Property cardinality analysis to extract truly tabular query results from Wikidata

Wolfgang Fahl[1], Tim Holzheim[1], Andrea Westerinen[2], Christoph Lange[1,3] and Stefan Decker[1,3]

[1]*RWTH Aachen University, Computer Science i5, Aachen, Germany*
[2]*OntoInsights LLC, Elkton, MD, United States*
[3]*Fraunhofer FIT, Sankt Augustin, Germany*

Abstract

Tabular views of data with tables, columns and rows as the key concepts are still a popular basis for data analysis and storage, used in relational database management systems and spreadsheet software. Graph based approaches are a superset of the tabular view and use vertices and edges/properties as the key concepts to manage the data. A common way to store graph data is using subject, predicate and object triples in a "triple store". For quite a few use cases, transforming the triple store data to a tabular view is needed since tabular systems are still widespread. The straightforward approach to generating such data using a "naive" query will, however, create unexpected results or even fail because of conceptual differences between the relational and the graph approaches regarding the handling of the cardinality/multiplicity of properties.

This work shows a systematic approach to analyze the property cardinalities of the graph data in an RDF/SPARQL triple store and to extract "truly tabular" data (with cardinalities of 1 in each column) by automatically generating appropriate queries. We propose a SPARQL query builder that simplifies the generation of queries that limit the result set to such "truly tabular" data.

Keywords

Wikidata, RDF, SPARQL

1. Introduction

In recent years Wikidata [1] has grown from an encyclopedic knowledge graph to a community curated general knowledge graph with subgraphs for diverse special interest subcommunities, such as scholarly articles, astronomical objects, people, Wikimedia assets, chemical compounds, etc., with niches for items such as Pokémon and "Game of Thrones" characters.

The work discussed in this paper is motivated by the frustration we experienced when trying to extract academic conference data from RDF datasets as supplied by the German National Library (in its GND dataset) or by Wikidata. Our experience with the GND dump[1] as shown in Table 1 shows that the RDF dataset has content that creates surprising effects when using a straightforward/naive query to select the intended data. E.g., of the 731K conference entries in

Wikidata'22: Wikidata workshop at ISWC 2022
0000-0002-0821-6995 (W. Fahl); 0000-0003-2533-6363 (T. Holzheim); 0000-0002-8589-5573 (A. Westerinen); 0000-0001-9879-3827 (C. Lange); 0000-0001-6324-7164 (S. Decker)
© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

[1]https://wiki.bitplan.com/index.php/Truly_Tabular_RDF/GND

CEUR-WS.org/Vol-3262/paper7.pdf

Table 1
GND authorities-kongress property multiplicity

property	gnd	total	unique	min	max	avg
eventId	gnd:gndIdentifier	731651	731651	1	1	1
title	gnd:preferredNameForTheConferenceOrEvent	731645	731645	0	1	0.999992
acronym	gnd:abbreviatedNameForTheConferenceOrEvent	3537	3206	0	4	0.00483
sameAs	owl:sameAs	769120	693077	0	20	1.05
variant	gnd:variantNameForTheConferenceOrEvent	632368	229268	0	41	0.86
date	gnd:dateOfConferenceOrEvent	710819	704949	0	9	0.971
areaCode	gnd:geographicAreaCode	797037	612631	0	11	1.089
place	gnd:placeOfConferenceOrEvent	659305	624667	0	18	0.901
topic	gnd:topic	5061	3520	0	6	0.00691
homepage	gnd:homepage	19011	18702	0	3	0.026
prec	gnd:homepage	12182	12106	0	3	0.0166
succ	gnd:homepage	11974	11929	0	3	0.0163

the dataset, only 3206 have a unique acronym, only 0.4% of the records have acronyms at all, and some conferences have up to 4 different ones. When querying the dataset with a "naive" query[2], we had single conferences having 1 query solution (known as a "binding set"), while others had up to 576 query solutions. We learned that we had to analyze our data for cardinalities and adapt the SPARQL query accordingly.

The problem showed up again when querying Wikidata, and the research question arose, "How could duplicate query solutions be avoided and queries be generated for tabular target systems?"

To better describe the problem, consider a novice user with a "tabular" (SQL) background attempting to query Wikidata to extract a table full of information of interest - data related to "Game of Throne" characters. Very likely, they are unfamiliar with the details of SPARQL multiset semantics (as outlined by Angles [2]).

In Wikidata, a "Game of Thrones" character (such as Jon Snow[3]) would be linked via an "instance of"[4] property to the type "Game of Thrones" character[5]. Retrieving all instances of Game of Thrones characters would simply check for all items of that type (117 at the time of writing this work). It is natural to now create a query to get a table of all Game of Thrones characters with columns for their most interesting aspects.

We might start using the Wikidata Query Service [3] and one of its examples. Let us select the first "Cats" sample from the many examples provided at Wikidata SPARQL examples web page[6]. We will refine the example for our purposes with properties related to Jon Snow.

We might thus create a "naive" query[7] without aggregates as it would work in SQL out of the box. The result of executing this query appears to be nicely tabular and can be exported to CSV to then be imported to our favorite relational database or spreadsheet. However, when

[2]https://wiki.bitplan.com/index.php/GOTExample2022#NaiveGOTQuery2
[3]https://www.wikidata.org/wiki/Q3183235
[4]https://www.wikidata.org/wiki/Property:P31
[5]https://www.wikidata.org/wiki/Q20086263
[6]https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/queries/examples
[7]https://wiki.bitplan.com/index.php/GOTExample2022#NaiveGOTQuery1

examining the results, the table does not contain 117 entries (all the documented Game of Thrones characters) but only 28.

SPARQL queries do not return NULL values as a SQL/relational database query language [4] would. The SPARQL query needs to be amended with the "OPTIONAL" keyword as shown in the second naive query[8] to return either a value or to leave a query variable unbound in case of no match.

Next, suppose a bit more information is needed and 3 more properties are requested. The result of this third naive query[9] leads to a new surprise for the novice. Suddenly the query result has 304 rows for our 117 characters. There are duplicate rows for each character and strange combinations of entries. A troublesome example seems to be Jorah Mormont[10] having 8 different values for "occupation".

Given this problem statement, we explore the basic concepts upon which this work is based in Section 2. Sections 3 and 4 overview how property cardinality analyses are performed (and can be automated), as well as the results of analyzing various Wikidata item types. Section 5 discusses related work; Section 6 concludes.

1.1. Data management design choices

Viewing the world in terms of tables vs. graphs makes a major difference in your approach to data management.

A tabular design is more straightforward than a graph design since it forces dealing with multi-valued properties in a very explicit manner. In each row, a column has exactly one entry. In a graph, this restriction is relaxed and multiple values per property (i.e., column) are possible. This calls for an implicit $1 : n$ relation for the property's domain and range. Unfortunately, introducing the $1 : n$ relation is often not an explicit design decision but happens accidentally (due to multi-valued data being entered by chance) or might be motivated by specific requirements. Note that the use of multi-valued properties can be a legitimate design choice.

Choosing a tabular design to avoid complexity is not easy in a graph environment, especially if data and schema are not well documented or controlled. This is the case with many datasets in an RDF/SPARQL based system, such as Wikidata or GND [5][11].

Using a "naive" approach for handling data in RDF/SPARQL directly translates SQL thinking to SPARQL. However, as discussed above, this leads to surprising results. Following the "principle of least astonishment" [6][12], it is desirable to get a more consistent result by applying the "truly tabular" approach.

[8] https://wiki.bitplan.com/index.php/GOTExample2022#NaiveGOTQuery2

[9] https://wiki.bitplan.com/index.php/GOTExample2022#NaiveGOTQuery3

[10] https://www.wikidata.org/wiki/Q3810007

[11] The main catalog of the German National Library has embraced an RDF ontology from 2012

[12] If a feature is accidentally misapplied by the user and causes what appears to him to be an unpredictable result, that feature has a high astonishment factor and is therefore undesirable. If a feature has a high astonishment factor, it may be necessary to redesign the feature

2. Basic concepts

2.1. Tables

Relational databases based on tables, columns and rows were proposed by E. F. Codd in 1970 [7]. Codd discusses the relational view (or model): *"It appears to be superior in several respects to the graph or network model . . . It provides a means of describing data with its natural structure only–that is, without superimposing any additional structure for machine representation purposes"*. The "natural structure" for many use cases thus is a tabular structure with rows and columns. SQL/SEQUEL [4] is the query language that has been in use for more than four decades for describing and querying data in a ubiquitous fashion.

- A table t is a subset of the cartesian product $D_1 \times \cdots \times D_n$. t has n attributes $a_1 \ldots a_n$. Each attribute a_k has a domain D_k, and any given row of r is an n-tuple $(t_1, \ldots, t_n)$ such that $t_k \in D_k$.
- A table is in first normal form (1NF) if all domains D_k are *atomic*, i.e., no domain D_k is composed of other atoms.

2.2. Graphs

A graph $G = (V, E)$ has a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E = \{e_1, e_2, \ldots, e_m\}$ where each edge e_k is a pair (v_i, v_j) defining the connected vertices. Graph data may be stored in RDF [8] triple stores and queried using the SPARQL query language defined by the W3C [9].

2.3. Classes/Types

The concepts of classes (also known as types) are defined in the W3C specifications for RDF: *Resources may be divided into groups called classes. The members of a class are known as instances of the class. Classes are themselves resources* [10]. Similarly for OWL, the initial specification stated that *Classes provide an abstraction mechanism for grouping resources with similar characteristics* [11]. This evolved in OWL2 to *Classes can be understood as sets of individuals* [12]. Note that classes/types are the vertices of a graph.

Wikidata uses the Property P31/"instance of" to assign classes to instances. Properties are the edges of a graph. SPARQL allows querying instances of a specific class as well as querying other properties.

2.4. SPARQL query results and cardinalities

A SPARQL query result is a solutions sequence (or "binding sets"), which might be unordered [13, 9]. A solution is defined by a mapping of variables to RDF terms. For in depth-definitions, see [2].

In the context of this work the concept of the cardinality (also called multiplicity or frequency) of properties is relevant. We define the cardinality of a property (also called max frequency, maxf) with respect to an instance as the maximum number of RDF triples selected by the basic graph pattern, *<instanceIdentifier> <propertyIdentifier> ?value*. (See GND Query to analyze

multiplicity[13]: for a concrete SPARQL example.) In general, a SPARQL graph pattern is written as *<subjIdentifierOrVariableName> <propertyIdentifierOrVariableName> <objectIdentifierOrVariableName>*.

Expanding on this pattern, all instances of a particular class can be queried using *?item a <classIdentifier>*. However, the problem is a bit more complex. You may want all instances of the class AND any of its subclasses. This requires using SPARQL property paths, and in Wikidata is written as *?item wdt:P31/wdt:P279* <classIdentifier>* (see Wikidata basic membership properties[14]). The query result is then grouped by the *?item* to count the property values per class instance.

2.5. Truly Tabular SPARQL SELECT query results

A Truly tabular SPARQL SELECT query result is a solution sequence where

- each RDF term (column) is functionally dependent on a property of a specific class
- each tuple of RDF terms (row) represents a single instance of a class so that the cardinality / entry count is limited by:

$$\prod_{col=0}^{n} \max(entrycount(table, column)) = 1$$

  I.e., for each row (an instance for which data is provided) and column (a property that has some value) in a table, there is only one value[15]).

For example, for a list of instances of people (class: human/Q5), there might be properties indicating their date of birth, their country of residence, a contact phone number and their marital status. Properties such as date of birth, country of residence and marital status should be single-valued for any specific person, whereas there may be multiple contact phone numbers. In this case, the single-valued properties would lead to "truly tabular" query results while the list of contact phone numbers column needs to be specified by its own RDF term in the solution mapping using the SPARQL GROUP_CONCAT aggregate to avoid multiple rows per person.

Obtaining such tabular query results directly is much easier than having to post-process the data to make it compatible with systems expecting tabular data.

Since often only a few instances of a specific class have multi-valued attributes (such as persons with multiple genders), it is simpler to handle the standard cases with single-valued attributes and account for the special cases with extra aggregate columns such as COUNT, MIN, MAX and GROUP_CONCAT.

Combining the tabular standard view on data with the aggregate view containing the special cases allows handling the special cases with standard tabular tools such as spreadsheets.

Our definition of "truly tabular" corresponds to the first normal form of relational database theory. In practice it is not clear whether the first normal form (1NF) is violated if an attribute holds a value that has a datatype that is non relational such as a string but the content is

[13]https://wiki.bitplan.com/index.php/Truly_Tabular_RDF/GND

[14]https://www.wikidata.org/wiki/Help:Basic_membership_properties

[15]Note that the value does not have to be atomic in the pure sense - see comments below

```
SELECT ?count (COUNT(?count) AS ?frequency) WHERE {
  SELECT ?item ?itemLabel (COUNT (?value) AS ?count)
  WHERE {
    # instance of Game of Thrones character
    ?item wdt:P31 wd:Q20086263.
    ?item rdfs:label ?itemLabel.
    FILTER (LANG(?itemLabel) = "en").
    # cause of death
    ?item wdt:P509 ?value.
  } GROUP BY ?item ?itemLabel
} GROUP BY ?count
ORDER BY DESC (?frequency)
```

Listing 1: SPARQL Query to count property data availability

multi-valued by using a delimiter separated list of entries. "A,B,C" could be interpreted as an atomic value or as a list of three items. If the latter, the table would not be in 1NF any more.

IBM DB2 designer Christopher J. Date states that "The 1970 paper fails to define the term atomic value adequately. This failure led to a massive misunderstanding in the database community at large as to what exactly it means to say a relation is in first normal form — a misunderstanding that persists, widely, to the present day." [14]

Therefore, the "true tabularity" of a table is in the eyes of the observer.

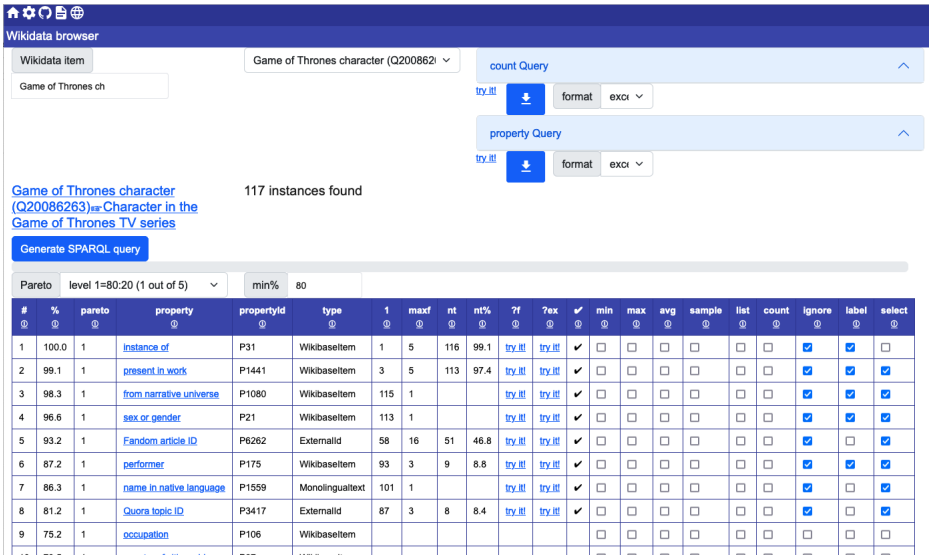
3. Analyzing property cardinalities

Analyzing the cardinalities of the properties related to a class is done in three steps. First a count query determines the total number of instances in the class, then a property count query selects all properties that any instance of the class might use along with the number of instances that actually provide data for that property. In the third step, a query per property is performed that reports the actual cardinality statistics per property. Listing 1 shows an example for the property “cause of death” of the class “Game of Thrones” character. (Note that the property P279 is not necessary since we know that there are no subclasses defined for the class Q20086263.)

To automate the cardinality analysis, we created a tool that will be enhanced to become a SPARQL query builder: http://wikidata.bitplan.com[16].

Figure 1 shows an exemplary screen shot after starting the analysis for the class Game of Thrones Character(Q20086263).

[16] http://wikidata.bitplan.com



#	%	pareto	property	propertyId	type	1	maxf	nt	nt%	?f	?ex	✔	min	max	avg	sample	list	count	ignore	label	select
1	100.0	1	instance of	P31	WikibaseItem	1	5	116	99.1	try it!	try it!	✔	☐	☐	☐	☐	☐	☐	☑	☑	☐
2	99.1	1	present in work	P1441	WikibaseItem	3	5	113	97.4	try it!	try it!	✔	☐	☐	☐	☐	☐	☐	☑	☑	☑
3	98.3	1	from narrative universe	P1080	WikibaseItem	115	1			try it!	try it!	✔	☐	☐	☐	☐	☐	☐	☑	☑	☑
4	96.6	1	sex or gender	P21	WikibaseItem	113	1			try it!	try it!	✔	☐	☐	☐	☐	☐	☐	☑	☑	☑
5	93.2	1	Fandom article ID	P6262	ExternalId	58	16	51	46.8	try it!	try it!	✔	☐	☐	☐	☐	☐	☐	☑	☐	☑
6	87.2	1	performer	P175	WikibaseItem	93	3	9	8.8	try it!	try it!	✔	☐	☐	☐	☐	☐	☐	☑	☑	☑
7	86.3	1	name in native language	P1559	MonolingualText	101	1			try it!	try it!	✔	☐	☐	☐	☐	☐	☐	☑	☐	☑
8	81.2	1	Quora topic ID	P3417	ExternalId	87	3	8	8.4	try it!	try it!	✔	☐	☐	☐	☐	☐	☐	☑	☐	☑
9	75.2	1	occupation	P106	WikibaseItem								☐	☐	☐	☐	☐	☐	☐	☐	☐

Figure 1: Truly Tabular Property Cardinality Analysis prototype screenshot

4. Results

4.1. Examples

The two figures 2, 3 show how "truly tabular" the instances of the classes scientific conference series (Q47258130)[17] and academic conference (Q2020153)[18] are. The examples are in the focus of the ConfIDent[15] project that partly funded this work. The explanation of the details of the columns of the property statistics are available on a wiki page[19]. 11 properties are available for more than 20% of the 4244 scientific conference series instances. The external IDs are not unique for a small percentage of corner cases of less than 0.5%. There may be up to three different titles in 6 cases.

13 properties are available for more than 20% of the 7765 academic conference instances. The external IDs are not unique for a small percentage of corner cases of less than 0.2%. There may be up to 14 different locations in 30 cases and multiple start and end dates and titles in less than 0.3% of the cases.

The distribution of the cardinality of properties for classes is a data quality indicator for how well the instance data of this class is curated. Table 2 shows a table of classes and the number of instances available in Wikidata as of 2022-07. The columns total, 80%, 20% and tail denote the number of properties in total and the number of instances where the percentage of instances where at least one value is available for the property is higher than 80% versus 20% according to the pareto principle. The tail column shows the percentage of properties being available in

[17] https://www.wikidata.org/wiki/Q47258130
[18] https://www.wikidata.org/wiki/Q2020153
[19] https://wiki.bitplan.com/index.php/Truly_Tabular_RDF/Info

scientific conference series (Q47258130) ☞series of scientific conferences (e.g., yearly) with the same name and topic

4244 instances found

Generate SPARQL query

Pareto level 1=80:20 (1 out of 5) min% 20.0

#	%	pareto	property	propertyId	type	1	maxf	nt	nt%
1	100.0	1	instance of	P31	WikibaseItem	4201	3	30	0.7
2	97.6	1	title	P1476	Monolingualtext	4132	3	6	0.1
3	97.5	1	DBLP venue ID	P8926	ExternalId	4129	2	2	
4	94.1	1	short name	P1813	Monolingualtext	6940	2	9	0.1
5	61.1	1	VIAF ID	P214	ExternalId	2114	3	4	0.2
6	60.9	1	has part(s)	P527	WikibaseItem	787	49	1795	69.5
7	47.6	1	Microsoft Academic ID	P6366	ExternalId	2004	2	10	0.5
8	42.7	1	GND ID	P227	ExternalId	1800	3	7	0.4
9	26.5	1	Library of Congress authority ID	P244	ExternalId	1121	2	1	0.1
10	25.8	1	WikiCFP conference series ID	P5127	ExternalId	1093	2	1	0.1
11	22.6	1	inception	P571	Time	957	1		

Figure 2: Truly Tabular analysis for Scientific Conferences Series in Wikidata

academic conference (Q2020153)☞conference for researchers to present and discuss their work

7765 instances found

Generate SPARQL query

Pareto level 1=80:20 (1 out of 5) min% 20.0

#	%	pareto	property	propertyId	type	1	maxf	nt	nt%
1	100.0	1	instance of	P31	WikibaseItem	7481	3	216	2.8
2	95.3	1	part of the series	P179	WikibaseItem	7342	3	19	0.3
3	95.3	1	location	P276	WikibaseItem	7333	14	30	0.4
4	94.2	1	country	P17	WikibaseItem	7266	1		
5	91.6	1	start time	P580	Time	7079	2	2	
6	91.5	1	end time	P582	Time	7074	2	3	
7	89.6	1	short name	P1813	Monolingualtext	6940	2	9	0.1
8	89.1	1	title	P1476	Monolingualtext	6886	3	18	0.3
9	84.2	1	described at URL	P973	Url	6534	3	2	
10	40.0	1	GND ID	P227	ExternalId	3086	3	6	0.2
11	27.5	1	VIAF ID	P214	ExternalId	2114	3	4	0.2
12	24.6	1	main subject	P921	WikibaseItem	1728	9	163	8.6
13	23.8	1	organizer	P664	WikibaseItem	1398	19	424	23.3

Figure 3: Truly Tabular analysis for class Academic Conference (Q2020153) in Wikidata

less than 20% of the cases.

Typically the encyclopedic classes such as country or continent with a small number of instances to curate have a long list of well defined properties. Alternately, a concept such as "single" (a record with a single or just a few music items) is an example of a probably crowd-

sourced class that has a lot of instances but only a few well curated properties. Human(Q5) is one of the classes with the highest number of instances. Only two properties P31/instance of and P21/sex or gender are available in more than 80% of the cases. There are only 12 properties which are available in 20% of the cases but 4647 properties in total that makes 99.7% of properties are rarely available.

Table 2
Pareto comparison of properties of some example classes

class	# instances	total	80%	20%	tail
country (Q6256)[20]	186	668	143	252	62%
continent (Q5107)[21]	13	251	9	143	53%
human (Q6256)[22]	10027613	4647	2	12	99.7%
million city (Q1637706)[23]	624	661	20	80	87%
single (Q134556)[24]	99048	451	3	13	97%

5. Related Work

A recent survey by Fiorelli [16] covers several systems for the triplification of tabular data. Such systems are an alternative to make sure that "truly tabular" results are imported.

Gleim [17] applies a compact trie-based representation of property and type co-occurrences to recommend properties to be used for items. This means that the recommender proposes columns that should be filled. The tabularity will only be influenced if more than one entry is added for the recommended property. The inclusion of the values of properties and especially their multiplicity are not considered in the property recommendation.

Lisena [18] handles the case when the number of properties that have multiple values grows (e.g., multilingual names, multilingual descriptions, a set of images, ...). A SPARQL query of these properties would return many results, one for each combination of values. A merging procedure is proposed such that the resulting JSON has a tree structure. The intended "list" aggregate option of our approach creates an equivalent result.

6. Conclusion

Property cardinality analysis is necessary to obtain SPARQL query results appropriate for tabular view target systems such as relational databases or spreadsheets. Property cardinality analysis can be useful for various applications such as the development of semantic alignments using OpenRefine[25], the analysis of the completeness of the description of an entity using tools such as Recoin[19], and the adjustment of the definition of Wikidata statements

We propose a property cardinality analysis and SPARQL query builder tool that simplifies the generation of queries to limit the result set to "truly tabular" data.

Even in its current experimental state, the analysis and query builder results are a strong basis for future work. Potential extensions include data quality analysis, instance-level ontology

[25]https://github.com/OpenRefine/OpenRefine

validation and benchmarking of SPARQL endpoints.

Further details and example are made available at https://wiki.bitplan.com/index.php/Truly_Tabular_RDF.

Acknowledgements. This research has been partly funded by a grant of the Deutsche Forschungsgemeinschaft (DFG). [26]

References

[1] D. Vrandečić, M. Krötzsch, Wikidata, Communications of the ACM 57 (2014) 78–85. doi:10.1145/2629489.

[2] R. Angles, C. Gutiérrez, The multiset semantics of SPARQL patterns, in: P. Groth, E. Simperl, A. J. G. Gray, M. Sabou, M. Krötzsch, F. Lécué, F. Flöck, Y. Gil (Eds.), The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I, volume 9981 of *Lecture Notes in Computer Science*, 2016, pp. 20–36. URL: https://doi.org/10.1007/978-3-319-46523-4_2. doi:10.1007/978-3-319-46523-4_2.

[3] Wikimedia Foundation, Wikidata query service, 2022. URL: https://query.wikidata.org/.

[4] D. D. Chamberlin, R. F. Boyce, SEQUEL: A structured english query language, in: G. Altshuler, R. Rustin, B. D. Plagman (Eds.), Proceedings of 1974 ACM-SIGMOD Workshop on Data Description, Access and Control, Ann Arbor, Michigan, USA, May 1-3, 1974, 2 Volumes, ACM, 1974, pp. 249–264. URL: https://doi.org/10.1145/800296.811515. doi:10.1145/800296.811515.

[5] A. Haffner, GND ontology, 2012. URL: https://d-nb.info/standards/elementset/gnd.

[6] M. Cowlishaw, The design of the REXX language, ACM SIGPLAN Notices 22 (1987) 26–35. doi:10.1145/24686.24687.

[7] E. F. Codd, A relational model of data for large shared data banks, Communications of the ACM 13 (1970) 377–387. doi:10.1145/362384.362685.

[8] R. Cyganiak, D. Wood, M. Lanthaler, RDF 1.1 concepts and abstract syntax, 2014. URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[9] S. Harris, A. Seaborne, SPARQL 1.1 query language, 2013. URL: https://www.w3.org/TR/sparql11-query/.

[10] D. Brickley, R. Guha, RDF schema 1.1: Classes, 2014. URL: https://www.w3.org/TR/rdf-schema/#ch_classes.

[11] S. Bechhofer, et al., OWL web ontology language reference, 2004. URL: https://www.w3.org/TR/owl-ref/#Class.

[12] C. Bock, A. Fokoue, P. Haase, et al., OWL 2 web ontology language structural specification and functional-style syntax (second edition), 2012. URL: https://www.w3.org/TR/owl2-syntax/#Classes.

[13] E. Prud'hommeaux, A. Seaborne, SPARQL query language for rdf, 2008. URL: https://www.w3.org/TR/rdf-sparql-query/.

[14] C. J. Date, Codd's first relational papers: a critical analysis, 2015. URL: https://www.dcs.warwick.ac.uk/~hugh/TTM/CJD-on-EFC%27s-First-Two-Papers.pdf.

[26] ConfIDent project; see https://gepris.dfg.de/gepris/projekt/426477583

[15] Technische Informationsbibliothek (TIB), ConfIDent – a service for open research information on conferences, 2020. URL: https://projects.tib.eu/en/confident/.

[16] M. Fiorelli, A. Stellato, Lifting tabular data to RDF: A survey, in: Metadata and Semantic Research, Springer International Publishing, 2021, pp. 85–96. doi:10.1007/978-3-030-71903-6_9.

[17] L. C. Gleim, R. Schimassek, D. Hüser, M. Peters, C. Krämer, M. Cochez, S. Decker, SchemaTree: Maximum-likelihood property recommendation for wikidata, in: The Semantic Web, Springer International Publishing, 2020, pp. 179–195. doi:10.1007/978-3-030-49461-2_11.

[18] P. Lisena, A. Meroño-Peñuela, T. Kuhn, R. Troncy, Easy web API development with SPARQL Transformer, in: C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, F. Gandon (Eds.), The Semantic Web – ISWC 2019, Springer International Publishing, Cham, 2019, pp. 454–470. doi:10.1007/978-3-030-30796-7_28.

[19] V. Balaraman, S. Razniewski, W. Nutt, Recoin, in: Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18, ACM Press, 2018, p. 1787.–1792. doi:10.1145/3184558.3191641.