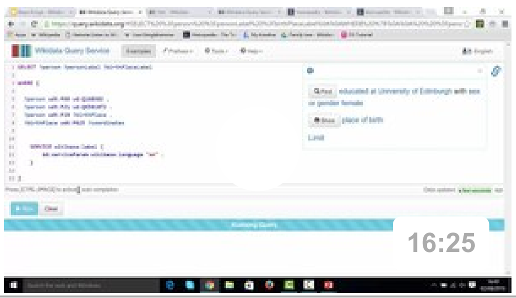
MediaWiki

Wikidata Query Service/User Manual

< Wikidata Query Service

Wikidata Query Service (WDQS) is a software package and public service designed to provide a SPARQL endpoint which allows you to query against the Wikidata data set.


A 15-min Wikidata Query Service tutorial

This page or other relevant documentation pages will be updated accordingly; it is recommended that you watch them if you are using the service.

If you're looking for a tutorial on the SPARQL syntax itself, we recommend starting here: SPARQL tutorial.

You can see examples of the SPARQL Queries on the SPARQL examples page.

Data set

The Wikidata Query Service operates on a data set from Wikidata.org, represented in RDF as described in the RDF dump format documentation.

The service's data set does not exactly match the data set produced by RDF dumps, mainly for performance reasons; the documentation describes a small set of differences.

You can download a weekly dump of the same data from:

https://dumps.wikimedia.org/wikidatawiki/entities/

Basics - Understanding SPO (Subject, Predicate, Object) also known as a Semantic Triple

spo or "subject, predicate, object" is known as a triple, or commonly referred to in Wikidata as a **statement** about data.

The statement "The United States capital is Washington DC" consists of the subject "United States" (Q30), the predicate "capital is" (P36), and an object "Washington DC" (Q61). This statement can be represented as three URIs:

```
<http://www.wikidata.org/entity/Q30> <http://www.wikidata.org/prop/direct/P36> <http://www.wikidata.org/entity/Q61> .
```

Thanks to the prefixes (see below), the same statement can be written in a more concise form. Note the dot at the end to represent the end of the statement.

```
wd:Q30 wdt:P36 wd:Q61 .
```

The **/entity/** (**wd:**) represents Wikidata entity (Q-number values). The **/prop/direct/** (**wdt:**) is a "truthy" property — a value we would expect most often when looking at the statement. The truthy properties are needed because some statements could be "truer" than others. For example, the statement "The capital of U.S. is New York City" is true — but only in the historical context of the year 1790. WDQS uses rank to determine which statements should be used as "truthy".

In addition to the truthy statements, WDQS stores all statements (both truthy and not), but they don't use the same **wdt:** prefix. U.S. capital has three values: DC, Philadelphia, and New York. And each of these values have "qualifiers" - additional information, such as start and end dates, that narrows down the scope of each statement. To store this information in the triplestore, WDQS introduces an automatic "statement" subject, which is essentially a random number:

```
wd:Q30 p:P36 <random_URI_1> .         # US "indirect capital" is <X>
<random_URI_1> ps:P36 wd:Q61 .        # The X's "real capital value" is Washington DC
<random_URI_1> pq:P580 "1800-11-17" . # The X's start date is 1800-11-17
```

See SPARQL tutorial - qualifiers for more information.

spo is also used as a form of basic syntax layout for querying RDF data structures, or any graph database or triplestore, such as the Wikidata Query Service (WDQS), which is powered by Blazegraph, a high performance graph database.

Advanced uses of a triple (spo) even including using triples as objects or subjects of other triples!

Basics - Understanding Prefixes

The subjects and predicates (first and second values of the triple) must always be stored as URI. For example, if the subject is Universe (Q1), it will be stored as **<https://www.wikidata.org/entity/Q1>**. Prefixes allow us to write that long URI in a shorter form: **wd:Q1**. Unlike subjects and predicates, the object (triple's third value) can be either a URI or a literal, e.g. a number or a string.

WDQS understands many shortcut abbreviations, known as prefixes. Some are internal to Wikidata, e.g. **wd, wdt, p, ps, bd**, and many others are commonly used external prefixes, like **rdf, skos, owl, schema**.

In the following query, we are asking for items where there is a statement of "P279 = Q7725634" or in fuller terms, selecting subjects that have a predicate of "subclass of" with an object of = "literary work".

The output variables:

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wds: <http://www.wikidata.org/entity/statement/>
PREFIX wdv: <http://www.wikidata.org/value/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX p: <http://www.wikidata.org/prop/>
PREFIX ps: <http://www.wikidata.org/prop/statement/>
PREFIX pq: <http://www.wikidata.org/prop/qualifier/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bd: <http://www.bigdata.com/rdf#>

# The below SELECT query does the following:
# Selects all the items(?s subjects) and their labels(?label)
# that have(WHERE) the statement(?s subject) has a direct property(wdt:) = P279 <subclasses of>
# with a value of entity(wd:) = Q7725634 <Literary Work>
# and optionally return the English labels

SELECT ?s ?label WHERE {
  ?s wdt:P279 wd:Q7725634 .
  OPTIONAL {
    ?s rdfs:label ?label FILTER (LANG(?label) = "en") .
  }
}
```

Extensions

The service supports the following extensions to standard SPARQL capabilities:

Label service

You can fetch the label, alias, or description of entities you query, with language fallback, using the specialized service with the URI <http://wikiba.se/ontology#label>. The service is very helpful when you want to retrieve labels, as it reduces the complexity of SPARQL queries that you would otherwise need to achieve the same effect.

The service can be used in one of the two modes: manual and automatic.

In automatic mode, you only need to specify the service template, e.g.:

```
PREFIX wikibase: <http://wikiba.se/ontology#>

SERVICE wikibase:label {
  bd:serviceParam wikibase:language "en" .
}
```

and WDQS will automatically generate labels as follows:

- If an unbound variable in `SELECT` is named `?`*`NAME`*`Label`, then WDQS produces the label (`rdfs:label`) for the entity in variable `?`*`NAME`*.
- If an unbound variable in `SELECT` is named `?`*`NAME`*`AltLabel`, then WDQS produces the alias (`skos:altLabel`) for the entity in variable `?`*`NAME`*.
- If an unbound variable in `SELECT` is named `?`*`NAME`*`Description`, then WDQS produces the description (`schema:description`) for the entity in variable `?`*`NAME`*.

In each case, the variable in `?`*`NAME`* should be bound, otherwise the service fails.

Automatic mode only inspects the projection of the query – for instance, in `SELECT ?aLabel (GROUP_CONCAT(?bLabel) AS ?bLabels)`, only the first label is recognized, and `SELECT *` is not supported by automatic mode at all. In such cases, you will need to use manual mode (see below).

You specify your preferred language(s) for the label with one or more of `bd:serviceParam wikibase:language "`*`language-code`*`"` triples. Each string can contain one or more language codes, separated by commas. WDQS considers languages in the order in which you specify them. If no label is available in any of the specified languages, the Q-id of the entity (without any prefix) is its label.

The Wikidata Query Service website automatically replaces `[AUTO_LANGUAGE]` with the language code of current user's interface. For example, if the user's UI is in French, the SPARQL's code `bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en"` will be converted to `bd:serviceParam wikibase:language "fr,en"` before being sent to the query service.

Example, showing the list of US presidents and their spouses:

```
SELECT ?p ?pLabel ?w ?wLabel WHERE {
  wd:Q30 p:P6/ps:P6 ?p .
  ?p wdt:P26 ?w .
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en" .
  }
}
```

Try it! (https://query.wikidata.org/#SELECT%20%3Fp%20%3FpLabel%20%3Fw%20%3FwLabel%20WHERE%20%7B%0A%20%20wd%3AQ30%20p%3AP6%2Fps%3AP6%20%3Fp%20.%0A%20%20%3Fp%20wdt%3AP26%20%3Fw%20.%0A%20%20SERVICE%20wikibase%3Alabel%20%7B%0A%20%20%20%20bd%3AserviceParam%20wikibase%3Alanguage%20%22en%22%20.%0A%20%20%7D%0A%7D)

In this example WDQS automatically creates the labels `?pLabel` and `?wLabel` for properties.

In the manual mode, you explicitly bind the label variables within the service call, but WDQS will still provide language resolution and fallback. Example:

```
SELECT * WHERE {
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "fr,de,en" .
    wd:Q123 rdfs:label ?q123Label ;
            skos:altLabel ?q123Alt ;
            schema:description ?q123Desc .
    wd:Q321 rdf:label ?q321Label .
  }
}
```

This will consider labels and descriptions in French, German and English, and if none are available, will use the Q-id as the label.

Geospatial search

The service allows to search for items with coordinates located within a certain radius of a central point or within a certain bounding box.

Search around point

Example:

```
#title: Airports within 100km from Berlin
#defaultView:Map
SELECT ?place ?placeLabel ?location ?dist WHERE {
  # Berlin coordinates
  wd:Q64 wdt:P625 ?berlinLoc .
  SERVICE wikibase:around {
    ?place wdt:P625 ?location .
    bd:serviceParam wikibase:center ?berlinLoc ;
                    wikibase:radius "100" ;
                    wikibase:distance ?dist .
  }
  # Is an airport
  FILTER EXISTS { ?place wdt:P31/wdt:P279* wd:Q1248784 }
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en" .
  }
} ORDER BY ASC(?dist)
```

Try it! (https://query.wikidata.org/#%23title%3A%20Airports%20within%20100km%20from%20Berlin%0A%23defaultView%3AMap%0ASELECT%20%3Fplace%20%3FplaceLabel%20%3Flocation%20%3Fdist%20WHERE%20%7B%0A%20%20%23%20Berlin%20coordinates%0A%20%20wd%3AQ64%20wdt%3AP625%20%3FberlinLoc%20.%0A%20%20SERVICE%20wikibase%3Aaround%20%7B%0A%20%20%20%20%3Fplace%20wdt%3AP625%20%3Flocation%20.%0A%20%20%20%20bd%3AserviceParam%20wikibase%3Acenter%20%3FberlinLoc%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20wikibase%3Aradius%20%22100%22%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20wikibase%3Adistance%20%3Fdist%20.%0A%20%20%7D%0A%20%20%23%20Is%20an%20airport%0A%20%20FILTER%20EXISTS%20%7B%20%3Fplace%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ1248784%20%7D%0A%20%20SERVICE%20wikibase%3Alabel%20%7B%0A%20%20%20%20bd%3AserviceParam%20wikibase%3Alanguage%20%22en%22%20.%0A%20%20%7D%0A%7D%20ORDER%20BY%20ASC%28%3Fdist%29)

The first line of the `around` service call must have format `?item predicate ?location`, where the result of the search will bind `?item` to items within the specified location and `?location` to their coordinates. The parameters supported are:

Predicate	Meaning
wikibase:center	The point around which search is performed. Must be bound for search to work.
wikibase:radius	Distance from the center. Currently the distance is always in kilometers, other units are not supported yet.
wikibase:globe	The globe which is being searched. Optional, default is Earth (wd:Q2) (https://www.wikidata.org/wiki/Q2).
wikibase:distance	The variable receiving distance information

Search within box

Example of box search:

```
#title: Schools between San Jose, CA and Sacramento, CA
#defaultView:Map
SELECT * WHERE {
  wd:Q16553 wdt:P625 ?SJloc .
  wd:Q18013 wdt:P625 ?SCloc .
  SERVICE wikibase:box {
    ?place wdt:P625 ?location .
    bd:serviceParam wikibase:cornerSouthWest ?SJloc ;
                    wikibase:cornerNorthEast ?SCloc .
  }
  FILTER EXISTS { ?place wdt:P31/wdt:P279* wd:Q3914 }
}
```

Try it! (https://query.wikidata.org/#%23title%3A%20Schools%20between%20San%20Jose%2C%20CA%20and%20Sacramento%2C%20CA%0A%23defaultView%3AMap%0ASELECT%20%2A%20WHERE%20%7B%0A%20%20wd%3AQ16553%20wdt%3AP625%20%3FSJloc%20.%0A%20%20wd%3AQ18013%20wdt%3AP625%20%3FSCloc%20.%0A%20%20SERVICE%20wikibase%3Abox%20%7B%0A%20%20%20%20%3Fplace%20wdt%3AP625%20%3Flocation%20.%0A%20%20%20%20bd%3AserviceParam%20wikibase%3AcornerSouthWest%20%3FSJloc%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20wikibase%3AcornerNorthEast%20%3FSCloc%20.%0A%20%20%7D%0A%20%20FILTER%20EXISTS%20%7B%20%3Fplace%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ3914%20%7D%0A%7D)

or:

```
#title: Schools between San Jose, CA and San Francisco, CA
#defaultView:Map
SELECT ?place ?location WHERE {
  wd:Q62 wdt:P625 ?SFloc .
  wd:Q16553 wdt:P625 ?SJloc .
  SERVICE wikibase:box {
    ?place wdt:P625 ?location .
    bd:serviceParam wikibase:cornerWest ?SFloc ;
                    wikibase:cornerEast ?SJloc .
  }
  FILTER EXISTS { ?place wdt:P31/wdt:P279* wd:Q3914 }
}
```

Try it! (https://query.wikidata.org/#%23title%3A%20Schools%20between%20San%20Jose%2C%20CA%20and%20San%20Francisco%2C%20CA%0A%23defaultView%3AMap%0ASELECT%20%3Fplace%20%3Flocation%20WHERE%20%7B%0A%20%20wd%3AQ62%20wdt%3AP625%20%3FSFloc%20.%0A%20%20wd%3AQ16553%20wdt%3AP625%20%3FSJloc%20.%0A%20%20SERVICE%20wikibase%3Abox%20%7B%0A%20%20%20%20%3Fplace%20wdt%3AP625%20%3Flocation%20.%0A%20%20%20%20bd%3AserviceParam%20wikibase%3AcornerWest%20%3FSFloc%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20wikibase%3AcornerEast%20%3FSJloc%20.%0A%20%20%7D%0A%20%20FILTER%20EXISTS%20%7B%20%3Fplace%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ3914%20%7D%0A%7D)

Coordinates may be specified directly:

```
#title: Schools between San Jose, CA and Sacramento, CA
#same as previous
#defaultView:Map
SELECT * WHERE {
  SERVICE wikibase:box {
    ?place wdt:P625 ?location .
    bd:serviceParam wikibase:cornerWest "Point(-121.872777777 37.304166666)"^^geo:wktLiteral ;
                    wikibase:cornerEast "Point(-121.486111111 38.575277777)"^^geo:wktLiteral .
  }
  FILTER EXISTS { ?place wdt:P31/wdt:P279* wd:Q3914 }
}
```

Try it! (https://query.wikidata.org/#%23title%3A%20Schools%20between%20San%20Jose%2C%20CA%20and%20Sacramento%2C%20CA%0A%23same%20as%20previous%0A%23defaultView%3AMap%0ASELECT%20%2A%20WHERE%20%7B%0A%20%20SERVICE%20wikibase%3Abox%20%7B%0A%20%20%20%20%3Fplace%20wdt%3AP625%20%3Fl ocation%20.%0A%20%20%20%20bd%3AserviceParam%20wikibase%3AcornerWest%20%22Point%28-121.872777777%2037.304166666%29%22%5E%5Egeo%3AwktLiteral%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20wikibase%3AcornerEast%20%22Point%28-121.486111111%2038.575277777%29%22%5E%5Egeo%3AwktLiteral%20.%0A%20%20%7D%0A%20%20FILTER%20EXISTS%20%7B%20%3Fplace%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ3914%20%7D%0A%7D)

The first line of the box service call must have format `?item predicate ?location`, where and the result of the search will bind `?item` to items within the specified location and `?location` to their coordinates. The parameters supported are:

Predicate	Meaning
wikibase:cornerSouthWest	The south-west corner of the box.
wikibase:cornerNorthEast	The north-east corner of the box.
wikibase:cornerWest	The western corner of the box.
wikibase:cornerEast	The eastern corner of the box.
wikibase:globe	The globe which is being searched. Optional, default is Earth (wd:Q2) (https://www.wikidata.org/wiki/Q2).

`wikibase:cornerSouthWest` and `wikibase:cornerNorthEast` should be used together, as well as `wikibase:cornerWest` and `wikibase:cornerEast`, and cannot be mixed. If `wikibase:cornerWest` and `wikibase:cornerEast` predicates are used, then the points are assumed to be the coordinates of the diagonal of the box, and the corners are derived accordingly.

Extended functions

Distance function

The function `geof:distance` returns distance between two points on Earth, in kilometers. Example usage:

```
#title: Airports within 100km from Berlin
SELECT ?place ?placeLabel ?location ?dist WHERE {
  # Berlin coordinates
  wd:Q64 wdt:P625 ?berlinLoc .
  SERVICE wikibase:around {
    ?place wdt:P625 ?location .
    bd:serviceParam wikibase:center ?berlinLoc ;
                    wikibase:radius "100" .
  }
  # Is an airport
  ?place wdt:P31/wdt:P279* wd:Q1248784 .
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en" .
  }
  BIND(geof:distance(?berlinLoc, ?location) AS ?dist) .
} ORDER BY ?dist
```

Try it! (https://query.wikidata.org/#%23title%3A%20Airports%20within%20100km%20from%20Berlin%0ASELECT%20%3Fplace%20%3FplaceLabel%20%3Flocation%20%3Fdist%20WHERE%20%7B%0A%20%20%23%20Berlin%20coordinates%0A%20%20wd%3AQ64%20wdt%3AP625%20%3FberlinLoc%20.%0A%20%20SERVICE%20wikibase%3Aaround%20%7B%0A%20%20%20%20%3Fplace%20wdt%3AP625%20%3Flocation%20.%0A%20%20%20%20bd%3AserviceParam%20wikibase%3Acenter%20%3FberlinLoc%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20wikibase%3Aradius%20%22100%22%20.%0A%20%20%7D%0A%20%20%23%20Is%20an%20airport%0A%20%20%3Fplace%20wdt%3AP31%2Fwdt%3AP279%2A%20wd%3AQ1248784%20.%0A%20%20SERVICE%20wikibase%3Alabel%20%7B%0A%20%20%20%20bd%3AserviceParam%20wikibase%3Alanguage%20%22en%22%20.%0A%20%20%7D%0A%20%20BIND%28geof%3Adistance%28%3FberlinLoc%2C%20%3Flocation%29%20AS%20%3Fdist%29%20.%0A%7D%20ORDER%20BY%20%3Fdist)

```
#title: Places around 0°,0°
SELECT * WHERE {
  SERVICE wikibase:around {
    ?place wdt:P625 ?location .
    bd:serviceParam wikibase:center "Point(0 0)"^^geo:wktLiteral ;
```

```
                    wikibase:radius "250" .
  }
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en" .
    ?place rdfs:label ?placeLabel .
  }
  BIND(geof:distance("Point(0 0)"^^geo:wktLiteral, ?location) AS ?dist) .
} ORDER BY ?dist
```

Try it! (https://query.wikidata.org/#%23title%3A%20Places%20around%200%C2%B0%2C0%C2%B0%0ASELECT%20%2A%20WHERE%20%7B%0A%20%20SERVICE%20wikibase%3Aaround%20%7B%0A%20%20%20%20%3Fplace%20wdt%3AP625%20%3Flocation%20.%0A%20%20%20%20bd%3AserviceParam%20wikibase%3Acenter%20%22Point%280%200%29%22%5E%5Egeo%3AwktLiteral%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20wikibase%3Aradius%20%22250%22%20.%0A%20%20%7D%0A%20%20SERVICE%20wikibase%3Alabel%20%7B%0A%20%20%20%20bd%3AserviceParam%20wikibase%3Alanguage%20%22en%22%20.%0A%20%20%20%20%3Fplace%20rdfs%3Alabel%20%3FplaceLabel%20.%0A%20%20%7D%0A%20%20BIND%28geof%3Adistance%28%22Point%280%200%29%22%5E%5Egeo%3AwktLiteral%2C%20%3Flocation%29%20AS%20%3Fdist%29%20.%0A%7D%20ORDER%20BY%20%3Fdist)

Coordinate parts functions

Functions `geof:globe`, `geof:latitude` & `geof:longitude` return parts of a coordinate - globe URI, latitude and longitude accordingly.

Decode URL functions

Function `wikibase:decodeUri` decodes (i.e. reverses percent-encoding) given URI string. This may be necessary when converting Wikipedia titles (which are encoded) into actual strings. This function is an opposite of SPARQL encode_for_uri (https://www.w3.org/TR/sparql11-query/#func-encode) function.

```
#title: Example usage of wikibase:decodeUri function
SELECT DISTINCT * WHERE {
  ?el wdt:P31 wd:Q5 .
  ?webRaw schema:about ?el ;
          schema:inLanguage "ru" ;
          schema:isPartOf <https://ru.wikipedia.org/> .
  BIND(URI(wikibase:decodeUri(STR(?webRaw))) AS ?webHyperlink) .
  BIND(wikibase:decodeUri(STR(?webRaw)) AS ?webString) .
} LIMIT 20
```

Automatic prefixes

Most prefixes that are used in common queries are supported by the engine without the need to explicitly specify them.

Extended dates

The service supports date values of type `xsd:dateTime` in the range of about 290B years in the past and in the future, with one-second resolution. WDQS stores dates as the 64-bit number of seconds since the Unix epoch.

Blazegraph extensions

Warning: As Blazegraph is not actively developed, it is planned to migrate Wikidata Query Service to another triplestore. The following may be discontinued at some point.

Blazegraph (https://wiki.blazegraph.com/wiki/index.php/About_Blazegraph) platform on top of which WDQS is implemented has its own set of SPARQL extension. Among them several graph traversal algorithms which are documented on Blazegraph Wiki (https://github.com/blazegraph/database/wiki/RDF_GAS_API), including BFS, shortest path, CC and PageRank implementations.

Please also refer to the Blazegraph documentation on query hints (https://github.com/blazegraph/database/wiki/QueryOptimization) for information about how to control query execution and various aspects of the engine.

There is no documentation in the BlazeGraph wiki about the bd:sample extension. It's documented only in a comment in the code (https://github.com/blazegraph/database/blob/3127706f0b6504838daae226b9158840d2df1744/bigdata-core/bigdata-rdf/src/java/com/bigdata/rdf/sparql/ast/eval/SampleServiceFactory.java).

Federation

We allow SPARQL Federated Queries (https://www.w3.org/TR/sparql11-federated-query/) to call out to a selected number of external databases. Please see the full list of federated endpoints on the dedicated page.

Example federated query:

- Items used: Lope de Vega (Q165257) (https://www.wikidata.org/wiki/Q165257)
- Properties used: BVMC person ID (P2799) (https://www.wikidata.org/wiki/Property:P2799)

```
SELECT ?workLabel WHERE {
  wd:Q165257 wdt:P2799 ?id .
  BIND(URI(CONCAT("https://data.cervantesvirtual.com/person/", ?id)) AS ?bvmcID) .
  SERVICE <http://data.cervantesvirtual.com/openrdf-sesame/repositories/data> {
    ?bvmcID <http://rdaregistry.info/Elements/a/otherPFCManifestationOf> ?work .
    ?work rdfs:label ?workLabel .
  }
}
```

Try it! (https://query.wikidata.org/#SELECT%20%3FworkLabel%20WHERE%20%7B%0A%20%20wd%3AQ165257%20wdt%3AP2799%20%3Fid%20.%0A%20%20BIND%28URI%28CONCAT%28%22https%3A%2F%2Fdata.cervantesvirtual.com%2Fperson%2F%22%2C%20%3Fid%29%29%20AS%20%3FbvmcID%29%20.%0A%20%20SERVICE%20%3Chttp%3A%2F%2Fdata.cervantesvirtual.com%2Fopenrdf-sesame%2Frepositories%2Fdata%3E%20%7B%0A%20%20%20%20%3FbvmcID%20%3Chttp%3A%2F%2Frdaregistry.info%2FElements%2Fa%2FotherPFCManifestationOf%3E%20%3Fwork%20.%0A%20%20%20%20%3Fwork%20rdfs%3Alabel%20%3FworkLabel%20.%0A%20%20%7D%0A%7D)

Please note that the databases that the federated endpoints serve use ontologies that may be very different from the Wikidata one. Please refer to the owner documentation links to learn about the ontologies and data access to these databases.

MediaWiki API

Please see full description on MediaWiki API Service documentation page.

MediaWiki API Service allows to call out to MediaWiki API from SPARQL, and receive the results from inside the SPARQL query. Example (finding category members):

- Items used: parking area (Q6501349) (https://www.wikidata.org/wiki/Q6501349)
- Properties used: topic's main category (P910) (https://www.wikidata.org/wiki/Property:P910)

```
SELECT * WHERE {
  wd:Q6501349 wdt:P910 ?category .
  ?link schema:about ?category ;
        schema:isPartOf <https://en.wikipedia.org/> ;
        schema:name ?title .
  SERVICE wikibase:mwapi {
    bd:serviceParam wikibase:api "Generator" ;
                    wikibase:endpoint "en.wikipedia.org" ;
                    mwapi:gcmtitle ?title ;
                    mwapi:generator "categorymembers" ;
                    mwapi:gcmprop "ids|title|type" ;
                    mwapi:gcmlimit "max" .
    # out
    ?subcat wikibase:apiOutput mwapi:title .
    ?ns wikibase:apiOutput "@ns" .
    ?item wikibase:apiOutputItem mwapi:item .
  }
}
```

Try it! (https://query.wikidata.org/#SELECT%20%2A%20WHERE%20%7B%0A%20%20wd%3AQ6501349%20wdt%3AP910%20%3Fcategory%20.%0A%20%20%3Flink%20schema%3Aabout%20%3Fcategory%20%3B%0A%20%20%20%20%20%20%20%20schema%3AisPartOf%20%3Chttps%3A%2F%2Fen.wikipedia.org%2F%3E%20%3B%0A%20%20%20%20%20%20%20%20schema%3Aname%20%3Ftitle%20.%0A%20%20SERVICE%20wikibase%3Amwapi%20%7B%0A%20%20%20%20bd%3AserviceParam%20wikibase%3Aapi%20%22Generator%22%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20wikibase%3Aendpoint%20%22en.wikipedia.org%22%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20mwapi%3Agcmtitle%20%3Ftitle%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20mwapi%3Agenerator%20%22categorymembers%22%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20mwapi%3Agcmprop%20%22ids%7Ctitle%7Ctype%22%20%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20mwapi%3Agcmlimit%20%22max%22%20.%0A%20%20%20%20%23%20out%0A%20%20%20%20%3Fsubcat%20wikibase%3AapiOutput%20mwapi%3Atitle%20.%0A%20%20%20%20%3Fns%20wikibase%3AapiOutput%20%22%40ns%22%20.%0A%20%20%20%20%3Fitem%20wikibase%3AapiOutputItem%20mwapi%3Aitem%20.%0A%20%20%7D%0A%7D)

Wikimedia service

Wikimedia runs the public service instance of WDQS, which is available for use at http://query.wikidata.org/.

The runtime of the query on the public endpoint is limited to 60 seconds. That is true both for the Web interface and the public SPARQL endpoint.

Web application

The Web application at the home page of http://query.wikidata.org/ allows you to edit and submit SPARQL queries to the query engine. The results are displayed as an HTML table. Note that every query has a unique URL which can be bookmarked for later use. Going to this URL will put the query in the edit window, but will **not** run it - you still have to click "Execute" for that.

One can also generate a short URL for the query via a URL shortening service by clicking the "Generate short URL" link on the right - this will produce the shortened URL for the current query.

The "Add prefixes" button generates the header containing standard prefixes for SPARQL queries. The full list of prefixes that can be useful is listed in the RDF format documentation. Note that most common prefixes work automatically, since WDQS supports them out of the box.

The Web application also features a simple entity explorer which can be activated by clicking on the "🔍" symbol next to the entity result. Clicking on the entity Q-id itself will take you to the entity page on wikidata.org.

Default views

Main article: Wikidata:SPARQL query service/Wikidata Query Help/Result Views

If you run the query in the WDQS Web application, you can choose which view to present by specifying the tag: `#defaultView:viewName` at the beginning of the query.

Display a title

If you run the query in the WDQS Web application, you can display a title on top of the results by specifying the tag: `#title: String you want to display` at the beginning of the query.

SPARQL endpoint

SPARQL queries can be submitted directly to the SPARQL endpoint with a GET or POST request to `https://query.wikidata.org/sparql`.

GET requests have the query specified in the URL, in the format `https://query.wikidata.org/sparql?query=(SPARQL_query)`, e.g. `https://query.wikidata.org/sparql?query=SELECT%20?dob%20WHERE%20{wd:Q42%20wdt:P569%20?dob.}`.

POST requests can alternatively accept the query in the body of the request, instead of the URL, which allows running larger queries without hitting URL length limits. (Note that the POST body must still include the `query=` prefix (that is, it should be `query=(SPARQL_query)` rather than just `(SPARQL query)`), and the SPARQL query must still be URL-escaped.)

The result is returned as XML by default, or as JSON if either the query parameter `format=json` is included in the URL, or the header `Accept: application/sparql-results+json` is provided with the request.

The JSON format is standard SPARQL 1.1 Query Results JSON Format (https://www.w3.org/TR/sparql11-results-json/).

It is recommended to use GET for smaller queries and POST for larger queries, as POST queries are not cached.

Supported formats

The following output formats are currently supported by the SPARQL endpoint:

Format	HTTP Header	Query parameter	Description
XML	Accept: application/sparql-results+xml	format=xml	XML result format, is returned by default. As specified in https://www.w3.org/TR/rdf-sparql-XMLres/
JSON	Accept: application/sparql-results+json	format=json	JSON result format, as in: https://www.w3.org/TR/sparql11-results-json/
TSV	Accept: text/tab-separated-values		As specified in https://www.w3.org/TR/sparql11-results-csv-tsv/
CSV	Accept: text/csv		As specified in https://www.w3.org/TR/sparql11-results-csv-tsv/
Binary RDF	Accept: application/x-binary-rdf-results-table		

Query limits

There is a hard query deadline configured which is set to **60 seconds**. There are also following limits:

- One client (user agent + IP) is allowed 60 seconds of processing time each 60 seconds
- One client is allowed 30 error queries per minute

Clients exceeding the limits above are throttled with HTTP code `429`. Use `Retry-After` header to see when the request can be repeated. If the client ignores `429` responses and continues to produce requests over the limits, it can be temporarily banned from the service. Clients who don't comply with the User-Agent policy may be blocked completely – make sure to send a good `User-Agent` header.

Every query will timeout when it takes more time to execute than this configured deadline. You may want to optimize the query or report a problematic query here.

Also note that currently access to the service is limited to 5 parallel queries per IP. The above limits are subject to change depending on resources and usage patterns.

Explain Query

Blazegraph allows to show query analysis that explains how the query has been parsed and which optimizations were applied. To see this information, add `explain=details` parameter to the query string, for example: `https://query.wikidata.org/sparql?query=SELECT%20?dob%20WHERE%20{wd:Q42%20wdt:P569%20?dob.}&explain=details`.

Namespaces

The data on Wikidata Query Service contains the main namespace, `wdq`, to which queries to the main SPARQL endpoint are directed, and other auxiliary namespaces, listed below. To query data from different namespace, use endpoint URL https://query.wikidata.org/bigdata/namespace/NAMESPACENAME/sparql.

Categories

Please see full description on Categories documentation page.

Wikidata Query Service also provides access to the category graph of select wikis. The list of covered wikis can be seen here: https://noc.wikimedia.org/conf/dblists/categories-rdf.dblist

The category namespace name is `categories`. The SPARQL endpoint for accessing it is https://query.wikidata.org/bigdata/namespace/categories/sparql.

Please see Categories page for detailed documentation.

DCAT-AP

The DCAT-AP data for Wikidata is available as SPARQL at https://query.wikidata.org/bigdata/namespace/dcatap/sparql endpoint.

The source for the data is: https://dumps.wikimedia.org/wikidatawiki/entities/dcatap.rdf

Example query to retrieve data:

```
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>

SELECT ?url ?date ?size WHERE {
  <https://www.wikidata.org/about#catalog> dcat:dataset ?dump .
  ?dump dcat:distribution [
    dct:format "application/json" ;
    dcat:downloadURL ?url ;
    dct:issued ?date ;
    dcat:byteSize ?size
  ] .
}
```

Linked Data Fragments endpoint

We also support querying the database using Triple Pattern Fragments (http://linkeddatafragments.org/) interface. This allows to cheaply and efficiently browse triple data where one or two components of the triple is known and you need to retrieve all triples that match this template. See more information at the Linked Data Fragments site (http://linkeddatafragments.org/concept/).

The interface can be accessed by the URL: `https://query.wikidata.org/bigdata/ldf (https://query.wikidata.org/bigdata/ldf)`. This service is implemented on the top of Blazegraph database, so it will have the same lag as the Query Service. Example requests:

- https://query.wikidata.org/bigdata/ldf?subject=http%3A%2F%2Fwww.wikidata.org%2Fentity%2FQ146 - all triples with subject cat (Q146)

- https://query.wikidata.org/bigdata/ldf?subject=&predicate=http%3A%2F%2Fwww.w3.org%2F2000%2F01%2Frdf-schema%23label&object=%22London%22%40en - all triples that have English label "London"
- https://query.wikidata.org/bigdata/ldf?predicate=http%3A%2F%2Fwww.wikidata.org%2Fprop%2Fdirect%2FP212&object=%22978-0-262-03293-3%22 All triples that have `978-0-262-03293-3` as the value for International Standard Book Number-13 (P212). The following shell command uses `curl` to build the same URL and obtain the same data.

```
$ curl \
  --get \
  -H 'Accept: application/rdf+xml' \
  --data-urlencode 'predicate=http://www.wikidata.org/prop/direct/P212' \
  --data-urlencode 'object="978-0-262-03293-3"' \
  'https://query.wikidata.org/bigdata/ldf'
```

Note that only full URLs are currently supported for the `subject`, `predicate` and `object` parameters.

By default, HTML interface is displayed, however several data formats are available, defined by `Accept` HTTP header.

Available data formats

Accept	Format
text/html	Default HTML browsing interface
text/turtle	Turtle (https://www.w3.org/TR/turtle/) format
application/ld+json	JSON-LD (https://www.w3.org/TR/json-ld/) format
application/n-triples	N-Triples (https://www.w3.org/TR/n-triples/) format
application/rdf+xml	RDF/XML (https://www.w3.org/TR/REC-rdf-syntax/) format

The data is returned in pages, page size being 100 triples. The pages are numbered starting from 1, and page number is defined by `page` parameter.

Standalone service

As the service is open source software, it is also possible to run the service on any user's server, by using the instructions provided below.

The hardware recommendations can be found in Blazegraph documentation (https://wiki.blazegraph.com/wiki/index.php/Hardware_Configuration).

If you plan to run the service against non-Wikidata Wikibase instance, please see further instructions.

Installing

In order to install the service, it is recommended that you download the full service package as a ZIP file, e.g. from Maven Central (http://search.maven.org/#search%7Cgav%7C1%7Cg%3A%22org.wikidata.query.rdf%22%20AND%20a%3A%22service%22), with group ID `org.wikidata.query.rdf` and artifact ID `service` , or clone the source distribution at https://github.com/wikimedia/wikidata-query-rdf/ and build it with "mvn package". The package ZIP will be in the `dist/target` directory under `service-VERSION-dist.zip`.

The package contains the Blazegraph server as a .war application, the libraries needed to run the updater service to fetch fresh data from the wikidata site, and scripts to make various tasks easier. (The GUI (https://gerrit.wikimedia.org/g/wikidata/query/gui/) is no longer included and must be set up separately if desired.)

By default, only the SPARQL endpoint at http://localhost:9999/bigdata/namespace/wdq/sparql is configured, and the default Blazegraph GUI is available at http://localhost:9999/bigdata/. Note that in the default configuration, both are accessible only from localhost. You will need to provide external endpoints and an appropriate access control if you intend to access them from outside.

Using snapshot versions

If you want to install an un-released snapshot version (usually this is necessary if released version has a bug which is fixed but new release is not available yet) and do not want to compile your own binaries, you can use either:

- https://github.com/wikimedia/wikidata-query-deploy - deployment repo containing production binaries. Needs `git fat` working. Check it out and do "`git fat pull`".
- Archiva snapshot deployments at https://archiva.wikimedia.org/#artifact/org.wikidata.query.rdf/service - choose the latest version, then Artifacts, and select the latest package for download.

Loading data

Warning: As of 2020, in Wikimedia servers it will take about 12 days to get all data in the dump imported, and another 12 days to make the query service catching up the lag.

Further install procedure is described in detail in the Getting Started document (https://github.com/wikimedia/wikidata-query-rdf/blob/master/docs/getting-started.md) which is part of the distribution, and involves the following steps:

1. Download recent RDF dump from https://dumps.wikimedia.org/wikidatawiki/entities/ (the RDF one is the one ending in `.ttl.gz`).
2. Pre-process data with the `munge.sh` script. This creates a set of TTL files with preprocessed data, with names like `wikidump-000000001.ttl.gz`, etc. See options for the script below.
3. Start Blazegraph service by running the `runBlazegraph.sh` script.
4. Load the data into the service by using `loadData.sh (https://github.com/wikimedia/wikidata-query-rdf/blob/master/dist/src/script/loadData.sh)`. Note that loading data is usually significantly slower than pre-processing, so you can start loading as soon as several preprocessed files are ready. Loading can be restarted from any file by using the options as described below.
5. After all the data is loaded, start the Updater service by using `runUpdate.sh`.

Loading categories

If you also want to load category data, please do the following:

1. Create namespace, e.g. `categories`: `createNamespace.sh categories`
2. Load data into it: `forAllCategoryWikis.sh loadCategoryDump.sh categories`

Note that these scripts only load data from Wikimedia wikis according to Wikimedia settings. If you need to work with other wiki, you may need to change some variables in the scripts.

Scripts

The following useful scripts are part of the distribution:

munge.sh

Pre-process data from RDF dump for loading.

Option	Required?	Explanation
-f *filename*	Required	Filename of the RDF dump
-d *directory*	Optional	Directory where the processed files will be written, default is current directory
-l *language*	Optional	If specified, only labels for the given language will be retained. Use this option if you need only one language, as it may improve performance, reduce the database size and simplify queries.
-s	Optional	If specified, the data about sitelinks is excluded. Use this option if you do not need to query sitelinks, as this may improve performance and reduce the database size.
-c *size*	Optional	Use this option to override the default chunk size. Too big chunks may time out when importing.

Example:

```
./munge.sh -c 50000 -f data/wikidata-20150427-all-BETA.ttl.gz -d data -l en -s
```

loadData.sh (https://github.com/wikimedia/wikidata-query-rdf/blob/master/dist/src/script/loadData.sh)

Load processed data into Blazegraph. Requires `curl` to be installed.

Option	Required?	Explanation
-n *namespace*	Required	Specifies the graph namespace into which the data is loaded, which should be **wdq** for WDQS data
-d *directory*	Optional	Directory where processed files are stored, by default the current directory
-h *host*	Optional	Hostname of the SPARQL endpoint, by default localhost
-c *context*	Optional	Context URL of the SPARQL endpoint, by default bigdata - usually doesn't need to be changed for WDQS
-s *start*	Optional	Number of the processed file to start with, by default 1
-e *end*	Optional	Number of the processed file to end with

Example:

```
./loadData.sh -n wdq -d `pwd`/data
```

runBlazegraph.sh

Run the Blazegraph service.

Option	Required?	Explanation
-d *directory*	Optional	Home directory of the Blazegraph installation, by default the same directory where the script is
-c *context*	Optional	Context URL of the SPARQL endpoint, by default bigdata - usually doesn't need to be changed for WDQS
-p *port*	Optional	Port number of the SPARQL service, by default 9999
-o *options*	Optional	Add options to the command line

Example:

```
./runBlazegraph.sh
```

Inside the script, there are two variables that one may want to edit:

```
# Q-id of the default globe
DEFAULT_GLOBE=2
# Blazegraph HTTP User Agent for federation
USER_AGENT="Wikidata Query Service; https://query.wikidata.org/";
```

Also, the following environment variables are checked by the script (all of them are optional):

Variable	Default	Explanation
HOST	localhost	Hostname for binding the Blazegraph service
PORT	9999	Port for binding the Blazegraph service
DIR	*directory where the script is located*	Directory where config files are stored
HEAP_SIZE	16g	Java heap size for Blazegraph
MEMORY	`-Xms${HEAP_SIZE} -Xmx${HEAP_SIZE}`	Full Java memory settings for Blazegraph
GC_LOGS	*see the source (https://github.com/wikimedia/wikidata-query-rdf/blob/master/dist/src/script/runBlazegraph.sh)*	GC logging settings
CONFIG_FILE	RWStore.properties	Blazegraph configuration file location
BLAZEGRAPH_OPTS	*empty*	Additional options, are passed as-is to the Java command line

runUpdate.sh

Run the Updater service.

Option	Required?	Explanation
-n *namespace*	Optional	Specifies the graph namespace into which the data is loaded, should be **wdq** for WDQS data. Default: **wdq**
-h *host*	Optional	Hostname of the SPARQL endpoint, by default localhost
-c *context*	Optional	Context URL of the SPARQL endpoint, by default bigdata - usually doesn't need to be changed for WDQS
-l *language*	Optional	If specified, only labels for given language will be retained. Use this option if you need only one language, as it may improve performance, reduce the database size and simplify queries.
-s	Optional	If specified, the data about sitelinks is excluded. Use this option of you do not need to query sitelinks, as this may improve performance and reduce the database size.
-t *secs*	Optional	Timeout when communicating to Blazegraph, in seconds.
-N	Optional	This option causes the script to ignore options that may cause problems when running second Updater while the first is already running. Use it when running secondary Updater e.g. to catch up specific item with `--ids` (see below).
-S	Optional	Log to console instead of log files. Useful when running the script from the command line for maintenance tasks.

It is recommended that the settings for the `-l` and `-s` options (or absence thereof) be the same for munge.sh and runUpdate.sh, otherwise data may not be updated properly.

Example:

```
./runUpdate.sh
```

Also, the following environment variables are checked by the script (all of them are optional):

Variable	Default	Explanation
UPDATER_OPTS	*empty*	Additional options, are passed as-is to the Java command line

Updater options

The following options works with Updater app.

They should be given to the `runUpdate.sh` script as additional options after `--`, e.g.: `runUpdate.sh -- -v`.

Options for the Updater

Option	Long option	Meaning
-v	--verbose	Verbose mode
-s TIMESTAMP	--start TIMESTAMP	Start data collection from certain timestamp, in `2015-02-11T17:11:08Z` or `20150211170100` format.
	--keepTypes	Keep all type statements
	--ids ID1,ID2,...	Update certain IDs and exit
	--idrange ID1-ID2	Update range of IDs and exit
-d SECONDS	--pollDelay SECONDS	How long to sleep when no new data is available
-t NUMBER	--threadCount NUMBER	How many threads to use when fetching Wikibase data
-b NUMBER	--batchSize NUMBER	How many changes to fetch from RecentChanges API
-V	--verify	Verify data after loading (SLOW! For debug use only)
-T SECONDS	--tailPollerOffset SECONDS	Use secondary trailing poller with given offset behind the main one
	--entityNamespaces NUMBER,NUMBER,...	List of Wikibase namespaces to check for changes
-W	--wikibaseUrl URL	URL to use when talking to Wikibase. E.g. `https://www.wikidata.org`. Must be set if updates do not come from Wikidata.
-U	--conceptUri URL	URL base of the URLs used to represent Wikibase entities in RDF. E.g. `http://www.wikidata.org`. Must be set if the Wikibase instance does not use Wikidata-based prefixes.
	--commonsUri URL	Commons URI for SDC support. Both `--conceptUri` and `--commonsUri` should be set for this to work.
	--wikibaseScheme SCHEME	URL scheme (http, https) to use when talking to Wikibase. (deprecated) Use wikibaseUrl above instead.
	--wikibaseHost HOSTNAME	Hostname to use when talking to Wikibase. (deprecated) Use wikibaseUrl above instead.
-I	--init	If specified together with start time, this time is marked in the database as most recent change time, and future requests would be using it as starting point even if no newer data has been found.
	--constraints	Load constraints violations for updated items via `constraintsrdf` API.
-K SERVERS	--kafka SERVERS	If specified, Updater will use Kafka as the update source and the specified servers as brokers. E.g. `kafka1001.eqiad.wmnet:9092,kafka1002.eqiad.wmnet:9092,kafka1003.eqiad.wmnet:9092`
-C NAME	--consumer NAME	Kafka consumer name. It is a good idea to set it to the host name or something based on it.
-c NAME1,NAME2	--cluster NAME1,NAME2	Kafka cluster names. If specified, the Kafka topic names will be prefixed by cluster names, so instead of `topic1` there would be `NAME1.topic1` and `NAME2.topic2`.
	--resetKafka	Reset Kafka offsets
	--apiPath	Path to the API, e.g. `/w/api.php` (default) or `/api.php`. (introduced in I78c222fe8b)
	--entityDataPath	Path to the entities, e.g. `/wiki/Special:EntityData/` (default) or `/Special:EntityData/`. (introduced in I78c222fe8b)

Configurable properties

The following properties are configurable via adding them to the script run command in the scripts above:

Name	Meaning	Default
wikibaseServiceWhitelist	Filename of remote service whitelist. Applies to Blazegraph.	whitelist.txt
org.wikidata.query.rdf.blazegraph.mwapi.MWApiServiceFactory.config	Config file for MWAPI integration	mwservices.json
com.bigdata.rdf.sail.sparql.PrefixDeclProcessor.additionalDeclsFile	Filename that contains additional prefix definitions. The syntax is as in SPARQL query. These definitions will be loaded by Blazegraph and available for all queries.	
wikibaseConceptUri	URL prefix for Wikibase data, which is used in RDF representation of entities. Needs to be set if the dataset does not use Wikidata prefixes.	http://www.wikidata.org
commonsConceptUri	URL prefix for Structured Data on Commons support. Both `wikibaseConceptUri` and `commonsConceptUri` should be set for it to work.	
wikibaseHost	Hostname of the wikibase instance. Applies to both Blazegraph and Updater. (deprecated) Use wikibaseConceptUri above.	www.wikidata.org
org.wikidata.query.rdf.blazegraph.inline.literal.WKTSerializer.noGlobe	Default globe value for coordinates that have no globe. "2" would mean that entity Q2 is the default globe. "0" means no default globe. Applies to Blazegraph.	0
org.wikidata.query.rdf.tool.rdf.RdfRepository.timeout	Timeout when communicating with RDF repository, in seconds. Applies to Updater.	-1
org.wikidata.query.rdf.tool.wikibase.WikibaseRepository.timeout	Timeout when communicating with wikibase repository, in milliseconds. Applies to Updater.	5000
http.userAgent	User agent that the service would use while calling other services	
http.proxyHost http.proxyPort https.proxyHost https.proxyPort	Proxy settings used while calling other services	
wikibaseMaxDaysBack	How many days back we can request the recent changes data from Updater. If the database is more than this number of days older, it should be reloaded from more recent dump.	30

Missing features

Below are features which are currently not supported:

- Redirects are only represented as owl:sameAs triple, but do not express any equivalence in the data and have no special support.

Contacts

If you notice anything wrong with the service, you can contact the Discovery team by email on the list `discovery@lists.wikimedia.org` or on the IRC channel `#wikimedia-discovery`.

Bugs can also be submitted to Phabricator and tracked on the Discovery Phabricator board.

See also

- WDQ to SPARQL syntax translator
- SPARQL Query examples
- Discovery team
- WDQS Implementation notes
- An introduction to SPARQL query syntax (http://programminghistorian.org/lessons/graph-databases-and-SPARQL)
- RDF dump format