DBpedia's Databus and strategic initiative to facilitate "1 Billion derived Knowledge Graphs by and for Consumers" until 2025

Status: Sept 9th, 2019, public, content: Databus Public Beta announcement and strategic impact, to be used as Databus documentation and follow-up discussion with DBpedia Association members, chapters and community to create a strategic agenda for DBpedia Databus Links: Document (direct link) and Version metadata

We are proud to announce that the DBpedia Databus website at https://databus.dbpedia.org and the SPARQL API at https://databus.dbpedia.org/(repo/sparql|yasgui) (docu) are in public beta now. The system is usable (eat-your-own-dog-food tested) following a "working software over comprehensive documentation" approach. Due to its many components (website, sparql endpoints, keycloak, mods, upload client, download client, and data debugging), we estimate approximately six months in beta to fix bugs, implement all features and improve the details. If you have any feedback or questions, please use the DBpedia Forum, the "report issues" button, or dbpedia@infai.org.

DBpedia Databus

The DBpedia Databus is a platform to capture invested effort by data consumers who needed better data quality (fitness for use) in order to use the data and give improvements back to the data source and other consumers. DBpedia Databus enables anybody to build an automated DBpedia-style extraction, mapping and testing for any data they need.

In computer architecture, a bus is a communication system that transfers data between components inside a computer, or between computers. The DBpedia Databus elevates this principle to the Web in the following manner:

- Databus is purely **intended for data consumers**, who can mirror the original data and their extracted or cleaned derivatives to facilitate easier consumption, deployment and further derivatives. Original (external) data publishers can become consumers by re-integrating derived value into their source data. Of course, they can also join the databus to support their consumer network.

- Data consumers can create a space on the Databus and add storage in form of URL - accessible (HTTP/S, FTP, etc.) files to their servers and post a minimal dataid.ttl metadata file (CC-0) with dcat:downloadURL to the Databus SPARQL API for discovery and network coordination. Consumers have full authority over storage on their servers. They announce dcat:downloadURL links, but can return 401 Unauthorized. This is useful for the coordination of internal projects with sensitive data and opens the possibility for selling. We plan to implement a unified access and payment layer.
- Any consumers can query the Databus with SPARQL to identify their own datasets alongside datasets of other consumers and download them via retrieving dcat:downloadURLs or the Databus Client into their local machines and applications.
- If consumers either spot quality issues (such as parse errors, wrong data, missing coverage) or would like extensions, they are able to add value to the data in the following manner:
    - [Consumer feedback] Alongside the metadata, consumers can include feedback links to forums, issue-trackers or code (e.g. on GitHub) where other consumers can specify the issue, fix the software, or provide test cases. This feedback is a valuable input for initial derivative, as it can improve data quality and speed up innovation.
    - [Derivatives] Managing data quality is pareto-efficient, 20% of the initial effort make up 80% of the result. Then effort increases steeply, in fact so steep that the original publisher will struggle. Commercial data providers will weigh effort against income and therefore triage what to improve. Open data providers without an income model normally do not have the resources to accommodate all user needs. Databus allows consumers to re-publish patched data or community extensions, which can be picked up by publishers from the bus to complement their data. These "Derive" processes are fully automatable and keep prov:wasDerivedFrom. New versions can be detected by querying the Databus, and processors can run and publish derivatives in third-party storage. Different from pipelines, **Derive operations form a data network** where effort and value is discoverable and can flow freely through the network up to the sources in a trickle-up manner.
    - [Mods] Mods are processors for statistical and semantic analysis, continuous integration (CI) testing and enrichment, or any other tasks. We implemented a demo mod that measures uptime of dcat:downloadURL of all storage. It is a prov:Activity and adds "onlinerate" and "weeklyonlinerate", and links to detailed reports to the SPARQL-accessible database. Mods are an effective way to

provide value for the whole Databus. Only one developer and one server (not necessarily provided by the developer) is required to add additional information to the datasets, producing a consistent layer of annotations to all data (no more messy user tagging!). DBpedia plans to implement and run a ParseMod that produces parse logs for all NTriples/RDF files on the bus, hoping for cleaner data for us to consume. Mods can also be completely selfish. For the DBpedia knowledge graph itself, we will implement a statistic collection mod, allowing us to better understand how our knowledge graph evolves. Others need to deploy and run it for themselves. Consumers can write mods that check new data before they ingest it to avoid application breaks on data updates. Results of mods can also be sold in the future to be able to scale them up and improve them.

Vision

Professional consumers of data worldwide have already built stable cleaning and refinement chains for all available datasets, but their efforts are invisible and not reusable. Deep, cleaned data silos exist beyond the reach of publishers and other consumers trapped locally in pipelines.

Data is not oil that flows out of inflexible pipelines. Databus breaks existing pipelines into individual components that together form a decentralized, but centrally coordinated data network in which data can flow back to previous components, the original sources, or end up being consumed by external components,

The Databus provides a platform for re-publishing these files with very little effort (leaving file traffic as only cost factor) while offering the full benefits of built-in system features such as automated publication, structured querying, automatic ingestion, as well as pluggable automated analysis, data testing via continuous integration, and automated application deployment **(software with data)**. The impact is highly synergistic, just a few thousand professional consumers and research projects can expose millions of cleaned datasets, which are on par with what has long existed in deep silos and pipelines.

DBpedia runs a stable extraction and transformation network that targets Wikipedia's content. In collaboration with our community, this effort is driven by developing best practices and tools that other projects are encouraged to copy and adapt. These clean and automatically retrievable datasets will synergistically raise the efficiency of feeding data into AI-driven analytics and fusion & recombination engines. One AI developer can directly use the recycled data garbage of his colleagues and recombine previous effort into a derived knowledge graph as input for their AI application. The synergies scale well over networks. If a whole department consistently uses the databus it can transfer their tools and datasets faster from one project to the other. If partners of these projects do the same, it is possible to transfer value across whole partner networks. With value (tools with data) transferred speedier, we hope to spur innovation for a new generation of Siris and Alexas, prediction, data cleaning, search, and question & answering systems operating on the bus.

Our vision is to commoditize knowledge graphs by automating cleaning and recombinations of datasets and facilitating low-code, use case specific deployment for each organisational department, each development team, even down to the level of non-technical domain experts and individual bachelor theses and student projects. While cleaner data will be available to everyone, we see a plethora of new business opportunities for database and use-case specific application providers, as well as system integrators and value shops, smoothing the last mile of in-house integration. Open data publishers - e.g. DBpedia - can efficiently coordinate with their consumer value network and re-integrate derivative value via linking and consumption.

A new era for decentralized collaboration on data quality

DBpedia was established around producing a queryable knowledge graph derived from Wikipedia content that's able to answer questions like "What have Innsbruck and Leipzig in common?" A community and consumer network quickly formed around this highly useful data, resulting in a large, well-structured, open knowledge graph that seeded the Linked Open Data Cloud -- which is the largest knowledge graph on earth. The main lesson learned after these 13 years is that current data "copy" or "download" processes are inefficient by a magnitude that can only be grasped from a global perspective. Consumers spend tremendous effort fixing errors on the client-side. If one unparseable line needs 15 minutes to find and fix, we are talking about 104 days of work for 10,000 downloads. Providers - on the other hand - will never have the resources to fix the last error as cost increases exponentially (20/80 rule).
Discarding faulty data often means that a substitute source has to be found, which is hours of research and might lead to similar problems. From the dozens of DBpedia Community meetings we held, such as the upcoming meeting at SEMANTiCS 2019 on Sept 12th, we can summarize that for each clean-up procedure, data transformation, linkset or schema mapping that a consumer creates client-side, dozens of consumers have invested the same effort client-side before him and none of it reaches the source or other consumers with the same problem. Holding the community meetings just showed us the tip of the iceberg. As a foundation we implemented a mappings wiki that allowed consumers to improve data quality centrally. A next advancement was the creation of the SHACL standard by our former CTO and board member Dimitris Kontokostas. SHACL allows consumers to specify repeatable tests on graph structures and datatypes, which is an effective way to systematically assess data quality. We established

the DBpedia Databus as a central platform to better capture decentrally created, client-side value by consumers. It is an open system, therefore value that is captured flows right back to everybody.

We would like to share some more experiences when working with data on the Databus:

- There are very few rules or specifications. We accept structured data in any file format (JSON, XML, CSV, TSV, Binary and various RDF files, e.g., RDF-Turtle, RDF-NTriples, RDF-XML, JSON-LD). If you would like to standardise or improve consistency, the first way is to write a mod that runs tests to check conformance, the second way is a popular application that fails, if standards are not adhered. Manuals and specs do not work well for data. On the web, JS validation is a must for HTML forms.
- Data publishers should approach the Databus in an agile manner. Your data will never be perfect and the Databus will help you with improving the data by enabling consumer feedback, derivatives and mods. It is a good idea to release data from the beginning and write your own mod to test and analyse it.
- Open Data providers should not be concerned, if other users upload their data into their space. After a while you can check, whether they created links, mappings or mods, that might benefit you, or whether they innovated something entirely unexpected.
- Consumers must properly attribute and should not hesitate to mirror and alter other data and put them on the Databus. Any data on the bus can use the same mechanisms and tools for all your data dependencies. The synergies are great, others can use these as well and improve them for you.

Databus incorporates features from DNS, Git, RSS, online forums and Maven to harness the full workpower of data consumers.

DNS feature: Stable Dataset ID

Databus hands out stable, persistent dataset IDs (similar to purl.org, w3id.org): Databus distinguishes between artifacts (abstract dataset identity), version snapshots (dcat:Dataset) containing files (a slight semantic change to dcat:Distribution). Databus is version-centric and adds Databus identifiers such as https://databus.dbpedia.org/dbpedia/databus/databus-data/2019.09.01 to annotate the version and files. Each version can be re-published in case the dcat:downloadURL changes. This is similar to updating purl.org or changing DNS A records. While DNS only allows lookup queries, we provide full graph queries, so you can not only resolve dcat:downloadURL, but also query over the cached FOAF profiles of users, versioning information, file statistics, and mod stats.

Git-like workflows

For small datasets such as ontologies, we recommend using Git as merge conflicts are relatively easy to check and resolve that way. Otherwise, the "merge" operation for data is normally called "data integration" and follows other principles. Once formats are changed or data from different sources is merged, diffs will not work anymore. We call it "Git-like" because we have /$username/ in all URIs, so there are personal, organisational, and foaf:Agent spaces like on GitHub.
There is also a "Derive" operation, which is similar to fork. If you are allowed to republish data, e.g. in a secure enterprise network or in the case of an open license, you can just fix and publish it. It is immediately usable and useful for others and also a great way to file-share with colleagues in the same project. We spend around 3 hours to clean up an RDF dump from geonames.org and are hosting it now on the databus. We did it because we need unified team access and auto-loading into FlexiFusion.

RSS, forums and issue trackers

Initially, we intended to implement RSS feeds for all new versions of datasets. However, pre-defined RSS feeds proved much less flexible than querying latest versions of specific files with SPARQL directly (DBpedia instance-types, transitive, en & de) . It is easy to implement RSS feeds, but it should be implemented by consumers who need them and added via Databus Mods, where needed. For publishers that would like to get in touch with and support their consumer communities, we intend to integrate forums and issue trackers, linking to existing forums is already possible.

Usage and Acknowledgements

The Databus is free with a fair use policy (during beta). Each announced file carries a metadata overhead of ~20 triples (digital signature, sha256sum) plus 20 triples for the dataset (signed license required). So in its initial design, it is best suited for fewer, but larger files. We consider a fair deal to post max. 1000 files per month with an average of 10MB per file.
If data publishers can host 10GB of open data, we are happy to host 20,000 triples. Stale metadata will be removed after a while. Different quotas or a small fee (please ask) for private "HTTP 401" files will apply as they use our platform and resources. Once the Databus leaves

beta, we will be able to fork the architecture, e.g. for libraries and enterprises. DBpedia is non-profit and accepts financial contributions. If sufficient income is established, we intend to use the resources to remove the quota for open data, and lobby national public bodies to provide (persistent) storage and processing to create a global data infrastructure by unifying and interconnecting national initiatives.

We are also accepting in-kind development work force and volunteers. Incentives for contributions are that we can focus on faster specialising and extending Databus in the direction that meets its contributors demands. Servers are currently covered via support by TIB Hannover, OpenLink Software, URZ Leipzig, SCADS, DWS (Uni Mannheim), over a dozen DBpedia chapters, the BMWi-funded projects PLASS and LOD-GEOSS, as well as the fees of the DBpedia Association members and generous additional donations by Diffbot and Ontotext. We would also like to thank the thousands of DBpedians that supported us with data issues, test cases, code patches, extensions, and applications.

Databus Download Client

The Databus was inspired by Apache Maven and MVNRepository and Maven Central, - a software build automation infrastructure - with influences from GitHub, Steam and DebianRepository. In addition to the SPARQL API (including an alpha collection feature), we implemented a data dependency plugin for Maven which complements automatic software dependency downloads with automatic download of input data for data and text analytics, training AIs, or extraction (as done by the DBpedia Extraction Framework).
While the favorite Databus file format is the line-based RDF-NTriples with decoded-IRIs compressed as .bz2, we have developed a Databus Client (alpha) with the goal of reaching **client-side interoperability in a manner on par with Web browsers**. Currently, the client features:

- Download and sha checksum verification
- Recompression of many different compressions into one (Apache Compress)
- Conversion of isomporhic formats such as RDF-XML, RDF-NTriples, JSONLD, RDF-Turtle into single format.
- Some experimental mapping capabilities like RDF to CSV

The client is open-source for anybody to extend. Our vision is to build (quasi-) isomorphic groups over mediatypes, such as RDF2RDF or CSV2TSV as well as JPG2PNG . In addition, we plan to provide a plugin mechanism to incorporate more sophisticated mapping engines as

RML, R2RML, R2R (for owl:equivalence translation) and XSLT. Any file containing mappings can already be published using the Databus, we didn't implement picking them up or connecting them to the matching datasets, yet.
We envision that anyone can just publish data in their preferred format,others publish mappings and yet others can use and improve these mappings. **The Databus Client will collect all information and handle all consumer preferences and requirements.**

As a sweet note here, we developed a pattern for low-code application deployment. We have a prototype of a 20 line docker, which take Databus dcat:downloadURL queries as configurable input, then download the data and deploy a Virtuoso SPARQL endpoint . **The approach is extensible to any software that consumes data.**

1 Billion interconnected, quality-controlled Knowledge Graphs until 2025

As we are inversing the paradigm form a publisher-centric view to a data consumer network, we will open the download valve to enable discovery and access to massive amounts of cleaner data than published by the original source. The main DBpedia Knowledge Graph - cleaned data from Wikipedia in all languages and Wikidata - alone has 600k file downloads per year complemented by downloads at over 20 chapter, e.g. http://es.dbpedia.org as well as over 8 million daily hits on the main Virtuoso endpoint. Community extension from the alpha phase such as DBkWik, LinkedHypernyms are being loaded onto the bus and consolidated and we expect this number to reach over 100 by the end of the year. Companies and organisations who have previously uploaded their backlinks here will be able to migrate to the databus. Other datasets are cleaned and posted. In two of our research projects LOD-GEOSS and PLASS, we will re-publish open datasets, clean them and create collections, which will result in DBpedia-style knowledge graphs for energy systems and supply-chain management.

Databus is a common, extensible metadata overlay over decentral data dumps, a global view with extensible tools to locally deploy any of it. Ending data silos has been a long standing vision of ours, and we believe that with the Databus, we are contributing a practical cost effective tool to achieve it. Please try it for yourself and download the first 10k files, all versions, ca. 250 GB: https://databus.dbpedia.org/yasgui/

```
PREFIX dataid: <http://dataid.dbpedia.org/ns/core#>
PREFIX dcat:   <http://www.w3.org/ns/dcat#>

SELECT ?downloadURL, ?stableFileId, ?dataset, ?stableVersionId, ?localpath, ?shasum
WHERE {
  # ?dataset URI points to the remote dataid.ttl file
  ?dataset a dataid:Dataset.
  ?dataset dcat:distribution/dcat:downloadURL ?downloadURL .
  ?dataset dcat:distribution/dataid:file ?stableFileId .
  ?dataset dcat:distribution/dataid:sha256sum ?shasum .
  ?dataset dataid:version ?stableVersionId .
  # when downloading use this ?localpath to solve conflicting file names
  BIND (replace(str(?stableVersionId), "https://databus.dbpedia.org/" , "") AS ?localpath)
}
LIMIT 10000
OFFSET 0
```

Of course not every download results in a knowledge graph and there are repeated downloads of new versions for the same dataset. Then there are recombinations and fusions as FlexiFusion and also linking engines as SILK and LIMES, who require two or more datasets as input and generate links between them. After the beta, we intend to publish anonymized SPARQL query logs, so anybody can produce and reproduce access statistics of the Databus.
We will then call to the community to start a discussion about what is a valid measure of data size and quality to count as a knowledge graph and also reintegrate any statistics produced by Databus consumers.

Consumers who also publish services on the Web.

DBpedia has a 13 year tradition to host the DBpedia Knowledge Graph and other free online services as well as exporting language-specific knowledge graphs and services to our DBpedia chapters and community. In the databus model Web publishing becomes consumption. Data needs to be loaded into databases first, DBpedia Spotlight requires to load training data. The Databus will lower maintenance cost for these free service as they now can auto-update directly from the databus. Before setting up a full dbpedia chapter was very difficult, sometimes data needed to be extracted from Wikipedia first. Now DBpedia keeps all releases centrally accessible on different servers with monthly updates (https://databus.dbpedia.org/dbpedia). Chapters consume and can focus on extensions or if necessary fix.

Scalability and sustainability. There is an inherent deterioration of Common goods, which are defined in economics as goods that are rivalrous and non-excludable. Similar to overfishing the ocean, database performance is limited by the allocated servers. Scalability incurs an increasing cost burden proportional to popularity as the number of daily concurrent requests grows. Virtuoso currently handles 8 million daily requests on the main endpoint at http://dbpedia.org/sparql with two average servers. OpenLink as well as the chapters provide these resources without compensation.
DBpedia is still community effort and it is unfair to expect 100% uptime or increasing performance.

1. As a first scalability solution, we advocate **that consumers replicate the infrastructure, that fits their needs best locally, on rented servers or clouds.** The Databus/client/docker combination will effectively lower the cost for docker deployment. The storage provider still has to carry the cost for traffic and bandwidth, which is low and can be cached, e.g. at existing public infrastructures or at Internet Service Provider (ISP) level (Clarification note: depending on your country, these are the organisations selling you internet and storing PetaBytes of your complete online accesses, if you don't use a VPN).

2. As a second scalability solution, we advocate **paid hosting and services**. Earning income should scale with the number of users. Twice the number of users, means twice the income which means you can afford twice the number of servers and hire extra staff for maintenance. There might be some extra, which you can keep or reinvest to improve. It is business 101.

Each databus version page also contains a services section, where free, **shared services** are displayed, which you can overfish. These are followed by **dedicated services**, which offer reliable hosting and extra features (depending on the provider).

Databus and the LOD Cloud

As a future research agenda, we are encouraging an open comparison of the Databus architecture to the LOD Cloud. Fundamentally, the development of the Databus was triggered by a (now offline) proof of concept implementation called LODVader in 2016, where we demonstrated that it is feasible to create an image of the LOD cloud (including ontologies) with dump downloads of RDF data.

In the design of the Databus, we included some concepts that are also established in LOD, as well as some concepts that we found difficult when deploying linked data.

Here is a list:

- Databus is consumer-centric, Linked Data is publisher-centric
- Content-negotiation or publishing in several formats is handled on consumer side via the Databus Client and local transformations.
- Provenance is handled by private-key signatures instead of domain ownership or URI namespace. We implemented WebID, but are still revising the system. The main problem we are facing is that via the Databus we are trying to automate all processes on servers with cron-jobs. Therefore, they require either private keys with no passwords or to add the password to a config file. All of our servers are shared between multiple root developers, so honestly all of the Databus developers have access to all other private keys and accounts. We are investigating how to manage signatures either client-side via browser certificate, or by integrating them into our https://www.keycloak.org/ server, which we also use for https://forum.dbpedia.org
- The Web and Linked Data are more forgiving with URI variants, while RDF and SPARQL on the other hand are very strict and often implemented on exact URI equality.
  URIs such as http://dbpedia.org//////resource/URI work in Linked Data and we investigated a lot on how Wikipedia implemented https://en.wikipedia.org/wiki/The_Ren_%26_Stimpy_Show allowing both & and %26 to find the canonical form. We implemented a preliminary of a URI eval mod on the Databus, to evaluate DBpedia URIs testing that they are the same between versions.
- Including versions into URIs is often problematic. However, Web Ontologies evolve under the same URI and you never know what you get with the next request (if it doesn't fail). Our bachelor student Denis Streitmatter implemented an ontology-tracker that checks the DBpedia ontology URL every 30 min, diffs it with the previous version, and publishes it on GitHub and the Databus including a DL-axiom and RDF diff: https://databus.dbpedia.org/denis/ontology/dbo-snapshots
  Ontologies are often small in size. In the future, we plan to run this tracker over any OWL ontology we can find, e.g. in https://lov.linkeddata.es/ . Changes in the ontology are particularly critical for application stability and it is difficult for clients to clearly communicate to each other which exact version they used. Databus versioning enables this and the ontology-tracker also consolidates the revision history.
- Link asymmetry: currently in LOD, when looking at two sources, either source A or source B publishes links in an asymmetric fashion. Discovery is difficult as it depends on

the coordination of players. In addition, it is difficult to post and find links by third parties. We believe that many datasets have converged towards stable IDs already, and that links should be symmetrical. The property to form large equivalence clusters is generally seen as horrible for automatic linking. Looking at it from another perspective though, it is clear that if we focus on very few links to existing clusters, that task should be sufficient and better manageable. Therefore, instead of linking to many other databases with high effort and potentially high impact errors, we actually just need one correct link. This goes together with the Databus methodology (see above) to listen to consumers and accept their help with data quality. Complementary to the Databus, we designed a scalable global ID index for clusters (data, lookup service (contributed by Alex Olieman), browse: https://global.dbpedia.org/id/55LmB). Our ID clustering engine uses standard algorithms, scales with SPARK, and can index Wikipedia/Wikidata within a few hours. It is definitely a community and consumer task to devise extensive testing for URI stability, change tracking, and link validation. It is an effective effort though, as the Databus provides a global view. Just one developer is necessary to implement a mod generating the statistical properties of VOID, such as `void:distinctSubjects`, which will consistently cover all datasets on the bus. No coordination is necessary and all mod results are retrievable via the SPARQL API as we will load the link from the result to the stable dataset, version, and file IDs. There still can be several global views (build your own index) on the linking of LOD, i.e. several truths, we just minimize the effort to debug and keep them consistent.

Impact

An important note: DBpedia's mission is to facilitate "Global and Unified Access to Knowledge Graphs". Linked Data and Linked Open Data is a vital part of this mission. Although we advocate dumps and replication, our contribution should be seen as a complementary effort to clean and manage LOD better. There is a great opportunity for sustainability. Open Data Projects that are able to host linked data during project lifetime can simply put their result in storage, which can be backuped. It will be available for local replication and usage. If the dataset is in high demand, national data infrastructures can decide to retrieve the dumps from storage and revive the hosting (e.g. by taking over the domain, rewriting the namespace, or using the ?subject=$URL pattern). Moreover, it does not pose a threat to decentralisation, when we provide a reliable super node infrastructure. Quite the contrary is the case - it will support smaller nodes and self-hosters to get cleaner data and ease their effort (data management and hosting is hard!), which is what we need to achieve to make the long tail of humanity's knowledge and data accessible for everyone.

We invite all members and the larger DBpedia and Data community to give feedback, and if positive help implement the strategic initiative.

This document was written by Sebastian Hellmann with contributions and feedback of members of the DBpedia Association Board as well as input from the community.

Global and Unified Access to Knowledge Graphs

Website: dbpedia.org
Twitter: @dbpedia
Get Involved: https://forum.dbpedia.org/

Contact

DBpedia Association *affiliated with*
Institut für Angewandte Informatik e. V. (InfAI)
Goerdelerring 9, 04109 Leipzig

Email: dbpedia@infai.org
Phone: +49 341 229037 0