Certain Answers in SPARQL using Maybe Tables
Documentation for experiments

Daniel Hernández, Claudio Gutierrez and Aidan Hogan

This document describes experiments, used settings, and results obtained with that settings. Also, this document can be useful to repeat the experiments with different settings in order to produce comparable results.

Our settings

Machine

We configure the query engines on each machine to correspond with vendor recommendations for memory usage. Our machine has an AMD Opteron Processor 4122, 24GB of RAM, and a single 240 GB Kingston SUV400S SSD disk. The operative system is Debian 8.8.

Engines

We used Virtuoso Open Source Edition 7.2.4.2 and Apache Jena Fuseki 2.6.0 (with Java SE 1.8.0_131).

The configuration of Virtuoso is done as usual in the config.ini file. The following parameters are set:

Argument	Value
NumberOfBuffers	1360000
MaxDirtyBuffers	1000000
ResultSetMaxRows	0
MaxQueryCostEstimationTime	0
MaxQueryExecutionTime	300

In Fuseki parameters are set in the following config.ttl file.

```
@prefix fuseki:  <http://jena.apache.org/fuseki#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix tdb:     <http://jena.hpl.hp.com/2008/tdb#> .
@prefix ja:      <http://jena.hpl.hp.com/2005/11/Assembler#> .
@prefix :        <#> .
```

```
<#service> rdf:type fuseki:Service ;
  fuseki:name                     "ds" ;       # http://host:port/ds
  fuseki:serviceQuery             "sparql" ;   # SPARQL query service
  fuseki:dataset                  <#dataset> .

<#dataset> rdf:type      tdb:DatasetTDB ;
  tdb:location "db" ;
  # Query timeout on this dataset (in milliseconds)
  ja:context [ ja:cxtName "arq:queryTimeout" ; ja:cxtValue "300000" ] .
```

Also, Fuseki is initialized in both machines passing `--Xmx12g` as a `JVM_ARGS` variable, assigning 12GB of RAM to the Java heap.

Environment variables

Several environments variables are required by scripts to locate folders and files. Each parameter can be set with the `export NAME=VALUE` directive of bash. The complete list of parameters is presented in the following table.

`DB_DIR`	Path to store databases.
`DATA_DIR_LOW`	Path to store Turtle files of data instances for Low Level Join queries.
`DATA_DIR_LOW_VARIANT`	Path to store Turtle files of data instances for Low Level Join queries with the variant that we used in Fuseki. This variant adds some extra triples that are not involved in queries. We argue that this variant does not affect the relative performances of queries.
`DATA_DIR_TPCH`	Path to store Turtle files of data instances for queries based in the TPCH-H benchmark.
`VIRTUOSO_HOME`	Path chosen to install Virtuoso.
`VIRTUOSO_INI`	Path to virtuoso.ini.
`JENA_HOME`	Path to chosen to the folder of Fuseki.
`OUTPUT_DIR`	Path where results are stored.

Queries and results

queries.tar.bz2

This compressed folder contains queries, grouped in two folders, namely low and tpch. The former contains queries for the *low level join* dataset. The latter contains queries for the TPC-H based dataset.

avg-time-by-scale.tar.bz2

This compressed folder contains the results of running the experiments in our settings. For each query q and blank rate r there is a TSV file where rows have two columns: the scale factor s and the average elapsed time t of evaluating the query q on a dataset generated with a scale factor s and blank rate r. Also, results are grouped in the folders low-fuseki, low-virtuoso, tpch-fuseki and tpch-virtuoso. For instance, the results for Fuseki and the TPC-H query labeled 4.2, in datasets with blank rate 5 is stored in the file tpch-fuseki/br005-q4.2.tsv.

tpch-fuseki-summary.tsv, tpch-virtuoso-summary.tsv

These tabular files contains data for results of TPC-H based queries. Each file correspond to an specific engine. The columns are: (A) the scale factor, (B) the blank rate, (C) the query, (D–F) the result get on three instances of the query.

tpch-raw-results.tar.bz2

This compressed folder contains the outputs generated by TPC-H based queries. Results are grouped in the folders tpch-fuseki and tpch-virtuoso. Each TPC-H based query is executed for three different set of parameters for each scale factor, blank rate and engine. The results are stored into a JSON file. For instance, the results for Fuseki and the TPC-H query labeled 4.2, in the dataset with scale factor 30 and blank rate 5, is stored in the file tpch-fuseki/ds-sf030-br005-q4.2.json.

Each JSON file is a list containing three dictionaries with the following keys:

`time`	The ellapsed time in seconds.
`query`	The query.
`params`	The parameters used to build the query.
`status`	The HTTP status of the response, e.g., 200.
`doc`	A JSON document if status is 200.
`body`	The response body if status is not 200.
`error`	The type of error.

ASK questions Query 4.1 for the TPC-H based set of queries allows using the approach of running several ASK queries in order to simplify the query resulting of the rewriting. The table below shows the results of these ASK queries for all instances of the dataset (the same results where obtained). Every ASK query was run in milliseconds, even in the the largest dataset instances.

ASK Query	Result
`ASK { [] :s_nation [ :n_name "ETHIOPIA" ] }`	true
`ASK { [] :s_nation [ :n_name "INDIA" ] }`	true
`ASK { [] :s_nation [ :n_name "IRAQ" ] }`	true
`ASK { [] :p_name ?pn . FILTER ( regex(?pn, "burnished") ) }`	true
`ASK { [] :p_name ?pn . FILTER ( regex(?pn, "blush") ) }`	true
`ASK { [] :p_name ?pn . FILTER ( regex(?pn, "bisque") ) }`	true
`ASK { [] :p_name ?pn . FILTER ( isBlank(?pn) ) }`	true
`ASK { [] :s_nation [ :n_name ?nn ] . FILTER ( isBlank(?nn) ) }`	false
`ASK { [] :n_name "ETHIOPIA" }`	true
`ASK { [] :n_name "INDIA" }`	true
`ASK { [] :n_name "IRAQ" }`	true
`ASK { [] :s_nation ?n . FILTER ( isBlank(?n) ) }`	true

Running the experiments

dbgen-low-level-joins.tar.bz2

This compressed folder contains the code to generate the datasets for experiments with Low Level Join queries. The following shell code show how to generate the dataset. Uncompress the folder and run the this code inside.

```
gcc dbgen.c -Wall -O2 -o dbgen
gcc dbgen_variant.c -Wall -O2 -o dbgen_variant
./dbgen.rb 1..10 [1,2,4,8]
```

dbgen-tpch.tar.bz2

This compressed folder contains the code to generate the datasets for queries based in the TPC-H benchmark. This tool requires generating datasets with the TPC-H dbgen tool before using the tool provided by the TPC. We cannot make all TPC-H materials available since the benchmark is protected by copyright and by waiver. However, it can be downloaded from the TPC-H website. In our experiment, we used dbgen version 2.4.0 to generate the data.

The instructions to compile dbgen are also provided in the documentation of the TPC-H package. After compiling it, instances are generated with scale factors of 0.1, 0.3, 0.6, 1, 3, 6 and 10. For instance, you can generate the TPC-H instance of scale factor 0.3 with the following command inside the TPC-H tools folder:

```
./dbgen -s 0.3
mkdir ${DATA_DIR_TPCH}/ds-sf003
mv *.tbl /${DATA_DIR_TPCH}/ds-sf003
```

```
rename s/\.tbl$/-sf003-br000.tbl/ ${DATA_DIR_TPCH}/ds-sf003/*.tbl
gzip /${DATA_DIR_TPCH}/ds-sf003/*.tbl
```

This procedure generates several data files. Each of them corresponds to a relation in the TPC-H model. These relations have no nulls, so we appended the label br000 to files in it.

To generate nulls in the data and to translate these datasets into the RDF data model, the following code is executed for each dataset inside the dbgen-tpch folder.

```
./generate_nulls.rb "${DATA_DIR_TPCH}/**/*.tbl.gz" 1..5
./tbl_to_rdf.rb "${DATA_DIR_TPCH}/**/*.tbl.gz"
```

At this moment, compressed Turtle files will be in the same folder that the generated TBL files.

load.tar.bz2

This compressed folder contains scripts that load the datasets created in the previous step in engine databases. For each dataset will be created a folder storing the files generated by the database engine. The following code lines load all the datasets generated.

```
./load_virtuoso_low.rb 1..10 [1,2,4,8]
./load_fuseki_low.rb 1..10 [1,2,4,8]
./load_virtuoso_tpch.rb [1,3,6,10,30,60,100] 0..10
./load_fuseki_tpch.rb [1,3,6,10,30,60,100] 0..10
```

run-bench.tar.bz2

This compressed folder contains the script to run the benchmarks. The following bash code runs the benchmarks.

```
./run_bench_low.rb virtuoso 1..10 [1,2,4,8]
./run_bench_low.rb fuseki 1..10 [1,2,4,8]
./run_bench_tpch.rb virtuoso [1,3,6,10,30,60,100] 0..5
./run_bench_tpch.rb fuseki [1,3,6,10,30,60,100] 0..5
./csv_serie.rb ${OUTPUT_DIR}/low/virtuoso 1500
./csv_serie.rb ${OUTPUT_DIR}/low/fuseki 1500
./csv_serie.rb ${OUTPUT_DIR}/tpch/virtuoso 600
./csv_serie.rb ${OUTPUT_DIR}/tpch/fuseki 600
```