Semantic Data Management
Knowledge Graphs Lab

Nima Kamali Lassem, Jule Grigat

June 8, 2025

B1 - TBox Generation

The generated tbox.ttl file ("BDMA12L-B2-Kamali-Lassem+Grigat-tbox.ttl") and the python code to generate the TBox ("BDMA12L-B1-KamaliLassem+Grigat-TBoxGeneration.py") for the following graph are attached in the ZipFolder.

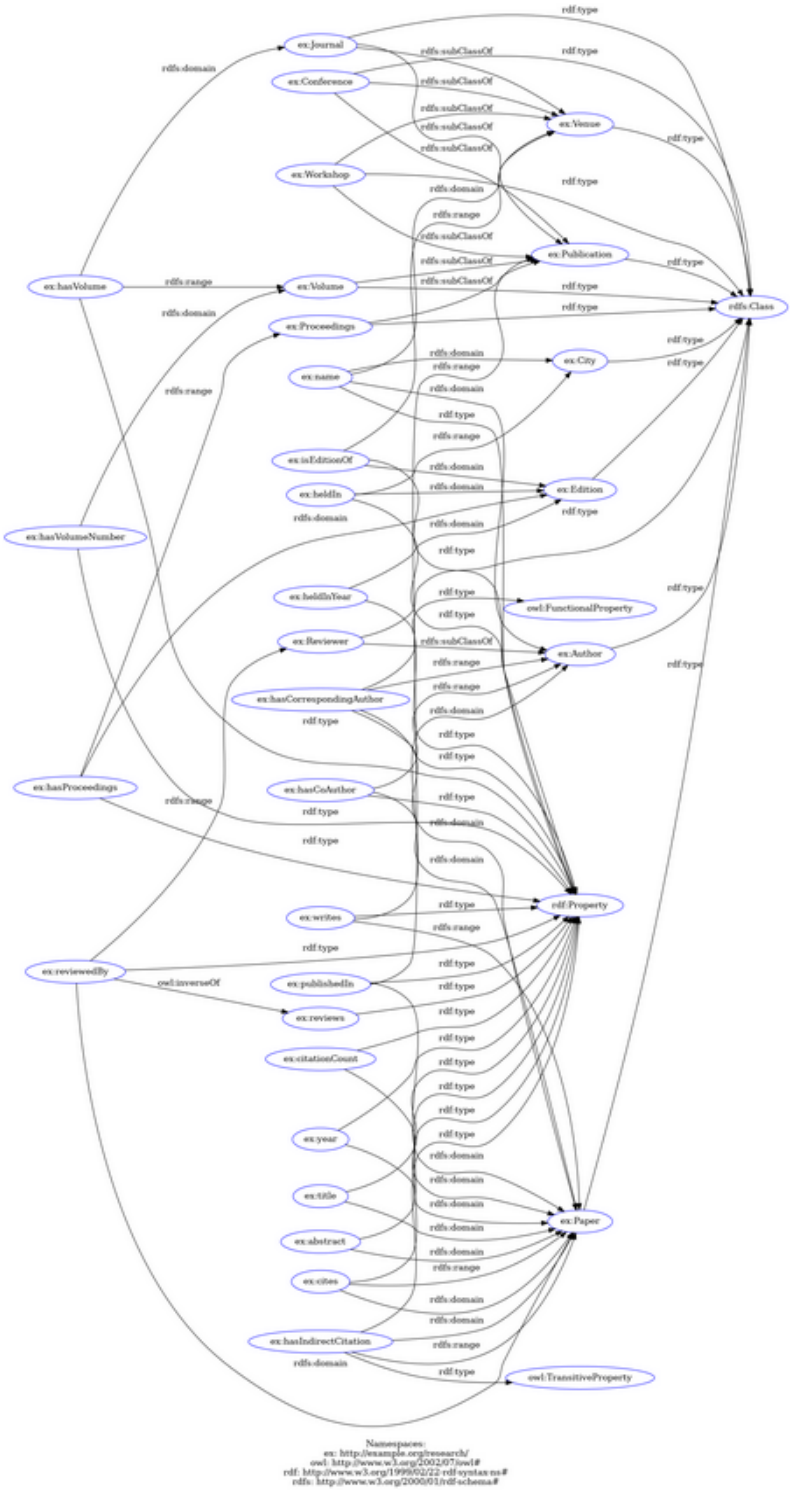

Figure 1: TBox Graph

The TBox-Graph was created with the help of the website "RDF-Grapher" [1], using the available classes and properties provided by the SemanticsScholar Database [2] in CSV-Files originating from the first part of the project. The construction of the TBox and the respective ABox will be further elaborated on in the respective chapters of this report. To enhance the semantic expressiveness and reasoning capabilities of the ontology for this part of the project, a selection of OWL constraints were additionally integrated into the TBox, that are not depicted in the graph. Firstly, the property ex:hasCorrespondingAuthor was declared as a functional property using the owl:FunctionalProperty, enforcing that each paper is linked to at most one corresponding author. Secondly, the review relation was enriched by defining ex:reviewedBy as the inverse of ex:reviews, using the owl:inverseOf relationship, that enables bidirectional navigation in potential SPARQL queries and inference engines, facilitating the retrieval of both the reviewed papers for a given reviewer/authos and the reviewers of an individual paper. Lastly, to make potential citation chains possible and support reasoning over the "cites" relation in general without misrepresenting the semantics of direct citations, a new transitive property ex:hasIndirectCitation was introduced and declared as an owl:TransitiveProperty. This property is distinct from ex:cites, which reflects immediate and intentional citation. By using ex:hasIndirectCitation for inferred citation paths, the schema enables analysis of influential papers while maintaining of direct citation semantics. This design allows reasoning engines to compute indirect citations cleanly, without overloading the meaning of ex:cites.

B2.1. - ABox generation

General Set up

To construct the ABox from "plain" CSV data, a Python-based script was created, leveraging the rdflib library. The data was sourced from the from the first Projects existing structured CSV files representing authors, papers, venues, and citations, as generated in the first part of the project. As in this second part of the project not all attributes of the papers nor authors are needed, only a selection of rows was parsed and converted into RDF triples using URIs designed according to the TBox schema, created as the first step in this project. It was ensured that each instance is linked to the appropriate RDF class, entirely inferred automatically through reasoning, which will be further elaborated on in chapter B2.3. Textual attributes such as papers title, papers abstract, authors name are represented as xsd:string literals, while numeric fields like citationCount and paperCount are explicitly cast to xsd:integer. Temporal values, in this case only the publication year are typed using xsd:gYear to support potential time-based queries, like paper per year and venue or similar analysis.

Data Preprocessing

To make sure that only clean and consistent data is used for RDF conversion, several preprocessing steps were applied before or while parsing trhough the CSV files. For numeric fields such as citationCount, paperCount, and hIndex, a utility function safe_int(value) was used to convert values to integers, replacing placeholders like "Unknown" with a default value of 0. This avoids ValueError exceptions during the creation of typed literals with XSD.integer. String-based identifiers, such as volume numbers and city names, were normalized using regular expressions. Volume strings were stripped of blancspaces and partially pimped by replacing spaces with dashes to make them URI-compatible. Similarly, city names had to be reduced to alphanumeric characters only, to ensure valid RDF URIs for use as instances of the class ex:City. In the case of multi-value fields in the CSVs, like otherAuthors or reviewers in the papers.csv, entries are originally stored as semicolon-separated strings. These were split using Python's str.split(";") function, and each individual value was transformed into a RDF proof resource URI using the

namespace pattern EX[f"author/author_id"]. Each resulting author URI was assigned a type (e.g., ex:Author) and then appropriately linked to the corresponding paper using predicates such as ex:hasAuthor or ex:reviewedBy.

Venues Hierarchy and Reviewers

To fully leverage the semantic power of the resulting knowledge graph, the representation of venues and publication types (such as journals, conferences, and workshops) was reorganized into a structured class hierarchy. This layered design allows for more flexible queries and can make great use of inferences. Publications are split semantically into two overarching categories: recurring venues and specific instances of those venues. Journals are modeled as recurring venues (ex:Journal) and contain individual volumes (ex:Volume), which represent publication units within a specific year. Each ex:Volume is treated as a subclass of ex:Publication and is linked to its journal via ex:hasVolume. Papers published in a journal volume are connected using the ex:publishedIn relation. Conferences and workshops are represented as recurring event venues (ex:Conference, ex:Workshop) and are also subclasses of ex:Venue. Each event has editions (ex:Edition), which are temporally specific instances linked to a particular city (ex:heldIn) and year (ex:heldInYear). Each ex:Edition is linked back to its venue via ex:isEditionOf and is associated with a set of proceedings (ex:Publication) through ex:hasProceedings. Papers presented at the event are linked to these proceedings using ex:publishedIn. This modeling pattern ensures a rich and inferable structure. For instance, if a paper is published in a proceedings node, its venue type (conference or workshop) can be deduced via reasoning over ex:isEditionOf and its superclass. Where supported by the GraphDBs reasoning engine, explicit rdf:type assertions were minimized whenever they could be derived through inference, as explained in the next part of this report. A volume is not explicitly typed as ex:Publication if the TBOX already defines ex:Volume rdfs:subClassOf ex:Publication

Constraints in the ABox

To ensure data quality and semantic correctness within the ABox, several validation constraints were implemented directly in the RDFLib script. First, a reviewer exclusion check was applied to guarantee that no author could act as a reviewer of their own paper. This reflects a realistic peer-review process and enforces the integrity of the ex:reviewedBy relation. During parsing, all author IDs (corresponding and co-authors) were collected, and any overlap with assigned reviewer IDs triggered an error, thus preventing self-review assignments. Second, two content-based constraints were added to validate the presence of critical metadata: each paper was required to have a non-empty title and abstract. These checks ensured that all ex:title and ex:abstract literals represented meaningful textual content and avoided the inclusion of incomplete or malformed paper instances in the knowledge graph. These procedural constraints complement the semantic structure defined in the TBox and contribute to a robust and queryable knowledge graph. Additionally, the ABox generation script included a simplified citation chain expansion. For all citation chains of length two (e.g., Paper A cites B and B cites C), the property ex:hasIndirectCitation was asserted to reflect this indirect influence. This avoids marking ex:cites as transitive while still supporting inference over citation networks.

B2.3. - Inference explanation

The constructed knowledge graph supports the RDFS entailment regime, which enables inference over class hierarchies and property declarations through the use of rdfs:subClassOf, rdfs:domain, and rdfs:range. For example, linking a paper to an author via the ex:hasAuthor property allows a reasoner to infer that the subject is an instance of ex:Paper and the object is an instance of

ex:Author. Similarly, the use of ex:publishedIn enables the inference that a paper is linked to a Publication, and that specific instances such as ex:Volume or ex:Proceedings are subclasses of it. While many types can be inferred from property usage, selected rdf:type declarations were explicitly included in the ABox — not redundantly, but once per instance — for key entities such as ex:Author, ex:Paper, ex:Edition, ex:Volume, and the main venue subclasses (ex:Journal, ex:Conference, ex:Workshop). This approach ensures that essential classes are recognizable to reasoners and visible in tools like GraphDB, without inflating the ABox with unnecessary type assertions. Other types, such as ex:Reviewer, are inferred at query time via the range specification of ex:reviewedBy in the TBox and were therefore not explicitly typed. Additionally, a derived property ex:hasIndirectCitation was incorporated as an owl:TransitiveProperty to support reasoning over possible multi-step citation paths, while preserving the intended semantics of direct citations expressed via ex:cites. These indirect citation links are precomputed and materialized during ABox generation, making efficient querying of citation networks and papers influences possible. Overall, the graph combines great semantic richness without a too complex structure to support both inference and performance.

B2.4. Knowledgegraph Analysis

Originating from the limited amount of instances we retrieved in the first part of the Project from the SemanticScholarDB [2] (due to a missing key) the used instances in this part of the project are also limited. Nevertheless, the rather small amount is fully representing all necessary query results and showing the completeness and functionality of the created graph. As shown in the following table the existing instances provide a solid foundation to show the richness of the graph.

Table 1: Summary Statistics of the Knowledge Graph

Metric	**Value**
Number of Classes	18
Number of Properties	45
Number of `ex:Author` Instances	267
Number of `ex:Reviewer` Instances	142
Number of `ex:Paper` Instances	56
Number of `ex:Publication` Instances	111
Number of `ex:Journal` Instances	29
Number of `ex:Conference` Instances	19
Number of `ex:Workshop` Instances	5
Number of `ex:Edition` Instances	53
Number of `ex:Volume` Instances	37
Number of `ex:Venue` Instances	48
Number of `ex:City` Instances	12
Number of `ex:cites` Triples	2079
Number of `ex:hasIndirectCitation` Triples	655
Number of `ex:reviewedBy` Triples	350
Number of `ex:hasAuthor` Triples	272
Number of `ex:publishedIn` Triples	130
Number of `ex:heldInYear` Triples	76

Continued on next page

Metric	Value
Number of `ex:abstract` Triples	56
Number of `ex:name` Triples	315
Number of `ex:paperCount` Triples	267
Number of `ex:hIndex` Triples	216

B3 Queries

The first Query retrieves, for each recurring venue (conference, journal, or workshop) and publication year, the number of distinct authors who published papers there. It follows an indirect path from each paper to its venue by first resolving the ex:publishedIn property (linking a Paper to a Proceedings), then using ex:hasProceedings and ex:isEditionOf to identify the overarching Venue. Inference is used implicitly through subclass reasoning: since Conference, Journal, and Workshop are all subclasses of Venue, a reasoner can infer that a given venue entity is of type ex:Venue even if only the subclass is explicitly declared. This enables the query to use a generic ?venue a ?venueType pattern while filtering by specific types like ex:Conference. Such reasoning allows querying across hierarchical class structures without requiring redundant assertions in the ABox.

```
PREFIX ex: <http://example.org/research/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?venueName ?venueType ?year (COUNT(DISTINCT ?author) AS ?
   authorCount)
WHERE {
  {
    ?paper ex:hasCorrespondingAuthor ?author .
  }
  UNION
  {
    ?paper ex:hasCoAuthor ?author .
  }

  ?paper a ex:Paper ;
         ex:publishedIn ?proceedings ;
         ex:year ?year .

  ?edition ex:hasProceedings ?proceedings ;
           ex:isEditionOf ?venue .

  ?venue a ?venueType ;
         ex:name ?venueName .

  FILTER(?venueType IN (ex:Conference, ex:Workshop, ex:Journal))
}
GROUP BY ?venueName ?venueType ?year
ORDER BY ?venueName ?year
```

Listing 1: Count the authors that have published in a venue per year

An excerpt of the results is listed in the following table and a full version as a CSV (BDMA12L-KamaliLassem+Grigat-Query1_Result) in the project folder.

#	**Venue Name**	**Venue Type**	**Year**	**Author Count**
1	ACM Asia Conference on Computer and Communications Security	Conference	2019	6
2	Alberto Mendelzon Workshop on Foundations of Data Management	Workshop	2018	1
3	Annual Conference on Innovation and Technology in Computer Science Education	Conference	2021	4
4	Chinese Control and Decision Conference	Conference	2020	6
5	EDBT/ICDT Workshops	Workshop	2016	2
6	IEEE International Conference on Fuzzy Systems	Conference	2015	3
7	IEEE International Symposium on Parallel & Distributed Processing, Workshops and PhD Forum	Workshop	2019	7
8	International Conference on Advanced Cloud and Big Data	Conference	2017	9
9	International Conference on Automated Software Engineering	Conference	2014	2
10	International Conference on Blockchain	Conference	2020	6

Table 2: List of venues with year and author count

The second query identifies authors whose papers have a strong indirect influence on the research network and who also serve as reviewers for other papers, ensuring they are engaged in both publishing and evaluating work within the community. It makes extensive use of semantic inference based on the defined ontology. First, the property ex:hasIndirectCitation is declared as transitive, allowing the query to detect not only direct citations but also multi-step influence chains, which are grouped and counted per paper. Second, ex:hasCorrespondingAuthor and ex:reviewedBy are defined with clear domain and range semantics, enabling inference of the involved classes (e.g., that a subject of ex:hasCorrespondingAuthor is a Paper and its object is an Author). Third, ex:reviewedBy is defined as the inverse of ex:reviews, so the query can rely on that inverse relationship to retrieve all papers an author has reviewed. Additionally, the structure of the publication venues is navigated using subclass reasoning: for instance, Volume is a subclass of Publication, and papers published in conference proceedings are linked through Edition and Venue entities. This allows the query to generically retrieve and filter by publication type, whether journal, conference, or workshop, using rdf:type inference. Overall, the query demonstrates how a semantically enriched ontology enables complex, meaningful queries that would not be possible—or would be much harder to express—in a non-inferable graph model.

```
PREFIX ex: <http://example.org/research/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?authorName ?paperTitle ?indirectInfluence ?venueType
WHERE {
  ?paper ex:hasIndirectCitation ?citedPaper .
  ?paper ex:hasCorrespondingAuthor ?author .
  ?author ex:name ?authorName .
  ?paper ex:title ?paperTitle .

  ?reviewedPaper ex:reviewedBy ?author .

  FILTER(?paper != ?reviewedPaper)

  ?paper ex:publishedIn ?publication .

  {
    # For conference and workshop papers
    ?edition ex:hasProceedings ?publication .
    ?edition ex:isEditionOf ?venue .
    ?venue a ?venueType .
    FILTER(?venueType IN (ex:Conference, ex:Workshop))
  }
  UNION
  {
    ?publication a ex:Volume .
    ?journal ex:hasVolume ?publication .
    ?journal a ex:Journal .
    BIND(ex:Journal as ?venueType)
  }
  {
    SELECT ?paper (COUNT(?indirectlyCited) as ?indirectInfluence)
    WHERE {
      ?paper ex:hasIndirectCitation ?indirectlyCited .
    }
    GROUP BY ?paper
```

```
  }
  FILTER(?indirectInfluence > 5)
}
ORDER BY DESC(?indirectInfluence) ?authorName
```

Listing 2: Authors whose papers have a strong indirect influence and who also serve as reviewers for other work

The results are listed in the following table and as a CSV (BDMA12L-KamaliLassem+Grigat-Query2_Result) in the project folder.

#	**Author Name**	**Paper Title**	**Indirect Influence**	**Venue Type**
1	C. Cattuto	Time-varying social networks in a graph database: a Neo4j use case	63	Workshop
2	Ran Wang	An Empirical Study on Recent Graph Database Systems	39	Conference
3	Yingying Zheng	Finding bugs in Gremlin-based graph database systems via Randomized differential testing	28	Conference
4	Sakshi Srivastava	Fraud detection in the distributed graph database	25	Journal
5	B. Kan	Topology Modeling and Analysis of a Power Grid Network Using a Graph Database	20	Journal
6	Zhisong Fu	GeaBase: A High-Performance Distributed Graph Database for Industry-Scale Applications	14	Conference
7	Byoung-Ha Yoon	Use of Graph Database for the Integration of Heterogeneous Biological Data	12	Journal
8	Konstantinos Semertzidis	Time Traveling in Graphs using a Graph Database	9	Workshop

Table 3: Papers and their indirect influence by venue type

References

1 RDF-Grapher. *Linked Data Finland*. https://www.ldf.fi/service/rdf-grapher

2 Semantic Scholar Database. https://www.semanticscholar.org/