Reconstructing Wikidata Statement Lifecycles for Temporal Maintenance Analysis*

Nicolas Ferranti[1,*,†], Axel Polleres[1,†]

[1]Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

Abstract

Wikidata evolves through additions, deletions, re-rankings, and changes to constraint definitions, yet many analyses rely on static snapshots. We present a pipeline that reconstructs truthy statement lifecycles from the full Wikidata meta-history dump, recording when statements are created or removed and by whom. From these lifecycles, we derive analysis-ready artifacts for editor–property activity and historical property constraints. We demonstrate the pipeline through property-centered editor specialization and temporal validation of reported repairs against the constraint definitions active at the time. The results reveal coherent property-maintenance portfolios and show that a non-negligible share of repairs cannot be aligned with a relevant historical constraint. Statement lifecycles therefore enable analyses unavailable from current query endpoints or static dumps alone.

Keywords

Wikidata, Knowledge Graph Evolution, Statement Lifecycles, Historical Analysis, Knowledge Graph Maintenance

1. Introduction

Wikidata [1] is one of the most prominent collaboratively maintained knowledge graphs (KGs) [2]. Its content is maintained by a community of human editors and bots. Factual statements, qualifiers, references, ranks, schema information, and constraints are constantly evolving.

This evolution affects several interacting layers of the graph. Besides factual statements, such as USA (Q30) capital (P36) Washington, D.C. (Q61), Wikidata contains schema-like statements such as instance of (P31), subclass of (P279), and subproperty of (P1647), as well as property constraint (P2302) statements [3]. Changes in any of these layers may affect how the graph is interpreted and validated.

Figure 1 illustrates this interaction for the capital (P36) property. The truthy capital claim depends on statement ranks, while qualifiers provide additional context. Its validation under a subject-type constraint may additionally depend on the surrounding instance of and subclass of statements. Thus, facts, schema information, and constraints may each have distinct editing histories.

This raises two problems for empirical analyses of evolving KGs. First, statements used together as evidence should overlap in time; otherwise, an analysis may rely on a temporally inconsistent configuration that was never simultaneously valid. Second, snapshots miss additions, deletions, re-rankings, and repairs occurring between releases, limiting historical validation and more granular studies of KG evolution.

Historical editing behavior may also reveal maintenance patterns invisible in static snapshots: editors may specialize in particular properties or repeatedly maintain related constraint definitions. However, Wikidata's full history is organized around page revisions rather than explicit statement lifecycles, making such analyses difficult without first reconstructing when statements became active or inactive and which users were involved.

Prior work has studied historical querying of Wikidata [4], evolution in open KGs [5], temporal KG construction [6], and user roles or sub-schema evolution in Wikidata [7, 8]. Our work is complementary:

ISWC 2026 Companion Volume, October 25–29, 2026, Bari, Italy

*Corresponding author.

†These authors contributed equally.

nicolas.ferranti@wu.ac.at (N. Ferranti); axel.polleres@wu.ac.at (A. Polleres)

https://nicolasferranti.github.io/me/ (N. Ferranti); https://aic.ai.wu.ac.at/~polleres/ (A. Polleres)

0000-0002-5574-1987 (N. Ferranti); 0000-0001-5670-1146 (A. Polleres)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

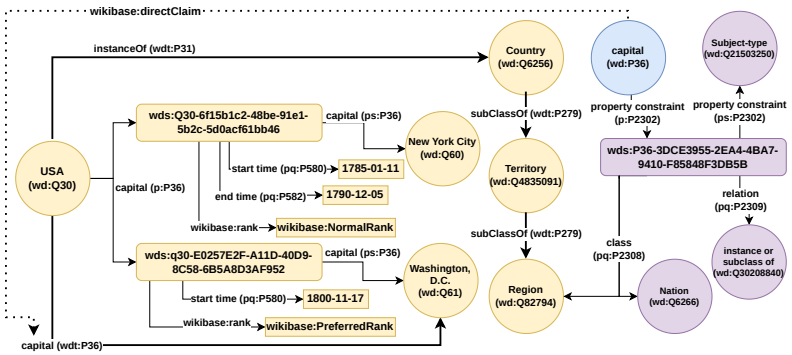


Figure 1: Simplified Wikidata RDF representation of two capital statements for the USA and a subject-type constraint, adapted from [3]. Full statements use `p:`/`ps:` predicates with qualifiers (`pq:`) and ranks, while `wdt:` exposes the truthy claim. Constraint validation may also depend on the P31/P279 type and subclass context.

we reconstruct Wikidata-native truthy statement lifecycles under rank semantics, retain creation and deletion provenance, and derive historical constraint definitions. We show the usefulness of these artifacts through property-centered editor specialization and temporally grounded constraint validation.

2. Proposed Approach

Statement Lifecycles. We use the term *statement lifecycle* for the statement-level history of Wikidata facts and schema-related statements. A lifecycle record represents a qualified statement together with its active interval and the users responsible for its creation and deletion. Conceptually, it has the MAPL [9, 10] form $property(entity, value)@Q$ where the context set Q contains qualifiers, references, and ranks.

Drawing from the ideas of Annotated RDF [11, 12], the full Wikidata edit history can be represented compactly as a sequence of *statement lifecycles* of the form $property(entity, value)@Q[t_c, t_d, u_c, u_d]$, where t_c, t_d are the creation and deletion times, and u_c, u_d are the corresponding users. If the statement remains active, u_d and t_d are left open.

This representation makes Wikidata's temporal dimension explicit at statement level. It allows us to determine when a truthy statement was active, who introduced or removed it, and whether related schema statements or constraints were simultaneously valid. We use these records for property-centered editing analysis and historical constraint maintenance and validation.

Historical Extraction Pipeline. We process the Wikidata page meta-history dump of May 2025.[1] Because the dump records successive page revisions rather than statement lifecycles, we compare consecutive revisions of each entity, reconstructing the RDF deltas, and recording statement creations and deletions with their timestamps and editors.

A central step is reconstructing the historical truthy view. Revisions contain full statements with ranks, whereas direct `wdt:` statements must be derived from Wikidata's rank semantics: preferred-rank statements are selected when present, otherwise normal-rank statements are selected. Adding, deleting, or re-ranking a full statement may therefore start or end a corresponding truthy lifecycle.

As summarized in Figure 2, the resulting lifecycle shards use the schema $s, p, o, t_c, u_c, t_d, u_d$. From them, we derive a truthy artifact for editor–property profiles and a constraint artifact containing historical property constraint (P2302) statements with their definition qualifiers. The latter can be compared with repaired-statement lifecycles. An entity-to-file index supports efficient retrieval without scanning all shards. Code, generated artifact specifications, and supporting materials are available online.[2]

[1] https://dumps.wikimedia.org/wikidatawiki/

[2] https://github.com/nicolasferranti/wikidata_statements_lifecycle_machine

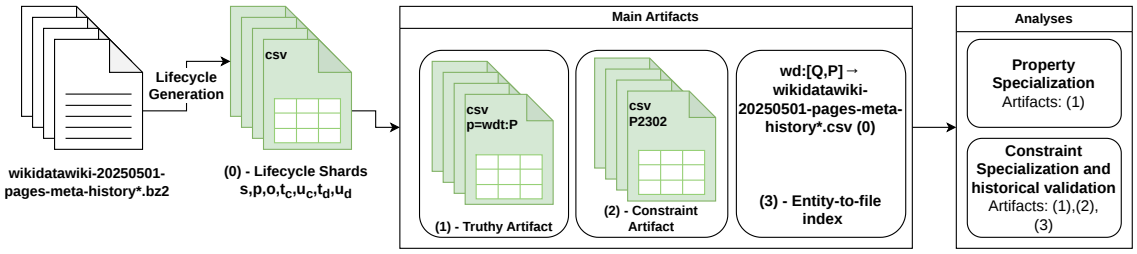
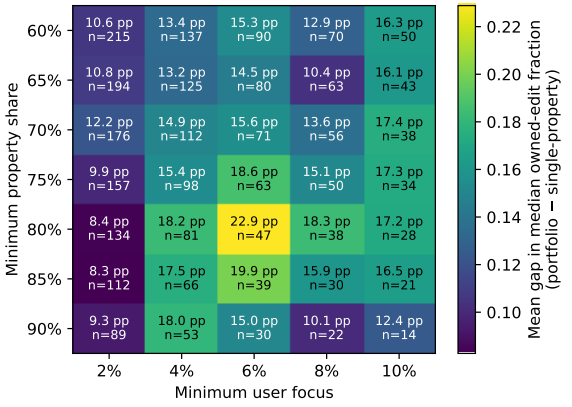


Figure 2: Pipeline from Wikidata page revisions to lifecycle shards, truthy and constraint artifacts, and an entity-to-file index.

Group	Users	Dom. frac.	Dom. edits	Prop. share
Single-property	333	0.178	4,059	0.914
Portfolio	79	0.347	37,787	0.911

(a)



(b)

Figure 3: Property specialization results. **(a)** Specialist groups. *Dom. frac.* is the median share of a user's edits on dominated properties; *Dom. edits* is the median number of such edits; *Prop. share* is the median share of a dominated property's edits made by the specialist. **(b)** Sensitivity over dominance thresholds: minimum property share and minimum user focus. Darker cells indicate larger portfolio–single-property gaps in median dominant-edit fraction; *n* reports the mean number of portfolio specialists.

3. Empirical Demonstrations

Property Specialization. Our first example uses the reconstructed truthy artifact to study whether Wikidata editors specialize in particular properties. We build an editor-property matrix in which each row records the number of truthy statement edits by user u involving property p. After excluding anonymous IP editors, identified bots, low-activity users, and rarely edited properties, the analysis contains 104.4K users, 9.7K properties, 2.6M user-property rows, and 623.6M edits.

We define specialization through a dominance criterion: user u dominates property p if p has at least 50 edits globally, u contributed at least 75% of all edits involving p (*property share*), and p accounts for at least 5% of u's own activity (*user focus*). As shown in Figure 3, this identifies 528 dominant user–property pairs involving 412 editors. Most specialists dominate a single property, but 79 editors dominate two or more properties. These *portfolio specialists* are fewer, but devote a larger median share of their activity to dominated properties than single-property specialists (34.7% versus 17.8%) and have substantially more dominant edits (37,787 versus 4,059). Their portfolios often form coherent groups, such as astronomical metadata, clinical-trial metadata, anime and manga identifiers, or domain-specific external identifiers.

This illustrates how lifecycle-based artifacts can reveal fine-grained maintenance roles that cannot be recovered from static graph snapshots alone.

Historical Constraint Maintenance and Validation. Our second example uses the constraint artifact to study constraint maintenance and to validate reported repairs against historically available constraint definitions. For the maintenance view, we extract human-created, curated property constraint

Table 1
Historical availability of constraint definitions for reported repairs from [13]. Columns 4 and 5 compare each repaired statement lifecycle with the reported constraint statement and with all reconstructed historical constraints of the same type and property, respectively. *Top property* identifies the property accounting for the largest share of no-intersection cases under the any-constraint setting. The final column counts, for that property, statement lifecycles that ended before its first reconstructed constraint was introduced. IRS=item-requires-statement; VRS=value-requires-statement.

Constr. type	QID	Reported repairs	Reported constr. statement no intersection	Any constr. statement no intersection	Top property		
					Name	Share	Lifecycles ended before first constraint
conflicts-with	Q21502838	496k	179k (36%)	112k (23%)	country (P17)	9%	10.5k
inverse	Q21510855	2.18M	178k (8%)	169k (8%)	category's main topic (P301)	21%	34.8k
IRS	Q21503247	16.2M	628k (4%)	613k (4%)	Ensembl transcript ID (P704)	33%	202k
one-of	Q21510859	15.1k	8.26k (55%)	8.26k (55%)	sex or gender (P21)	62%	5.08k
symmetric	Q21510862	928k	36.8k (4%)	36.0k (4%)	said to be the same as (P460)	43%	15.6k
VRS	Q21510864	15.1M	1.27M (8%)	1.24M (8%)	subclass of (P279)	59%	733k

(P2302) additions and build user–constraint-type and user–target-entity profiles. After excluding anonymous IP editors and identified bots, we obtain 62,317 human-created constraint-related statement additions by 835 users, covering 32 constraint types and 9,864 target entities. Among users with at least 20 constraint additions, most are generalists: 150 users work across several constraint types and target properties, while 12 specialize in a constraint type and only one specializes in a target entity.

For the validation view, we compare reported historical repairs from [13] with the constraint definitions reconstructed from the lifecycle artifact. We restrict the experiment to six constraint types whose validation does not require class-hierarchy reasoning or complex qualifier handling. For each repaired triple, we retrieve its lifecycle and test whether it overlaps with a constraint definition active at the time. We report two checks: *reported id*, which tests only the constraint identifier reported in the repair dataset, and *any id*, which tests any reconstructed constraint of the same type and property. As shown in Table 1, a non-negligible share of reported repairs has no temporal intersection with a corresponding historical constraint. Some cases predate the first reconstructed related constraint, indicating that they are not necessarily wrong repairs, but repairs for which the temporal evidence needed for validation is missing. This illustrates how statement lifecycles help separate historically grounded constraint maintenance from explanations that are only valid retrospectively.

4. Conclusions and Outlook

We presented a statement lifecycle machine for reconstructing truthy Wikidata statement lifecycles from the full historical dump, including their active intervals and creation and deletion provenance. We demonstrated its usefulness through property-centered editor specialization and historical constraint maintenance and validation, showing how reported repairs can be checked against constraint definitions active at the time.

Statement lifecycles provide a reusable basis for temporally grounded analysis of evolving KGs, including time-aware training graphs, realistic link-prediction benchmarks, and historically aligned repair data. Although class and hierarchy changes are reconstructed as evolving graph components, they were not yet incorporated into the presented use cases. Future work will therefore address hierarchy-aware constraint validation, schema-level specialization, and community-level maintenance patterns.

Declaration on Generative AI

During the preparation of this work, the author(s) used GPT 5.4 in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

Funding Note

This research was funded in part by the Austrian Science Fund (FWF) [10.55776/COE12].

References

[1] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, D. Vrandecic, Introducing wikidata to the linked data web, in: P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. A. Knoblock, D. Vrandecic, P. Groth, N. F. Noy, K. Janowicz, C. A. Goble (Eds.), The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I, Lecture Notes in Computer Science, Springer, 2014, pp. 50–65. URL: https://doi.org/10.1007/978-3-319-11964-9_4. doi:10.1007/978-3-319-11964-9_4.

[2] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, R. Navigli, A. N. Ngomo, S. M. Rashid, A. Rula, L. Schmelzeisen, J. F. Sequeda, S. Staab, A. Zimmermann, Knowledge graphs, CoRR abs/2003.02320 (2020). URL: https://arxiv.org/abs/2003.02320. arXiv:2003.02320.

[3] N. Ferranti, J. F. De Souza, S. Ahmetaj, A. Polleres, Formalizing and validating wikidata's property constraints using shacl and sparql, Semantic Web 15 (2024) 2333–2380.

[4] T. Pellissier Tanon, F. Suchanek, Querying the edit history of wikidata, in: European Semantic Web Conference, Springer, 2019, pp. 161–166.

[5] A. Polleres, R. Pernisch, A. Bonifati, D. Dell'Aglio, D. Dobriy, S. Dumbrava, L. Etcheverry, N. Ferranti, K. Hose, E. Jiménez-Ruiz, et al., How does knowledge evolve in open knowledge graphs?, Transactions on Graph Data and Knowledge 1 (2023) 11–1.

[6] M. Hofer, M. M. Töpfer, C. Rost, E. Rahm, Dbpedia-tkg: Capturing wikipedia's evolution as temporal knowledge graphs, in: European Semantic Web Conference, Springer, 2025, pp. 262–279.

[7] A. Piscopo, E. Simperl, Who models the world? collaborative ontology creation and user roles in wikidata, Proceedings of the ACM on Human-Computer Interaction 2 (2018) 1–18.

[8] S. Baroncini, M. Martorana, M. Scrocca, Z. Smiech, A. Polleres, Analysing the evolution of community-driven (sub-) schemas within wikidata., in: Wikidata@ ISWC, 2022.

[9] D. L. Martin, P. F. Patel-Schneider, Wikidata constraints on MARS, in: L. Kaffee, O. Tifrea-Marciuska, E. Simperl, D. Vrandecic (Eds.), Proceedings of the 1st Wikidata Workshop (Wikidata 2020) co-located with 19th International Semantic Web Conference(OPub 2020), Virtual Conference, November 2-6, 2020, volume 2773 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: https://ceur-ws.org/Vol-2773/paper-12.pdf.

[10] M. Marx, M. Krötzsch, V. Thost, Logic on MARS: ontologies for generalised property graphs, in: C. Sierra (Ed.), Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, ijcai.org, 2017, pp. 1188–1194. doi:10.24963/IJCAI.2017/165.

[11] O. Udrea, D. R. Recupero, V. S. Subrahmanian, Annotated rdf, ACM Trans. Comput. Logic 11 (2010). URL: https://doi.org/10.1145/1656242.1656245. doi:10.1145/1656242.1656245.

[12] N. Lopes, A. Polleres, U. Straccia, A. Zimmermann, AnQL: SPARQLing up annotated RDFS, in: Proceedings of the 9th International Semantic Web Conference (ISWC 2010), volume 6496 of *LNCS*, Springer, Shanghai, China, 2010.

[13] T. Pellissier Tanon, C. Bourgaux, F. Suchanek, Learning how to correct a knowledge base from the edit history, in: The World Wide Web Conference, 2019, pp. 1465–1475.