RAW-JENA: Approximate Query Processing for SPARQL Endpoints

Julien Aimonier-Davat[1], Minh-Hoang Dang[1], Pascal Molli[1], Brice Nédelec[1] and Hala Skaf-Molli[1]

[1]*Nantes Université, CNRS, LS2N, UMR 6004, F-44000 Nantes, France*

Abstract

Sampling-based Approximate Query Processing (S-AQP) has many important use cases for RDF, including computing large-scale statistics, embeddings, join orderings, approximate aggregations, summaries, and exploratory queries. However, current SPARQL endpoints have no support for S-AQP, and many queries just time out on public SPARQL endpoints. In this demonstration, we present RAW-JENA: an extension of Apache Jena to support S-AQP for conjunctive SPARQL queries relying on random walks. RAW-JENA delivers partial random results and cardinality estimates in a pay-as-you-go fashion.

Keywords

SPARQL, Sampling, Approximate query processing, Random walks

1. Introduction

Public SPARQL endpoints cannot fully execute many SPARQL queries due to their quotas and fair-use policies. After 60 seconds, they stop their execution and return only partial results, forbidding many use cases such as building summaries or computing large-scale statistics. This constitutes a major issue for building decentralized ecosystems of public SPARQL endpoints.

Sampling-based Approximate Query Processing (S-AQP) [1] tackles this issue for use cases such as computing large-scale statistics [2, 3], knowledge graph embeddings [4], join orders [5], approximate aggregations [6], summaries [7], and exploratory queries [1]. By confining query execution to samples of large datasets, S-AQP drastically reduces execution time, delivering approximate results with error estimates. To support S-AQP in SPARQL, the evaluation of such queries *must* both (i) return random samples, and (ii) comply with fair-use policies of public SPARQL endpoints. Ad-hoc methods already exist to sample knowledge graphs. For example, a user could draw random triples from Wikidata [2] by repeatedly executing `SELECT * {?s ?p ?o} OFFSET r LIMIT 1`, where r is a random number between 0 and the dataset size ($0 < r < 12B$). However, all queries time out when r is above $100M$. Some triple stores propose home-made methods for sampling triple patterns[1,2], but these solutions are limited to single triple patterns and the underlying complexity is not established.

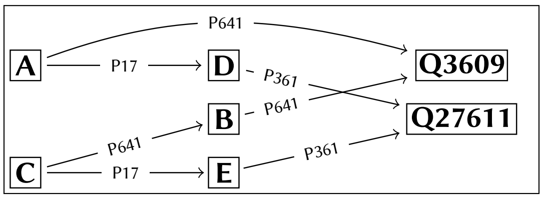
In this demonstration, we introduce RAW-JENA, an open source extension of Apache Jena that efficiently supports S-AQP for conjunctive SPARQL queries. RAW-JENA evaluates a query

[1]https://docs.stardog.com/query-stardog/sampling-service#sampling-service

[2]https://docs.openlinksw.com/virtuoso/rndsalltr/

```
SELECT ?x1 ?x3 WHERE {
  ?x1  wdt:P641 wd:Q3609  .        # tp3
  ?x1  wdt:P17   ?x3          .        # tp2
  ?x3  wdt:P361 wd:Q27611 .        # tp1
}
```

(a) Query Q_1: All cycling races of Central America along with their respective country.



(b) RDF Graph G_1: A, B, and C are cycling sports; D and E are countries of Central America [11]

Figure 1: Query Q_1 evaluated on the RDF graph G_1 provides cardinality insights [11].

Q over a dataset D by drawing random walks guided by Q, as defined in Wander Join [8]. Random walks have two desirable properties: (i) As each random walk is *independent*, a user can distribute its query execution into multiple requests, hence collecting and merging random walks in a pay-as-you-go fashion without impairing the fair-use policy of public SPARQL endpoints. (ii) As each random walk enjoys a *logarithmic time complexity*, endpoints can draw thousands of random walks per second: the widely used BTree indexes of RDF stores allow endpoints to compute each random walk in $\mathcal{O}(|Q| \log |D|)$ where $|Q|$ is the number of triple/quad patterns in Q and $|D|$ is the number of triples/quad in D.

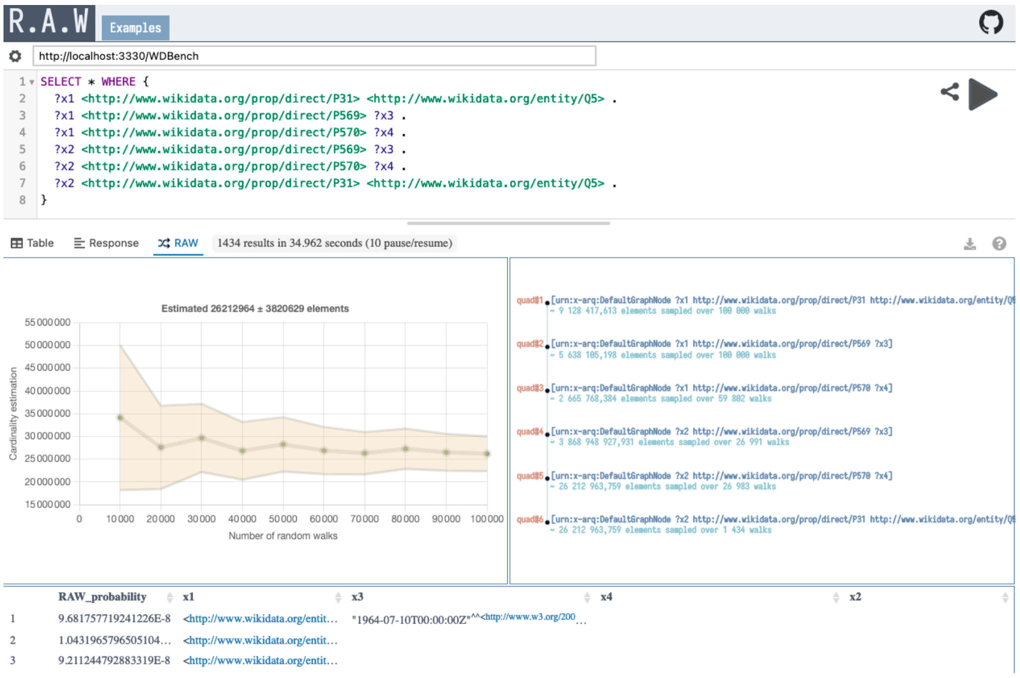
RAW-JENA produces random partial results instead of complete results. Without cardinality estimates, users cannot appreciate the quality of provided samples. In Figure 2, RAW-JENA returns 1434 random results after 35 seconds. By adding an estimate of the total number of results ($26 \pm 3M$), the user acknowledges that her sample represents but a tiny fraction of the whole results. Provided estimates prove highly accurate as long as random walks remain paired with their probability of being drawn [9]. Despite starting from $35 \pm 15M$, RAW-JENA improved over time, reaching $26 \pm 3M$ while the actual number of results is $25M$.

For the demonstration, we load the dump of Wikidata provided by WDBench [10] that comprises $1.2B$ triples. We let users choose a query among the 16 that time out on Apache Jena. We execute it using RAW-JENA to highlight (i) the benefits of S-AQP for ecosystems of public SPARQL endpoints and (ii) the feasibility of implementing S-AQP on well-known engines.

2. RAW-JENA: RAndom Walks for Apache Jena

RAW-JENA is an open source extension of Apache Jena available on the GitHub platform at https://github.com/GDD-Nantes/raw-jena. It efficiently supports Sampling-based Approximate Query Processing (S-AQP) for conjunctive SPARQL queries. This Section provides a formal framework for sampling and applies it on a small toy example; then it details the benefits of RAW-JENA on a larger larger example extracted from WDBench [10]; finally it describes the demonstration protocol for the live session.

Sampling principles. Let Q be a SPARQL conjunctive query, and $J = \langle tp_1, ..., tp_n \rangle$ be the join order to perform random walks. A random walk $\gamma_i = \langle t_1, ..., t_n \rangle$ is computed over an RDF graph G by randomly picking t_1 in $[\![tp_1]\!]_G$, and each subsequent t_i ($i > 1$) in $[\![t_{i-1} \bowtie tp_i]\!]_G$.



http://localhost:3330/WDBench

```
SELECT * WHERE {
  ?x1 <http://www.wikidata.org/prop/direct/P31> <http://www.wikidata.org/entity/Q5> .
  ?x1 <http://www.wikidata.org/prop/direct/P569> ?x3 .
  ?x1 <http://www.wikidata.org/prop/direct/P570> ?x4 .
  ?x2 <http://www.wikidata.org/prop/direct/P569> ?x3 .
  ?x2 <http://www.wikidata.org/prop/direct/P570> ?x4 .
  ?x2 <http://www.wikidata.org/prop/direct/P31> <http://www.wikidata.org/entity/Q5> .
}
```

1434 results in 34.962 seconds (10 pause/resume)

```
quad#1 [urn:x-arq:DefaultGraphNode ?x1 http://www.wikidata.org/prop/direct/P31 http://www.wikidata.org/entity/Q5
       ~ 9 128 417,613 elements sampled over 100 000 walks
quad#2 [urn:x-arq:DefaultGraphNode ?x1 http://www.wikidata.org/prop/direct/P569 ?x3]
       ~ 5 638 105,198 elements sampled over 100 000 walks
quad#3 [urn:x-arq:DefaultGraphNode ?x1 http://www.wikidata.org/prop/direct/P570 ?x4]
       ~ 2 665 768,384 elements sampled over 59 802 walks
quad#4 [urn:x-arq:DefaultGraphNode ?x2 http://www.wikidata.org/prop/direct/P569 ?x3]
       ~ 3 868 948 927,931 elements sampled over 26 991 walks
quad#5 [urn:x-arq:DefaultGraphNode ?x2 http://www.wikidata.org/prop/direct/P570 ?x4]
       ~ 26 212 963,759 elements sampled over 26 983 walks
quad#6 [urn:x-arq:DefaultGraphNode ?x2 http://www.wikidata.org/prop/direct/P31 http://www.wikidata.org/entity/Q5
       ~ 26 212 963,759 elements sampled over 1 434 walks
```

	RAW_probability	x1	x3	x4	x2
1	9.681757719241226E-8	<http://www.wikidata.org/entit…	"1964-07-10T00:00:00Z"^^<http://www.w3.org/200…		
2	1.0431965796505104…	<http://www.wikidata.org/entit…			
3	9.211244792883319E-8	<http://www.wikidata.org/entit…			

Figure 2: Screenshot of RAW-JENA processing the query Q_{604} of WDBench [10].

Once computed, the cardinality of Q is estimated as the inverse probability of sampling γ_i [8], with $P(\gamma_i) = |[\![tp_1]\!]_G|^{-1} \prod_{i=2}^{n} |[\![t_{i-1} \bowtie tp_i]\!]_G|^{-1}$. For instance, let us consider the query Q_1 and the RDF graph G_1 depicted in Figure 1. Following the join order tp_3, tp_2, tp_1, the random walk γ_1 is computed as follows:

$$\begin{array}{c|llll} tp_3 & \text{draw } t_1 & = (\mathbf{A}, P641, Q3609) & \in [\![(?x1, P641, Q3609)]\!]_{G_1} \\ tp_2 & \text{draw } t_2 & = (A, P17, \mathbf{D}) & \in [\![(\mathbf{A}, P17, ?x3)]\!]_{G_1} \\ tp_1 & \text{draw } t_3 & = (D, P361, Q27611) & \in [\![(\mathbf{D}, P361, Q27611)]\!]_{G_1} \end{array}$$

The cardinality of Q_1 is then estimated as the inverse probability of sampling γ_1:
$P(\gamma_1)^{-1} = |[\![(?x1, P641, Q3609)]\!]_{G_1}| \cdot |[\![(\mathbf{A}, P17, ?x3)]\!]_{G_1}| \cdot |[\![(\mathbf{D}, P361, Q27611)]\!]_{G_1}| = 2 \cdot 1 \cdot 1 = 2$
Note that a random walk may fail if it becomes impossible to sample t_i for some $i \leq n$. In this case, its probability $P(\gamma_i)$ of being sampled is 0. For instance, if $t_1 = (B, P641, Q3609)$ is picked in $[\![(?x1, P641, Q3609)]\!]_{G_1}$ instead of $(A, P641, Q3609)$, then the random walk fails because $[\![(B, P17, ?x3)]\!]_{G_1} = \varnothing$. To improve the quality of estimates, we compute a set of k random walks $\Gamma = \langle \gamma_1, ..., \gamma_k \rangle$, and the cardinality of Q is estimated as $|\Gamma|^{-1} \sum_{i=1}^{|\Gamma|} P(\gamma_i)^{-1}$.

RAW-JENA in action (https://youtu.be/We5-rG6uxN8). We executed the query Q_{604} of WDBench [10] presented in Figure 2 that searches for people with the same dates of birth and death. Q_{604} expects $25M$ results, but times out on the public Wikidata SPARQL endpoint after 60 seconds; and takes longer than 2 hours to complete on Apache Jena.

Each time the user presses the play button, RAW-JENA computes either $10k$ random walks or as many as possible before reaching 60 seconds of execution time. By repeating the operation, the web client merges the results iteratively in a pay-as-you-go fashion.
The bottom panel of Figure 2 displays both succeeded and failed random walks γ_i with their respective probability of being drawn $P(\gamma_i)$. Increasing the number of random walks allows for displaying more accurate estimates. The left panel of Figure 2 presents the evolution of cardinality estimates and confidence intervals with respect to the number of random walks. The 1^{st} iteration provides a rough estimate of $35 \pm 15M$ expected results. After 35 seconds of execution time and $100k$ random walks, the 10^{th} iteration provides a more accurate estimate of $26 \pm 3M$ of expected results and 1434 actual random results. The right panel of Figure 2 displays the join order with (i) the estimated cardinality of the partial query up to each triple/quad pattern, and (ii) the number of random walks that reached each triple/quad pattern. Among $100k$ random walks launched on $quad_1$, only 1434 reached $quad_6$. This hints that the join order might be suboptimal since $quad_5$ is very selective and makes random walks fail.
This example shows how S-AQP can be used for exploratory queries and optimization. Nevertheless, S-AQP also provides opportunities for computing large-scale statistics, embeddings, or summaries.

Live demonstration. During the session, we start a local RAW-JENA server on the dataset of WDBench [10] that comprises $1.2B$ triples extracted from Wikidata. We present the Web interface of RAW-JENA depicted in Figure 2. The user can choose a query among a predefined set of conjunctive queries that timed out using Apache Jena; or even type her own.
We show that using RAW-JENA, after a second of execution time, the user already gets meaningful insights on her query execution. Together, we analyze query plan Web view that highlights where random walks succeeded or failed. We deduce a better join order, emphasizing the benefits of S-AQP for this first use case.
Afterwards, we further show the advantages of pay-as-you-go approaches by starting and stopping the query execution multiple times until reaching the 60 seconds timeout mark of public SPARQL endpoints. The Web client manages to receive and aggregate incoming data, thus providing an estimate of the expected number of results. The cardinality estimation view allows users to observe the accuracy evolution over time. The estimates and confidence intervals converge towards the actual number of results of her query. This demonstrates the approximate aggregations use case [6] that proves useful for numerous other use cases such as exploratory queries [1].
Finally, we show that RAW-JENA returns both failed and succeeded random walks along with their probability of being drawn which enables use cases such as the computation of knowledge graph embeddings [4], or summaries [7].

3. Conclusion

S-AQP has many important use cases crucial for building and maintaining ecosystems of public SPARQL endpoints. RAW-JENA demonstrates the feasibility of implementing S-AQP on a representative SPARQL endpoint: Apache Jena. As each random walk is independent and enjoys

a logarithmic upper bound on its time complexity, users can collect and merge samples across multiple executions of the same query. Such a pay-as-you-go approach perfectly fits the fair-use policies of public SPARQL endpoints.
In future works, we plan to support full SPARQL queries and to show how the SPARQL standard could evolve to integrate sampling as SQL did with the `TABLESAMPLE` clause.

Acknowledgments

This work is supported by the French ANR project DeKaloG (Decentralized Knowledge Graphs) `ANR-19-CE23-0014`, and the French Labex CominLabs project MiKroloG (The Microdata Knowledge Graph).

References

[1] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. I. Jordan, S. Madden, B. Mozafari, I. Stoica, Knowing when you're wrong: Building fast and reliable approximate query processing systems, in: International Conference on Management of Data, SIGMOD, 2014.
[2] A. Soulet, F. M. Suchanek, Anytime large-scale analytics of Linked Open Data, in: 18th International Semantic Web Conference, ISWC, 2019.
[3] J. Debattista, S. Londoño, C. Lange, S. Auer, Quality assessment of linked datasets using probabilistic approximation, in: 12th European Semantic Web Conference on The Semantic Web. Latest Advances and New Domains, 2015.
[4] P. Ristoski, H. Paulheim, RDF2Vec: RDF graph embeddings for data mining, in: 15th International Semantic Web Conference, ISWC, 2016.
[5] V. Leis, B. Radke, A. Gubichev, A. Kemper, T. Neumann, Cardinality estimation done right: Index-based join sampling, in: Cidr, 2017.
[6] Y. Wang, A. Khan, X. Xu, S. Ye, S. Pan, Y. Zhou, Approximate and interactive processing of aggregate queries on knowledge graphs: A demonstration, in: 31st ACM International Conference on Information & Knowledge Management, 2022.
[7] L. Heling, M. Acosta, Estimating characteristic sets for RDF dataset profiles based on sampling, in: The Semantic Web: 17th International Conference, ESWC, 2020.
[8] F. Li, B. Wu, K. Yi, Z. Zhao, Wander Join and XDB: online aggregation via random walks, ACM Transactions Database Systems (2019).
[9] Y. Park, S. Ko, S. S. Bhowmick, K. Kim, K. Hong, W. Han, G-CARE: A framework for performance benchmarking of cardinality estimation techniques for subgraph matching, in: International Conference on Management of Data, SIGMOD, 2020.
[10] R. Angles, C. B. Aranda, A. Hogan, C. Rojas, D. Vrgoč, WDBench: A Wikidata graph query benchmark, in: International Semantic Web Conference, 2022.
[11] J. Aimonier-Davat, H. Skaf-Molli, P. Molli, M.-H. Dang, B. Nédelec, Join ordering of SPARQL property path queries, in: The Semantic Web: 20th International Conference, ESWC, 2023.