The dblp Knowledge Graph and SPARQL Endpoint

Marcel R. Ackermann
Schloss Dagstuhl – Leibniz Center for Informatics, dblp computer science bibliography, Trier, Germany

Hannah Bast
University of Freiburg, Department of Computer Science, Freiburg, Germany

Benedikt Maria Beckermann
Schloss Dagstuhl – Leibniz Center for Informatics, dblp computer science bibliography, Trier, Germany

Johannes Kalmbach
University of Freiburg, Department of Computer Science, Freiburg, Germany

Patrick Neises
Schloss Dagstuhl – Leibniz Center for Informatics, dblp computer science bibliography, Trier, Germany

Stefan Ollinger
Schloss Dagstuhl – Leibniz Center for Informatics, dblp computer science bibliography, Trier, Germany

Abstract

For more than 30 years, the dblp computer science bibliography has provided quality-checked and curated bibliographic metadata on major computer science journals, proceedings, and monographs. Its semantic content has been published as RDF or similar graph data by third parties in the past, but most of these resources have now disappeared from the web or are no longer actively synchronized with the latest dblp data. In this article, we introduce the *dblp Knowledge Graph (dblp KG)*, the first semantic representation of the dblp data that is designed and maintained by the dblp team. The dataset is augmented by citation data from the OpenCitations corpus. Open and FAIR access to the data is provided via daily updated RDF dumps, persistently archived monthly releases, a new public SPARQL endpoint with a powerful user interface, and a linked open data API. We also make it easy to self-host a replica of our SPARQL endpoint. We provide an introduction on how to work with the dblp KG and the added citation data using our SPARQL endpoint, with several example queries. Finally, we present the results of a small performance evaluation.

2012 ACM Subject Classification Information systems → Digital libraries and archives; Information systems → Graph-based database models; Computing methodologies → Knowledge representation and reasoning

Keywords and phrases dblp, Scholarly Knowledge Graph, Resource, RDF, SPARQL

Digital Object Identifier 10.4230/TGDK.2.2.3

Category Resource Paper

Supplementary Material All described data is available under the CC0 1.0 Universal license.[a]
Dataset (dblp KG, monthly snapshot): `https://doi.org/10.4230/dblp.rdf.ntriples` [15]
Dataset (dblp SPARQL query service, daily updated source triples): `https://sparql.dblp.org/download/`
Service (dblp SPARQL query service, user interface): `https://sparql.dblp.org/`
Service (dblp SPARQL query service, endpoint): `https://sparql.dblp.org/sparql`
Software (QLever SPARQL engine, source code): `https://github.com/ad-freiburg/qlever`
archived at `swh:1:dir:594019ed1ffe83fef266ac215ba69dff30185d72`
Text (dblp KG, ontology reference documentation): `https://dblp.org/rdf/docu/`

Funding *Benedikt Maria Beckermann*: DFG-LIS project Unknown Data (GEPRIS 460676019)
Stefan Ollinger: consortium NFDI4DataScience (GEPRIS 460234259); consortium NFDIxCS (GEPRIS 501930651)

Acknowledgements The dblp team would like to thank Silvio Peroni, Ralf Schenkel, and Tobias Zeimetz for the many fruitful discussions and practical help with the specification of the dblp RDF schema. We would also like to thank the many members of the dblp community who sent us their comments, thoughts, criticisms, and suggestions from working with the early versions of the dblp RDF data. Many thanks to Michael Wagner and Michael Didas from the Dagstuhl Publishing team for their support in creating a sustainable workflow for publishing and preserving persistent dblp RDF snapshot releases.

Received 2024-06-28 **Accepted** 2024-11-02 **Published** 2024-12-18

Editors Aidan Hogan, Ian Horrocks, Andreas Hotho, Lalana Kagal, and Uli Sattler

Special Issue Resources for Graph Data and Knowledge

[a] `https://creativecommons.org/publicdomain/zero/1.0/`

1 Introduction

The ever-increasing volume of academic research requires advanced methods for managing and accessing the wealth of information about scholarly publications. To harness the full potential of modern research information systems, it is essential to represent knowledge in a structured, interlinked, and semantically rich manner. Knowledge graphs [23] allow such a representation by providing structured and interlinked data and improving the ability to understand the interconnected nature of scholarly knowledge.

The *dblp computer science bibliography* is a comprehensive online reference for bibliographic information on important computer science publications. It was launched in 1993 by Michael Ley at the University of Trier and has developed from a small experimental website about databases and logic programming (hence, “dblp”) into a popular open data service for the entire computer science community [26]. As of June 2024, dblp indexes over 7.2 million publications written by more than 3.5 million authors. The database indexes more than 57,000 journal volumes, more than 58,000 conference and workshop proceedings, and more than 150,000 monographs.

Although the term “open data” had not yet been coined in 1993, dblp was open from the very beginning. Individual data entries have always been freely accessible, and complete dump downloads of the entire dblp data in its own custom XML format have been available since at least 2002 [27, 28]. Since 2015, dblp XML snapshots are archived as persistent monthly releases [16].

1.1 Related work

The idea of providing access to dblp data as linked open data is not new. In the very first iteration of the linked open data cloud from 2007 (see Figure 1), dblp was already linked as one of the few early data sources [3, 11]. However, these were always independent contributions from the international computer science community and not an original contribution of the dblp team. These earlier contributions were based on snapshots (current at the time) of the public XML dump export and were generally not updated after creation. Given the continuous additions and maintenance by the dblp team, which make dblp a “living” dataset, these conversions were quickly out of sync with the live data. Many of the external live services, in particular SPARQL endpoints, have vanished from the web in the meantime.

The probably earliest RDF conversion of the dblp dataset has been exercised as a benchmark example for the declarative mapping language D2RQ in 2004 [10]. The data was later released together with an accompanying SPARQL endpoint using the D2R Server tool [9].[1] This conversion provided entities for up to 800,000 articles and 400,000 authors and hasn’t been updated since 2007.

[1] `https://web.archive.org/web/*/http://www4.wiwiss.fu-berlin.de/dblp/` (archived)

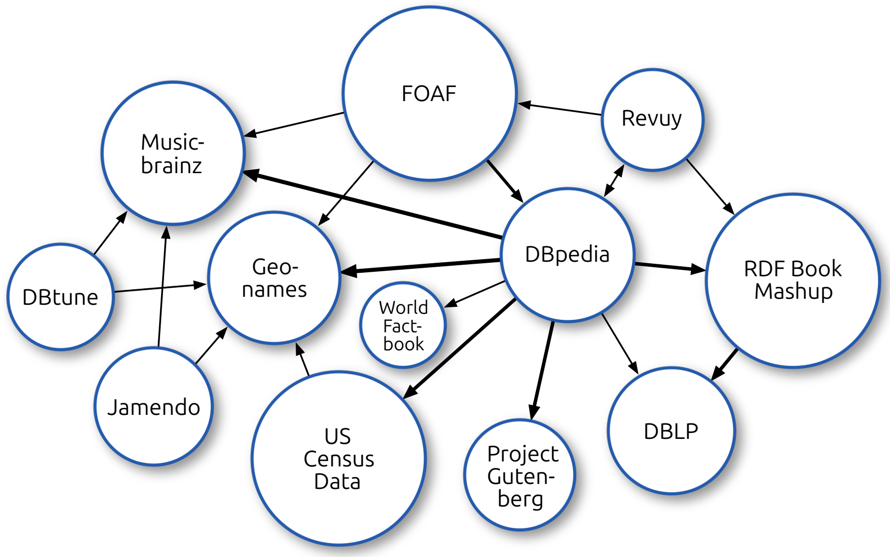


Figure 1 The first snapshot of the LOD cloud, from May 2007, according to `https://lod-cloud.net/`.

At about the same time, further conversions and SPARQL endpoints were the D2R Server in the context of the *Faceted DBLP*[2] search engine [17] and the *RKBExplorer*[3] [21, 20]. These RDF datasets have been actively used for a long time and are a subject or component of numerous computer science publications. E.g., a simple Internet search using Google Scholar with query `"fu-berlin.de/dblp" OR "dblp.l3s.de/d2r" OR "dblp.RKBExplorer.com"` finds at least 380 results,[4] with citing papers still being published today in 2024.

Other graph datasets used dblp data as a starting point to build improved and extended semantic information. For example, the *SwetoDblp* ontology and dataset[5] augmented dblp XML data with relationships to other entities such as publishers and affiliations [2, 1]. The *GraphDBLP* tool[6] models the dblp data from 2016 as a graph database and, in doing so, allows for performing graph-based queries and social network analyses [31, 30]. The *COLINDA* dataset[7] provided a linked data collection of 15,000 conference events, augmenting dblp proceedings data with location, start and end times, geodata and further links [39, 38]. More recently, the *EVENTSKG* knowledge graph[8] provided a semantic description of publications, submissions, start date, end date, location, and homepage for events of top-prestigious conference series in different computer science communities [18]. Semantically structured metadata on scientific events was later also made accessible via the *ConfIDent* platform[9] [22, 19].

Independently of dblp and beyond the discipline of computer science, several international efforts have been launched in recent years to provide open scientific knowledge graphs. *Wikidata*[10] is the collaborative, omnithematic, and multilingual knowledge graph hosted by the Wikimedia

[2] `https://web.archive.org/web/*/http://dblp.l3s.de/d2r/` (archived)
[3] `https://web.archive.org/web/*/dblp.RKBExplorer.com` (archived)
[4] Accessed on 2024-06-12.
[5] `https://web.archive.org/web/*/http://knoesis.wright.edu/library/ontologies/swetodblp` (archived)
[6] `https://github.com/fabiomercorio/GraphDBLP`
[7] `https://web.archive.org/web/*/http://www.colinda.org/` (archived)
[8] `http://w3id.org/EVENTSKG-Dataset/ekg`
[9] `https://www.confident-conference.org/`
[10] `https://www.wikidata.org`

Foundation [40]. Within Wikidata, the *WikiCite* project[11] aims to create an open, collaborative repository of bibliographic data. *OpenCitations*[12] maintains and publishes open citation data as linked open data, thereby providing the first truly open alternative to proprietary citation indexes [35]. Initially an outcome of the EU Horizon 2020, the *OpenAIRE Graph*[13] was one of the first comprehensive research knowledge graphs [29]. Since then, OpenAIRE has consolidated its organizational structure and the OpenAIRE Graph is now the authoritative source for the European Open Science Cloud (EOSC).[14] *OpenAlex*[15] is a recent open infrastructure service, built on the data of the now abandoned Microsoft Academic Graph[16]. OpenAlex is a massive, cross-disciplinary research knowledge graph of publications, authors, venues, institutions, and concepts [36]. Furthermore, the *Open Research Knowledge Graph (ORKG)* aims to make scientific knowledge fully human- and machine-actionable by describing research contributions in a structured manner, e.g., by connecting research papers, datasets, and used methods [25]. The ORKG aims to build a community of contributors in order to collect, curate, and organize descriptions of scientific contributions in a crowd-sourcing manner.

1.2 Our contribution

In this article, we introduce the *dblp Knowledge Graph (dblp KG)*. The dblp KG aims to make all semantic relationships modeled in the dblp computer science bibliography explicit. In contrast to previous approaches, the dblp KG is not merely based on a one-time snapshot of dblp data, but is actively synchronized with the current data. In particular, the dblp KG thus benefits from the continuing curation work of the dblp team.

The dblp KG aims to complement other open knowledge graphs by bringing in dblp's unique strengths in author disambiguation, semantic enrichment of bibliographies, and its role as a directory of computer science journals and conferences. We demonstrate this by augmenting our dataset with identifiers and citation data from the OpenCitations corpus.

Open and FAIR access to the data is provided via daily updated RDF dumps, persistently archived monthly releases, a new public SPARQL endpoint with a powerful user interface, and a linked open data API. We also make it easy to self-host a replica of our SPARQL endpoint.

The rest of this article is organized as follows. In Section 2, we introduce the ontology of the dblp KG and present statistics about the graph. Section 3 describes the different ways to access the dblp KG and the citation data. Section 4 provides an introduction on how to work with the dblp KG and the added citation data using our SPARQL endpoint, with several example queries, and complemented by a small performance evaluation. Section 5 concludes the article with a short discussion and an outlook.

2 dblp as a Knowledge Graph

Since its earliest stages, the semantic organization of bibliographic metadata has been a primary concern of the dblp team's editorial work. This includes linking publications with their true authors, research papers with their proceedings, and conference events with the history of their conference series. The dblp team puts a lot of (often manual) effort into providing such information

[11] https://www.wikidata.org/wiki/Wikidata:WikiCite
[12] https://opencitations.net
[13] https://graph.openaire.eu
[14] https://eosc-portal.eu/
[15] https://openalex.org/
[16] https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/

as accurately, completely, and up-to-date as possible. Editors manually annotate individual entries with further metadata, like alternative names, external identifiers, or links to relevant web resources. However, most semantic relations have only been provided implicitly on the (once manually crafted) dblp HTML webpages, and so far have not been made explicit in a machine-friendly way.

The dblp Knowledge Graph (dblp KG) aims to make these semantic relations explicit and machine-actionable. This includes structured information already available via APIs, like the authorship of publications [28], as well as information that has not been published explicitly before. In the current, second major iteration of the knowledge graph, this additional information includes the concrete linkage of published works with the publication venue they appeared in (conferences, journals, etc.), metadata about these venues, and information about known relations between venues. Future iterations of the graph will expand the model even further. This will include such information as metadata about conference events within a conference series, and author affiliations. A simplified excerpt of the current graph is shown in Figure 2.

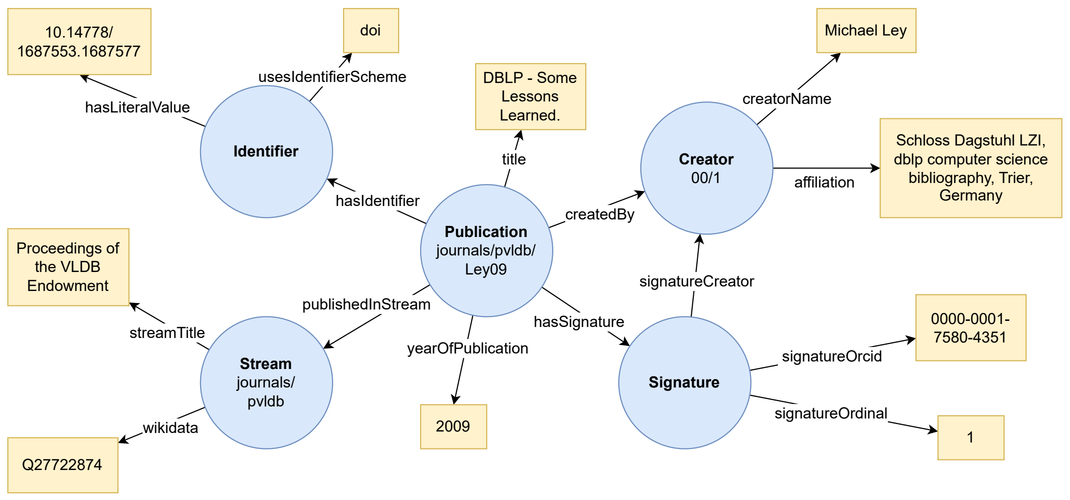


Figure 2 Simplified excerpt from the dblp knowledge graph. The excerpt is centered on the paper "DBLP – Some Lessons Learned" from Michael Ley [28].

2.1 The dblp ontology

As there already is a whole range of ontologies that model bibliographic information about scientific works (e.g., see [14, 32, 24]), the dblp ontology is explicitly not intended to replace them. Instead, it is designed to model the way dblp handles and provides bibliographic metadata, including all possible quirks and oddities that may arise from dblp's unique approach. For example, dblp's author disambiguation uses certain "pseudo-author entities" (described in more detail in Section 2.1.1.3 below) to model cases where the true authorship of a work is currently unknown or ambiguous. Also, dblp's records are incompatible with the more fine-grained FRBR model [34] that is standard in the library community.[17] Therefore, it was not viable to reuse existing ontologies, as is usually recommended. However, links to related types and predicates from existing ontologies are provided in the dblp RDF schema whenever possible.

[17] In particular, the dblp data model generally does not distinguish between the FRBR layers *Work* and *Expression* and does not address the layers *Manifestation* or *Item* at all.

In the remainder of this section, *entity* refers to any resource accessible via an IRI, a literal, or an anonymous node. It is synonymous with the term *resource* as defined in [12]. Entities in the dblp ontology are assigned certain core and reification *types* that stem from the dblp internal data model, and relations between entities are modeled using *predicates*. The types of the dblp ontology and their connections are described below, and a simplified view of the ontology is presented in Figure 3. The full dblp ontology reference documentation can be found at `https://dblp.org/rdf/docu/`.

Table 1 IRI prefixes for the core entities of the dblp KG.

Type	IRI Prefix
Publication	`https://dblp.org/rec/`
Creator	`https://dblp.org/pid/`
Stream	`https://dblp.org/streams/`

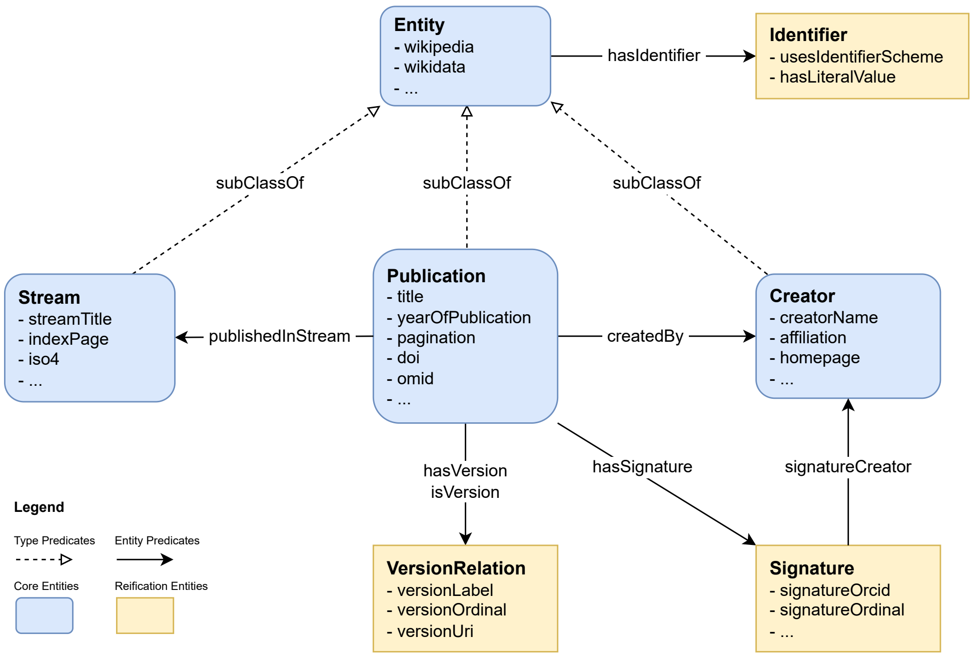


Figure 3 Excerpt from the dblp ontology showing the relationships between core and reification types. Each of the entity boxes shows the type at the top, followed by a list of predicates of entities of that type. The figure shows only a small selection of all predicates and omits most of the finer-grained subtypes.

2.1.1 Core entities

The current iteration of the dblp ontology contains named entities for publications, their creators, and the publication venues (which we call streams) they appeared in. These core entities have persistent IRIs and are accessible via open data APIs and as HTML web pages within the dblp website. For an overview of the IRI prefixes of the different types, see Table 1.

2.1.1.1 Entities

The dblp ontology defines an abstract supertype `dblp:Entity` as a parent to all core entity types. In the dblp ontology, this type represents any core entity that can be associated with an identifier. The main purpose of this abstract supertype is to provide a common `rdfs:range` subject for predicates in the dblp RDF schema.

2.1.1.2 Publications

Entities of type `dblp:Publication` represent any academic work indexed in dblp. This includes traditionally published articles, authored or edited volumes, and (more recently) also published data artifacts.

Like on the dblp websites, publications are linked to their authors or editors (modeled as `dblp:Creator` entities, see Section 2.1.1.3). This is done redundantly in two ways. First, there is a direct link towards authors and editors using the predicate `dblp:createdBy`. Additionally, special reification entities called *signatures* (of type `dblp:Signature`, see Section 2.1.2) are provided using the predicate `dblp:hasSignature`. This redundancy enables convenient and elegant queries via the first option when nothing other than the link from a publication to its creators is needed, and can provide more in-depth metadata about the authorship via the signature entities if required.

Publications are also linked to their publication venue (of type `dblp:Stream`, see Section 2.1.1.4), such as the conference or journal in which they are published. The link is modeled via the predicate `dblp:publishedInStream`. There are no dedicated reification entities for these links in the current iteration of the dblp ontology. Related metadata, such as issue or volume numbers, is given as literal values via predicates on the publications. In the future, reification entities might be introduced here.

To provide external identifiers, publications are linked to identifier entities (`datacite:Identifier`, see Section 2.1.2) using the `datacite:hasIdentifier` predicate. Redundantly and for convenience, links to the IRIs of the most important external identifiers are provided via direct predicates, namely DOIs (`dblp:doi`), ISBN (`dblp:isbn`), Wikidata entity (`dblp:wikidata`), and OpenCitations Meta IDs (`dblp:omid`).

Publication entities also carry further metadata fields, such as their titles, the year of publication, or pagination information.

All publications in dblp are classified by a rudimentary system of publication types. Similar to many modeling decisions made at dblp in the early days, these types were originally derived from classic BibTeX, but have evolved. Publication types are modeled as subtypes of `dblp:Publication`, like `dblp:Inproceedings` for conference publications, `dblp:Book` for monographs, or `dblp:Data` for research data and artifacts. A list of all types can be found in Table 2. We are aware that due to the evolving publication landscape, a BibTeX-inherited classification might no longer be a best fit for modern publication practices, and many of the decisions behind the dblp type classification system are disputable.

2.1.1.3 Creators

The type `dblp:Creator` represents any individual or group listed as the author or editor of a publication. Analogous to the case of `dblp:Publication` entities, creators are linked to their publications redundantly in two ways: First, they are linked directly via predicates (such as `dblp:creatorOf`) and indirectly via `dblp:Signature` reification entities.

Creator entities also carry metadata such as their names, alternative names, current and former affiliations, and homepages. Manually curated identifiers are provided via identifier entities (`datacite:Identifier`) linked using the `datacite:hasIdentifier` predicate. For convenience, ORCID IRIs (`dblp:orcid`) and Wikidata IRIs (`dblp:wikidata`) are provided via direct predicates.

Table 2 List of all publication types within the dblp ontology.

Publication Type	Description
Inproceedings	Conference and workshop publications
Article	Journal articles
Book	Monographs and PhD theses
Editorship	Edited volumes, prefaces, and editorials
Incollection	Chapters within a monograph
Reference	Reference material and encyclopedia entries
Data	Research data and artifacts
Informal	Preprints, non-peer-reviewed and other publications
Withdrawn	Withdrawn publications

The standard creator subtype (c.f. Table 3) used for individual authors or editors is `dblp:Person`. In some cases, where a listed author of a publication is not a single person but represents a known group or consortium, the type `dblp:Group` is used.

One major contribution of the dblp team is the continuous work to identify and disambiguate the "true authors" behind the plain character strings given in bibliographic metadata. This work often leaves a fair number of disambiguation cases unresolved as the information at hand does not allow for a reliable decision. These situations are handled by introducing certain pseudo-persons that represent more than one individual and are fully known to be ambiguous. Publications assigned to such a pseudo-person are known to have their true author not yet determined, and the collected bibliography of such a pseudo-person is known to not represent the coherent scholarly work of an actual person.

For example, assume we have several publications written by people called "Jane Doe". Further, assume that we know for some of those publications that they are written by two different individuals, called "Jane Doe 0001" and "Jane Doe 0002" (following dblp's scheme to distinguish different individuals with the same name). These two individuals will be modeled using the subtype `dblp:Person`. However, for the remaining "Jane Doe" publications, the true authorship is currently unknown. In that case, the remaining publications will be linked to neither "Jane Doe 0001" nor "Jane Doe 0002", but to a different pseudo-person "Jane Doe" of type `dblp:AmbiguousCreator`.

For all purposes, `dblp:AmbiguousCreator` entities are used and referenced just like normal, unambiguous creator entities in dblp, and they are linked to publications, signatures, etc. in the usual way. However, when retrieved in complex queries, their ambiguous nature should be understood and results should be handled accordingly. E.g., if an `dblp:AmbiguousCreator` is retrieved as a common coauthor, there is no guarantee that this is really the same person linking both authors. `dblp:AmbiguousCreator` entities do provide several unique predicates, like `dblp:possibleActualCreator` and `dblp:proxyAmbiguousCreator` that link between ambiguous and actual creators that may be related.

Please be aware that due to the continuous curation work done by the dblp editorial team, long-standing disambiguation cases can be (and are regularly) resolved at any time. Hence, `dblp:proxyAmbiguousCreator` entities and their links to publications and signatures are among the most volatile content of the dblp Knowledge Graph.

2.1.1.4 Streams

In dblp, we use the term *stream* to refer to any journal, conference series, book series, or repository that acts as a regular source for publications. Such streams are modeled using the type `dblp:Stream`. Publications are linked to the streams they appeared in using the predi-

Table 3 List of all creator types within the dblp ontology.

Creator Type	Description
Person	An individual person
Group	A group or organisation
AmbiguousCreator	An unknown number of unidentified individuals of the same name

cate `dblp:publishedInStream`. A single publication might be linked to multiple streams in that way. For example, an HCI conference paper might appear both in the stream of its conference event series `<https://dblp.org/streams/conf/hci>` as well as, say, the LNCS book series `<https://dblp.org/streams/series/lncs>` that publishes the conference proceedings.

`dblp:Stream` entities may be linked to other `dblp:Stream` entities using `dblp:relatedStream` or one of its subpredicates. These include hierarchical relations (`dblp:subStream` and `dblp:superStream`) in cases of streams that take place or are published as part of another stream, and temporal relations (`dblp:predecessorStream` and `dblp:successorStream`) in cases where streams merge with or are replaced by another stream.

Stream entities have further metadata like their (past, current, or alternative) titles, homepage URLs, a URL of their dblp index page, or their ISO4 journal title abbreviation. Identifiers are again provided via identifier entities (`datacite:Identifier`) linked using the `datacite:hasIdentifier` predicate. For convenience, ISSN IRIs (`dblp:issn`) and Wikidata IRIs (`dblp:wikidata`) are provided via direct predicates.

The dblp ontology uses the following subtypes of `dblp:Stream` (see Table 4): `dblp:Journal` for periodically published journals, `dblp:Conference` for conference or workshop series, and `dblp:Series` for series of published volumes like monographs and proceedings. Only very recently, we expanded the dblp data model also to include a fourth, new subtype `dblp:Repository` for sources of research data and artifacts, such as Zenodo.[18]

Table 4 List of all stream types within the dblp ontology.

Stream Type	Description
Journal	Periodically published journals
Conference	Conference or workshop series
Series	Series of monographs or proceedings volumes
Repository	Sources of research data and artifacts

2.1.2 Reification entities

Reification entities, also known as linking entities, link core entities to other core or external entities and provide further metadata about that link. Reification entities are represented by blank nodes in the dblp Knowledge Graph.

Identifiers. Identifier entities (of type `datacite:Identifier`) are used to annotate identifiable entities in the dblp KG with external identifiers, such as DOI or ORCID. These external identifiers allow users to connect the dblp KG entities with information from other knowledge graphs. The type `datacite:Identifier` is reused from and defined in the DataCite Ontology [37].

[18] `https://zenodo.org/`

A dblp KG core entity is linked to identifier entities using the predicate `datacite:hasIdentifier`. Identifier entities are linked to their identifier schema (such as `datacite:doi`) via the predicate `datacite:usesIdentifierScheme`. They are linked to their literal value stating the actual ID string via the predicate (`litre:hasLiteralValue`). It is important to note that identifier entities do not link directly to the IRIs of the external identifier to support a wider range of identifier schemas.

Signatures. In dblp, we use the term *signature* to refer to the reification entity (of type `dblp:Signature`) that links a publication (of type `dblp:Publication`) to one of its authors (of type `dblp:Creator`). The purpose of these entities is to provide more context to this otherwise simple link. Signature entities may link to an ORCID IRI that has been stated in the publication's metadata using the `dblp:signatureOrcid` predicate and provide the relative position of a publication's creator in the complete creator list using the `dblp:signatureOrdinal` predicate. We aim to provide additional context via the signature entities in future iterations of the dblp KG, such as affiliation information provided in the publication.

To distinguish between the roles of an editor and an author for a published work, the two subtypes `dblp:AuthorSignature` and `dblp:EditorSignature` are used.

Version Relations. With the recent inclusion of the publication type `dblp:Data`, an optional hierarchy between publications has been introduced to dblp to model cases where one publication is an instance of another publication. In dblp, we call the instanced publication a *version*, while the instantiated publication is called a *concept*. These relations are represented by the type `dblp:VersionRelation`. Publications are linked to version-relation entities via the predicates `dblp:versionConcept` and `dblp:versionInstance`. For convenience, the redundant predicate `dblp:isVersionOf` is provided to directly link between the `dblp:Publication` entities in cases when nothing other than this link is needed.

The relation contains further metadata like a label for the instance (such as "Version 1.3"), their relative order compared to other instances of the same concept, and an identifying IRI of the concept.

2.2 Key statistics

This section presents several key statistics of the dblp Knowledge Graph to give an overview of its content and dimension. Table 5 shows the number of entities per type, and Table 6 shows the number of identifiers by schema. The SPARQL queries used to create the statistics can be found online[19], each with a direct link to our SPARQL endpoint that will then execute the corresponding query. The statistics in this paper are based on the dump from September 11th, 2024.

The majority of publication entities are conference proceedings papers (47.43%) and journal articles (39.22%). Informal publications (9.25%), such as preprint publications on arXiv, also form a significant share of publications. The other publication types each form less than 3% of the corpus.

Less than 0.01% of all creator entities are of type `dblp:Group` because we aim to present individual authorship where possible. Individual authors of type `dblp:Person` form the vast majority (99.58%) of creator entities. The manually identified `dblp:AmbiguousCreator` entities only form a small fraction (0.41%).

In contrast to many other research fields, where journals are the predominant medium, conferences play a crucial role in disseminating research in computer science. Many conferences only take place once or twice, while journals tend to be much more long-lived. In addition,

[19] `https://github.com/dblp/kg/wiki/Paper-TGDK-2024#statistics-queries`

Table 5 Number of entities in the dblp KG by type, as of September 11th, 2024.

(a) Count of the main dblp type entities.

Type	Count	%
Signature	24,559,211	41.09
Identifier	24,157,894	40.42
Publication	7,446,698	12.46
Creator	3,595,686	6.02
Stream	8,864	0.01
VersionRelation	899	< 0.01

(b) Count of the publication subtype entities.

Publication Type	Count	%
Inproceedings	3,532,088	47.43
Article	2,920,903	39.22
Informal	689,075	9.25
Book	154,185	2.07
Editorship	61,847	0.83
Incollection	43,220	0.58
Reference	27,366	0.37
Withdrawn	10,425	0.14
Data	7589	0.10

(c) Count of the creator subtype entities.

Creator Type	Count	%
Person	3,580,548	99.58
AmbiguousCreator	14,774	0.41
Group	364	0.01

(d) Count of the stream subtype entities.

Stream Type	Count	%
Conference	6,753	76.18
Journal	1,885	21.27
Series	220	2.48
Repository	6	0.07

conferences are often divided into workshop series which may also form their own entities. All this explains why there are three times as many conference entities (76.18%) as journal entities (21.27%) in the dblp KG. Currently, there are only 6 repository streams in the dblp KG because the support for data publications, which are modeled to be published in repositories, has only recently been added to dblp.

The almost 6 million DOIs are the most often occurring identifier in the dblp KG. In addition to about 170,000 distinct ORCID IRIs manually linked to creators using `dblp:orcid` in the dblp KG, we also link to more than one million distinct ORCID IRIs on signatures via `dblp:signatureOrcid`. Other than ORCIDs linked to creator entities, ORCIDs linked to signatures are automatically harvested from metadata and have not been manually verified by the dblp team.

3 How to access the dblp Knowledge Graph and the citation data

We provide a variety of ways to access the dblp Knowledge Graph and the associated citation data: via RDF dumps, via a public SPARQL endpoint with an associated user interface, via SPARQL queries embedded into the dblp website, via a linked open data API, and by providing an easy way to set up one's own SPARQL endpoint. Each of these is briefly described in one of the following subsections. All data is released openly under the CC0 1.0 Universal license.[20]

[20] `https://creativecommons.org/publicdomain/zero/1.0/`

Table 6 Count of external identifier entities in the dblp KG by type, as of September 11th, 2024.

Identifier Type	Count	%
doi	6,205,594	47.35
omid	5,312,165	40.53
wikidata	685,613	5.23
arxiv	409,965	3.14
orcid	167,117	1.28
isbn	90,777	0.69
handle	44,665	0.34
dnb	41,713	0.32
google scholar	30712	0.23
urn	24386	0.19
ieee	14876	0.11
gnd	13387	0.10
zbmath	12215	0.09
acm	11486	0.09
loc	10266	0.08
math genealogy	9938	0.08
linkedin	5764	0.04
twitter	4162	0.03
issn	3567	0.03
research gate	2356	0.02
github	2157	0.02
isni	1412	0.01
viaf	1071	< 0.01
oclc	448	< 0.01
lattes	442	< 0.01
repec	244	< 0.01
gepris	128	< 0.01
openalex	10	< 0.01
gitlab	10	< 0.01

3.1 RDF dumps of the dblp Knowledge Graph

We provide daily updated RDF exports of the dblp KG in RDF/XML, N-Triples, and Turtle formats.[21] These are useful for tools and services that need the latest version of the data. Further, we publish persistent monthly releases of the dblp KG in N-Triples format [15] and recommend using these persistent releases for reproducible experiments and similar purposes. We also provide serializations of the RDF schema of the dblp KG ontology described in Section 2.1. This schema is rarely changed, and all previous versions of the schema are persistently available.

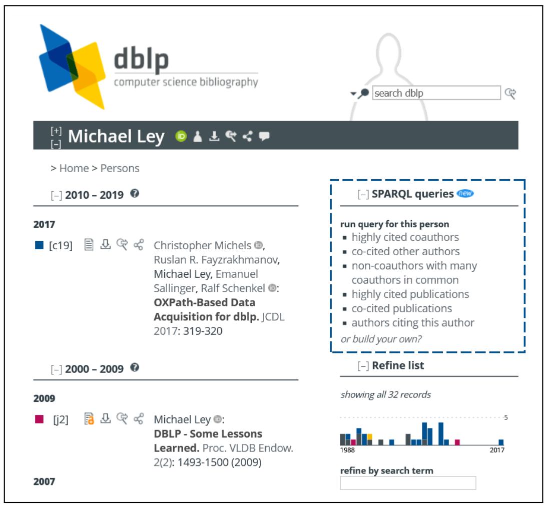
3.2 RDF dumps of the dblp KG with citation data

We also provide daily updated dblp RDF exports augmented with citation data[22], obtained from the OpenCitations project, which provides open-access citation data for publications across all areas of science [13]. OpenCitations assigns each publication an identifier (called OMID), which is also provided in the dblp KG; see Section 2.1.1. We filter the whole OpenCitations corpus to the subset of citations that are concerned with publications listed in dblp and provide the combined graph. The connection between dblp and OpenCitation entities is done via the predicate `dblp:omid` (see Section 2.1.1.2). Please be aware that while the OpenCitations data is updated only infrequently, the dblp KG is updated daily. Thus, the combined dataset is also updated daily.

3.3 Public SPARQL endpoint with associated user interface

We provide a public SPARQL endpoint for the combined data described in Section 3.2 under `https://sparql.dblp.org/sparql`, powered by the QLever SPARQL engine [6] [8].[23] The SPARQL endpoint is updated daily, in sync with the daily releases of these datasets. The endpoint

[21] `https://dblp.org/rdf/`
[22] `https://sparql.dblp.org/download/`
[23] `https://github.com/ad-freiburg/qlever`


dblp
computer science bibliography

search dblp

[+] [–] Michael Ley

> Home > Persons

[–] 2010 – 2019

2017

[c19] Christopher Michels, Ruslan R. Fayzrakhmanov, Michael Ley, Emanuel Sallinger, Ralf Schenkel: **OXPath-Based Data Acquisition for dblp.** JCDL 2017: 319-320

[–] 2000 – 2009

2009

[j2] Michael Ley: **DBLP - Some Lessons Learned.** Proc. VLDB Endow. 2(2): 1493-1500 (2009)

2007

[–] SPARQL queries new

run query for this person
- highly cited coauthors
- co-cited other authors
- non-coauthors with many coauthors in common
- highly cited publications
- co-cited publications
- authors citing this author

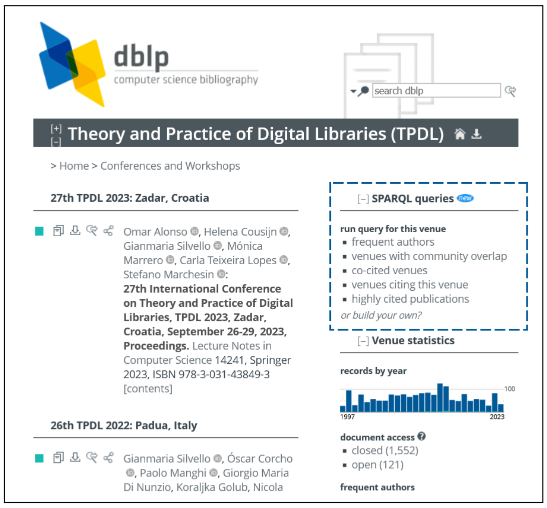
or build your own?

[–] Refine list

showing all 32 records

refine by search term

(a) Example of person queries.


dblp
computer science bibliography

search dblp

[+] [–] Theory and Practice of Digital Libraries (TPDL)

> Home > Conferences and Workshops

27th TPDL 2023: Zadar, Croatia

Omar Alonso, Helena Cousijn, Gianmaria Silvello, Mónica Marrero, Carla Teixeira Lopes, Stefano Marchesin: **27th International Conference on Theory and Practice of Digital Libraries, TPDL 2023, Zadar, Croatia, September 26-29, 2023, Proceedings.** Lecture Notes in Computer Science 14241, Springer 2023, ISBN 978-3-031-43849-3 [contents]

26th TPDL 2022: Padua, Italy

Gianmaria Silvello, Óscar Corcho, Paolo Manghi, Giorgio Maria Di Nunzio, Koraljka Golub, Nicola

[–] SPARQL queries new

run query for this venue
- frequent authors
- venues with community overlap
- co-cited venues
- venues citing this venue
- highly cited publications

or build your own?

[–] Venue statistics

records by year

document access
- closed (1,552)
- open (121)

frequent authors

(b) Example of venue queries.

Figure 4 Example SPARQL queries embedded into the dblp website.

conforms with the SPARQL 1.1 Protocol[24], currently still with minor deviations. These are documented in the dblp KG Wiki[25] and will be fixed in the near future. The SPARQL endpoint makes it very easy to build services or tools on top of the dblp KG. Section 4.6 briefly analyzes its performance.

The endpoint also comes with a user interface, available under `https://sparql.dblp.org`, for the interactive formulation and execution of SPARQL queries. The user interface provides various features to help people that are unfamiliar with the intricacies of the dataset or of the SPARQL query language, most notably context-sensitive suggestions (autocompletion) after each keystroke. This is explained in more detail in Section 4.2, for the example query from Section 4.1. The user interface also features a button, which opens a searchable list of example queries.

3.4 Setting up your own SPARQL endpoint

The SPARQL endpoint described in the previous section is publicly and freely available. To enable a continuous service, there is a fixed timeout for each query, and at some point, we might also introduce rate limits or quotas. For users with high query volumes or other special requirements, we provide instructions for setting up their own SPARQL endpoint and user interface,[26] with the exact same functionality as described in Section 3.3, using the exact same dataset. This setup requires only a few commands and works with standard hardware; see Section 4.6.

3.5 SPARQL queries embedded into the dblp website

Beyond the examples of the query interface, we also provide links with preformulated SPARQL queries embedded into various pages across the dblp website, as shown in Figure 4. In particular, each author page now features a box with links to related SPARQL queries, such as a query to calculate the most highly cited co-authors of the author described on the page. Similarly, each

[24] `https://www.w3.org/TR/sparql11-protocol/`
[25] `https://github.com/dblp/kg/wiki/Known-Issues`
[26] `https://github.com/dblp/kg/wiki/SPARQL-server-setup`

Table 7 Recognized file extensions and MIME types.

Format	API file extension	MIME type for content negotiation
RDF/XML	.rdf	application/rdf+xml
N-Triples	.nt	application/n-triples
Turtle	.ttl	text/turtle
HTML	.html	text/html
dblp XML	.xml	application/xml

conference or journal page features a box with links to related SPARQL queries, such as a query to calculate frequent authors for this conference, or conferences with a large overlap regarding authors. These embedded queries are useful in three respects: (1) they provide interesting information that complements the information provided on the respective page, (2) they draw attention to the SPARQL endpoint for users that might otherwise miss this opportunity, and (3) it is an easy and motivating way to learn by example what is in the dblp KG and how to query it.

3.6 Linked Open Data API

Finally, we provide an API for individual pieces of RDF data for creators, publications, and streams. This data is guaranteed to always be up-to-date with the current state of the dblp database. The URLs of this API all follow the same structure: a dblp resource IRI, followed by a file extension corresponding to the requested file format as given in Table 7. For example, for the creator resource IRI `https://dblp.org/pid/71/4882` there is

`https://dblp.org/pid/71/4882.rdf` for retrieving RDF/XML,
`https://dblp.org/pid/71/4882.nt` for retrieving N-Triples,
`https://dblp.org/pid/71/4882.ttl` for retrieving Turtle,
`https://dblp.org/pid/71/4882.html` for the dblp HTML website.

Similarly, for publications, there is

`https://dblp.org/rec/conf/semweb/AuerBKLCI07.rdf` for retrieving RDF/XML,
`https://dblp.org/rec/conf/semweb/AuerBKLCI07.nt` for retrieving N-Triples,
`https://dblp.org/rec/conf/semweb/AuerBKLCI07.ttl` for retrieving Turtle
`https://dblp.org/rec/conf/semweb/AuerBKLCI07.html` for the dblp HTML website,

and for stream entities, there is

`https://dblp.org/streams/conf/semweb.rdf` for retrieving RDF/XML,
`https://dblp.org/streams/conf/semweb.nt` for retrieving N-Triples,
`https://dblp.org/streams/conf/semweb.ttl` for retrieving Turtle,
`https://dblp.org/streams/conf/semweb.html` for the dblp HTML website.

4 SPARQL queries and performance

This section provides an introduction on how to work with the dblp KG and the added citation data using the SPARQL endpoint we provide. We provide four example queries: a basic one (Section 4.1), a more advanced one (Section 4.3), a federated query that queries two endpoints (Section 4.4), and a query that uses both the dblp KG and the added citation data (Section 4.5). We use the basic example query to explain how the autocompletion works (Section 4.2). The section closes with a brief performance evaluation of a selection of SPARQL queries (Section 4.6). All queries discussed in this section can also be found on the web[27], each with a direct link to our SPARQL endpoint that will then execute the corresponding query.

4.1 A basic SPARQL query

SPARQL is the standard query language for querying RDF data. The language can be seen as a variant of SQL (the standard query language for relational databases), adapted to the RDF data model. Namely, just like RDF data is a set of triples, the core of a typical SPARQL query is a set of triples, with variables in some places. Following is an example query that asks for the titles of all papers in dblp published until 1940, their authors, and the year of publication:

```
PREFIX dblp: <https://dblp.org/rdf/schema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?title ?author ?year WHERE {
  ?paper dblp:title ?title .
  ?paper dblp:authoredBy ?author_id .
  ?author_id rdfs:label ?author .
  ?paper dblp:yearOfPublication ?year .
  FILTER (?year <= "1940"^^xsd:gYear)
}
ORDER BY DESC(?year) ASC(?title)
```

Run this query

Conceptually, the result of a SPARQL query is a table. For the query above, that table has three columns (labeled `?title`, `?author`, and `?year`), and one row for each possible assignment to these three variables such that all the corresponding triples in the query body exist and all the additional constraints (in this case, the `FILTER` condition) are fulfilled. For example, the first five result rows for the query above are as follows:

?title	?author	?year
A Correction to Lewis and Langford's Symbolic Logic.	J. C. C. McKinsey	1940
A Formulation of the Simple Theory of Types	Alonzo Church	1940
Einkleidung der Mathematik in Schröderschen Relativkalkül	Leopold Lowenheim	1940
Elimination of Extra-Logical Postulates.	Willard Van Orman Quine	1940
Elimination of Extra-Logical Postulates.	Nelson Goodman	1940

Note that if a paper has k authors, there are k rows for that paper in the result (as in rows four and five above). If a paper had k_1 authors and k_2 titles, there would be $k_1 \cdot k_2$ result rows for that paper, one for each combination. Such Cartesian products are unexpected for many SPARQL beginners and can lead to very large results, in particular, when making mistakes in the query formulation.

[27] `https://github.com/dblp/kg/wiki/Paper-TGDK-2024#example-queries`

4.2 SPARQL autocompletion

Writing a correct SPARQL query requires knowledge about the SPARQL query language in general as well as about the structure of the concrete knowledge graph. For the example query above, the following skills are required:

1. Getting the general syntax right: how to write a `SELECT` statement, how to write a `FILTER` expression, how to write an `ORDER BY` clause.
2. Knowing the names of the RDF types and entities needed for the query, here: `dblp:title`, `dblp:authoredBy`, and `dblp:yearOfPublication`.
3. Knowing the right `PREFIX` definitions and where to put them (the first few lines in the query above).

The user interface helps with all of these by offering incremental context-sensitive autocompletion after each keystroke. We recommend to go to `https://sparql.dblp.org` and try the following instructions live.

By simply typing “S”, the UI suggests the whole template for a SELECT clause (needed for most queries). After having typed the first variable `?paper` in the body of the SPARQL query, the UI will suggest predicate names, which are searchable by typing a prefix such as “ti” (for `title`). When selecting a predicate, the corresponding PREFIX statement will be automatically added at the top of the query. After entering the variable `?paper` a second time, the UI will suggest only those predicates that would lead to a non-empty result together with the already typed `?paper dblp:title ?title`. This narrows down the selection considerably. Continuing this way, the user can type a query from left to right, top to bottom relatively easily with minimal input and minimal knowledge of the syntax and the details of the knowledge graph. The details of this mechanism, along with example queries for other RDF datasets, are described in [7].

4.3 A more advanced SPARQL query

The following query returns all papers published at STOC 2018, and for each paper the number of all its authors, the number of its ORCID-certified authors, and the ratio between the two. The results are ordered by that ratio, largest first. The query makes use of the signatures in the dblp KG (see Section 2) as well as of several more advanced SPARQL features.

```
PREFIX dblp: <https://dblp.org/rdf/schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?paper (COUNT(?signature) AS ?num_authors) (COUNT(?orcid) AS ?num_orcid)
              (ROUND(100 * ?num_orcid / ?num_authors) AS ?perc) WHERE {
  ?paper dblp:publishedInStream <https://dblp.org/streams/conf/stoc> .
  ?paper dblp:yearOfEvent "2018"^^xsd:gYear .
  ?paper dblp:hasSignature ?signature .
  OPTIONAL { ?signature dblp:signatureOrcid ?orcid }
}
GROUP BY ?paper
ORDER BY DESC(?perc)
```

Run this query

Here are the top-5 results:

?paper	?num_authors	?num_orcid	?perc
https://dblp.org/rec/conf/stoc/Cheraghchi18	1	1	100
https://dblp.org/rec/conf/stoc/Filos-RatsikasG18	2	2	100
https://dblp.org/rec/conf/stoc/BergBKMZ18	5	4	80
https://dblp.org/rec/conf/stoc/ChattopadhyayKL18	4	3	75
https://dblp.org/rec/conf/stoc/ByrkaSS18	3	2	67

Let us break down the main components of this query:

- Each paper has one signature per author, that is, the pattern `?paper dblp:hasSignature ?signature` will have one match for each author of each paper.
- A signature might or might not have an ORCID associated with it. The `OPTIONAL` ensures that no signature will be left out, but if there is no ORCID, the value for `?orcid` is undefined.
- The `GROUP BY` groups the information by paper, that is, there will be one row per paper in the result. As a consequence, any other variable used in the `SELECT` clause has to be aggregated so that we get one value for each paper: `COUNT(?signature)` counts the number of authors, `COUNT(?orcid)` counts the number of ORCIDs that are not undefined, and `?perc` is computed as the ratio between the two, expressed as a percentage and rounded to the nearest integer.
- The `ORDER BY` ensures that the results are ordered by the percentage, highest percentage first. Note that by default, the result of a SPARQL query is unordered (and an endpoint can produce them in an arbirtrary order).

4.4 Federated queries

Federated queries request data from more than one SPARQL endpoint. By design, RDF and SPARQL are particularly well suited for such queries because there is no dataset-specific schema (conceptually, every RDF dataset is just a set of triples) and because all identifiers are globally unique (just like IRIs). Connections between datasets are established by reusing identifiers from the other dataset or by having extra triples that relate the identifiers to each other.

For example, the following query returns all SIGIR authors that exist in Wikidata with a link to dblp (via Wikidata's `wdt:P2456` predicate), and the location of their birthplace (which can then be shown on a map).

```
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX dblp: <https://dblp.org/rdf/schema#>
SELECT ?author_dblp ?author_name ?num_papers ?location WHERE {
  { SELECT ?author_dblp (COUNT(?paper) AS ?num_papers) WHERE {
    ?paper dblp:authoredBy ?author_dblp .
    ?paper dblp:publishedInStream <https://dblp.org/streams/conf/sigir> .
  } GROUP BY ?author_dblp }
  ?author_dblp dblp:primaryCreatorName ?author_name .
  ?author_dblp dblp:wikidata ?person_wd .
  SERVICE <https://query.wikidata.org/sparql> {
    # ?person_wd wdt:P2456 [] .
    ?person_wd wdt:P19 ?place_of_birth .
    ?place_of_birth wdt:P625 ?location .
  }
}
ORDER BY DESC(?num_papers)
```

Run this query ↗

Let us break down the three main components of this query:

- The first four lines of the query body are a so-called subquery, which is a full SPARQL query enclosed in `{ ... }`. The result of that subquery is a table with one row per dblp author and two columns: the author ID from dblp and the number of papers from that author in dblp.
- The next two lines augment that table by two more columns: the name of the author and the Wikidata IRI of that author. Note that both of these predicates are *functional*, that is, for each distinct subject there is at most one object.
- The remaining four lines of the query body are a SPARQL query to another SPARQL endpoint, with the URL `https://query.wikidata.org/sparql`. The result is a table with one row per entity in Wikidata that has a birthplace (these are mostly people) and three columns: the IRI of that person, their birthplace, and the coordinates of that birthplace.[28] If the IRIs for `?person_wd` from Wikidata are compatible with the IRIs for `?person_wd` from dblp (which they are), the join of the two tables then gives the desired result.
- The reason for the commented out first line of the `SERVICE` query is as follows. Without that line (or when it's commented out), the `SERVICE` query produces a large result, namely a table of all people in Wikidata together with their birthplace and the respective coordinates (3.5 million rows at the time of this writing). Transferring this result to the dblp SPARQL endpoint would be very expensive, and there are two ways to avoid that. One way makes use of the fact that the part of the query before the `SERVICE` gives only relatively few results, namely one row per author who has published at SIGIR (around one thousand at the time of this writing). The corresponding matches for `person_wd` can be sent to the Wikidata SPARQL endpoint using a VALUES clause to restrict the result of the `SERVICE` query.[29] QLever indeed automatically performs this optimization for small sub-results, so also for the given query (The exact threshold is configurable). The other way is to comment in the commented out line, which would restrict the result of the `SERVICE` query to only those persons with a dblp ID (around 71K at the time of this writing). For this particular query, the second way has the disadvantage that the query excludes dblp authors who do have an entry in Wikidata, but where the `wdt:P2456` predicate, which links authors to their dblp identifier, is missing (18 authors at the time of this writing). The second way would however be necessary if we wanted to perform the query for all 3.6 million dblp authors (not limited to SIGIR).

4.5 Querying both the dblp KG and the citation data

It is very natural to query the combined data from the dblp KG (Section 3.1) and the added citation data (Section 3.2). We could have set up a separate endpoint for each of these datasets and then use federated queries as shown in the previous section. However, there is always an overhead associated with federated queries because potentially large amounts of data have to be transferred between endpoints in one of the standard serialization formats. We therefore decided to provide both datasets in a single endpoint, as explained in Section 3.3.

Here is a typical example query to our endpoint that makes use of both datasets. It results in a list of all publications of Donald E. Knuth with at least one citation, ordered by the number of citations (most cited paper first).

[28] We here assume that both of these predicates are functional, which may not always be true. This could be addressed by a slightly more complex query, or by accepting that there will be multiple rows for the same person.

[29] This optimization is even discussed and suggested in the official SPARQL standard, see `https://www.w3.org/TR/sparql11-federated-query/#values`.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cito: <http://purl.org/spar/cito/>
PREFIX dblp: <https://dblp.org/rdf/schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?publ (COUNT(?cite) as ?count_cite) (SAMPLE(?label) as ?sample_label)
WHERE {
  ?publ rdf:type dblp:Publication .
  ?publ dblp:authoredBy ?author .
  ?publ rdfs:label ?label .
  ?author dblp:creatorName "Donald E. Knuth" .
  ?publ dblp:omid ?omid .
  ?cite cito:hasCitedEntity ?omid .
}
GROUP BY ?publ
ORDER BY DESC(?count_cite)
```

`Run this query`

This query is easy to understand given the concepts already explained in the previous sections. The second to last pattern of the query connects each publication to its OMID (the identifier used by OpenCitations, see Section 2.1.1). The last pattern produces one match for `?cite` for every citation. Grouping by `?publ` and using `COUNT(?cite)` for aggregation gives us the number of citations per publication.

Note that the result will only include publications that have an OMID and at least one citation. For the query above, this is true for only 100 of Donald Knuth's 184 publications in dblp. To include all publications, the last two patterns could be included in an `OPTIONAL { ... }`, see Section 4.3.

4.6 Performance

Our SPARQL endpoint and its user interface described in Section 3.3, as well as the embedded queries described in Section 3.5, are all powered by the QLever SPARQL engine. QLever is free and open-source software (FOSS), provided under a permissive license. QLever's primary design goal is to be efficient even on very large knowledge graphs (with up to hundreds of billions of triples), on a single machine using (relatively cheap) standard hardware. Compared to other knowledge graphs, the dblp KG is medium-sized (around 400 million triples), even if combined with the OpenCitations data (giving a total of around 1.2 billion triples). But even on a graph of this size, queries can be expensive and an efficient engine is key for a good user experience.

Indexing time. Like all SPARQL engines except the most basic ones, QLever precomputes special index data structures based on the input data, in order to enable fast queries. This pre-computation is called *indexing*. On a PC with an AMD Ryzen 9 7950X processor with 16 cores, 128 GB of RAM, and a 2 TB NVMe SSD, indexing the dblp KG takes around 4 minutes, while indexing the combined data takes around 12 minutes. Indexing times for QLever are roughly proportional to the number of triples in the input data.

Query times. Table 8 shows the query times (including the time to download the complete result) for a selection of six queries, against a SPARQL endpoint running on the same machine as above. The queries were chosen manually to cover a spectrum of queries that users typically ask and

Table 8 Query times in seconds and result sizes (number of rows × number of columns) for a selection of six queries on the dblp KG. Clicking on the query name takes you to the full query. The time for downloading the full result is included, hence the larger time for the fifth query.

Query	Result size	QLever	Comment
All papers published in SIGIR ↗	6,264 x 3	0.02 s	Two simple joins, nothing special
Number of papers by venue ↗	19,954 x 2	0.02 s	Scan of a single predicate with GROUP BY and ORDER BY
Author names matching REGEX ↗	513 x 3	0.05 s	Joins, GROUP BY, ORDER BY, FILTER REGEX
All papers in DBLP until 1940 ↗	70 x 4	0.11 s	Three joins, a FILTER, and an ORDER BY
All papers with their title ↗	7,167,122 x 2	4.2 s	Simple, but must materialize large result (problematic for many SPARQL engines)
All predicates ordered by size ↗	68 x 3	0.01 s	Conceptually requires a scan over all triples, but huge optimization potential

different complexities regarding the query processing. This is not a complete evaluation and just meant to give an impression. For a more extensive performance evaluation and for a comparison against other SPARQL engines, see the QLever Wiki and the publications listed there.[30]

We remark that all queries except the first are non-trivial and pose significant performance challenges to other SPARQL engines. The second query requires a scan over all 7.1M publication-venue pairs in the data. The third query requires the materialization of over 10K strings and a REGEX evaluation on each. The fourth query filters out 63 publications from over 7.2M. The fifth query requires the materialization and downloading of 7.2M paper IDs and titles. The sixth query conceptually requires a scan of the complete dataset.

5 Discussion and outlook

In this article, we introduced the dblp Knowledge Graph (dblp KG), an up-to-date semantic representation of the knowledge contained in the dblp computer science bibliography. We also introduced our new public SPARQL endpoint as a powerful new tool to explore dblp's bibliographic data and to create new insights. One particular advantage of the dblp KG is that it can be easily combined with other scholarly graphs using common identifiers. We have demonstrated this by combining the dblp and OpenCitations data in our query service. Just as easily, dblp could be joined with with, e.g., biographical data from Wikidata or the subject area classification from the ORKG.

The dblp KG is already in active use by the community. The linked open data live API (Section 3.6) alone receives more than one million requests from more than 25,000 IPs each month. The dblp KG RDF dump is downloaded about one thousand times each month. In recent research,

[30] Go to `https://github.com/ad-freiburg/qlever/wiki/` and search for "performance".

[5] uses the dblp KG to create a dataset for training and testing of question answering over Knowledge Graph (KGQA) systems. In [41], a natural language interface is built for the dblp KG, while [33] evaluates their universal question-answering platform using the dblp KG. In [4], an entity linking method has been proposed which links entities mentioned in text to their corresponding unique identifiers in the dblp KG.

Building upon the current iteration of the dblp KG and expanding its capabilities is an ongoing endeavor of the dblp team. A particular priority is the further utilization of only weakly structured semantic information listed on the dblp website, such as event dates or publisher information, as well as making it machine-actionable. The immediate next steps ahead are already clear: In the current DFG-LIS project *SmartER affiliations*,[31] the dblp team is intensifying its coverage of author affiliation information. Future iterations of the dblp KG will expand its model to add institution entities as first-class citizens to the graph and link affiliation information for authors and signatures. Also, the event history of conference and workshop series, together with metadata about the time and date of events, is contained aplenty in the dblp webpages and will be added to the dblp KG.

Having said that, there are several limitations that are probably out of the scope of what the dblp team can deliver. For example, the breakdown of person names into first, last, or middle name parts, gender information, or the annotation of the language a published work is written in, is out of reach since dblp has no comprehensive, reliable open data source for this kind of information. For the same reason, we cannot provide finer-grained type classifications, e.g., there will be no distinguishing between conference and workshop series, no distinguishing of full-paper from poster contributions, and no distinguishing of editorial articles from book review articles. Furthermore, email addresses or contact information (even if stated on published articles) will not be added to the dblp KG because of their privacy-sensitive nature. Finally, we have deliberately removed all links to authors and editors from `dblp:Withdrawn` publications to allow authors to exercise their right to be forgotten.

References

1 Boanerges Aleman-Meza. Swetodblp. `https://lod-cloud.net/dataset/sweto-dblp`, 2007. Accessed on 2024-04-22.

2 Boanerges Aleman-Meza, Farshad Hakimpour, Ismailcem Budak Arpinar, and Amit P. Sheth. Swetodblp ontology of computer science publications. *J. Web Semant.*, 5(3):151–155, 2007. `doi:10.1016/J.WEBSEM.2007.03.001`.

3 Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. Dbpedia: A nucleus for a web of open data. In *The Semantic Web, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2007. `doi:10.1007/978-3-540-76298-0_52`.

4 Debayan Banerjee, Arefa, Ricardo Usbeck, and Chris Biemann. Dblplink: An entity linker for the DBLP scholarly knowledge graph. In *ISWC 2023 Posters and Demos: 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023*, volume 3632 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023. URL: `https://ceur-ws.org/Vol-3632/ISWC2023_paper_428.pdf`.

5 Debayan Banerjee, Sushil Awale, Ricardo Usbeck, and Chris Biemann. DBLP-QuAD: A question answering dataset over the DBLP scholarly knowledge graph. In *BIR 2023: 13th International Workshop on Bibliometric-enhanced Information Retrieval ECIR 2023, Dublin, Ireland, April 2, 2023*, volume 3617 of *CEUR Workshop Proceedings*, pages 37–51. CEUR-WS.org, 2023. URL: `https://ceur-ws.org/Vol-3617/paper-05.pdf`.

6 Hannah Bast and Björn Buchhold. QLever: A query engine for efficient SPARQL+Text search. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 647–656. ACM, 2017. `doi:10.1145/3132847.3132921`.

7 Hannah Bast, Johannes Kalmbach, Theresa Klumpp, Florian Kramer, and Niklas Schnelle. Efficient and effective SPARQL autocompletion on very large knowledge graphs. In *Proceedings of CIKM 2022, Atlanta, GA, USA, Octo-*

[31] `https://gepris.dfg.de/gepris/projekt/515537520`

ber 17-21, 2022, pages 2893–2902. ACM, 2022. doi:10.1145/3511808.3557093.

8 Hannah Bast, Johannes Kalmbach, Claudius Korzen, and Theresa Klumpp. Knowledge graphs. In Omar Alonso and Ricardo Baeza-Yates, editors, *Information Retrieval: Advanced Topics and Techniques*, volume 60 of *ACM Books*. Association for Computing Machinery, New York, NY, USA, 1 edition, 2025. doi:10.1145/3674127.

9 Christian Bizer. DBLP bibliography database in RDF (fu berlin). https://lod-cloud.net/dataset/fu-berlin-dblp, 2007. Accessed on 2024-04-22.

10 Christian Bizer and Andy Seaborne. D2RQ - treating non-RDF databases as virtual RDF graphs. In *Proceedings of ISWC 2004) Posters, Hiroshima, Japan, November 7-11, 2004*. Springer, 2004. URL: http://iswc2004.semanticweb.org/posters/PID-SMCVRKBT-1089637165.pdf.

11 LOD W3C SWEO Community. Sweoig/taskforces/communityprojects/linkingopendata. https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/, April 2007. Accessed on 2024-04-22.

12 Richard Cyganiak, David Wood, and Markus Lanthaler. Rdf 1.1 concepts and abstract syntax. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/, februrary 2014. Accessed on 2024-05-23.

13 Marilena Daquino, Silvio Peroni, David M. Shotton, Giovanni Colavizza, Behnam Ghavimi, Anne Lauscher, Philipp Mayr, Matteo Romanello, and Philipp Zumstein. The opencitations data model. In *The Semantic Web - ISWC 2020, Athens, Greece, November 2-6, 2020, Proceedings, Part II*, volume 12507 of *Lecture Notes in Computer Science*, pages 447–463. Springer, 2020. doi:10.1007/978-3-030-62466-8_28.

14 Bruce D'Arcus and Frederick Giasson. The bibliographic ontology. https://www.dublincore.org/specifications/bibo/, May 2016. Accessed on 2024-05-13.

15 dblp Team. dblp computer science bibliography – Monthly Snapshot RDF/N-Triple Release. doi:10.4230/dblp.rdf.ntriples.

16 dblp Team. dblp computer science bibliography – Monthly Snapshot XML Release. doi:10.4230/dblp.xml.

17 Jörg Diederich. DBLP in RDF (l3s). https://lod-cloud.net/dataset/l3s-dblp, 2007. Accessed on 2024-04-22.

18 Said Fathalla, Christoph Lange, and Sören Auer. EVENTSKG: A 5-star dataset of top-ranked events in eight computer science communities. In *The Semantic Web - 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2-6, 2019, Proceedings*, volume 11503 of *Lecture Notes in Computer Science*, pages 427–442. Springer, 2019. doi:10.1007/978-3-030-21348-0_28.

19 Julian Franken, Aliaksandr Birukou, Kai Eckert, Wolfgang Fahl, Christian Hauschke, and Christoph Lange. Persistent identification for conferences. *Data Sci. J.*, 21:11, 2022. doi:10.5334/DSJ-2022-011.

20 Hugh Glaser. DBLP computer science bibliography (rkbexplorer). https://lod-cloud.net/dataset/l3s-dblp, 2007. Accessed on 2024-04-22.

21 Hugh Glaser and Ian Millard. RKB explorer: Application and infrastructure. In *Proceedings of the Semantic Web Challenge 2007 co-located with ISWC 2007 + ASWC 2007, Busan, Korea, November 13th, 2007*, volume 295 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007. URL: https://ceur-ws.org/Vol-295/paper13.pdf.

22 Stephanie Hagemann-Wilholt, Margret Plank, and Christian Hauschke. ConfIDent – an open platform for FAIR conference metadata. In *21st International Conference on Grey Literature "Open Science Encompasses New Forms of Grey Literature", Hannover, Germany, October 22-23, 2019*, volume 21 of *GL Conference Series*, pages 47–51, 2019. doi:10.15488/9424.

23 Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. *Knowledge Graphs*. Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool Publishers, 2021. doi:10.2200/S01125ED1V01Y202109DSK022.

24 Google Inc., Yahoo Inc., Microsoft Corporation, and Yandex. Schema.org v26.0. https://schema.org/version/26.0, February 2024. Accessed on 2024-05-13.

25 Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D'Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019, Marina Del Rey, CA, USA, November 19-21, 2019*, pages 243–246. ACM, 2019. doi:10.1145/3360901.3364435.

26 Michael Ley. Die trierer informatik-bibliographie DBLP. In *Informatik '97, Informatik als Innovationsmotor, 27. Jahrestagung der Gesellschaft für Informatik, Aachen, 24.-26. September 1997*, Informatik Aktuell, pages 257–266. Springer, 1997. doi:10.1007/978-3-642-60831-5_34.

27 Michael Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *String Processing and Information Retrieval, 9th International Symposium, SPIRE 2002, Lisbon, Portugal, September 11-13, 2002, Proceedings*, volume 2476 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2002. doi:10.1007/3-540-45735-6_1.

28 Michael Ley. DBLP – Some lessons learned. *Proc. VLDB Endow.*, 2(2):1493–1500, 2009. doi:10.14778/1687553.1687577.

29 Paolo Manghi, Alessia Bardi, Claudio Atzori, Miriam Baglioni, Natalia Manola, Jochen Schirrwagen, and Pedro Principe. The openaire research graph data model, April 2019. doi:10.5281/zenodo.2643198.

30 Fabio Mercorio, Mario Mezzanzanica, Vincenzo Moscato, Antonio Picariello, and Giancarlo Sperlì.

A tool for researchers: Querying big scholarly data through graph databases. In *Machine Learning and Knowledge Discovery in Databases, ECML PKDD 2019, Würzburg, Germany, September 16-20, 2019*, volume 11908 of *Lecture Notes in Computer Science*, pages 760–763. Springer, 2019. doi:10.1007/978-3-030-46133-1_46.

31 Mario Mezzanzanica, Fabio Mercorio, Mirko Cesarini, Vincenzo Moscato, and Antonio Picariello. Graphdblp: a system for analysing networks of computer scientists through graph databases - graphdblp. *Multim. Tools Appl.*, 77(14):18657–18688, 2018. doi:10.1007/S11042-017-5503-2.

32 United States Library of Congress. Bibframe 2 ontology. http://id.loc.gov/ontologies/bibframe-2-3-0/, 2016. Accessed on 2024-05-13.

33 Reham Omar, Ishika Dhall, Panos Kalnis, and Essam Mansour. A universal question-answering platform for knowledge graphs. *Proc. ACM Manag. Data*, 1(1):57:1–57:25, 2023. doi:10.1145/3588911.

34 Silvio Peroni and David Shotton. Frbr-aligned bibliographic ontology (fabio). http://www.sparontologies.net/ontologies/fabio, 2012. Accessed on 2024-05-13.

35 Silvio Peroni and David M. Shotton. Opencitations, an infrastructure organization for open scholarship. *Quant. Sci. Stud.*, 1(1):428–444, 2020. doi:10.1162/QSS_A_00023.

36 Jason Priem, Heather A. Piwowar, and Richard Orr. Openalex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. In *Proceedings of the 26th International Conference on Science and Technology Indicators (STI 2022), Granada, Spain, Sptember 7-9, 2022*, 2022. doi:10.5281/zenodo.6936227.

37 David Shotton and Silvio Peroni. Datacite ontology. http://www.sparontologies.net/ontologies/datacite, September 2022. Accessed on 2024-05-13.

38 Selver Softic. Colinda - conference linked data. https://lod-cloud.net/dataset/colinda, 2015. Accessed on 2024-04-22.

39 Selver Softic, Laurens De Vocht, Erik Mannens, Martin Ebner, and Rik Van de Walle. COLINDA: modeling, representing and using scientific events in the web of data. In *Proceedings of DeRiVE 2015, Protoroz, Slovenia, May 31, 2015*, volume 1363 of *CEUR Workshop Proceedings*, pages 12–23. CEUR-WS.org, 2015. URL: https://ceur-ws.org/Vol-1363/paper_2.pdf.

40 Denny Vrandecic. The rise of wikidata. *IEEE Intell. Syst.*, 28(4):90–95, 2013. doi:10.1109/MIS.2013.119.

41 Ruijie Wang, Zhiruo Zhang, Luca Rossetto, Florian Ruosch, and Abraham Bernstein. Nlqxform-ui: A natural language interface for querying DBLP interactively. *CoRR*, abs/2403.08475, 2024. doi:10.48550/arXiv.2403.08475.