Declaratively Populating Wikibases with WBML

Eléna LIU [1], Jakub DUCHATEAU and Christophe DEBRUYNE
Montefiore Institute, University of Liège, Liège, Belgium
ORCiD ID: Eléna Liu https://orcid.org/0009-0007-5175-4901, Jakub Duchateau https://orcid.org/0009-0009-5090-8192, Christophe Debruyne https://orcid.org/0000-0003-4734-3847

Abstract. Purpose: This paper addresses the lack of a general declarative approach for generating Wikibase content from heterogeneous data sources. Existing declarative mapping approaches mainly target Resource Description Framework (RDF), while Wikibase-oriented solutions are often limited in scope or procedural in nature. We therefore propose the Wikibase Mapping Language (WBML), a declarative mapping language tailored to the Wikibase statement model.
Methodology: WBML was designed by reusing selected RML concepts for logical sources and iterations, while introducing Wikibase-specific constructs. We implemented a Python prototype that translates WBML into intermediate RDF Mapping Language (RML) and RDF representations, executes them with Morph-KGC, and ingests the results into Wikibase.
Findings: The study shows that declaratively generating Wikibase statements from heterogeneous sources is feasible. We also reported on a demonstration using complex JSON and a simpler, real-world use case for our institution, using CSV files. WBML supports the generation of entities, fingerprints, statements, qualifiers, references, and ranks.
Value: This work provides a basis for declarative construction of Wikibase knowledge graphs from heterogeneous sources. It demonstrates how established ideas from the RDF mapping community can be adapted to other types of knowledge graphs. Future work includes supporting newer RML features not yet supported by reused components and managing updates.

Keywords. Wikibase, KG Construction, Declarative ETL

1. Introduction

Knowledge Graphs (KGs) have become an important technology (and technology stack) for modern data management, and Wikibase[2], the open-source software underlying Wikidata [1], has emerged as a widely adopted platform for building them, spanning domains from public administration [2] to life sciences [3]. The authors in [2] have also presented several other such bespoke deployments of Wikibase. As Wikibase deployments grow, so does the need for tooling to populate instances from the heterogeneous data sources that organizations typically work with.

[1]Corresponding author: `elena.liu@student.uliege.be`.
[2]`https://wikiba.se/`

As we will discuss in Section 2, existing approaches have limitations. Declarative languages such as the Relational Databases to RDF Mapping Language (R2RML) and the RDF Mapping Language (RML) target RDF and do not naturally accommodate Wikibase's "statement-centric" model, in which each statement can carry qualifiers, references, and ranks. And Wikibase also has its own approach for generating opaque identifiers for things (called Items) and properties. And Wikibase-specific tools either rely on "low-level" operational syntax or require custom imperative code. Only T2WML [4,5] offers the closest declarative solution, but it is limited to tabular data, and one needs to know the item and property identifiers to be reused beforehand.

To address this gap, we propose WBML (Wikibase Mapping Language), a declarative mapping language that generates Wikibase statements from heterogeneous sources, including JSON, CSV, and SQL. WBML reuses RML's abstractions for logical sources and iterations while introducing dedicated mapping constructs for entities, fingerprints, statements, qualifiers, references, and ranks. We implement a Python prototype and demonstrate its capabilities on a fairly complex dataset and a real-world institutional use case. Both datasets are presented in this paper, as the second use case relies on simpler data to be ingested into a Wikibase instance.

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 presents the WBML design. Sections 4 and 5 describe execution and implementation. Section 6 presents the demonstration and use case. Section 7 discusses the approach and its limitations, and Section 8 concludes with directions for future work.

2. Related Work

2.1. Wikibase as RDF

RDF graphs model knowledge as sets of triples that connect resources and values via properties, optionally organized in (named) graphs. In Wikibase, however, the data model[3] that conceptually represents the graph is statement-centric: entities are described through statements that not only express a claim (property and value for an entity), but may also include qualifiers, references, and ranks. Thus, where the unit of RDF is a triple or quad, Wikibase was conceived to be more fine-grained and to support statements about statements.

While not important for this paper, it is interesting to note that the RDF representation of Wikibase instances is fairly convoluted.[4] Wikibase SPARQL endpoints provide both "simple" triples for the claims, and triples representing the reified statements with subjects, predicates, objects, references, and qualifiers.

2.2. RDF Generation

A large body of work has addressed the problem of declaratively generating RDF from existing data sources. [6] The goal is to describe, at the mapping level, how data in a source should be transformed into RDF triples using a chosen vocabulary, rather than implementing the transformation imperatively in custom code. This problem is closely

[3] `https://www.mediawiki.org/wiki/Wikibase/DataModel`
[4] `https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format/pl`

related to Wikibase statement generation, since both involve expressing how source data should be transformed into graph-structured knowledge, but using different graph types. The goal of this section is not to provide an overview of the state of the art in RDF generation; an excellent overview is presented in [6], but to present some important initiatives and their rationale.

R2RML[5] is the W3C Recommendation for expressing customized mappings from relational databases to RDF datasets. It provides a declarative mechanism for defining how rows in relational tables are transformed into RDF resources and triples, and its mappings are themselves represented as RDF graphs.

RML [7,8] extends R2RML to heterogeneous sources, including CSV, JSON, XML, and databases, enabling the transformation and integration of RDF from multiple sources into a single mapping. While conceived for generating RDF, its abstractions for source access, iteration, and term construction provide useful building blocks for a declarative Wikibase generation pipeline. YARRRML [9] is a human-readable serialization for expressing declarative mappings that can be translated into RML. Its main contribution is not a new target model, but making mappings easier to write and maintain.

SPARQL-Generate [10] extends SPARQL's syntax with source-specific generator functions to produce RDF directly from non-RDF sources. Facade-X [11], more recently, takes a different approach; it first projects a source into a generic RDF-shaped metamodel, which unmodified SPARQL 1.1 CONSTRUCT queries can then transform into the target graph. Despite this difference, both require that the source be "queryable" as RDF before generation can occur. This works well when the target itself is RDF, since the query language and the target data model coincide.

2.3. (Declarative) Approaches to Wikibase Statement Generation and Population

T2WML [4,5], which stands for Table-to-Wikidata Mapping Language, is a mapping language for transforming tabular data into Wikibase statements. T2WML can directly produce Wikibase statements; however, it is not general-purpose and is limited to tabular data, making it less suitable for heterogeneous sources such as JSON, SQL, or NoSQL.

RaiseWikibase [12] addresses the problem of bulk data insertion into Wikibase, focusing on efficient integration and transaction management. Unlike the Wikibase API, RaiseWikibase supports transactions, allowing rollback of multiple write operations. Its main limitation is that it provides abstractions for writing directly to the relational database underlying Wikibase, rather than mapping sources to Wikibase statements. Consequently, RaiseWikibase is better understood as a system-oriented ingestion solution that can be adopted through declarative approaches.

QuickStatements[6] is used for batch creation and editing of statements in Wikibases. It offers a mechanism for bulk insertion via a compact, tool-specific command syntax, making it well-suited for straightforward import and curation tasks. However, this syntax is relatively low-level and operational, relying on Wikibase identifiers and edit commands rather than on a human-readable declarative description of how source data should be mapped. QuickStatements can encode complex Wikibase statements, but its linear command syntax makes complex statement structures less explicit and harder to

[5] `https://www.w3.org/TR/r2rml/`

[6] `https://www.wikidata.org/wiki/Help:QuickStatements`

manage than in a declarative mapping approach. In other words, QuickStatements, like RaiseWikibase, can be used by declarative approaches as a component for data ingestion.

WikibaseIntegrator[7] is a Python library that provides an interface to the Wikibase API. Unlike direct database-level approaches, it follows the official application interface, thereby aligning better with the Wikibase approach. Its main drawback is that the integration logic must be implemented programmatically. I.e., it does not provide the declarative abstraction needed to specify mappings of heterogeneous sources onto Wikibase statements.

We furthermore note that only QuickStatements, WikibaseIntegrator, and WBML support qualifiers, references, and ranks. As for T2WML and RaiseWikibase, they do not support ranks.

2.4. Conclusion

Several declarative approaches to KG construction already exist. However, these approaches primarily focus on generating RDF from heterogeneous sources. By contrast, declarative approaches that directly target the Wikibase statement model remain limited; among them, T2WML is a notable example, although it is primarily designed for tabular data and does not support ranks. This observation motivated the study that led to this paper.

3. Wikibase ML

We developed the Wikibase Mapping Language[8] (WBML), a declarative mapping language designed specifically to map (semi-)structured data sources onto Wikibase instances. The design of WBML was guided by two goals: adopting concepts from existing mapping languages where possible and introducing new constructs where the Wikibase data model requires them.

3.1. The Gist of WBML

The mapping in Figure 1 illustrates the core constructs of Wikibase Mapping Language.

A `wbml:StatementsMap` is the central construct: it binds a logical source (adopted from RML) to a Wikibase entity and declares what statements to generate for that entity. The `wbml:entityMap` identifies each entity via an entityTemplate (an internal key resolved to a Wikibase QID). A `wbml:labelMap` produces multilingual labels directly on the entity fingerprint. Each `wbml:statementMap` generates one Wikibase statement.

This mapping generates: 1) one Wikibase item per person as well linking it to an item representing its type, 2) an English label for each item drawn from the “name” field, 3) a “nationality” string statement for each person; and 4) “knows” item statement linking persons to one another, illustrating how WBML handles entity-to-entity relationships across sources (here the same source).

As discussed above, the Wikibase data model differs from RDF. As such, we cannot adopt RML constructs (triples maps, term maps, etc.) for generating Wikibase state-

[7] `https://github.com/LeMyst/WikibaseIntegrator`
[8] `http://w3id.org/dre/wbml#`

ments. Moreover, whereas one generates IRIs and blank nodes for RDF, Wikibase uses opaque entity identifiers (which are generated by a separate component). Rather than "abusing" RML to generate Wikibase statements, we introduce dedicated constructs such as `wbml:statementMap`, `wbml:qualifierMap`, and `wbml:referenceMap` to declaratively capture Wikibase's data model.

There are parts of RML that we can reuse as is. WBML adopts RML's notion of logical sources for data iteration. Constructs such as `rml:logicalSource`, `rml:source`, `rml:referenceFormulation`, and `rml:iterator` are used unchanged, allowing WBML mappings to iterate over JSON, CSV, and SQL sources in the same way RML does. This reuse keeps the language familiar to practitioners already knowledgeable in RML and avoids duplicating established solutions.

This choice was not merely one of familiarity. Alternative RDF-generation paradigms based on SPARQL, such as SPARQL-Generate and Facade-X, require that the source be already queryable as RDF. Applied to Wikibase, this would mean first designing an RDF vocabulary to represent Wikibase's statement-centric metamodel (items, statements, qualifiers, references, and ranks) so that SPARQL could then query and recon-

File

```
[
{ "id": 1, "name": "Bob",  "nationality": "British",  "knows": 2 },
{ "id": 2, "name": "Jane", "nationality": "American", "knows": 3 },
{ "id": 3, "name": "Zach", "nationality": "Canadian" }
]
```

WBML Mapping

```
<#PersonMapping> a wbml:StatementsMap ;
  rml:logicalSource [
    rml:source "demonstrators/test/input_data/persons.json" ;
    rml:referenceFormulation ql:JSONPath ;
    rml:iterator "$[*]"
  ] ;
  wbml:entityMap [
    wbml:entityTemplate "Person_{id}" ;
    wbml:class [
      a wbml:Class ;
      wbml:entityTemplate "Person" ;
      wbml:label "Person"@en
    ]
  ] ;
  wbml:labelMap [
    wbml:reference "name" ;
    wbml:language "en"
  ] ;
  wbml:statementMap [
    wbml:propertyMap [
      a wbml:Property ;
      wbml:entityTemplate "nationality" ;
      wbml:label "nationality"@en ;
      wbml:propertyType wbml:string
    ] ;
    wbml:valueMap [ wbml:reference "nationality" ]
  ] ;
  wbml:statementMap [
    wbml:propertyMap [
      a wbml:Property ;
      wbml:entityTemplate "knows" ;
      wbml:label "knows"@en ;
      wbml:propertyType wbml:item
    ] ;
    wbml:valueMap [
      wbml:entityTemplate "Person_{knows}"
    ]
  ] .
```

Resulting Wikibase KG

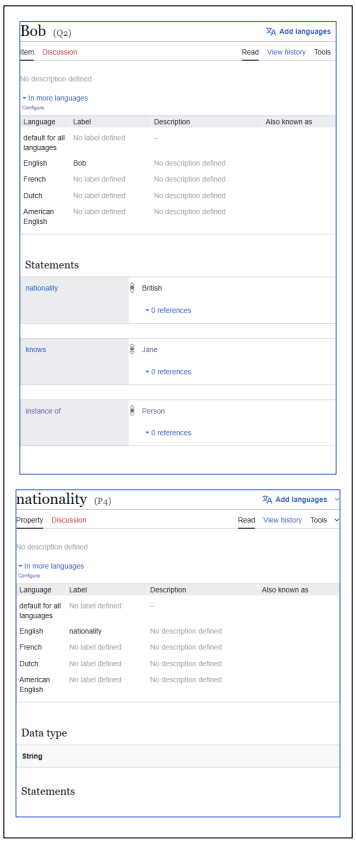

Figure 1. Example of a simple WBML mapping to transform data from JSON into a Wikibase KG.

struct it. This reintroduces exactly the kind of reified, complex representation discussed in Section 2.1, only shifted from Wikibase's RDF export format to the mapping author's input vocabulary. Moreover, this new vocabulary should also include a mapping between identifiers and Wikibase identifiers for entities and statements. RML's logical source and iterator constructs, by contrast, operate directly on the heterogeneous sources. In other words, RML lets us abstract Wikibase's data model at the mapping level, rather than first representing that model in RDF. A further practical reason for this choice is that RML-IO and RML-LV are well-defined; using those constructs unchanged means adopting an already standardized mechanism.

3.2. WBML Features

We assume the reader is familiar with RML and thus proceed with the description of the WBML constructs. We start with `wbml:EntityMap` that are connected to a Statements Map with the predicate `wbml:entityMap`. Much like `rml:SubjectMap`, an Entity Map is responsible for generating the Wikibase entities used in statements. We chose "EntityMap" as both items (Q) and properties (P) are the subjects of statements.

```
<#PokemonMapping> a wbml:StatementsMap ;
  rml:logicalSource [
    # Omitted for brevity
  ] ;
  wbml:entityMap [
    wbml:entityTemplate "Pokemon_{id}" ;
    wbml:entityType wbml:Item ; # Default wbml:Item, can also be wbml:Property
    wbml:class [
      wbml:entityTemplate "Pokemon" ;
      wbml:label "Pokemon"@en ;
      wbml:description "A fictional creature"@en
    ] ;
  ] ;
  # To be continued
.
```

Wikibase generates opaque identifiers for Items of the form Q*n* where *n* is a positive integer. With `wbml:entityTemplate`, we will associate values with such identifiers, which are then stored in an associative array on disk for reuse at later times.

Items can be associated with one or more classes via `wbml:class`. Similarly, the `wbml:entityTemplate` for classes is used to find an existing Item that represents the class. If no such item exists, one is created with the labels, descriptions, and aliases provided in the mapping. Labels, descriptions, and aliases are also updated if they change. Whenever a user knows which are the classes to use, i.e., the Q identifier, then `wbml:classId` can be used to provide an explicit Q-identifier. Finally, Wikibase has no built-in "instance of" relationship, and one is created and stored in the associative map if necessary.

In a Wikibase, an entity can have at most one label and one description per language, whereas multiple aliases may be provided in the same language. The labels, aliases, and descriptions make up an entity's *fingerprint*.[9]

```
wbml:labelMap [ wbml:reference "name.french" ; wbml:language "fr" ] ;
wbml:descriptionMap [ wbml:reference "description" ; wbml:language "en" ] ;
```

Generating statements is fairly straightforward. The generation of a property-value pair for an entity is based on RML-like predicate and object maps, which we call property and value maps, to conform to Wikibase's nomenclature. WBML also allows properties to be declared via an entity template, with the difference that the entity type for property maps is `wbml:Property`, which yields P-identifiers. Value maps can generate values for which data types can be provided and, in the case of an entity, even declare the value's type using the same `wbml:class` construct.

```
wbml:statementMap [
  wbml:propertyMap [
    wbml:entityTemplate "PokedexNumber" ;
    wbml:label "Pokdex number"@en ;
    wbml:propertyType wbml:quantity ;
  ] ;
  wbml:valueMap [ wbml:reference "id" ; wbml:datatype xsd:integer ]
] ;
```

In Wikibase, a qualifier is a propertyvalue pair attached to a statement to refine its meaning, whereas a reference is a set of one or more propertyvalue pairs attached to a statement to record its provenance. While we avoided discussing "snaks" in the previous section, it is worth noting that, in Wikibase terminology, the propertyvalue components that make up a reference are usually called snaks. However, to avoid this terminology, since users are not confronted with it when using a Wikibase, we propose using the term *reference component*. The example below shows how we add a qualifier to a statement ("is hidden ability", which can be true or false), and one reference that consists of two components: the source-based and the date.

```
wbml:statementMap [
  wbml:propertyMap [ wbml:entityTemplate "HasAbility" ; ] ;
  wbml:valueMap [ wbml:entityTemplate "Ability_{ability_name}" ; ] ;
  wbml:qualifierMap [
    wbml:propertyMap [ wbml:entityTemplate "IsHiddenAbility" ; ] ;
    wbml:valueMap [ wbml:reference "ability_flag" ; wbml:datatype xsd:boolean ]
  ] ;
  wbml:referenceMap [
    wbml:referenceRecord [
      wbml:propertyMap [ wbml:entityTemplate "StatedIn" ; ] ;
      wbml:valueMap [ wbml:constant "pokedex.json" ; ]
    ] ;
```

[9] If multiple descriptions are needed for an item, we presume that these would have (potentially) different references and qualifiers, and therefore should be represented with statements. But as this is about Wikibase data modeling, this is out of the scope of this study.

```
      wbml:referenceRecord [
        wbml:propertyMap [ wbml:entityTemplate "StatedOn" ; ] ;
        wbml:valueMap [ wbml:constant "2023-01-01" ; wbml:datatype xsd:date ]
      ]
    ] ;
  ]
```

A rank is a statement-level annotation that indicates the relative status or priority of a statement among multiple statements for the same property (preferred, normal, or deprecated). WBML also provides support for a rank map, but we refer the reader to the source code for examples.

Now that we have covered the most important constructs of WBML, we will describe additional constructs and their rationale for design.

3.3. Joining Statements Maps

Similar to RML, we allow Statements Maps to be joined. To this end, WBML adopts the idea of a *Referencing Object Map*, but translates this for Statement Maps. This approach allows us to generate interrelated Wikibase statements from data that comes from different sources.

```
<#ItemMapping> a wbml:StatementsMap ;
  # rml:logicalSource [...]
  # wbml:entityMap [...]
  # ...
  wbml:statementMap [
    wbml:propertyMap _:HasItemCategory ;
    wbml:valueMap [
      wbml:parentStatementsMap <#ItemCategoryMapping> ;
      wbml:joinCondition [ wbml:child "type" ; wbml:parent "type" ] ;
    ]
  ] .
```

This mapping creates statements for items whose value is not read directly from the item source but is obtained by joining to another Statements Map. Concretely, it links each item to its corresponding item category by matching the child field “type” in the item data with the parent field “type” in the category mapping.

3.4. Property Ranges and XSD Datatypes

The reader might have noticed that in WBML, mapping authors use the RDF terms `wbml:item`, `wbml:quantity`, `wbml:string`, ... in the declaration of new properties, but that the value maps use XSD data types. Our rationale is that XSD datatypes are richer and provide more control. For example, integers, doubles, and floats are all mapped onto Wikibase’s quantity type. Using XSD datatypes to generate values allows us not only to validate the data but also to format the lexical representation for ingestion.

3.5. Reusing Existing Entity Identifiers

We have introduced `wbml:entityTemplate`, which may contain one or more references enclosed in curly braces for string generation. If no such references are present, then we

have a constant "key." Those strings are then used to populate a dictionary that relates those values to P and Q identifiers.

There are cases where one knows the Items and properties beforehand and thus can provide them as constants. To this end, we support using `wbml:propertyId`.

```
wbml:propertyMap [ wbml:propertyId "P42" ; ] ;
```

At the time of writing, we have chosen to use strings rather than IRIs, as the availability of a Wikibase's SPARQL endpoint depends on whether the extension is provided, and the mapping would rely on convoluted approaches where the IRI needs to be parsed as a string for a particular token to be looked up via Wikibase's API.

3.6. Reusing IRIs from Existing Vocabularies and Ontologies

WBML also supports the "reuse" of existing vocabularies for generating properties. This was motivated by the fact that there are quite a few prevalent ontological relationships that are not "natively" supported by Wikibase, such as `rdf:type`, `rdfs:subClassOf`, etc. We allow mapping authors to use these IRIs as constants using `wbml:property`.

```
wbml:statementMap [
  # wbml:propertyMap [ wbml:entityTemplate "type" ; ] ;
  wbml:property rdf:type .
  wbml:valueMap [ wbml:entityTemplate "Pokemon" ; ] ;
] ;
```

Currently, the only "built-in" property that is supposed to be supported is `rdf:type`, whose correspondence must be provided as input; otherwise, a new P will be generated for this property. We assume this supports the syntactic sugar offered by `wbml:class`. This supposition also allows mapping authors to create type declarations with references and ranks using a Statement Map.

4. Executing WBML Mappings

A WBML processor requires a connection to a Wikibase instance, a WBML mapping, and, optionally, a dictionary (generated by a previous run) that acts as a look-up file. WBML iterates over records selected by a Logical Source and applies the mapping rules to each record. For every iteration, it generates Wikibase statements and fingerprints. In this sense, the behavior of WBML is similar to that of RML, but there are a couple of important differences.

First, Wikibase instances work with opaque identifiers. We use the values generated by `wbml:entityTemplate` to find (or create) a new entity identifier (P or Q) and relate this identifier to the value. In that sense, the behavior is similar to the R2RML Term Generation Rule for a blank node.[10] In WBML, we return a Q or P-identifier that is unique to the natural RDF lexical form corresponding to the value.

[10] See Section 11.2 of `https://www.w3.org/TR/r2rml/`

Secondly, if no look-up file is provided, then new items and properties are generated. This is why we also allow mapping authors to execute the mapping with a flag to look for existing items and properties using exact matching, given a preferred language. Such matches are then stored in the look-up file. Algorithm 1 details how WBML currently manages fingerprints. Indeed, we raise errors when the generated fingerprints are not compliant with Wikibase. When values already exist in Wikibase but differ, we do not update them; instead, we show a warning. This feature is merely a convenience, though declarative approaches to entity reconciliation will be considered for future work.

Algorithm 1 Managing fingerprints of an entity

```
for all generated entities (P or Q) s do
    if |labels for a language| > 1 then
        Raise Error
    end if
    if |descriptions for a language| > 1 then
        Raise Error
    end if
    id ← find id in look-up file
    if id = null then
        if label ≠ null ∧ search_wikibase = true then
            id ← find id with exact match label and preferred language in Wikibase
        end if
        if id = null then
            id ← id of newly created entity
        end if
    end if
    e ← getEntity(id)
    if e already has a (different) label for a language then
        Ignore, show warning
    end if
    if e already has a (different) description for a language then
        Ignore, show warning
    end if
    Add missing labels, descriptions, and aliases
end for
```

As for the statements, we verify their existence for reuse and, where necessary, complement them with ranks and references.

5. Implementation

The implementation of WBML follows from its design choices and partial reuse of RML concepts. The process is illustrated in Figure 2, and the pipeline works as follows:

1. First, WBML mappings are translated into an intermediate RML mapping of SPARQL CONSTRUCT queries implemented in Python using RDFLib. For example, the node template `"Pokemon_id"` is used to create an RML term map that relies on generating the URN `"urn:wikibase:item:Pokemon_id"`. Here, the important point is that all mappings are conceptually transformed into RML triple

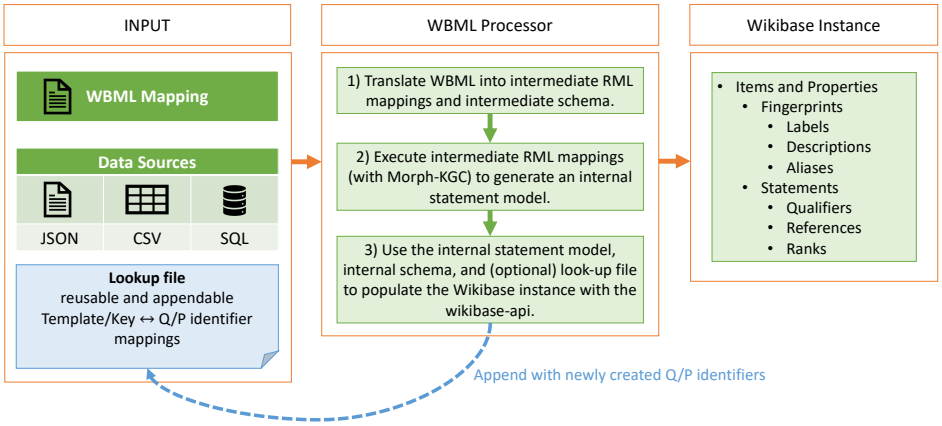

Figure 2. WBML Implementation's Execution Pipeline

maps that generate reified statements, i.e., a triples map for generating instances of statements with a subject, predicate, object, rank, qualifiers, and references.

2. This intermediate RML is not exposed to the user, but allows us to generate RDF with existing RML technologies. In the second step, we use Morph-KGC [13] to execute the resulting mappings and generate RDF in this internal format.
3. Finally, a Python ingestion component, using wikibase-api[11], transforms this intermediate output into Wikibase content by creating and updating statements, fingerprints, Items, and Properties where required. This component also maintains and reuses an associative array that stores the Wikibase identifiers generated for keys and templates, thereby ensuring consistent reuse of newly created entities (Items and Properties) across mapping executions.

While WBML is currently executed through an internal translation to intermediate RML and RDF representations, its declarative nature is preserved because these representations are not exposed at the WBML level. Users specify only the correspondence between source data and Wikibase structures. Unlike state-of-the-art approaches, which even declarative approaches rely on prior knowledge of P and Q identifiers, our approach enables users to include ontological information in the mapping (or in a separate graph combined with the mapping).

The prototype, together with demonstration examples, is available on GitHub and released under the MIT License.[12]

5.1. Discussion

Our implementation currently relies on Morph-KGC, mainly because its Python implementation made it convenient to use in a prototype. However, this choice also has its limitations: Morph-KGC supports R2RML and older RML versions, but full compliance with the latest RML specification is not guaranteed. This may become relevant if

[11] `https://pypi.org/project/wikibase-api/`
[12] `https://github.com/Hearthlia/ATFE9009-wikibase-rdf-pipeline`

WBML is extended with features such as RML Logical Views (LV) [14], which would allow combining multiple sources into a single view and applying functions over that view. Supporting such extensions is left for future work. Conceptually, adopting RML LV should be trivial, since the logical source would point to an RML Logical View.

6. Demonstration and Use Case

In this section, we report on a demonstration and a use case.

6.1. Pokédex Demonstrator

As a demonstrator, we use data from the Pokédex's JSON representation. We have chosen this dataset as it allows us to demonstrate practically all of WBML's capabilities on a complex, multivalued dataset with deeply nested JSON structures.

The dataset covers four interrelated entity types: Pokémon, Moves, Types, and Items. Each Pokémon entry includes multilingual labels (English, Japanese, Chinese, and French), a description, a Pokédex number, base stats such as HP, one or more elemental types, abilities with a qualifier indicating whether the ability is hidden, and evolution chains with qualifiers that capture the evolution condition. Move entries include numeric properties (e.g., accuracy) and a categorical property for move category, alongside a type association. Type entries describe type effectiveness relationships (effective against, ineffective against, no effect against) as item-valued statements linking types to one another. Item entries include multilingual labels, descriptions, an item number, and an item category, all resolved via a join.

The mapping illustrates the range of WBML's capabilities: item-valued statements, multilingual fingerprints (labels, descriptions), qualifiers on statements, references on statements, and joins via `wbml:parentStatementsMap`. All Wikibase properties and "class" items are declared in the mapping and created on demand, so no manual setup of the target Wikibase instance is required before running the pipeline.

The mapping is available in the repository, and can be run by following the instructions in the README file. The result of running this mapping for a local Wikibase instance can be seen in Figure 3. It is important to note, however, that the entity IDs (Q and P identifiers) may differ across Wikibase instances.

6.2. An Institutional Research Register

At the authors' institution, the Technology Transfer Office (TTO) aims to build a register of tools, platforms, and equipment, along with their associated research units. The problem is that this information is governed independently by individual research units and laboratories, and this information needs to be integrated using a common model. The goal is to populate a Wikibase instance that serves as an authoritative source, so that when external parties inquire, for example, about the availability of specific equipment for an experiment, the TTO can efficiently direct them to the appropriate unit.

The project proceeds in several phases. The first phase focused on designing a common ontology. Ontology constructions consisted of defining the classes, properties, labels, and descriptions needed for the register. Domain experts "modeled" the ontology in

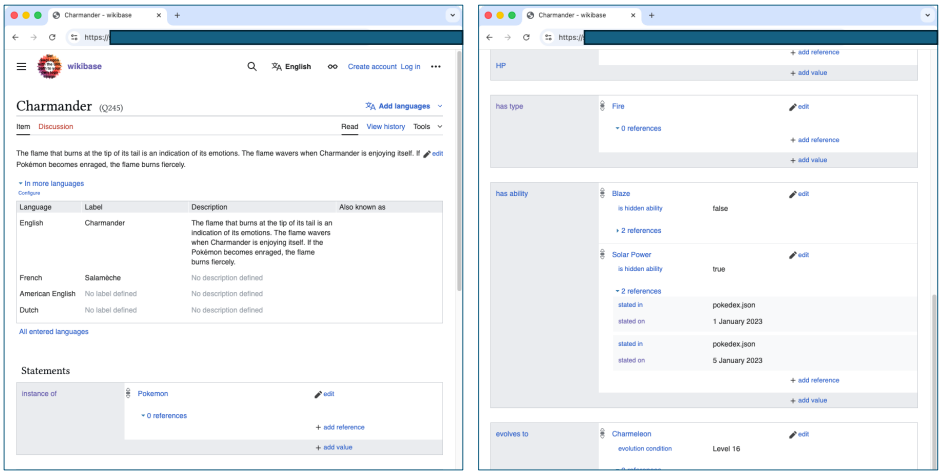

Figure 3. Local Wikibase instance updated after running the Pokédex demonstrator.

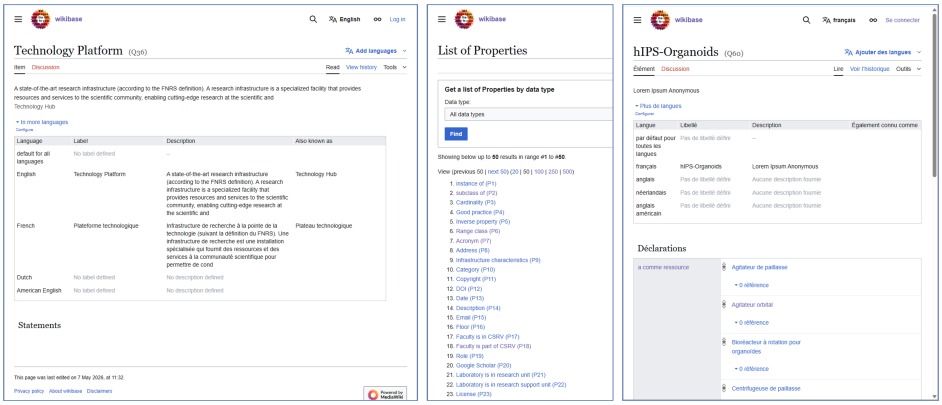

Figure 4. Local Wikibase for the Institutional Research Register (anonymized). The first two screenshots show an example of a "class" and its properties generated from CSV files in phase 1. The third screenshot shows an example of a technological platform and its equipment, generated with data from phase 1. This screenshot is in French as no translated labels and descriptions were provided.

Notion[13], producing tables of classes with their metadata, inter-class relationships, and class-to-value relationships. For this demonstrator, the CSV exports from Notion served as the data sources from which WBML generated the corresponding Wikibase items and properties.

This demonstrator mapping converts a domain model into a Wikibase KG. It reads CSV files exported from Notion, which contain both French and English content. Six statement maps handle the different entity types: one creates Wikibase items for each class with multilingual labels, aliases, and descriptions; a second builds the "class hi-

[13] `https://www.notion.so/`

erarchy" by creating and using a property based on `rdfs:subClassOf`. Three further maps handle data properties, splitting them by their declared range into string, URL, and time property types, and creating corresponding Wikibase properties with labels and descriptions in both languages. The final map handles object properties, including statements on cardinality, good practices, and their range class (linked as a reference to the corresponding class item).

In the second phase, some people have also started gathering information about the technological platforms of certain research units in Notion. Dumps were transformed into Wikibase statements using a similar approach, and the same lookup file was used to reuse the "classes" and properties.

7. Discussion

Currently, we have adopted RML for logical iterations via logical sources, i.e., pulling data from files and databases. Our approach can be easily extended to RML Logical Views (LV) [14]. RML LV is an extension to RML that lets one define a flattened ("relational"-like) and format-agnostic view over one or more existing data sources. Functions can even be applied to such views in order to preprocess the data. What is interesting is that views yield logical iterations, which are used to generate RDF. As such, our approach could incorporate RML LV and functions applied to views. In theory, support for Logical Views relies solely on adopting a compliant engine. While RML-compliant processors exist, such as SDM-RDFizer [15], our prototype was built in Python using Morph-KGC for purely pragmatic reasons.

7.1. Limitations

We currently assume that values are generated through logical iterations. Wikibase supports special "snaks" to indicate that a statement has no value or no known value. This requires including special keywords in WBML. Related to that, not all Wikibase types are supported. We currently support Wikibase strings, times, quantities, and URLs. But we do not (yet) support Wikibase-specific data types, such as LaTeX formulas and linguistic data types. This requires research beyond the widely used XSD datatypes.

From the implementation's perspective, we did not rely on Wikibase extensions (such as the SPARQL endpoint) or on modules developed by others. Our approach relies on a Python library that wraps Wikibase's PHP API. As such, there is no support for transaction management. A solution would have been to directly engage with the database, but that would have required a database connection and not a Wikibase account, which we deem dangerous in practice. Therefore, we believe it is more appropriate to connect to a Wikibase instance via a Wikibase account, though we should investigate data ingestion with transaction management. To the best of our knowledge, RaiseWikibase [12] was the only tool that explicitly considered transaction management.

8. Conclusions

This paper presents WBML, a declarative mapping language for populating Wikibase instances from heterogeneous data sources. By reusing established RML concepts for log-

ical sources and iterations, and introducing dedicated constructs for the Wikibase statement model, WBML addressed the lack of a flexible, declarative approach for generating Wikibase statements from heterogeneous sources and for ingesting them into a Wikibase instance. The feasibility of WBML was demonstrated through two use cases: a fairly complex JSON dataset with multivalued references (i.e., keys containing multiple values) and a simpler real-world institutional use case using CSV files. Together, these cases illustrate WBML's ability to handle joins across sources, multilingual fingerprints, and statement-level annotations.

As for future work, we will look at extending WBML to generate statements that fall outside RDF and to manage Wikibase instances. More concretely, we want to propose constructs for such as "no value" and "unknown value" snaks. We deem it important to seek out use cases where these make sense rather than propose them, as we believe there are practical implications when data needs to be managed. E.g., what happens when an unknown value becomes known? We will also investigate the problems of declarative entity reconciliation and the declarative expression of updates and deletions. For the former, we could look into related work on entity resolution, such as SILK [16]. SILK proposes a Link Specification Language (LSL) that allows one to specify how entities from different Linked Data datasets should be compared for similarity. As for the latter, prior work has already combined LDES [17] with RML to generate evolving datasets. However, LDES's underlying model is append-only: it supports the continuous addition of new elements and tagging elements as deleted, but does not natively support modifying or removing *members*[14] from a graph. When the source data explicitly signals a deletion (e.g., a status flag), the RML mapping can detect it and emit a delete-typed member accordingly. Implicit deletion, by contrast, only works if diffs are computed between complete snapshots of the source. Where LDES focuses on evolving datasets and communicating diffs, we work on a whole knowledge graph. It might nonetheless be interesting to explore how constructs for updating and deleting statements could be incorporated into WBML, and how Wikibase's statement metadata (e.g., ranks) could support them.

To finalize, WBML demonstrates that declarative mapping approaches need not be confined to RDF and that established ideas from the RDF mapping community can be meaningfully adapted to other types of KGs, in this case, Wikibase.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT and Grammarly to: Grammar and spelling check, paraphrase, and reword. While the initial prototype was developed mostly by hand for the MEng thesis, Claude was used for debugging, restructuring the codebase to separate the project from the demonstrator, and ensuring data anonymization for the review. After using these tools, the authors reviewed and edited the content as needed, and take full responsibility for the publication's content.

References

[1] Vrandecic D, Krötzsch M. Wikidata: a free collaborative knowledgebase. Commun ACM. 2014;57(10):78-85.

[14] In LDES, the unit is called a member and a single member can group many triples.

[2] Diefenbach D, De Wilde M, Alipio S. Wikibase as an Infrastructure for Knowledge Graphs: The EU Knowledge Graph. In: Hotho A, Blomqvist E, Dietze S, Fokoue A, Ding Y, Barnaghi PM, et al., editors. The Semantic Web - ISWC 2021 - 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24-28, 2021, Proceedings. Lecture Notes in Computer Science. Springer; 2021. p. 631-47. Available from: https://doi.org/10.1007/978-3-030-88361-4_37.

[3] Turki H, Shafee T, Hadj Taieb MA, Aouicha MB, Vrandecic D, Das D, et al. Wikidata: A large-scale collaborative ontological medical database. J Biomed Informatics. 2019;99. Available from: https://doi.org/10.1016/j.jbi.2019.103292.

[4] Szekely PA, Garijo D, Bhatia D, Wu J, Yao Y, Pujara J. T2WML: Table To Wikidata Mapping Language. In: Kejriwal M, Szekely PA, Troncy R, editors. Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019, Marina Del Rey, CA, USA, November 19-21, 2019. ACM; 2019. p. 267-70. Available from: https://doi.org/10.1145/3360901.3364448.

[5] Szekely PA, Garijo D, Pujara J, Bhatia D, Wu J. T2WML: A Cell-Based Language to Map Tables into Wikidata Records. In: Suárez-Figueroa MC, Cheng G, Gentile AL, Guéret C, Keet CM, Bernstein A, editors. Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas) co-located with 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26-30, 2019. CEUR Workshop Proceedings. CEUR-WS.org; 2019. p. 45-8. Available from: https://ceur-ws.org/Vol-2456/paper12.pdf.

[6] Van Assche D, Delva T, Haesendonck G, Heyvaert P, De Meester B, Dimou A. Declarative RDF graph generation from heterogeneous (semi-)structured data: A systematic literature review. J Web Semant. 2023;75:100753. Available from: https://doi.org/10.1016/j.websem.2022.100753.

[7] Dimou A, Vander Sande M, Colpaert P, Verborgh R, Mannens E, Van de Walle R. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data. In: Bizer C, Heath T, Auer S, Berners-Lee T, editors. Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014. CEUR Workshop Proceedings. CEUR-WS.org; 2014. Available from: https://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[8] Iglesias-Molina A, Van Assche D, Arenas-Guerrero J, De Meester B, Debruyne C, Jozashoori S, et al. The RML Ontology: A Community-Driven Modular Redesign After a Decade of Experience in Mapping Heterogeneous Data to RDF. In: Payne TR, Presutti V, Qi G, Poveda-Villalón M, Stoilos G, Hollink L, et al., editors. The Semantic Web - ISWC 2023 - 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023, Proceedings, Part II. Lecture Notes in Computer Science. Springer; 2023. p. 152-75. Available from: https://doi.org/10.1007/978-3-031-47243-5_9.

[9] Heyvaert P, De Meester B, Dimou A, Verborgh R. Declarative Rules for Linked Data Generation at Your Fingertips! In: Gangemi A, Gentile AL, Nuzzolese AG, Rudolph S, Maleshkova M, Paulheim H, et al., editors. The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers. Lecture Notes in Computer Science. Springer; 2018. p. 213-7. Available from: https://doi.org/10.1007/978-3-319-98192-5_40.

[10] Lefrançois M, Zimmermann A, Bakerally N. Flexible RDF Generation from RDF and Heterogeneous Data Sources with SPARQL-Generate. In: Ciancarini P, Poggi F, Horridge M, Zhao J, Groza T, Suárez-Figueroa MC, et al., editors. Knowledge Engineering and Knowledge Management - EKAW 2016 Satellite Events, EKM and Drift-an-LOD, Bologna, Italy, November 19-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science. Springer; 2016. p. 131-5. Available from: https://doi.org/10.1007/978-3-319-58694-6_16.

[11] Asprino L, Daga E, Gangemi A, Mulholland P. Knowledge Graph Construction with a *Façade*: A Unified Method to Access Heterogeneous Data Sources on the Web. ACM Trans Internet Techn. 2023;23(1):6:1-6:31. Available from: https://doi.org/10.1145/3555312.

[12] Shigapov R, Mechnich J, Schumm I. RaiseWikibase: Fast Inserts into the BERD Instance. In: Verborgh R, Dimou A, Hogan A, d'Amato C, Tiddi I, Bröring A, et al., editors. The Semantic Web: ESWC 2021 Satellite Events - Virtual Event, June 6-10, 2021, Revised Selected Papers. Lecture Notes in Computer Science. Springer; 2021. p. 60-4. Available from: https://doi.org/10.1007/978-3-030-80418-3_11.

[13] Arenas-Guerrero J, Chaves-Fraga D, Toledo J, Pérez MS, Corcho Ó. Morph-KGC: Scalable knowledge graph materialization with mapping partitions. Semantic Web. 2024;15(1):1-20. Available from: https://doi.org/10.3233/SW-223135.

[14] de Vleeschauwer E, Maria P, De Meester B, Colpaert P. RML-view-to-CSV: A Proof-of-Concept

Implementation for RML Logical Views. In: Chaves-Fraga D, Dimou A, Iglesias-Molina A, Serles U, Van Assche D, editors. Proceedings of the 5th International Workshop on Knowledge Graph Construction co-located with 21th Extended Semantic Web Conference (ESWC 2024), Hersonissos, Greece, May 27, 2024. CEUR Workshop Proceedings. CEUR-WS.org; 2024. Available from: https://ceur-ws.org/Vol-3718/paper2.pdf.

[15] Iglesias EA, Vidal M. Results for Knowledge Graph Creation Challenge 2025: SDM-RDFizer. In: Chaves-Fraga D, Dasoulas I, Debruyne C, Dimou A, Serles U, Assche DV, editors. Proceedings of the 6th International Workshop on Knowledge Graph Construction co-located with 22nd Extended Semantic Web Conference (ESWC 2025), Portorož, Slovenia, June 1, 2025. CEUR Workshop Proceedings. CEUR-WS.org; 2025. Available from: https://ceur-ws.org/Vol-3999/short3.pdf.

[16] Volz J, Bizer C, Gaedke M, Kobilarov G. Discovering and Maintaining Links on the Web of Data. In: Bernstein A, Karger DR, Heath T, Feigenbaum L, Maynard D, Motta E, et al., editors. The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings. Lecture Notes in Computer Science. Springer; 2009. p. 650-65. Available from: https://doi.org/10.1007/978-3-642-04930-9_41.

[17] Van Assche D, Rojas JA, De Meester B, Colpaert P. Incremental Knowledge Graph Construction from Heterogeneous Data Sources. Semantic Web. 2026;17(2). Available from: https://doi.org/10.1177/22104968251412270.