wbsheets - Google Sheets extension for Wikibase editing

Kesäniemi, Joonas[1]

[1]*Aalto University, Espoo, Finland*

Abstract

In this paper we introduce a new tool for editing Wikibase data in tabular format called wbsheets. The goal of the tool is to be approachable to wide audience of data editors for which reason it is implemented as a Google Sheets extension. The editable subsets of data are defined using the EntitySchema Wikibase extension as ShEx shape expressions. Schemas and property metadata is used by the wbsheets to generate validation rules in the Sheets documents as well provide autocomplete support against SPARQL endpoints for linkimg cell data to external resources. Finally, data can be synchronized with the connected Wikibase instance directly from Google Sheets with support for custom rank selection and collision detection. wbsheets can be used to either import data from Wikibase to Sheets for editing or map an existing Sheets document to an EntitySchema. The latter approach was used in the case study with the National Archives of Finland where an existing spreadsheets of thousands of corrections based on feedback from the public was mapped to an EntitySchema, reconciliated, validated and synchronized to a custom Wikibase instance.

Keywords

Knowledge graph, Wikibase, Linked data management, Tabular data

1. Introduction

Wikibase is the technology behind Wikidata, the free, collaborative knowledge graph with more than 100000000 data items [1] and counting. Wikibase incorporates many useful features for data management such as authentication and authorization, automatically assigned public identifiers, versioning with provenance information and commenting facilities. Expanding the use of Wikibase beyound Wikidata has been part of the Wikimedia Linked Open Data strategy [1] since 2021 and the results are starting to show as more and more projects are experimenting and deploying Wikibase-based data management solutions. The forerunner in these activities has been the library world with lesson learnt from OCLC [2] in 2019. Other more recent examples of standalone Wikibase instances can be found various domains such as EU knowledge [3], library cataloguing [4] and research data [5],[6] and [7]. And the growing usage creates opportunities for new tools such as the wbsheets presented in the paper.

The data model of Wikibase is based on Items and Properties, which are identified by Q and P numbers respectively. Both can be described using multilingual labels, descriptions and aliases as well as with statements. Properties contain additional data type information. A statement is

Fifth International Workshop on Knowledge Graph Construction@ESWC2024

joonas.kesaniemi@aalto.fi (K. Joonas)
0000-0002-0877-7063 (K. Joonas)

[1]Based on statistics from the Wikidata website https://wikidata.org. Accessed 11.03.2024.

used to make a claim about an Item or Property in question by assigning a Value to a Property, e.g. Berlin<Item> capital of<Property> Germany<Value>. Additional details, such as time range when the statement is valid, can be added with Qualifiers. Finally each statement can contain multiple references to record the provenance of the claim. Each statement is associated with a rank that can be used to prioritize statements.

The UI that ships with the Wikibase is handy for making individual edits to statements that involve properties with data types such as string, quantity or point int time, since it provides convenient input widget for each type. However, this is not case with statements that are used for linking things, such as properties with external-id or URL data type, as it is not possible to query for possible values via the UI, so the user needs to use an external system to look up an copy/paste the values. For properties with wikibase-item data type, it is possible use an autocomplete feature, but the item query is executed against the whole Wikibase instance, which can make it difficult to find the correct value. It is also very easy to add statements that do not make any sense (e.g. Finland is part of Sega Genesis). Editing a larger amount of Items requires the use of custom scripts or general data cleaning and transformation tools with Wikibase support.

This paper presents an ongoing work that allows user to add and edit Wikibase Items using Google Sheets. The idea is to provide the data in a familiar tabular format and take advantage of the collaboration (e.g. sharing and real-time collaborative editing) and validation features built in to the Google Sheets. wbsheets uses a combination of custom Property metadata and a Google Sheets extension to generate validation rules, provide fine-grained autocomplete queries against external data endpoints, and two-way synchronization of data between the Sheets document and Wikibase. wbsheets can be used to populate an empty spreadsheet with data from the Wikibase as well as edit or create Wikibase items by mappings an existing spreadsheet (e.g. Excel import) to Wikibase. Editing is based on ShEx shape expressions that are created using the EntitySchema Wikibase extension [2]. wbsheets specific Wikibase properties are used to for example assign classes to Wikibase Items and EntitySchemas to classes. wbsheets does not require any installations or programming experience from the data editors and it makes it easy to integrate externally managed domain ontologies to Wikibase. However, the setup of the wbsheets does require Wikibase and ShEx know-how, and the prototype implementation still uses a custom server-side API for some of the Wikibase processing, so it currently not straight-forward to use it against any Wikibase instance.

2. Related work

A custom property can be used to map Wikibase content to existing ontologies or other RDF resources. The downside of such approach is that it is always an instance specific customization and requires special handling of certain properties when accessing the data through the SPARQL endpoint. WikibaseRDF extension [8] is an effort which brings these features closer to the core of Wikibase. It can be used to create similar mappings to existing external RDF resources with a customized set of mapping predicates (e.g. owl:sameAs, owl:equivalentProperty and

[2]https://www.mediawiki.org/wiki/Extension:EntitySchema

owl:equivalentClass). These configuration can then be used to generate mapping triples with external mapping predicates to the triplestore and accessed through the SPARQL endpoint.

When it comes to tooling related reading and writing data to the Wikibase, there are programming libraries for different languages [3] that work as wrappers for the Wikibase API, but they have a substantial learning curve for users without prior programming experience. There are two main tools for making changes to Wikibase using a tabular data format: QuickStatements [4] and OpenRefine [5].

Quickstatements is based on command sequences that can be expressed in CSV format. It supports adding and deletion of statements, labels, descriptions and aliases as well as addition of qualifiers and references. QuicksStatements does not currently support setting ranks to statements [6]. The header columns of the QuickStatements CSV file uses Wikibase property numbers and they must be expressed in a specific format. For example S1545 would add a reference using property P1545 (series ordinal) [7] to the statement that was defined in the same line before it. Although the format is easy to create and modify with any spreadsheet editor, it does require the user to keep track of all the ids and their corresponding labels for both the headers and the column values. However, due to the low-level and general command structure, there are other tools that can be used in conjunction with more specialized programs, such as zotkat [8] that translates exports from Zotero [9] reference manager to QuickStatements. Also OpenRefine support QuickStatements exports.

OpenRefine is a general data manipulation tool for cleaning, transforming and extending data in various formats and external services. Wikibase integration is provided by an extension that ships by default with OpenRefine. The extension provides a visual tool for mapping tabular data to statements, labels and aliases with a look that resembles the Wikibase's own editor. OpenRefine support only addition and updates to statements. The mapping editor provides a way to configure how data is merged in case a statement with matching properties or qualifiers already exists as part of the item. All the new statements are added with the rank "Normal". OpenRefine provides a powerful reconciliation features for turning strings (or numbers) to things using for example services that supports the Reconciliation Service API [10], SPARLQ endpoints or local datasets. Modifications and additions are also checked againts a set of quality assurance rules that include Wikibase Quality Constraint [11] based check as well as some custom ones. OpenRefine has many useful and powerful features for cleaning, transforming and linking data, but those come with a learning curve, which might be too much for some users. Although OpenRefine runs in a browser and is implemented using a client server architecture, it is still meant to be installed for single machine usage only.

[3]https://www.wikidata.org/wiki/Wikidata:Tools/For$_{p}$*rogrammers*

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

[5]https://openrefine.org/

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

[7]https://www.wikidata.org/wiki/Property:P1545

[8]https://github.com/UB-Mannheim/zotkat

[9]https://www.zotero.org/

[10]https://reconciliation-api.github.io/specs/draft/

[11]https://gerrit.wikimedia.org/g/mediawiki/extensions/WikibaseQualityConstraints

3. wb-sheets

This section describes the architecture and basic functionality of the wbheets. The solution is still a prototype, but it has enough features that it can be used for solving real world problems (see Case study section). wbsheets development has been done in the context of Semantic Computing Research Group (Seco) at the Aalto University.

As mentioned in the introduction, Wikibase has been used lately more and more as the tool of choice for maintaining datasets that are modelled in a entity-based way [9]. wbsheet has a familiar and easily approachable spreadsheet interface for doing mass editing with features that support the linking of local data to the external domain ontologies and vocabularies. Google Sheets was selected as the implementation platform mainly due to its popularity and powerful Apps Script scripting features [12].

The architecture of wbsheets consists of Wikibase with SPARQL endpoint and EntitySchema extension [13], a custom wbsheets API, and the wbsheets Google Sheets extension. The extension consists of React based frontend and an Apps Script back-end. With Sheets API all modifications to the spreadsheet must happen in the back-end, which in turn calls the wbsheets API. wbsheets API is a NodeJS application that uses wikibase-edit [14] to execute the changes and shexjs [15] for ShEx parsing.

The current prototype supports adding and modifying items, labels, descriptions and statements without qualifiers or references. The support for multilingual content is limited and all the language dependent values are submitted using the default language configured for the document (see Connecting to Wikibase instance below).

3.1. Setting up Wikibase

In order to use wbsheets with Wikibase, the instance must be first populated with wbsheets specific properties and at least one Item presenting a class and one EntitySchema. All of them are needed for building the configuration for the Sheets extension. There are six required wbsheets properties:

uri mapping for mapping Wikibase identifier to URIs.

instance of for denoting the class of an item.

has schema for linking class Item to EntitySchema.

endpoint url for expressing the url of the SPARQL endpoint where the values for the property should be fetched.

query template for string template to use for searching resources for properties with endpoint url.

recon template for string template to use for reconciliation for properties with endpoint url.

[12]https://developers.google.com/apps-script/guides/sheets
[13]https://www.mediawiki.org/wiki/Extension:EntitySchema
[14]https://github.com/maxlath/wikibase-edit
[15]https://github.com/shexjs/shex.js

The EntitySchema represent an editable ”slice” of the item. From the ShEx node constraints only literal facets (e.g. length, pattern and minExclusive) and value sets with stems are handled at the moment in addition to any cardinality information expressed as part of the triple constraints. All other information from the schema is skipped. See listing 1 for an example of a simple EntitySchema in ShEx.

```
<DeathRecord> {
  wdp:P10 xsd:string {1:10};
  wdp:P11 URI [<http://ldf.fi/ammo/>~] ?;
  wdp:P12 xsd:string {1} ;
}
```

Listing 1: Example ShEx EntitySchema with property references, a stem and cardinalities.

3.2. Using the Google Sheets extension

The UI of the wb-sheets consists of five views: Connect, Import, Column editor, Data editor and Synchronization. The basic functionality of each view will be described in the proceeding sections.

Connecting to Wikibase instance

Before any operation can be completed against the Wikibase instance, the user must first configure the document with an url of the instance and select the EntitySchema they want to edit. Available EntitySchemas are retrieved from configured Wikibase instance. Connection does not currently contain any authentication setup, so only openly available Wikibases can be used. Configurations also include the default language used for Wikibase item labels and labels for external resources and mapping of P numbers to the wbsheet specific properties (e.g. P10 equals uri mapping).

Import

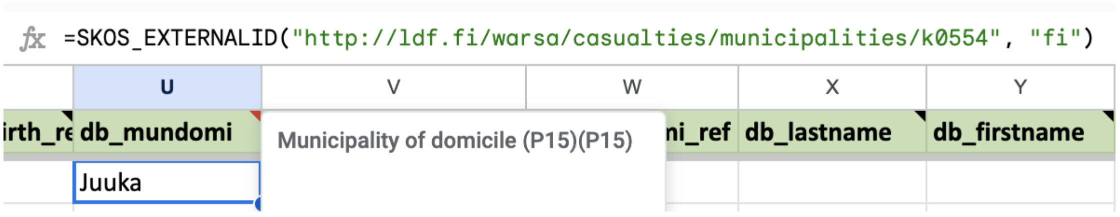
Import view can be used in a cases where the user wants edit existing data. First, user filters the data using properties available from the EntitySchema selected in the in the connect phase. It is also possible to just provide a list of Q numbers to import. Whenever filters are changed the size of the result table is calculated. This information can be used to estimate how long the import operation will last, but also to make sure that the resulting data does not exceed the cells per spreadsheet limit of Google Sheets. Sheet also has a row limit and new sheets are automatically added when the sheet is full. Finally, the user can select which properties are fetched for editing and begin the import process.

Column editor

Column editor is used to manage the column to Wikibase Property mappings. For imported data, the editor can only be used to list available properties and the mappings are created as part of the import process. Editing is not available for now to keep the implementation simple, because adding new columns to imported data would also mean fetching data for those columns. For new data, the user can map columns to properties by highlight the column and clicking the name of the property. Two generated properties ”Wikibase ID” and ”Wikibase label” are always added to the list of available properties and they must always be mapped in order for the synchronization to work. Columns do not need to be mapped in a particular order and is by no means necessary to map all columns or all properties. When a property mapping is applied to a column, a note is added to header row with information about the property such as P number, data type and possible endpoint for fetching property values. Mapping also generates a set of validation rules for the column based on the property. In addition to simple number and date validations, more complex rules are generated for columns with associated property data types wikibase-item, external-id or url. They must for example contain Q numbers and URIs with optionally required prefix (stem). If the property can have multiple values, it can be mapped to as many times to different columns as specified by the maximum cardinality of the property in the EntitySchema. For example, a property ”Author” might be mapped to three columns with headers ”author1”, ”author2”, and ”author3”. The column-property mapping is stored as part of the Sheets document and it is used when editing data for providing contextual information to the data editor view.

Data editor

Basic data editing does not involve any special handling by the wb-sheets. Users can use all functionality of the Google Sheets, such as notes and styling, normally to facilitate the editing process. The data editor has three functions: ”link cells”, ”search and link cells”, and ”search”. Linking is applied to all selected cells. The purpose of linking is to turn plain text cell values in columns that are mapped to properties with wikibase-item, url or external-id into values that include wb-sheets’ custom formulas (see 1. Let’s assume that a column named ”occupation” is mapped to a property with external-id data type and the values must be taken from AMMO ontology [10]. Let’s further assume that the initial value of the cell is ”brush maker”, which do not match the validation rule, since it is not a properly formatted wb-sheets reference to an external resource. If the user selects the cell and clicka ”Search and link cells”, the backend will query the configured SPARQL endpoint for resources with label ”brush maker” and if only one result is found, the value of the cell will be replaced by a formula ”SKOSEXTERNALID(”http://ldf.fi/ammo/harjantekija”, ”en”) and the label will stay the same. Now the cell value is valid and it contains both the URI and the label of the external resource. ”Link cells” works similarly, but it does not make a query, but checks the existence of an URI that is either the value of the cell as for URL datatype, or created by combining the URI prefix configured for the property with the value of the cell for external-ids data type.



```
=SKOS_EXTERNALID("http://ldf.fi/warsa/casualties/municipalities/k0554", "fi")
```

	U	V	W	X	Y
irth_re	db_mundomi	Municipality of domicile (P15)(P15)	ni_ref	db_lastname	db_firstname
	Juuka				

Figure 1: caption

Synchronization

wbsheets uses Google Sheets feature called developer metadata [16] to keep track of the status and possible revision identifier for each row. The revision identifier is stored if the data is

[16]https://developers.google.com/sheets/api/guides/metadata

imported from the Wikibase and can be used to detect edit collisions, if a newer version is available at the server. The row status indicates whether the local data is unmodified, edited or has newer data. With the Synchronization view, the user can write the data back to the Wikibase and handle possible edit collisions. Thanks to the status metadata, only rows with changes are sent to the server.

User can control the way how statements and their ranks are handled as part of the write operation. If statements with the same property already exist, the new data is either added to the item, or all existing statements with that property are replaced with it. The latter option might not be in the ethos of Wikidata, but it does make it straight-forward to keep statements clear of clutter when that is needed. The other option is related to ranks and can be set to either normal, preferred or preferred+. Normal and preferred rank modes works as the name suggests and they set the rank of new statements to either "Normal" or "Preferred". The preferred+ mode works similarly, but also changes the rank of all the statements with the same property to "deprecated". When combined with the "add" property mode, it is easy to retain the alternative values for a property, but only have one preferred value. wbsheets checks for edit collisions by default, but it can also be disabled. Rows with edit collisions are highlighted in the table and they can be handled with the resolve part of the synchronization view. The user can choose to either push their changes to Wikibase and ignore the collision or fetch the new data to the table for selected or all the rows that have collisions.

4. Case study: Finnish National Archives

The development of wb-sheets has been done in co-operation with the National Archives of Finland. The initial ideas related to mapping Wikibase's data model to the spreadsheet and the functionality of the Google Sheets extension were developed together with the potential end-users from the Archives in a series of workshops. The motivation behind National Archive's interest in the development comes from the fact that they receive significant amount of feedback from the public related to the WarSampo portal[**?**], a popular destination for information about Finland in the second World War.

Generating the WarSampo dataset is a complex process that includes multiple preprocessing and mapping tasks**??**. In practice, any changes to the dataset would require assistance from the the researchers that were part of the original WarSampo project, which is not sustainable in the long run. The goal is to give the Natinal Archives the tools to maintain the data themselves and wb-sheets is a part of that work.

Based on the analysis of the feedback, it was clear that most of the feedback dealt with details of individual soldiers. This is natural, since the feedback often comes from the family or relatives of the deceased. It also allowed us to focus our maintenance efforts on a small subset of the data, namely the DeathRecords**??**. The RDF dataset is maintained in a Gitlab instance with hooks in place that update DeathRecord specific named graphs behind SPARQL endpoints for test and production environment on a commit to a specific git branch. A custom component was developed that reads the data from the Wikibase's SPARQL endpoint with the changes and from the Gitlab repository and then creates a new version of the dataset where the properties than overlap are overridden using the data coming from the Wikibase.

After setting up the Wikibase instance as described in the previous sections, the DeathRecord dataset was imported using a Python scripts that took advantage of RaiseWiki?? for speedier creation of new Items for a around 100000 persons. The details of the ingestion process and the data model are out of the scope of this article. The National Archives employed a person whose task was to manually process feedback email into something that can be used to make changes to the WarSampo data. Our first idea was to use the browser to make changes directly to Wikibase, but the vanilla UI was too cumbersome to use for properties that needed to be linked to existing WarSampo resources outside the current dataset. These properties were implemented using an external identifier type, which required user to enter the local name of the RDF resource as a value. For example, in order to set Helsinki (http://ldf.fi/warsa/casualties/municipalities/k0005) as the municipality of birth, the user would need to set the value to ”k0005”. This could have been solved by also importing all the referenced data to the Wikibase, but that would have made the setup process a lot more complex. In the end, we ended compiling all the changes in a spreadsheet, which was first mapped with Wikibase properties with wb-sheets' Column Editor as explained in the previous section. Then properties with links to other resources were mapped to URIs using the Data editor and finally the data was synchronized to the Wikibase instance using override existing values settings.

Most of the time went to the gathering and structuring of feedback into the spreadsheet form. The final table contained one or more changes for roughly 1000 different Items. Applying the wb-sheets based mappings and updating the data took less than a working day.

5. Conclusions and future work

There are not that many options available for editing Wikibase data in a tabular format, which do not require coding. wbsheets utilizes the familiar and easily approachable spreadsheets format and the extensibility and collaborative feature of Google Sheets to provide a new take on an old idea of editing graph based data in a table. The current implementation is in a prototype stage, but it has already been applied successfully in real world use case in collaboration with National Archives of Finland. One of the major limitation of wbsheets is lack support of qualifiers and references, but support for both is under development. From an extension adoption point of view, it would beneficial to remove the wbsheets-api, because that would allow wbsheets to integrated with any Wikibase instance more easily. The user experience could improved for example by allowing items belonging to multiple schemes to be editable within the same spreadsheet in different sheets, providing a work-around to the fact that only one value endpoint per property can be processed and adding support for Reconciliation API. The usage of Google Sheets API is limited by quotas and we have not yet investigated what kind of usage is possible within the free plan. Google's services are also not suitable for all datasets for example due to nature of the data or organizational policies.

Acknowledgments

This project has received funding from the Memory Foundation for the Fallen (Kaatuneiden Muistosäätiö).

References

[1] LinkedOpenData/Strategy2021/Wikibase - Meta — meta.wikimedia.org, https://meta.wikimedia.org/wiki/LinkedOpenData/Strategy2021/Wikibase, 2021. [Accessed 11-03-2024].

[2] J. Godby, K. Smith-Yoshimura, B. Washburn, K. K. Davis, C. F. Eslao, S. Folsom, X. Li, M. McGee, K. Miller, H. Moody, et al., Creating library linked data with wikibase: Lessons learned from project passage (2019).

[3] D. Diefenbach, M. D. Wilde, S. Alipio, Wikibase as an infrastructure for knowledge graphs: The eu knowledge graph, in: A. Hotho, E. Blomqvist, S. Dietze, A. Fokoue, Y. Ding, P. Barnaghi, A. Haller, M. Dragoni, H. Alani (Eds.), The Semantic Web – ISWC 2021, Springer International Publishing, Cham, 2021, pp. 631–647.

[4] S. Zapounidou, L. Ioannidis, M. Gerolimos, E. Koufakou, C. Bratsas, Entity management using rda and wikibase: A case study at the national library of greece, Journal of Library Metadata (2024) 1–21.

[5] M. Koho, L. P. Coladangelo, L. Ransom, D. Emery, Wikibase model for premodern manuscript metadata harmonization, linked data integration, and discovery, J. Comput. Cult. Herit. 16 (2023). URL: https://doi.org/10.1145/3594723. doi:10.1145/3594723.

[6] C. Shimizu, P. Hitzler, S. Gonzalez-Estrecha, J. Goeke-Smith, D. Rehberger, C. Foley, A. Sheill, The wikibase approach to the enslaved.org hub knowledge graph, in: T. R. Payne, V. Presutti, G. Qi, M. Poveda-Villalón, G. Stoilos, L. Hollink, Z. Kaoudi, G. Cheng, J. Li (Eds.), The Semantic Web – ISWC 2023, Springer Nature Switzerland, Cham, 2023, pp. 419–434.

[7] L. Rossenova, P. Duchesne, I. Blümel, Wikidata and Wikibase as complementary research data management services for cultural heritage data, in: The 3rd Wikidata Workshop, Workshop for the Scientific Wikidata Community, 2022. URL: http://ceur-ws.org/Vol-3262/paper15.pdf.

[8] GitHub - ProfessionalWiki/WikibaseRDF: Wikibase extension that allows defining RDF mappings for Wikibase Entities — github.com, https://github.com/ProfessionalWiki/WikibaseRDF, ???? [Accessed 11-03-2024].

[9] F. Ibekwe-Sanjuan, B. Geoffrey, Implications of big data for knowledge organization., Knowledge organization 44 (2017) 187–198.

[10] M. Koho, L. Gasbarra, J. Tuominen, H. Rantala, I. Jokipii, E. Hyvönen, Ammo ontology of finnish historical occupations, CEUR Workshop Proceedings 2375 (2019) 91–96. International Workshop on Open Data and Ontologies for Cultural Heritage, ODOCH ; Conference date: 03-06-2019 Through 03-06-2019.