Federated Query Optimization Method based on SPARQL Endpoint Features for Efficient Retrieval

Soichiro Watanabe

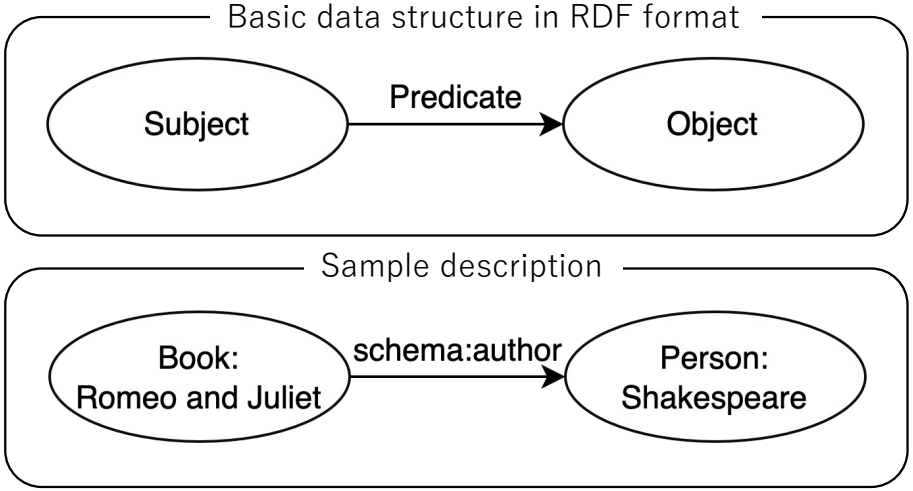
2023/10/26

Outline

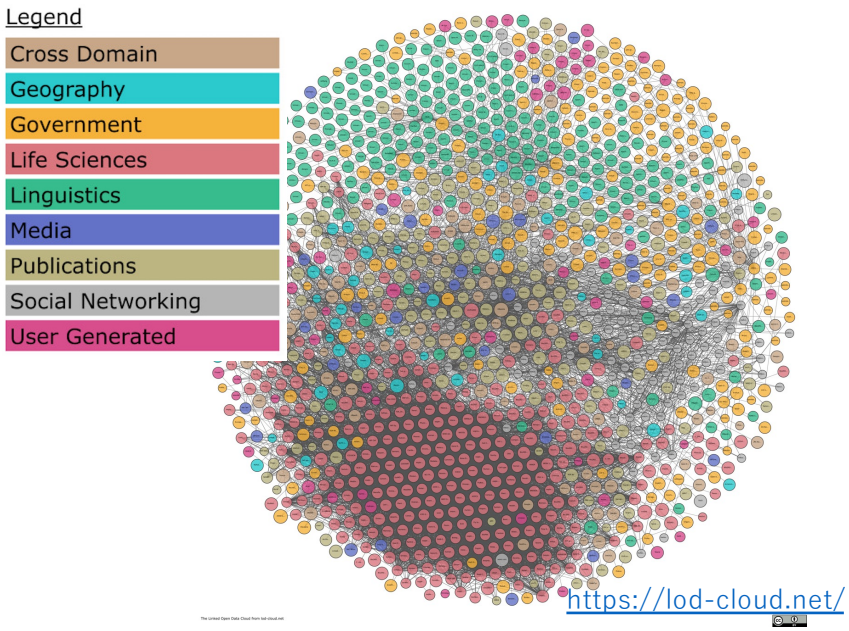
- Background :
  - It is important to be able to search across some fields for LOD datasets. SPARQL querying frequently time out.
- Purpose :
  - Efficiently retrieve SPARQL query results without timeouts
  - Related Work 1 (SPARQL optimization)
- Research Topic : Federated Query
  - Federated Query Structure
  - Related Work 2 (Federated Query Rewriting Technique using VALUES Clause)
- Approach : Federated Query Optimization Method based on SPARQL Endpoint Features
- Evaluation :
  - Comparative evaluation of the proposed method and 2 query execution methods.
  - Discuss the results
- Conclusion
  - Future Works

Linked Open Data(LOD)

- A mechanism for publishing and sharing metadata in a machine-readable format that can be used in combination each other.
- Described in Resource Description Framework(RDF) format.



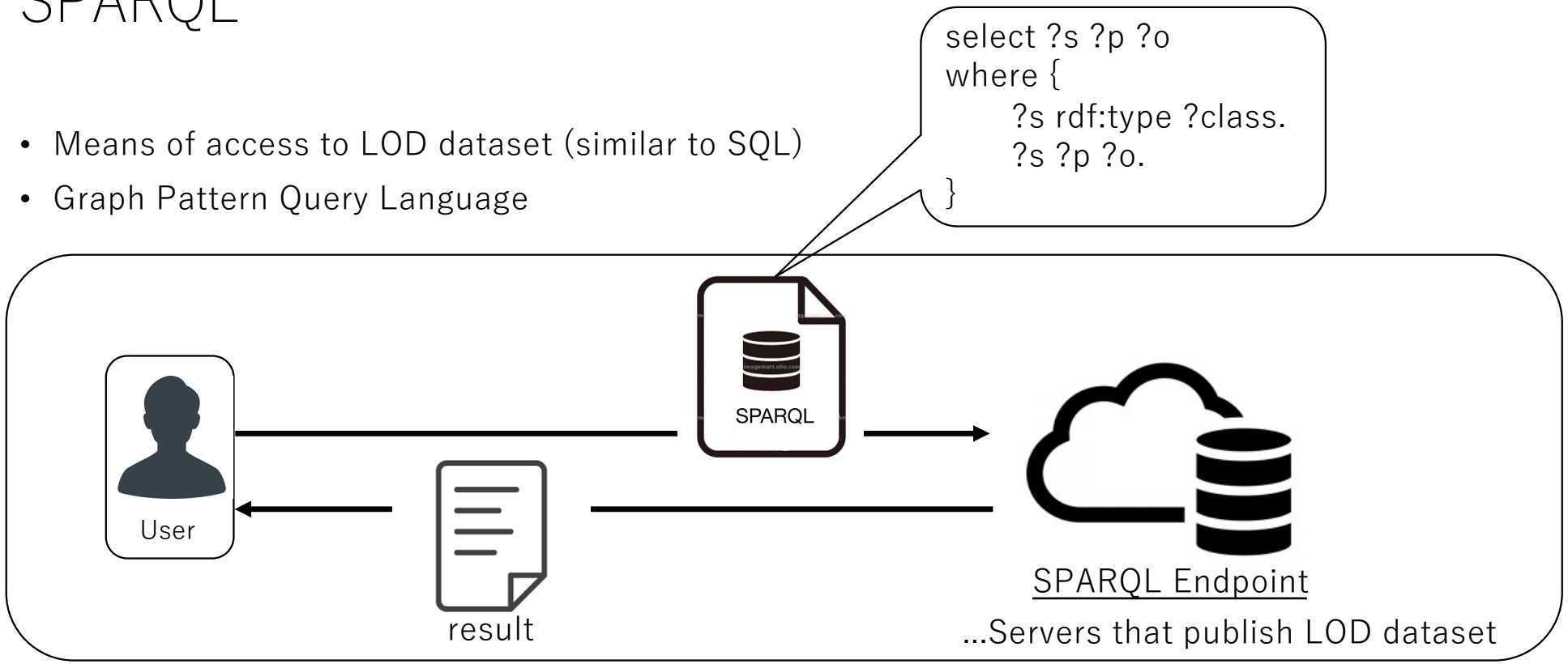
- Attracting attention as a mechanism to promote the use of data across a variety of fields. [1]



https://lod-cloud.net/

SPARQL

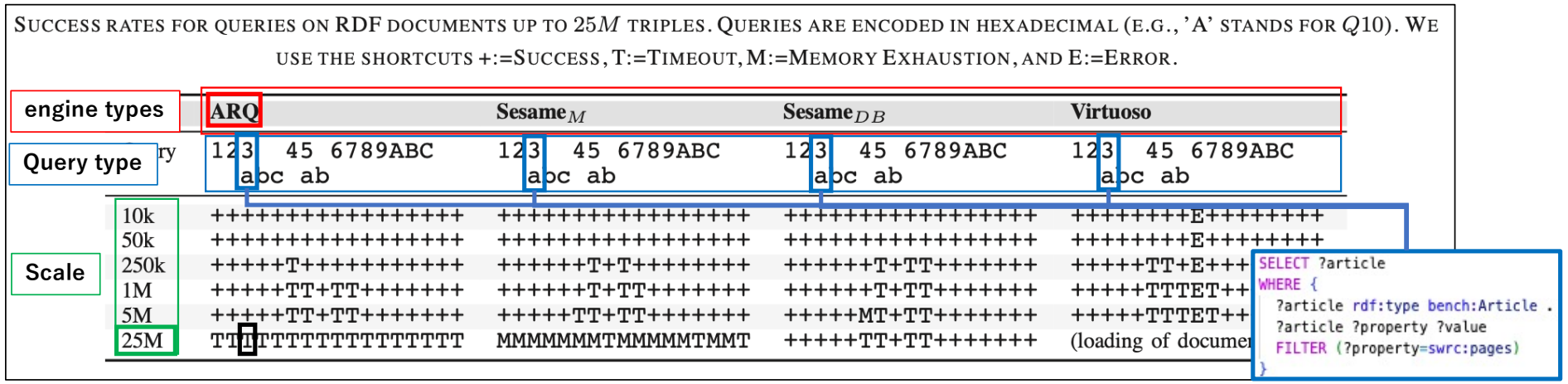
- Means of access to LOD dataset (similar to SQL)
- Graph Pattern Query Language



By using SPARQL, users can retrieve the desired information from LOD dataset

Problem

Timeouts in SPARQL queries



A work to measure the processing power of the SPARQL Endpoint engine (Like a DB engine in SQL)[2]

- **It actually happens that the query results are not returned from the SPARQL Endpoint .** ("**T**" in the above table; **T**imeout)
- **Whether a timeout occurs or not depends on the following three factors**
  1. SPARQL Endpoint engine types
  2. SPARQL Endpoint scale
  3. SPARQL query type

In SPARQL queries,
Timeouts are an important issue

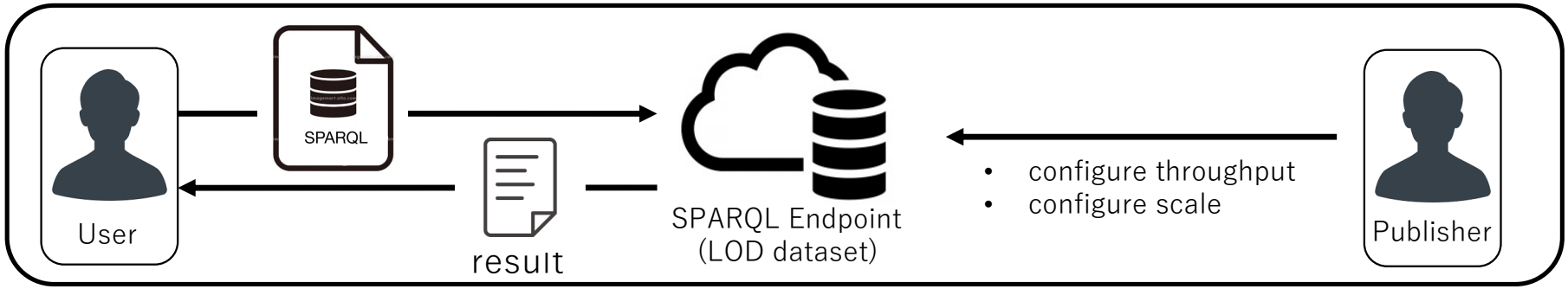
Efficiently retrieve SPARQL query results without timeouts

Whether a timeout occurs depends on the following items[2]

1. SPARQL Endpoint engine throughput
2. SPARQL Endpoint scale

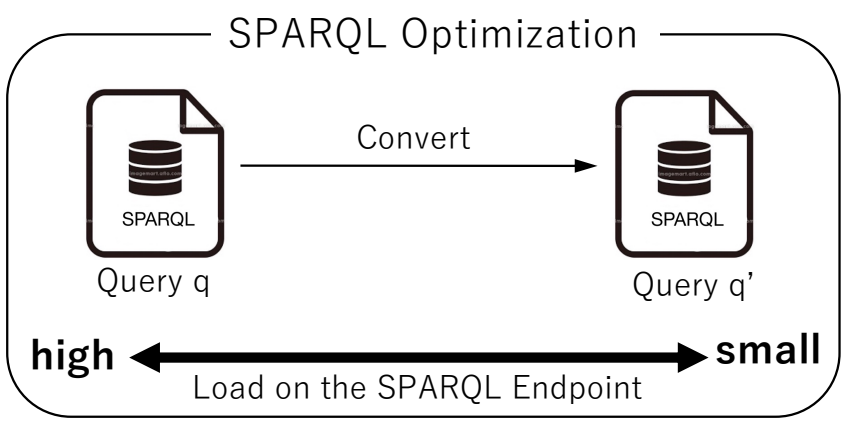
LOD dataset publishers set up
→ We cannot approach

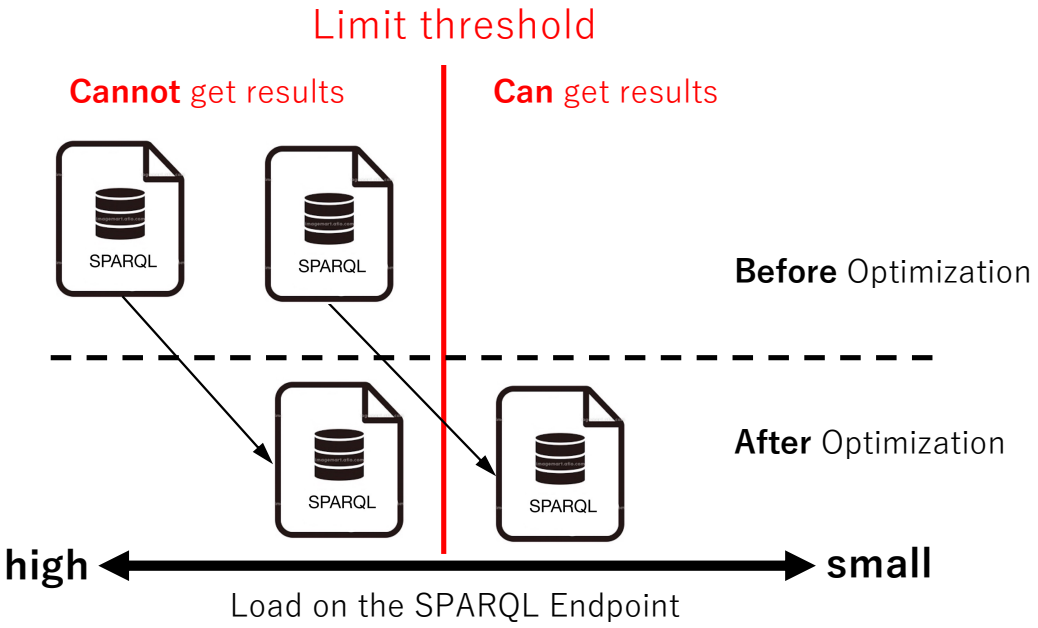
3. **The content of SPARQL query**



We process against the content of SPARQL query
→ Aim to obtain results efficiently without timeouts.

SPARQL optimization [3][4]





Result :

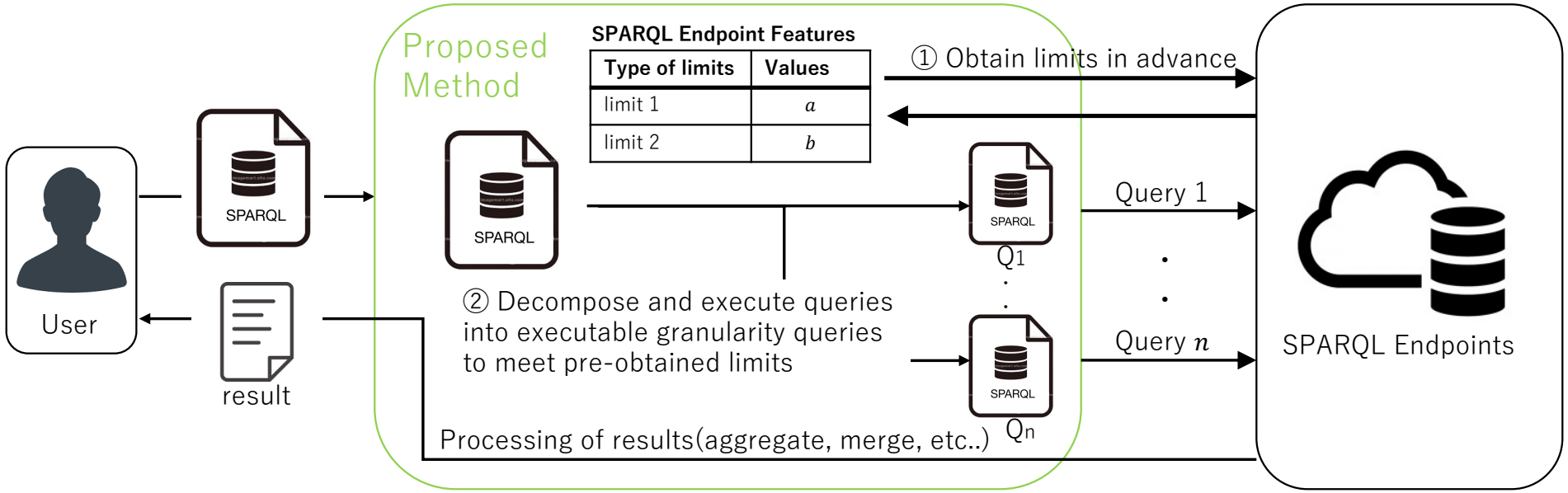
- **Reduced server(SPARQL endpoint) load**
- Some SPARQL queries increase load
  → suitable optimization technique should be chosen

Limit thresholds are different for each SPARQL endpoint
→ Optimization required for each SPARQL endpoint

Query Optimization Method based on SPARQL Endpoint Features

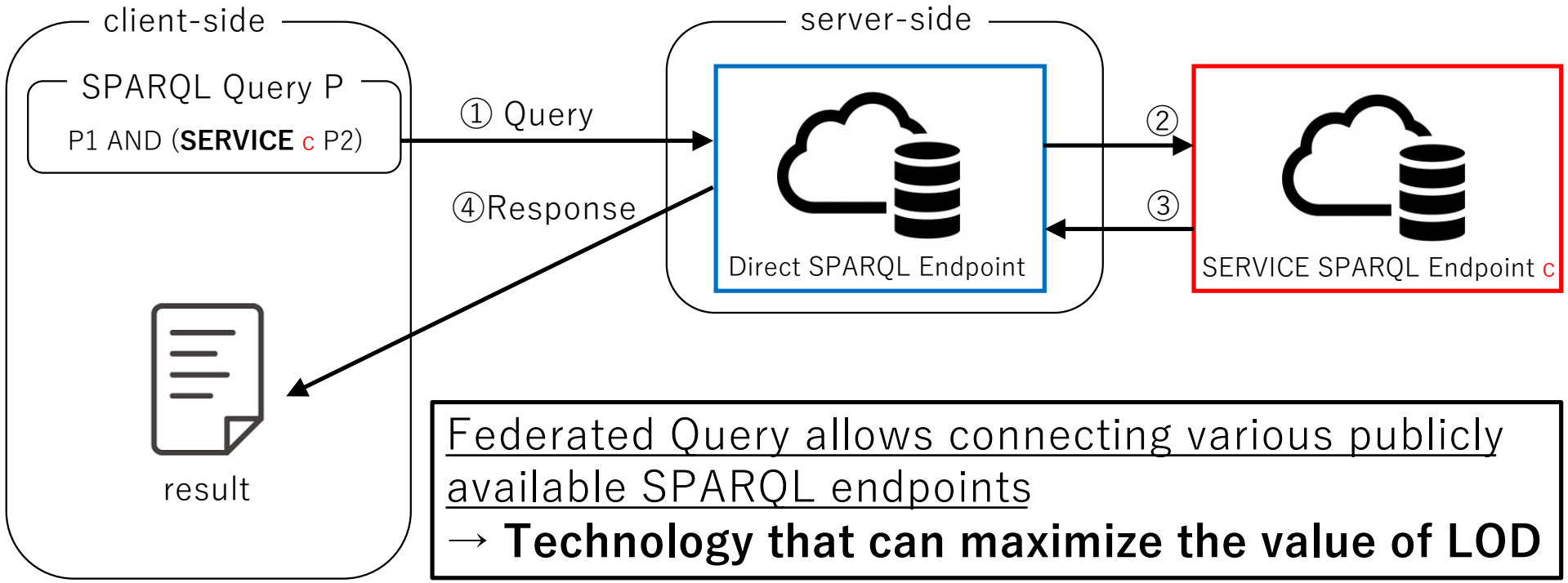
① **Obtain the various limits (SPARQL Endpoint Features) set on public SPARQL endpoints in advance by repeatedly querying using SPARQL**

② **Decompose and execute the query to meet the acquired limits.**



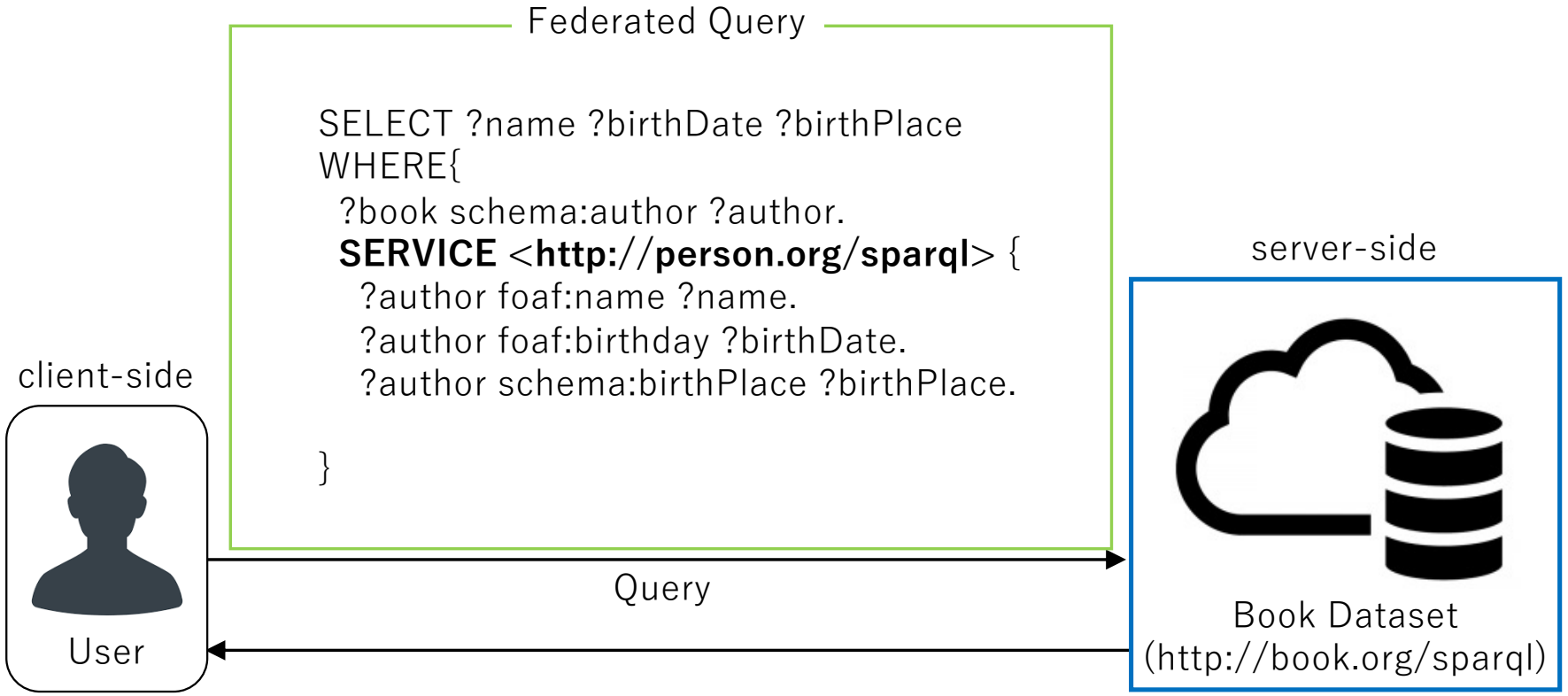
Federated Query

Multiple SPARQL endpoints can be queried using the **SERVICE** clause [5]

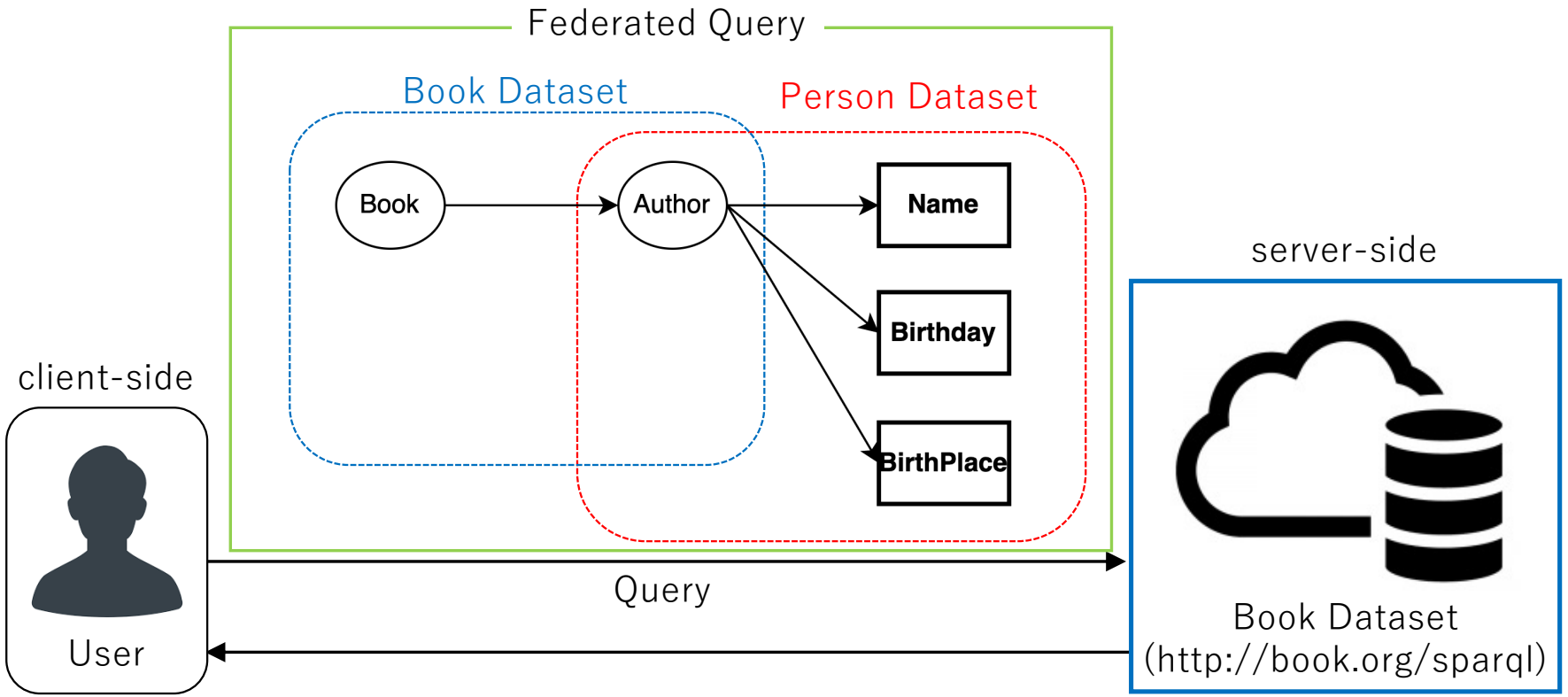


Federated Query allows connecting various publicly available SPARQL endpoints
→ **Technology that can maximize the value of LOD**

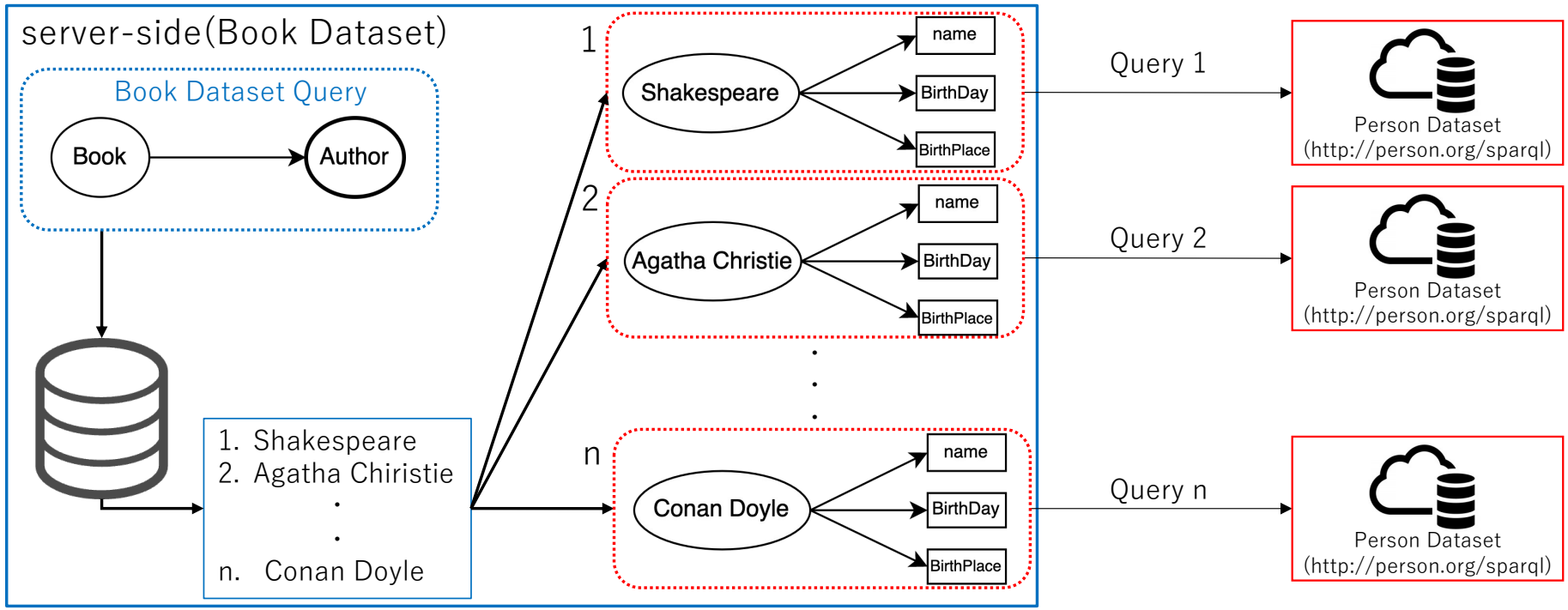
Example of Federated Query



Example of Federated Query



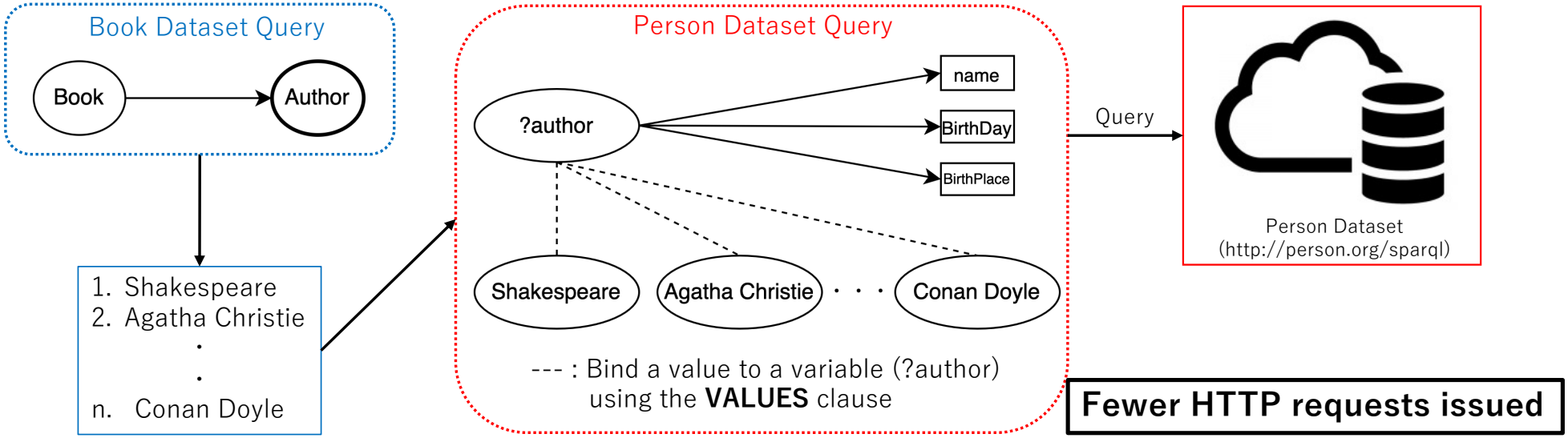
Behavior of the SPARQL endpoint being queried



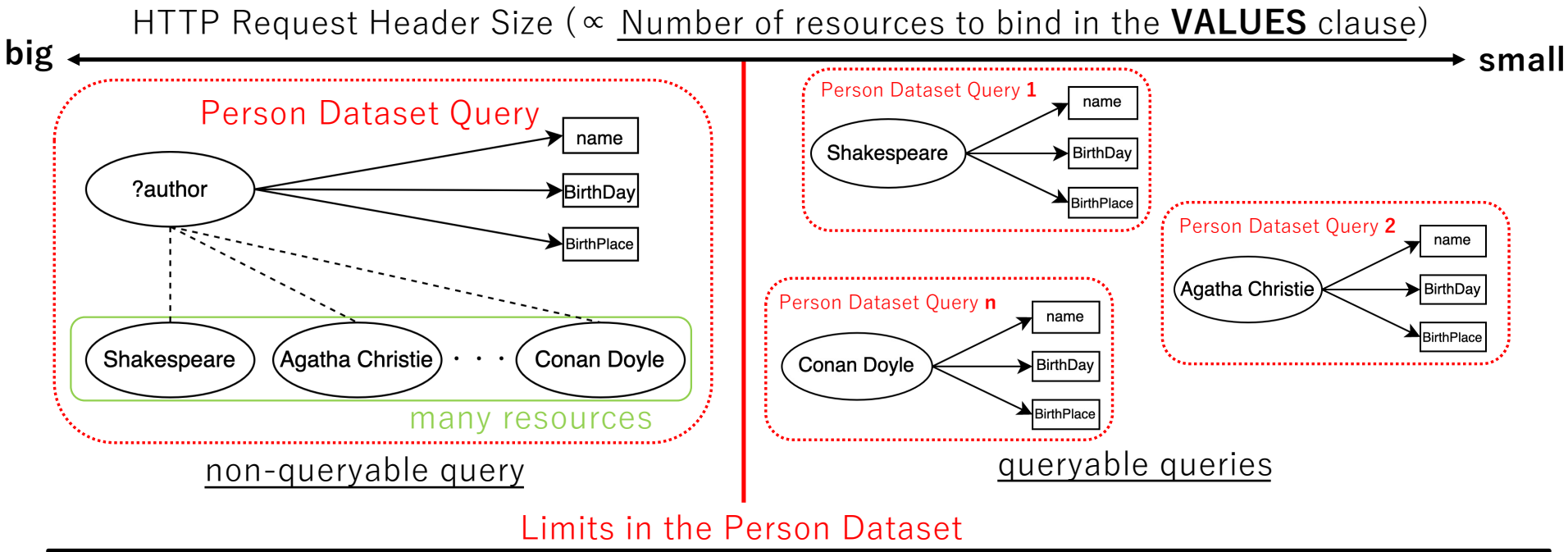
Challenges in Federated Query:
Increased number of queries n (number of results retrieved from the Book Dataset)
→ high load on the server side → increased possibility of timeouts

Strategies for executing federated queries in SPARQL 1.1 [6]

- A study of various techniques to decompose queries in SPARQL Federated Query to get complete results by avoiding endpoint limitations.
- Part of the technique is to rewrite the Federated Query using the **VALUES** clause[7].



Example of an issue in related research[6]



Different limits for each public LOD dataset
→ Need to decompose query to match limits for each LOD dataset

Federated Query Optimization Method based on SPARQL Endpoint Features

- SPARQL Endpoint Features：
  - A set of limits on SPARQL queries, different for each SPARQL endpoint
  - For instance …
    A) Permissible HTTP request methods(GET or POST)
    B) Constraints on the number of resources VALUES can bind within an executable SPARQL query
    C) Limitation on the length of a single resource that VALUES can bind within an executable SPARQL query
    D) Length restriction for executable SPARQL queries(= HTTP Request Header / HTTP Request Body Size)

Proposed Method

① Get Features for each SPARQL endpoint by repeatedly querying

② Decompose the Federated Query given as input to satisfy the pre-obtained Features

Get features for each SPARQL endpoint by repeatedly querying

Example：

GET Constraints on the number of resources VALUES can bind within an executable SPARQL query

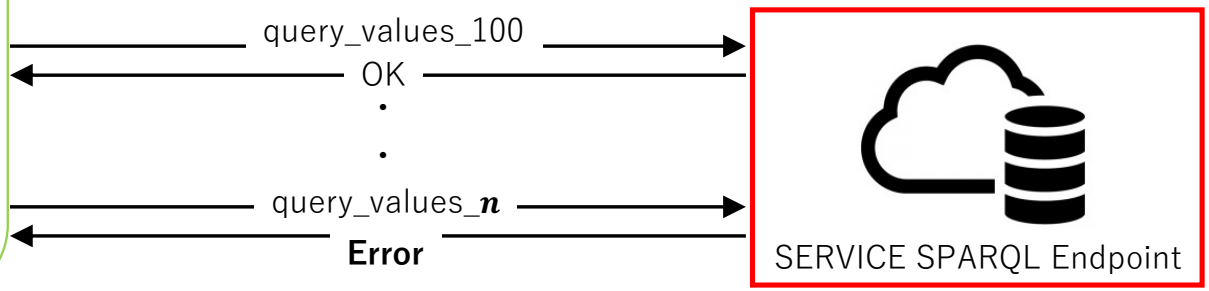
query_values_n

```
SELECT * WHERE {
    VALUES ?s {
        ex:resource_1
        ex:resource_2
            .
            .
        ex:resource_n
    }
    ?s ?p ?o
}
```

n

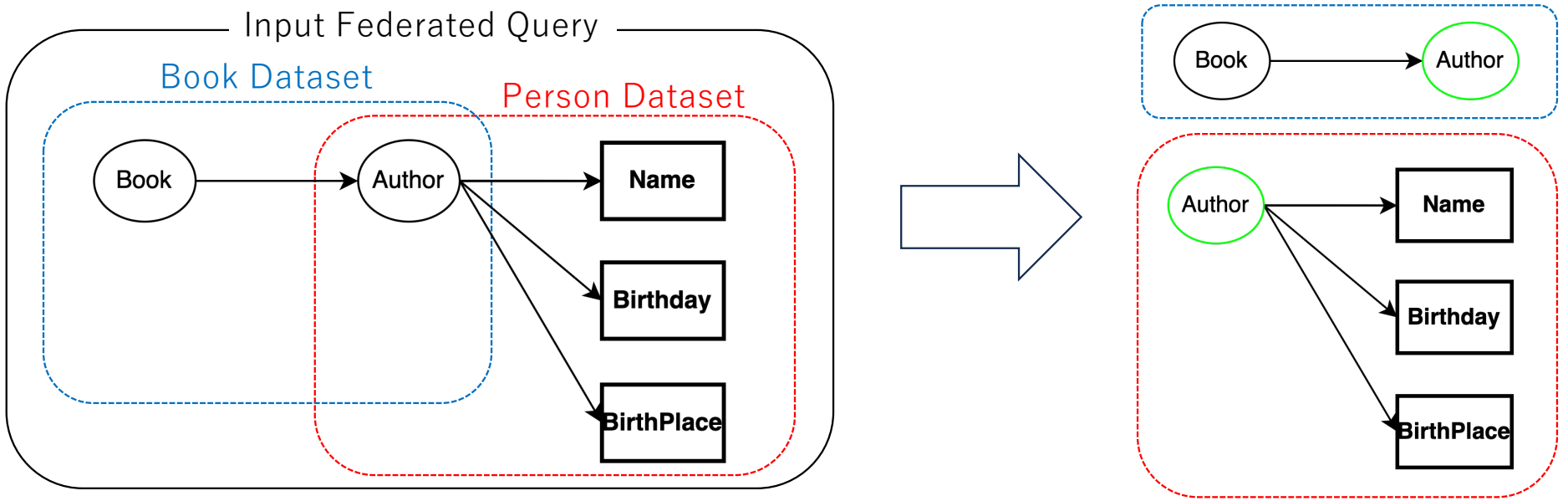
Proposed System

- Increase n until the response from the SERVICE SPARQL endpoint becomes an error
- Threshold is the number n of queries for which a response was obtained at the end



Decompose the Federated Query given as input to satisfy the pre-obtained Features

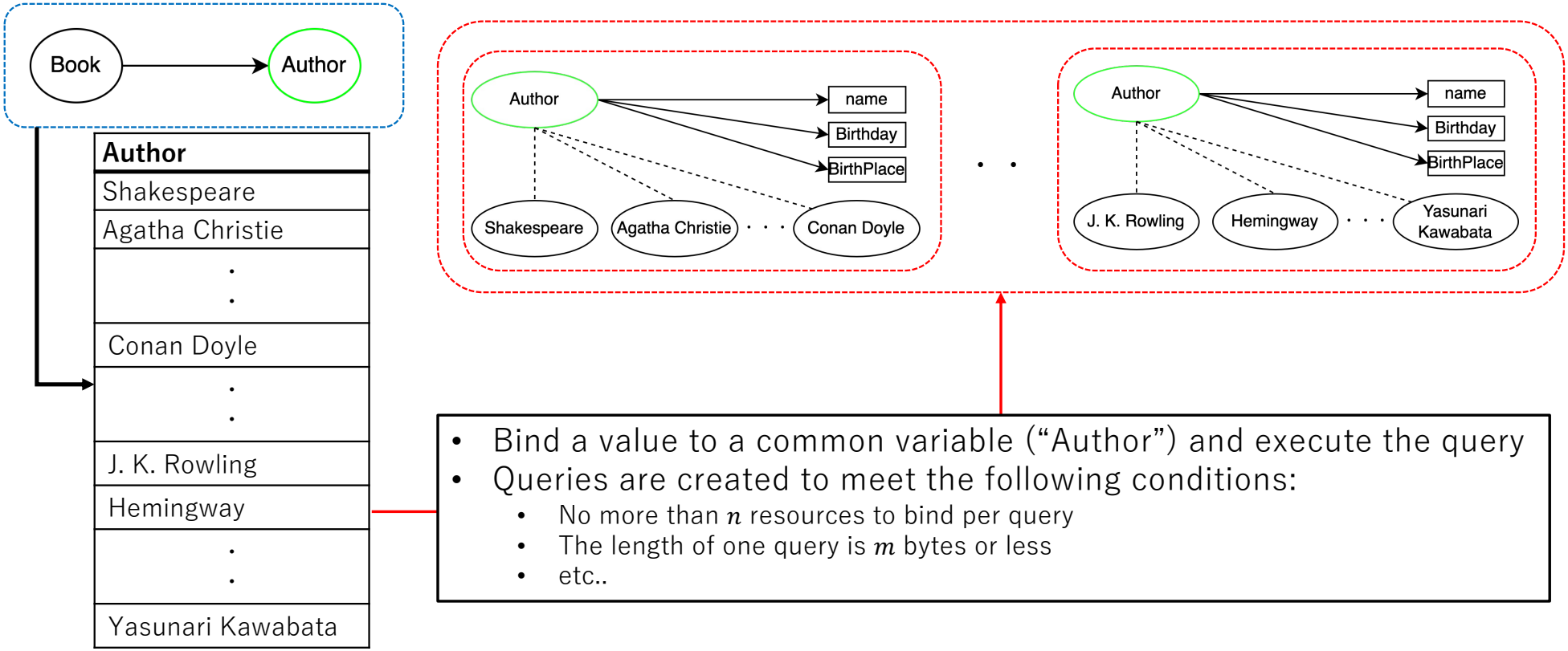
1. Parsing input Federated Query



- **Decompose into queries for each SPARQL endpoint**
- **Get the common variable (“Author” in the above figure)**

Decompose the Federated Query given as input to satisfy the pre-obtained Features

2. Create and execute each query to satisfy pre-obtained Features



Experiment

- Execute Federated Query with the following three methods and compare the execution time and availability of results
  1. Simply execute Federated Query (EXISTING)
  2. Rewrite the query using the VALUES clause (VALUES)
  3. Proposed Method (PROPOSED)

- Types of Federated Query prepared
  - manually created queries with relatively high utility
  - mechanically created queries with relatively low utility

- Evaluation Experiment
  - Ubuntu server 22.04 LTS
  - Apache jena fuseki 4.7.0 using 64GB memory
  - Ruby 2.7.7

Queries are here

https://github.com/ichichi-3180/mtsr2023/

Evaluation Result

Manually created

queryLabel	number of successes / number of trials, average execution time(second)		
	EXISTING	VALUES	PROPOSED
Query1	0/10,	5/10, 63.43932973099872	5/10, 72.34730386268348
Query2	10/10, 7.355361410416663	0/10,	0/10,
Query3	0/10,	0/10,	0/10,
Query4	1/10, 335.407594993012	0/10,	10/10, 61.23718678732403
Query5	0/10,	0/10,	10/10, 9.801166327246888
Query6	9/10, 437.27629097028534	0/10,	10/10, 170.16292382902466

Mechanically created

queryLabel (Second Service Endpoint)	number of successes / number of trials, execution time(second)		
	EXISTING	VALUES	PROPOSED
assaultlily-rdf []	0/10,	0/10,	10/10, 3.6007316867122428
Cultural Japan []	0/10,	0/10,	10/10, 0.20709268788341434
DBpedia []	0/10,	10/10, 2.4891252658562735	10/10, 3.1280574607662857
DBpedia Japanese []	10/10, 18.008064862154423	0/10,	0/10,
Earthquake LOD []	10/10, 39.91150904921815	0/10,	10/10, 0.9186438637319952
Japan Search []	0/10,	0/10,	10/10, 1.6497687711846083
jrslod []	10/10, 9.802138139726594	0/10,	10/10, 0.433084353688173
Linked Open Vocabularies []	0/10,	0/10,	10/10, 4.227437379048206
Media Art Database []	0/10,	0/10,	0/10,
Web NDL Authorities []	10/10, 9.497704373137093	0/10,	10/10, 0.40398354569915684
Onsen LOD []	0/10,	0/10,	10/10, 0.19432361498475076
OWL de ramen ontology []	0/10,	10/10, 5.092421144200489	10/10, 5.1769265659153465
☆pikopiko planet☆space []	0/10,	0/10,	0/10,
PrismDB []	0/10,	0/10,	10/10, 1.4926710256142541
Wikidata []	0/10,	0/10,	10/10, 44.732915345299986

Discussion

- Of the queries used in this evaluation, the PROPOSED method was the one that obtained 100% (10/10) of the results for the most queries (16/21).
- The PROPOSED method was significantly faster than the EXSITING method and was able to retrieve results almost as fast as VALUES method.
- There were queries for which results could not be obtained even with the PROPOSED method.
  - Query2:
    Due to character encoding in Japanese URIs from Japan Search
  - Query3:
    The number of Solutions obtained from MADB is too large (could not be handled by the machine specs of the experimental environment).
  - DBpedia Japanese, pikopiko planet:
    The query used for execution met all the pre-obtained limits but could not be executed.
    → There may be additional items required in the limits that should be obtained in advance.
  - Media Art Database:
    Comparison method is a timeout, PROPOSED is an error specific to the SPARQL endpoint(403)

Conclusion

- **The evaluation shows that PROPOSED method improves execution time and accuracy**

- **Future Works**
  - PROPOSED Method needs to be more updated based on the causes of queries that could not be executed

  - Need to experiment with more queries
    - I investigated Wikidata Query Log [7] and LSQ [8]
      - Few queries contain SERVICE clauses
        → Even though the SERVICE clause is used, it specifies a server down.
      - All triples have been specified in the graph pattern.(?s ?p ?o)
        → Depends on machine power rather than number of HTTP requests issued.

    - Need to collect query logs from different endpoints
      → Currently we have part of logs from the Media Art Database

    - Manual creation of further queries

References

1. LOD cloud. http://cas.lod-cloud.net/
2. Schmidt, M., Hornung, T., Lausen, G., & Pinkel, C. (2009, March). SP2Bench: a SPARQL performance benchmark. In 2009 IEEE 25th International Conference on Data Engineering (pp. 222-233). IEEE.)
3. Groppe, J., Groppe, S., Kolbaum, J. “Optimization of SPARQL by using core SPARQL”. Proc.of the ICEIS, pp.107–112 (2009)
4. Kenta Hirayama, Ken Kaneiwa. “SPARQLにおける集約の形式化とクエリ書き換え”. 人工知能学会研究資料 SIG-SWO-048-03. 2019. In Japanese.
5. SPARQL 1.1 Federated Query W3C Recommendation 21 March 2013, https://www.w3.org/TR/sparql11-federated-query/
6. Buil-Aranda, C. et al.: “Strategies for executing federated queries in SPARQL1. 1”. In: The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II 13. Springer International Publishing, 2014. p. 390-405.
7. Wikidata SPARQL Logs, https://iccl.inf.tu-dresden.de/web/Wikidata_SPARQL_Logs/en
8. Linked SPARQL Queries, http://lsq.aksw.org/

Thank you for listening !