR2RML, and CSVW annotations for tabular data.

7. Conclusions and future work

In this paper we propose a benchmark for virtual knowledge graph access using real data from the transport domain. The benchmark design considers variables that span all of its resources (queries, mappings and data) in order to test the capabilities and performance of the processors. GTFS-Madrid-Bench satisfies requirements that are an extension of those already identified in existing OBDA benchmarks. Besides, metrics have been established for each step of the workflow of virtualised knowledge graph access.

As already discussed, the main objective of this benchmark is not to provide a ranking of engines, but to provide a set of resources that can be useful for: (i) practitioners who choose the engine that best fits their use cases and (ii) developers of virtual knowledge graph access engines to improve their tools and compare their results with other proposals. As such, we expect this benchmark to be a stepping stone in this area where much research and development has been done for decades, but there is a need for more mature applications to be used in real-world environments. Indeed, our experimental study has shown that there are still relevant open issues, such as SPARQL conformance, semantic preservation in the translation from SPARQL queries to the query languages used to query raw data (CSV, JSON, XML),

[28] https://fno.io/.

and the application of query evaluation optimization techniques. With the Madrid-GTFS-Bench we intend to contribute to the community by providing not only the baseline that can be used to improve the development of the current engines, but also the possibility to use it to test new approaches and techniques over the next years.

The design of this benchmark has been a complex task, since it had to cover all of the identified requirements and, at the same time, work on a very general scenario with a mix of OBDA and OBDI approaches. On the one hand, some of the current OBDA proposals work with SQL datasets and in general conform to most of the features of the SPARQL language. Throughout the experiments we realized that the OBDA proposals that are designed to work with other formats support fewer features of the query language, and in general they have issues in their query translation process. There is a lot of room for improvement in these proposals, such as generating more efficient queries, which has been done in the SQL-based OBDA proposals. On the other hand, we were only able to include in the benchmark the Ontario OBDI proposal, even though other OBDI proposals have been published in the literature since it was not feasible to execute them because of lack of documentation. In all cases, the evaluation of our benchmark queries with the different engines exposed the need for improvements in their current releases, in terms of efficiency and correctness of the results.

It is also worth mentioning that the benchmark is easily extensible to be used with other data formats and engines. That is, if in the future there is a requirement to evaluate an engine that supports a different data format, the only need is to create a script that translates the data sources from CSV to the corresponding format.

Future work includes the development of mapping translation techniques that involve the different mapping language specifications. In the context of virtualized knowledge graph access it would be very useful to help in the development and maintenance of mappings, so as to avoid inconsistencies among mappings and errors in the evaluation. Another line of work is to improve the data generation process to ensure that scaled data is well aligned with the domain data model.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work presented in this paper is supported by the project Semantics for PerfoRmant and scalable INteroperability of multimodal Transport (SPRINT H2020-826172) and by the Spanish Ministerio de Economía, Industria y Competitividad and EU FEDER funds under the DATOS 4.0: RETOS Y SOLUCIONES - UPM Spanish national project (TIN2016-78011-C4-4-R) and by an FPI grant (BES-2017-082511).

Appendix A. Completeness of query evaluation

See Tables 14–19.

Table 14
Completeness of benchmark queries in experiment configurations with GTFS-1 dataset. Minus means that the processor is not able to execute the query (i.e. generates an error) or it does not evaluate the query within the timeout duration.

Dataset	Source	Tool	Queries																
			q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-1	SQL	Ontario	58540	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
		Morph-RDB	58540	765	–	13	84	1	2	–	151439	1	–	6	734	2234	26	–	855	–
		Ontop	58540	765	734	–	0	–	0	–	151439	–	–	–	734	2234	26	0	855	–
	MongoDB	Morph-xR2RML	0	0	0	0	0	0	0	0	0	0	0	0	0	2364	0	0	855	0
	CSV	Morph-RDB	58540	765	–	13	–	1	–	–	–	1	–	6	734	–	–	–	855	–
		Morph-CSV	58540	765	–	13	–	1	2	–	151439	1	–	6	734	–	–	–	855	128
		Ontario	0	–	734	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	XML	Ontario	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	JSON	Ontario	58540	–	734	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MINEXTJ	Ontario	0	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MAXEXTJ	Ontario	0	–	734	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	RDF	Virtuoso	58540	765	734	13	28	1	2	4728	151439	1	130	6	734	2364	26	34	855	64

Table 15
Completeness of benchmark queries in experiment configurations with GTFS-5 dataset. Minus means that the processor is not able to execute the query (i.e. generates an error) or it does not evaluate the query within the timeout duration.

Dataset	Source	Tool	Queries																
			q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-5	SQL	Ontario	176830	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
		Morph-RDB	176830	3161	–	65	350	1	62	–	–	1	–	65	1325	11170	4949	–	4275	–
		Ontop	176830	3161	2104	–	0	–	0	–	0	–	–	–	1325	11170	4828	0	4275	–
	MongoDB	Morph-xR2RML	0	0	0	0	0	0	0	0	0	0	643	0	0	6593	0	0	3357	0
	CSV	Morph-RDB	176830	3161	–	65	–	1	–	–	–	1	–	–	1325	–	–	–	4275	–
		Morph-CSV	176830	6310	–	65	–	1	0	–	–	1	–	65	1325	–	–	–	4275	0
		Ontario	0	–	2104	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	XML	Ontario	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	JSON	Ontario	0	–	2104	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MINEXTJ	Ontario	0	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MAXEXTJ	Ontario	0	–	2104	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	RDF	Virtuoso	176830	3161	2104	65	350	1	62	23640	359113	1	650	65	1325	11170	4949	2080	4275	624

Table 16
Completeness of benchmark queries in experiment configurations with GTFS-10 dataset. Minus means that the processor is not able to execute the query (i.e. generates an error) or it does not evaluate the query within the timeout duration.

Dataset	Source	Tool	Queries																
			q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-10	SQL	Ontario	353660	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
		Morph-RDB	353660	6312	–	130	700	1	67	–	–	1	–	130	2650	22340	8641	–	8550	–
		Ontop	353660	6312	4207	–	0	–	0	–	0	–	–	–	2650	22340	8521	0	8550	–
	MongoDB	Morph-xR2RML	0	0	0	0	0	0	0	0	0	0	1292	0	0	11348	0	0	5508	0
	CSV	Morph-RDB	353660	6312	–	130	–	1	–	–	–	1	–	–	2650	–	–	–	8550	–
		Morph-CSV	353660	12620	–	130	–	1	0	–	0	1	–	130	2650	–	–	–	8550	0
		Ontario	0	–	4207	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	XML	Ontario	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	JSON	Ontario	0	–	4207	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MINEXTJ	Ontario	0	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MAXEXTJ	Ontario	0	–	4207	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	RDF	Virtuoso	353660	6312	4207	130	350	1	67	47280	718317	1	1300	130	2650	22340	8641	1820	8550	1300

Table 17
Completeness of benchmark queries in experiment configurations with GTFS-50 dataset. Minus means that the processor is not able to execute the query (i.e. generates an error) or it does not evaluate the query within the timeout duration.

Dataset	Source	Tool	Queries																
			q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-50	SQL	Ontario	–	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
		Morph-RDB	1768300	31550	–	650	3500	1	59	–	–	1	–	650	13250	111700	21958	–	42750	–
		Ontop	1768300	31550	21034	–	0	–	0	–	0	–	–	–	13250	111700	17537	0	42750	–
	MogoDB	Morph-xR2RML	0	0	0	0	0	0	0	0	0	0	5058	0	0	–	0	0	–	0
	CSV	Morph-RDB	1768300	31550	–	650	–	1	–	–	–	1	–	–	1325	–	–	–	–	–
		Morph-CSV	1768300	63100	–	650	–	1	0	–	0	1	–	650	13250	–	–	–	42750	0
		Ontario	0	–	21034	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	XML	Ontario	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	JSON	Ontario	0	–	21034	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MINEXTJ	Ontario	0	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MAXEXTJ	Ontario	0	–	21034	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	RDF	Virtuoso	1768300	31550	21034	650	1750	1	59	236400	3591503	1	6500	650	13250	111700	21958	2730	42750	6500

Table 18
Completeness of benchmark queries in experiment configurations with GTFS-100 dataset. Minus means that the processor is not able to execute the query (i.e. generates an error) or it does not evaluate the query within the timeout duration.

Dataset	Source	Tools	Queries																
			q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-100	SQL	Ontario	–	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
		Morph-RDB	3536600	63100	–	1300	7000	1	67	–	–	1	–	1300	26500	223400	35502	–	85500	–
		Ontop	3536600	63100	42067	–	0	–	0	–	0	–	–	–	26500	223400	31080	0	85500	–
	MongoDB	Morph-xR2RML	0	0	0	0	0	0	0	0	0	0	10336	0	0	–	0	0	–	0
	CSV	Morph-RDB	–	63100	–	1300	–	1	–	–	–	1	–	–	–	–	–	–	–	–
		Morph-CSV	3536600	126200	–	1300	–	1	0	–	–	1	–	1300	26500	–	–	–	85500	0
		Ontario	0	–	42067	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	XML	Ontario	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	JSON	Ontario	0	–	42067	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MINEXTJ	Ontario	0	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MAXEXTJ	Ontario	0	–	42067	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	RDF	Virtuoso	3536600	63100	42067	1300	3500	1	67	472800	7183874	1	13000	1300	26500	223400	35502	1690	85500	13000

Table 19
Completeness of benchmark queries in experiment configurations with GTFS-500 dataset. Minus means that the processor is not able to execute the query (i.e. generates an error) or it does not evaluate the query within the timeout duration.

Dataset	Source	Tool	Queries																
			q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18
GTFS-500	SQL	Ontario	–	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
		Morph-RDB	–	315499	–	6500	35000	1	53	–	–	1	–	6500	132500	1117000	38749	–	427500	–
		Ontop	0	315499	210334	–	0	–	0	–	0	–	–	–	132500	1117000	34323	0	427500	–
	MongoDB	Morph-xR2RML	0	0	0	0	0	0	0	0	0	0	–	0	0	–	0	0	–	0
	CSV	Morph-RDB	–	–	–	–	–	–	–	–	–	1	–	–	–	–	–	–	–	–
		Morph-CSV	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
		Ontario	0	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	XML	Ontario	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	JSON	Ontario	–	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MINEXTJ	Ontario	0	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	MAXEXTJ	Ontario	0	–	–	–	–	–	–	0	–	–	–	–	–	0	–	–	–	–
	RDF	Virtuoso	17683000	315499	210334	6500	17500	1	53	2364000	35919991	1	65000	6500	132500	1117000	38749	2340	427500	65000

Appendix B. GTFS-Madrid-Bench queries

Listing 1: Prefixes

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX gtfs: <http://vocab.gtfs.org/terms#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX schema: <http://schema.org/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

Listing 2: Query 1 - List all shapes with some of their data.

```
SELECT * WHERE {
        ?shape a gtfs:Shape .
        ?shape geo:lat ?shape_pt_lat .
        ?shape geo:long ?shape_pt_lon .
        ?shape gtfs:pointSequence ?shape_pt_sequence .
}
```

Listing 3: Query 2 - List all stops with some of their data including geographic coordinates, where the latitude is bigger than its mean

```
SELECT * WHERE {
        ?stop a gtfs:Stop .
        OPTIONAL { ?stop dct:description ?stopDescription . }
        OPTIONAL { ?stop gtfs:wheelchairAccessible ?wheelchairAccesible }
        ?stop geo:lat ?stopLat .
        ?stop geo:long ?stopLong .
        FILTER (?stopLat > %LAT%) .
}
```

Listing 4: Query 3 - Find the accessibility information for the stations, if available

```
SELECT * WHERE {
        ?stop a gtfs:Stop .
        ?stop gtfs:locationType ?location .
        OPTIONAL { ?stop dct:description ?stopDescription . }
        OPTIONAL { ?stop geo:lat ?stopLat ; geo:long ?stopLong . }
        OPTIONAL { ?stop gtfs:wheelchairAccessible ?wheelchairAccessible .  }
        FILTER (?location=<http://transport.linkeddata.es/resource/LocationType/2>)
}
```

Listing 5: Query 4 - List all agencies and their routes with some of their data

```
SELECT * WHERE {
        ?route a gtfs:Route .
        OPTIONAL { ?route gtfs:shortName ?routeShortName . }
        OPTIONAL { ?route gtfs:longName ?routeLongName . }
        OPTIONAL { ?route dct:description ?routeDescription . }
        ?route gtfs:agency ?agency .
        ?agency a gtfs:Agency .
        ?agency foaf:page ?agencyPage .
        ?agency foaf:name ?agencyName .
        OPTIONAL { ?agency foaf:phone ?agencyPhone . }
}
```

Listing 6: Query 5 - Services that have been added on a specific day

```
SELECT * WHERE {
        ?service a gtfs:Service .
        ?service gtfs:serviceRule ?serviceRule .
        ?serviceRule a gtfs:CalendarDateRule .
        ?serviceRule dct:date ?date .
        ?serviceRule gtfs:dateAddition "true"^^xsd:boolean .
        FILTER(?date > %DATE%)
}
```

Listing 7: Query 6 - Check the number of routes of a particular agency

```
SELECT (count(?route) as ?nRoutes) WHERE {
        ?route a gtfs:Route .
        ?route gtfs:agency ?agency .
        FILTER (?agency=%AGENCY%)
}
```

Listing 8: Query 7 - List all wheelchair accessible stops along a particular route, with some of their additional data

```
SELECT DISTINCT ?routeShortName ?routeDescription ?tripShortName
?stopDescription ?stopLat ?stopLong WHERE {
        ?route a gtfs:Route .
        OPTIONAL { ?route gtfs:shortName ?routeShortName . }
        OPTIONAL { ?route dct:description ?routeDescription . }
        ?trip a gtfs:Trip .
        OPTIONAL { ?trip gtfs:shortName ?tripShortName . }
        ?trip gtfs:service ?service .
        ?trip gtfs:route ?route .
        ?stopTime a gtfs:StopTime .
        ?stopTime gtfs:trip ?trip .
        ?stopTime gtfs:stop ?stop .
        ?stop a gtfs:Stop .
        OPTIONAL { ?stop dct:description ?stopDescription . }
        OPTIONAL { ?stop geo:lat ?stopLat ; geo:long ?stopLong . }
        ?stop gtfs:wheelchairAccessible gtfsaccessible:1 .
        FILTER (?route=%ROUTE%)
}
```

Listing 9: Query 8 - List the routes and their related trips, services, stops and stop times with some of their additional data, if available.

```
SELECT * WHERE {
        ?route a gtfs:Route .
        OPTIONAL { ?route gtfs:shortName ?routeShortName . }
        OPTIONAL { ?route dct:description ?routeDescription . }
        ?trip a gtfs:Trip .
        OPTIONAL { ?trip gtfs:shortName ?tripShortName . }
        ?trip gtfs:service ?service .
        ?trip gtfs:route ?route .
        ?stopTime a gtfs:StopTime .
        ?stopTime gtfs:trip ?trip .
        ?stopTime gtfs:stop ?stop .
        ?stop a gtfs:Stop .
        OPTIONAL {?stop dct:description ?stopDescription . }
        ?service a gtfs:Service .
        ?service gtfs:serviceRule ?serviceRule .
}
```

Listing 10: Query 9 - Trips and associated shapes where lat is bigger than its average and some of their additional data

```
SELECT * WHERE {
        ?trip a gtfs:Trip .
        OPTIONAL { ?trip gtfs:shortName ?tripShortName . }
        ?trip gtfs:service ?service .
        ?trip gtfs:route ?route .
        ?trip gtfs:shape ?shape .
        ?shape a gtfs:Shape .
        ?shape geo:lat ?lat .
        FILTER (?lat > %LAT%)
}
```

Listing 11: Query 10 - Number of trips that have a duration over 30 minutes

```
SELECT (count(distinct ?trip) as ?count) WHERE {
        ?trip a gtfs:Trip .
        ?stopTime a gtfs:StopTime .
        ?stopTime gtfs:trip ?trip .
        ?stopTime gtfs:departureTime ?departureTime .
        FILTER (?departureTime >= "00:30:00"^^xsd:duration)
}
```

Listing 12: Query 11 - Trips that are available on a certain date and some of their additional data

```
SELECT * WHERE {
    ?service a gtfs:Service .
    ?service gtfs:serviceRule ?calendarRule .
    ?trip gtfs:service ?service .
    ?calendarRule a gtfs:CalendarRule .
    ?calendarRule schema:startDate ?startDate .
    ?calendarRule schema:endDate ?endDate .
    FILTER (?startDate <%DATE%) .
    FILTER (?endDate > %DATE%) .
    FILTER NOT EXISTS {
        ?service gtfs:serviceRule ?calendarDateRule .
        ?calendarDateRule a gtfs:CalendarDateRule .
        ?calendarDateRule dct:date %DATE% .
        ?calendarDateRule gtfs:dateAddition "false"^^xsd:boolean
    }
}
```

Listing 13: Query 12 - Number of stops that are wheelchair-accessible grouped by route and some of their additional data

```
SELECT ?longName (count(?name) as ?count) WHERE {
        ?route a gtfs:Route .
        ?route gtfs:longName ?longName .
        ?trip a gtfs:Trip .
        ?trip gtfs:route ?route .
        ?stopTime a gtfs:StopTime .
        ?stopTime gtfs:trip ?trip .
        ?stopTime gtfs:stop ?stop .
        ?stop a gtfs:Stop .
        ?stop foaf:name  ?name .
        ?stop gtfs:wheelchairAccessible gtfsaccessible:1 .
}
GROUP BY ?longName
```

Listing 14: Query 13 - All the accesses of the stations

```
SELECT  * WHERE {
    ?stop a gtfs:Stop .
    ?stop gtfs:parentStation ?parStation .
    OPTIONAL {?stop foaf:name ?accName} .
    ?stop gtfs:locationType gtfslocation:2 .
    ?parStation a gtfs:Stop .
    ?parStation foaf:name ?name
}
```

Listing 15: Query 14 - All stops times and their related routes and stops order by their sequence

```
SELECT * WHERE {
    ?stopTime a gtfs:StopTime .
    ?stopTime gtfs:trip ?trip .
    ?stopTime gtfs:stop ?stop .
    ?stopTime gtfs:stopSequence ?sequence .
    ?stop a gtfs:Stop .
    ?trip a gtfs:Trip .
    ?trip gtfs:route ?route .
    OPTIONAL {?stop foaf:name ?stopName}
} ORDER BY ?sequence
```

Listing 16: Query 15 - Everything that contains a specific string in the object placeholder (any property)

```
SELECT * WHERE {
        ?stop a gtfs:Stop .
        ?stop ?p ?str .
        FILTER regex (?str, %STRING%)
}
```

Listing 17: Query 16 - For all the routes, all the calendar changes during a specific month

```
SELECT * WHERE {
        ?trip a gtfs:Trip .
        ?trip gtfs:service ?service .
        ?trip gtfs:route ?route .
        ?service a gtfs:Service .
        ?service gtfs:serviceRule ?serviceRule .
        ?serviceRule a gtfs:CalendarDateRule .
        ?serviceRule  dct:date ?servDate .
        ?serviceRule  gtfs:dateAddition "true"^^xsd:boolean .
        FILTER (?servDate >= %DATE1%) .
        FILTER (?servDate <= '%DATE2%) .
}
```

Listing 18: Query 17 - Trips with their start and end time of the frequencies and associated routes

```
SELECT ?routeName ?routeType ?trip ?startTime ?endTime WHERE {
    ?trip a gtfs:Trip .
    ?trip gtfs:route ?route .
    ?frequency a gtfs:Frequency .
    ?frequency gtfs:startTime ?startTime .
    ?frequency gtfs:endTime ?endTime .
    ?frequency gtfs:trip ?trip .
    ?route a gtfs:Route .
    ?route gtfs:shortName ?routeName .
    ?route gtfs:routeType ?routeType .
}
```

Listing 19: Query 18 - All routes that have trips on Sunday

```
SELECT * WHERE {
        ?service a gtfs:Service .
        ?service gtfs:serviceRule ?serviceRule .
        ?serviceRule a gtfs:CalendarRule .
        ?serviceRule gtfs:sunday "true"^^xsd:boolean .
        ?trip gtfs:service ?service .
        ?trip gtfs:route ?route  .
        { ?route gtfs:longName ?longName }  UNION
        { ?route gtfs:shortName ?shortName }    .
}
```

Appendix C. GTFS-Madrid-Bench mappings

Listing 20: Prefixes

```
prefixes:
  rr: http://www.w3.org/ns/r2rml#
  foaf: http://xmlns.com/foaf/0.1/
  xsd: http://www.w3.org/2001/XMLSchema#
  rdfs: http://www.w3.org/2000/01/rdf-schema#
  dc: http://purl.org/dc/elements/1.1/
  rev: http://purl.org/stuff/rev#
  gtfs: http://vocab.gtfs.org/terms#
  geo: http://www.w3.org/2003/01/geo/wgs84_pos#
  schema: http://schema.org/
  dct: http://purl.org/dc/terms/
  rml: http://semweb.mmlab.be/ns/rml#
  ql: http://semweb.mmlab.be/ns/ql#
  rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
  mad: http://transport.linkeddata.es/madrid/metro/
  gtfsres: http://transport.linkeddata.es/resource/
```

Listing 21: Routes TriplesMap

```
routes:
 sources:
  - [ROUTES.format]
 s: mad:routes/$(route_id)
 po:
  - [a, gtfs:Route]
  - [gtfs:shortName, $(route_short_name)]
  - [gtfs:longName, $(route_long_name)]
  - [dct:description, $(route_desc)]
  - [gtfs:routeType, gtfsres:RouteType/$(route_type)~iri]
  - [gtfs:routeUrl, $(route_url)~iri]
  - [gtfs:color, $(route_color)]
  - [gtfs:textColor, $(route_text_color)]
  - p: gtfs:agency
    o:
      - mapping: agency
        condition:
          function: equal
          parameters:
            - [str1, $(agency_id)]
            - [str2, $(agency_id)]
```

Listing 22: Calendar_Date TriplesMap

```
calendar_rules:
 sources:  - [CALENDAR.format]
 s: mad:calendar_rules/$(service_id)
 po:
  - [a, gtfs:CalendarRule]
  - [gtfs:monday, $(monday), xsd:boolean]
  - [gtfs:tuesday, $(tuesday), xsd:boolean]
  - [gtfs:wednesday, $(wednesday), xsd:boolean]
  - [gtfs:thursday, $(thursday), xsd:boolean]
  - [gtfs:friday, $(friday), xsd:boolean]
  - [gtfs:saturday, $(saturday), xsd:boolean]
  - [gtfs:sunday, $(sunday), xsd:boolean]
  - [schema:startDate, $(start_date), xsd:date]
  - [schema:endDate, $(end_date), xsd:date]
```

Listing 23: Service_Calendar_Date TriplesMap

```
services2:
 sources:
  - [CALENDAR_DATES.format]
 s: mad:services/$(service_id)
 po:
  - [a, gtfs:Service]
  - p: gtfs:serviceRule
    o:
      - mapping: calendar_date_rules
        condition:
          function: equal
          parameters:
            - [str1, $(service_id)]
            - [str2, $(service_id)]
```

Listing 24: Agency TriplesMap

```
agency:
 sources:
    - [AGENCY.format]
 s: mad:agency/$(agency_id)
 po:
   - [a, gtfs:Agency]
   - [foaf:page, $(agency_url)~iri]
   - [foaf:name,$(agency_name)]
   - [gtfs:timeZone,$(agency_timezone)]
   - [dct:language,$(agency_lang)]
   - [foaf:phone,$(agency_phone)]
   - [gtfs:fareUrl,$(agency_fare_url)~iri]
```

Listing 25: Calendar_Date_Rules TriplesMap

```
calendar_date_rules:
 sources:
  - [CALENDAR_DATES.format]
 s: http://transport.linkeddata.es/madrid/metro/calendar_date_rule/$(service_id)-$(date)
 po:
  - [a, gtfs:CalendarDateRule]
  - [dct:date, $(date), xsd:date]
  - [gtfs:dateAddition, $(exception_type), xsd:boolean]
```

Listing 26: Stop_Times TriplesMap

```
stoptimes:
 sources:
  - [STOP_TIMES.format]
 s: mad:metro/stoptimes/$(trip_id)-$(stop_id)-$(arrival_time)
 po:
  - [a, gtfs:StopTime]
  - [gtfs:arrivalTime, $(arrival_time),xsd:duration]
  - [gtfs:departureTime, $(departure_time),xsd:duration]
  - [gtfs:stopSequence, $(stop_sequence),xsd:integer]
  - [gtfs:headsign, $(stop_headsign)]
  - [gtfs:pickupType, gtfsres:PickupType/$(pickup_type)~iri]
  - [gtfs:dropOffType, gtfsres:DropOffType/$(drop_off_type)~iri]
  - [gtfs:distanceTraveled, $(shape_dist_traveled)]
  - p: gtfs:trip
    o:
      - mapping: trips
        condition:
          function: equal
          parameters:
            - [str1, $(trip_id)]
            - [str2, $(trip_id)]
  - p: gtfs:stop
    o:
      - mapping: stops
        condition:
          function: equal
          parameters:
            - [str1, $(stop_id)]
            - [str2, $(stop_id)]
```

Listing 27: Frequencies TriplesMap

```
frequencies:
 sources:
  - [FREQUENCIES.format]
 s: mad:frequency/$(trip_id)-$(start_time)
 po:
  - [a, gtfs:Frequency]
  - [gtfs:startTime,$(start_time)]
  - [gtfs:endTime,$(end_time)]
  - [gtfs:headwaySeconds,$(headway_secs),xsd:integer]
  - [gtfs:exactTimes,$(exact_times),xsd:boolean]
  - p: gtfs:trip
    o:
      - mapping: trips
        condition:
          function: equal
          parameters:
            - [str1, $(trip_id)]
            - [str2, $(trip_id)]
```

Listing 28: Trips TriplesMap

```
trips:
 sources:
  - [TRIPS.format]
 s: mad:trips/$(trip_id)
 po:
  - [a, gtfs:Trip]
  - [gtfs:headsign, $(trip_headsign)]
  - [gtfs:shortName, $(trip_short_name)]
  - [gtfs:direction, $(direction_id)]
  - [gtfs:block, $(block_id)]
  - [gtfs:wheelchairAccessible,gtfsres:WheelchairBoardingStatus/$(wheelchair_accessible)~iri]
  - p: gtfs:service
    o:
      - mapping: services1
        condition:
          function: equal
          parameters:
            - [str1, $(service_id)]
            - [str2, $(service_id)]
      - mapping: services2
        condition:
          function: equal
          parameters:
            - [str1, $(service_id)]
            - [str2, $(service_id)]
  - p: gtfs:route
    o:
      - mapping: routes
        condition:
          function: equal
          parameters:
            - [str1, $(route_id)]
            - [str2, $(route_id)]
  - p: gtfs:shape
    o:
      - mapping: shapes
        condition:
          function: equal
          parameters:
            - [str1, $(shape_id)]
            - [str2, $(shape_id)]
```

Listing 29: Feed_Info TriplesMap

```
feed:
 sources:
  - [FEED_INFO.format]
 s: mad:feed/$(feed_publisher_name)
 po:
  - [a, gtfs:Feed]
  - [dct:publisher,$(feed_publisher_name)]
  - [foaf:page,$(feed_published_url)~iri]
  - [dct:language,$(feed_lang)]
  - [schema:startDate,$(feed_start_date), xsd:date]
  - [schema:endDate,$(feed_end_date), xsd:date]
  - [schema:version,$(feed_version)]
```

Listing 30: Stops TriplesMap

```
stops:
 sources:
  - [STOPS.format]
 s: mad:stops/$(stop_id)
 po:
  - [a,gtfs:Stop]
  - [gtfs:code,$(stop_code)]
  - [dct:identifier,$(stop_id)]
  - [foaf:name,$(stop_name)]
  - [dct:description,$(stop_desc)]
  - [geo:lat,$(stop_lat),xsd:double]
  - [geo:long,$(stop_lon),xsd:double]
  - [gtfs:zone,$(zone_id)]
  - [foaf:page,$(stop_url)~iri]
  - [gtfs:locationType, gtfsres:LocationType/$(location_type)~iri]
  - [gtfs:timeZone,$(stop_timezone)]
  - [gtfs:wheelchairAccessible,gtfsres:WheelchairBoardingStatus/$(wheelchair_boarding)~iri]
  - p: gtfs:parentStation
    o:
      - mapping: stops
        condition:
          function: equal
          parameters:
            - [str1, $(parent_station)]
            - [str2, $(stop_id)]
```

Listing 31: Shapes TriplesMap

```
shapes:
 sources:
  - [SHAPES.format]
 s: mad:shape/$(shape_id)-$(shape_pt_sequence)
 po:
  - [a, gtfs:Shape]
  - [geo:lat,$(shape_pt_lat),xsd:double]
  - [geo:long,$(shape_pt_lon),xsd:double]
  - [gtfs:pointSequence,$(shape_pt_sequence)]
  - [gtfs:distanceTraveled,$(shape_dist_traveled)]
```

Listing 32: Service_Calendar TriplesMap

```
services1:
 sources:
  - [CALENDAR.format]
 s: mad:services/$(service_id)
 po:
  - [a, gtfs:Service]
  - p: gtfs:serviceRule
    o:
      - mapping: calendar_rules
        condition:
          function: equal
          parameters:
            - [str1, $(service_id)]
            - [str2, $(service_id)]
```

References

[1] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, J. Data Semant. X (2008) 133–173.

[2] Ó. Corcho, F. Priyatna, D. Chaves-Fraga, Towards a new generation of ontology based data access, Semant. Web 11 (1) (2020) 153–160.

[3] C. Bizer, A. Schultz, The Berlin SPARQL benchmark, Int. J. Semantic Web Inf. Syst. 5 (2) (2009) 1–24.

[4] D. Lanti, M. Rezk, G. Xiao, D. Calvanese, The NPD benchmark: Reality check for OBDA systems, OpenProceedings.org, 2015.

[5] M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte, T. Tran, Fedbench: A benchmark suite for federated semantic data query processing, in: International Semantic Web Conference, Springer, 2011, pp. 585–600.

[6] A. Hasnain, Q. Mehmood, S.S. e Zainab, M. Saleem, C. Warren, D. Zehra, S. Decker, D. Rebholz-Schuhmann, Biofed: federated query processing over life sciences linked open data, J. Biomed. Semant. 8 (1) (2017) 13.

[7] G. Montoya, M.-E. Vidal, O. Corcho, E. Ruckhaus, C. Buil-Aranda, Benchmarking federated SPARQL query engines: Are existing testbeds enough? in: International Semantic Web Conference, Springer, 2012, pp. 313–324.

[8] K.M. Endris, P.D. Rohde, M.-E. Vidal, S. Auer, Ontario: Federated query processing against a semantic data lake, in: Database and Expert Systems Applications, in: Lecture Notes in Computer Science, Springer, Cham, 2019.

[9] M.N. Mami, D. Graux, S. Scerri, H. Jabeen, S. Auer, Querying data lakes using spark and presto, in: International World Wide Web Conference, ACM, 2019, pp. 3574–3578.

[10] D. Chaves-Fraga, K.M. Endris, E. Iglesias, Ó. Corcho, M. Vidal, What are the parameters that affect the construction of a knowledge graph? in: OTM Confederated International Conferences "on the Move To Meaningful Internet Systems", Springer, 2019.

[11] J. Mora, O. Corcho, Towards a systematic benchmarking of ontology-based query rewriting systems, in: International Semantic Web Conference, Springer, 2013, pp. 376–391.

[12] M. Acosta, M.-E. Vidal, T. Lampo, J. Castillo, E. Ruckhaus, ANAPSID: an adaptive query processing engine for SPARQL endpoints, in: International Semantic Web Conference, Springer, 2011, pp. 18–34.

[13] D. Lanti, G. Xiao, D. Calvanese, VIG: Data scaling for OBDA benchmarks, Semant. Web 10 (2018) 1–21.

[14] The W3C SPARQL Working Group, SPARQL 1.1 Overview, W3C, 2013, http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

[15] D.L. McGuinness, F. Van Harmelen, et al., OWL Web ontology language overview, W3C Recomm. 10 (10) (2004) 2004.

[16] D. Chaves-Fraga, E. Ruckhaus, F. Priyatna, M.-E. Vidal, O. Corcho, Enhancing OBDA query translation over tabular data with morph-CSV, 2020, arXiv:2001.09052.

[17] J. Tennison, G. Kellogg, I. Herman, Model for Tabular Data and Metadata on the Web, W3C, 2015, http://www.w3.org/TR/2015/REC-tabular-data-model-20151217/.

[18] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, RML: A generic language for integrated RDF mappings of heterogeneous data., in: LDOW, 2014.

[19] P. Heyvaert, D. Chaves-Fraga, F. Priyatna, O. Corcho, E. Mannens, R. Verborgh, A. Dimou, Conformance test cases for the RDF mapping language (RML), in: Iberoamerican Knowledge Graphs and Semantic Web Conference, Springer, 2019, pp. 162–173.

[20] P. Heyvaert, B. De Meester, A. Dimou, R. Verborgh, Declarative rules for linked data generation at your fingertips!, in: European Semantic Web Conference, Springer, 2018, pp. 213–217.

[21] A. Schwarte, P. Haase, K. Hose, R. Schenkel, M. Schmidt, Fedx: Optimization techniques for federated query processing on linked data, in: International Semantic Web Conference, Springer, 2011, pp. 601–616.

[22] M.A. Sharaf, P.K. Chrysanthis, A. Labrinidis, K. Pruhs, Algorithms and metrics for processing multiple heterogeneous continuous queries, ACM Trans. Database Syst. 33 (1) (2008) 5.

[23] M. Acosta, M.-E. Vidal, Y. Sure-Vetter, Diefficiency metrics: measuring the continuous efficiency of query processing approaches, in: International Semantic Web Conference, Springer, 2017, pp. 3–19.

[24] J. Mora, R. Rosati, O. Corcho, Kyrie2: Query rewriting under extensional constraints in $\{\mathcal{ELHIO}\}$, in: International Semantic Web Conference, Springer, 2014, pp. 568–583.

[25] A. Chebotko, S. Lu, F. Fotouhi, Semantics preserving SPARQL-to-SQL translation, Data Knowl. Eng. 68 (10) (2009) 973–1000.

[26] M. Vidal, E. Ruckhaus, T. Lampo, A. Martínez, J. Sierra, A. Polleres, Efficiently joining group patterns in SPARQL queries, in: Extended Semantic Web Conference, 2010, pp. 228–242.

[27] J. Pérez, M. Arenas, C. Gutiérrez, Semantics and complexity of SPARQL, ACM Trans. Database Syst. 34 (3) (2009) 16:1–16:45.

[28] G. Xiao, R. Kontchakov, B. Cogrel, D. Calvanese, E. Botoeva, Efficient handling of SPARQL OPTIONAL for OBDA (extended version), 2018, CoRR abs/1806.05918, URL: arXiv:1806.05918.

[29] R. Angles, C. Gutiérrez, Negation in SPARQL, in: Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management, 2016.

[30] K.M. Endris, M. Galkin, I. Lytra, M.N. Mami, M.-E. Vidal, S. Auer, MULDER: querying the linked data web by bridging RDF molecule templates, in: International Conference on Database and Expert Systems Applications, Springer, 2017, pp. 3–18.

[31] M. Rodriguez-Muro, M. Rezk, Efficient SPARQL-to-SQL with R2RML mappings, J. Web Semant. 33 (2015) 141–169.

[32] F. Priyatna, O. Corcho, J. Sequeda, Formalisation and experiences of -based SPARQL to SQL query translation using morph, in: International World Wide Web Conference, 2014.

[33] F. Michel, L. Djimenou, C.F. Zucker, J. Montagnat, Translation of relational and non-relational databases into RDF with xR2RML, in: 11th International Confenrence on Web Information Systems and Technologies, WEBIST'15, 2015, pp. 443–454.

[34] Y. Khan, A. Zimmermann, A. Jha, V. Gadepally, M. D'Aquin, R. Sahay, One size does not fit all: Querying web polystores, IEEE Access 7 (2019) 9598–9617.

[35] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Ontop: Answering SPARQL queries over relational databases, Semant. Web 8 (3) (2017) 471–487.

[36] B. De Meester, A. Dimou, R. Verborgh, E. Mannens, An ontology to semantically declare and describe functions, in: European Semantic Web Conference, Springer, 2016, pp. 46–49.

[37] C. Debruyne, D. O'Sullivan, R2RML-F: Towards sharing and executing domain logic in R2RML mappings, in: International World Wide Web Conference, in: Linked Data on the Web Workshop, 2016.

[38] F. Priyatna, D. Chaves-Fraga, A. Alobaid, O. Corcho, morph-GraphQL: GraphQL Servers Generation from R2RML Mappings (S), in: Proceedings of the 31st International Conference on Software Engineering and Knowledge Engineering, KSI Research Inc. and Knowledge Systems Institute Graduate School, 2019, http://dx.doi.org/10.18293/seke2019-055.

[39] C. Sundara, S. Das, R. Cyganiak, R2RML: RDB to RDF Mapping Language, W3C, 2012, http://www.w3.org/TR/2012/REC-r2rml-20120927/.

[40] J.F. Sequeda, D.P. Miranker, Ultrawrap: SPARQL execution on relational data, Web Semant.: Sci. Serv. Agents (2013) http://dx.doi.org/10.1016/j.websem.2013.08.002, WWW URL: http://www.sciencedirect.com/science/article/pii/S1570826813000383.

[41] M. Arenas, A. Bertails, E. Prud'hommeaux, J. Sequeda, A direct mapping of relational data to RDF, W3C Recomm. 27 (2012) 1–11.

[42] J. Slepicka, C. Yin, P.A. Szekely, C.A. Knoblock, KR2RML: An Alternative interpretation of R2RML for heterogenous sources, in: International Workshop on Consuming Linked Data, 2015.

[43] A. Chortaras, G. Stamou, Mapping diverse data to RDF in practice, in: International Semantic Web Conference, Springer, 2018, pp. 441–457.

[44] M. Lefrançois, A. Zimmermann, N. Bakerally, A SPARQL extension for generating RDF from heterogeneous formats, in: The Semantic Web, Springer International Publishing, 2017.