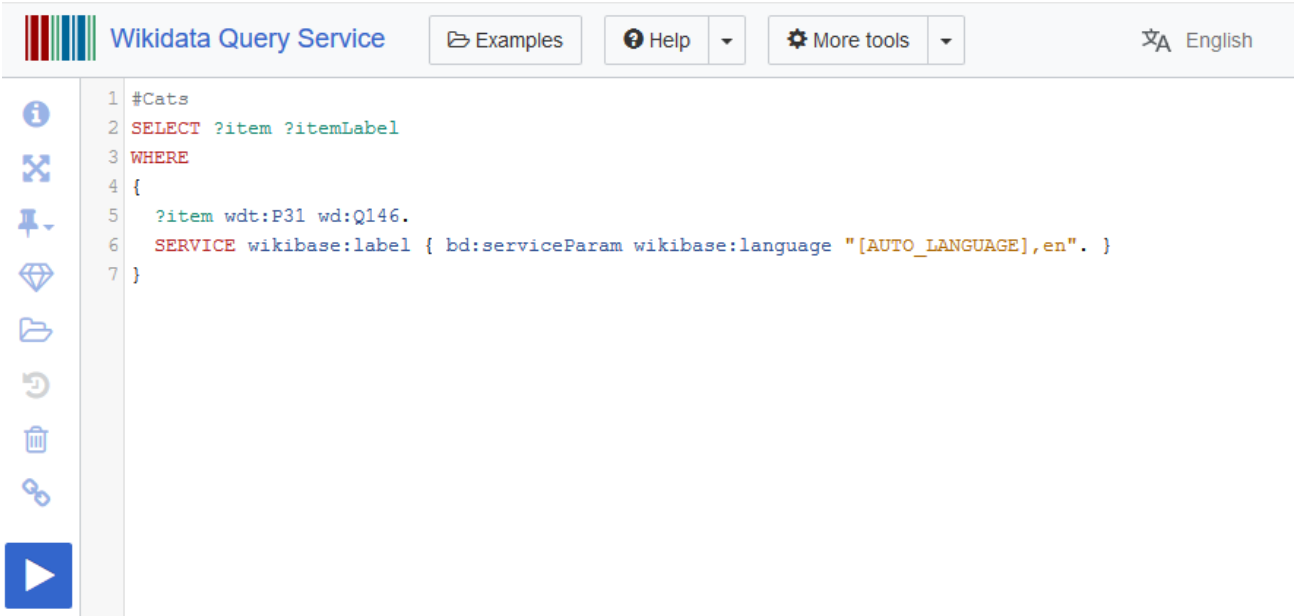
October 23, 2019

Your own Wikidata Query Service, with no limits

The Wikidata Query Service allows anyone to use SPARQL to query the continuously evolving data contained within the Wikidata project, currently standing at nearly 65 millions data items (concepts) and over 7000 properties, which translates to roughly 8.4 billion triples.



You can find a great write up introducing SPARQL, Wikidata, the query service and what it can do here. But this post will assume that you already know all of that.

Guide

Here we will focus on creating a copy of the query service using data from one of the regular TTL data dumps and the query service docker image provided by the wikibase-docker git repo supported by WMDE.

Somewhere to work

Firstly we need a machine to hold the data and do the needed processing. This blog post will use a "n1-highmem-16" (16 vCPUs, 104 GB memory) virtual machine on the Google Cloud Platform with 3 local SSDs held together with RAID 0.

This should provide us with enough fast storage to store the raw TTL data, munged TTL files (where extra triples are added) as well as the journal (JNL) file that the blazegraph query

service uses to store its data.

This entire guide will work on any instance size with more than ~4GB memory and adequate disk space of any speed. For me this setup seemed to be the fastest for the entire process to run.

If you are using the cloud console UI to create an instance then you can use the following options:

- Select Machine type n1-highmem-16 (16 vCPU, 104 GB memory)
- Bootdisk: Google Drawfork Debian GNU / Linux 9
- Firewall (optional): Allow HTTP and HTTPS traffic if you want to access your query service copy externally
- Add disk: Local SSD scratch disk, NVMe (select 3 of them)

If you are creating the instance via the command line your command will look something like the one below. All other defaults should be fine as long as the user you are using can create instances, and you are in the correct project and region.

```
gcloud compute instances create wikidata-query-1 --machine-type=n1-highmem-
```

Initial setup

Setting up my SSDs in RAID

As explained above I will be using some SSDs that need to be setup in a RAID, so I'll start with that following the docs provided by GCE. If you have no need to RAID then skip this step.

```
sudo apt-get update
sudo apt-get install mdadm --no-install-recommends
sudo mdadm --create /dev/md0 --level=0 --raid-devices=3 /dev/nvme0n1 /dev/r
sudo mkfs.ext4 -F /dev/md0
sudo mkdir -p /mnt/disks/ssddata
sudo mount /dev/md0 /mnt/disks/ssddata
sudo chmod a+w /mnt/disks/ssddata
```

Packages & User

Docker will be needed to run the wdqs docker image, so follow the instructions and install that.

This process will require some long running scripts so best to install tmux so that we don't loose where they are.

```
sudo apt-get install tmux
```

Now lets create a user that is in the docker group for us to run our remaining commands via. We will also use the home directory of this user for file storage.

```
sudo adduser sparql
sudo usermod -aG docker sparql
sudo su sparql
```

Then start a tmux session with the following command

```
tmux
```

And download a copy of the wdqs docker image from docker hub.

```
docker pull wikibase/wdqs:0.3.6
```

Here we use version 0.3.6, but this guide should work for future versions just the same (maybe not for 0.4.x+ when that comes out).

Download the TTL dump

The main location to find the Wikidata dumps is on https://dumps.wikimedia.org partially buried in the wikidatawiki/entities/ directory. Unfortunately downloads from here have speed restrictions, so you are better of using one of the mirrors.

I found that the your.org mirror was the fastest from us-east4-c GCE region. To download simply type the following and wait. (At the end of 2018 this was a 47GB download).

Firstly make sure we are in the directory which is backed by our local SSD storage which was mounted above.

```
cd /mnt/disks/ssddata
```

And download the latest TTL dump file which can take around 30 minutes to an hour.

```
wget http://dumps.wikimedia.your.org/wikidatawiki/entities/latest-all.ttl.g
```

Munge the data

UPDATE 2020: To munge quicker using hadoop read this blog post.

Munging the data involves running a script which is packaged with the query service over the TTL dump that we have just downloaded. We will use Docker and the wdqs docker image to do this, starting a new container running bash.

```
docker run --entrypoint=/bin/bash -it --rm -v /mnt/disks/ssddata:/stuff wik
```

And then running the munge script within the internal container bash. This will take around 20 hours and create many files.

```
./munge.sh -c 50000 -f /stuff/latest-all.ttl.gz -d /stuff/mungeOut
```

The -c option was introduced for use in this blog post and allows the chunk size to be selected. If the chunk size is too big you may run into import issues. 50000 is half of the default.

Once the munge has completed you can leave the container that we started and return to the virtual machine.

```
exit
```

Run the service

In order to populate the service we first need to run it using the below command. This mounts the directory containing the munged data as well as the directory for storing the service JNL file in. This will also expose the service on port 9999 which will be writable, so if you don't

want other people to access this check your firewall rules. Don't worry if you don't have a dockerData directory, as it will be created when you run this command.

```
docker run --entrypoint=/runBlazegraph.sh -d \
-v /mnt/disks/ssddata/dockerData:/wdqs/data \
-v /mnt/disks/ssddata:/mnt/disks/ssddata \
-e HEAP_SIZE="128g" \
-p 9999:9999 wikibase/wdqs:0.3.6
```

You can verify that the service is running with curl.

```
curl localhost:9999/bigdata/namespace/wdq/sparql
```

Load the data

Using the ID of the container that is now running, which you can find out by running "docker ps", start a new bash shell alongside the service.

```
docker exec -it friendly_joliot bash
```

Now that we are in the container running our query service we can load the data using the loadData script and the previously munged files.

```
/wdqs/loadData.sh -n wdq -d /mnt/disks/ssddata/mungeOut
```

Once the the data appears to be loaded, you can try out a shorter version of the cats query from the service examples.

```
curl localhost:9999/bigdata/namespace/wdq/sparql?query=%23Cats%0ASELECT%20%
```

Timings

Loading data from a totally fresh TTL dump into a blank query service is not a quick task currently. In production (wikidata.org) it takes roughly a week, and I had a similar experience while trying to streamline the process as best I could on GCE.

For a dump taken at the end of 2018 the timings for each stage were as follows:

- Data dump download: 2 hours
- Data Munge: 20 hours
- Data load: 4.5 days
- Total time: ~5.5 days

Various parts of the process lead me to believe that this could be done faster as throughout CPU usage was pretty low and not all memory was utilized. The loading of the data into blazegraph was by far the slowest step, but digging into this would require someone that is more familiar with the blazegraph internals.