

MCP-over-SPARQL: Giving AI the tools to query knowledge graphs

Tarek Bouzelat¹, Thomas Francart¹, Badisse Dahamna², Flavien Disson², Romain Lelong², Catherine Letord² and Stéfan J. Darmoni²

¹*Sparna, Tours, France*

²*Département de Santé Numérique (DéSaN), CHU de Rouen, Rouen, France*

Abstract

In this paper we present MCP-over-SPARQL, a Model Context Protocol server that provides tools enabling a client AI to successfully write and execute a valid SPARQL query on a knowledge graph. The server connects an MCP-capable AI client to a SPARQL endpoint together with other tools for retrieving an overview of the graph, curated query examples, and the detailed knowledge graph structure from a SHACL specification. In addition, it exposes a reconciliation tool to retrieve entity URIs from a label and a type. This provides the necessary primitives for an AI client to turn a natural language question into a SPARQL query, using a correct graph structure and valid URI identifiers. The system reduces schema guessing and returns data from the graph. The approach is demonstrated using the Standardized Drug Nomenclature of France digital health agency, and shows how it supports controlled, schema-aware querying of domain-specific knowledge graphs. The objective of this project is to build a hybrid AI system for drug knowledge.

Keywords

Model Context Protocol, SPARQL, SHACL, Knowledge Graph, Large Language Model, Entity Reconciliation

1. Introduction

Knowledge graphs are sources of trustworthy structured data, but leveraging this data directly in SPARQL [1] requires understanding the underlying schema, the available predicates, and the URI identifiers used in the graph. Yet, data publishers often want to maximise the reuse of their knowledge graph, and making such datasets queryable by AI systems is an interesting dissemination channel towards a non-technical audience.

Large Language Models (LLM) can query external sources when they are connected to tools through the Model Context Protocol [2]. In this demo, we present MCP-over-SPARQL [3], an MCP server driven by a SHACL [4] specification that describes the underlying knowledge graph structure. The server exposes this structure through a small set of schema-grounded tools (shown in Figure 1) that let an MCP-capable client discover the schema, inspect relevant types, reconcile named entities, and execute SPARQL queries. The PARTAGES project, led by the Health Data Hub (prime contractor) and bringing together over 30 academic, hospital, and industrial partners, including the Department of Digital Health of the Rouen University Hospital (RUH), aims to develop digital commons to support the use of generative large language models (LLMs) in healthcare. Several open-source deliverables are planned, including a French-language medical language model, a corpus of synthetic hospital reports written by physicians, seven specialized models addressing practical use cases, and a federated evaluation platform, deployed across 20 health institutions, including the RUH.

Posters, Demos, Blue Sky, and Tutorials at SEMANTiCS 2026, September 2026, Ghent, Belgium

✉ tarek.bouzelat@sparna.fr (T. Bouzelat); thomas.francart@sparna.fr (T. Francart)

🆔 0000-0003-4648-0065 (T. Francart); 0000-0003-0762-2518 (B. Dahamna); 0000-0002-4051-2284 (C. Letord);

0000-0002-7162-318X (S.J. Darmoni)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

2. System Architecture

MCP-over-SPARQL sits between an MCP-capable LLM client and a SPARQL endpoint. Rather than exposing only a SPARQL endpoint service, it exposes the tools depicted in Figure 1 below.

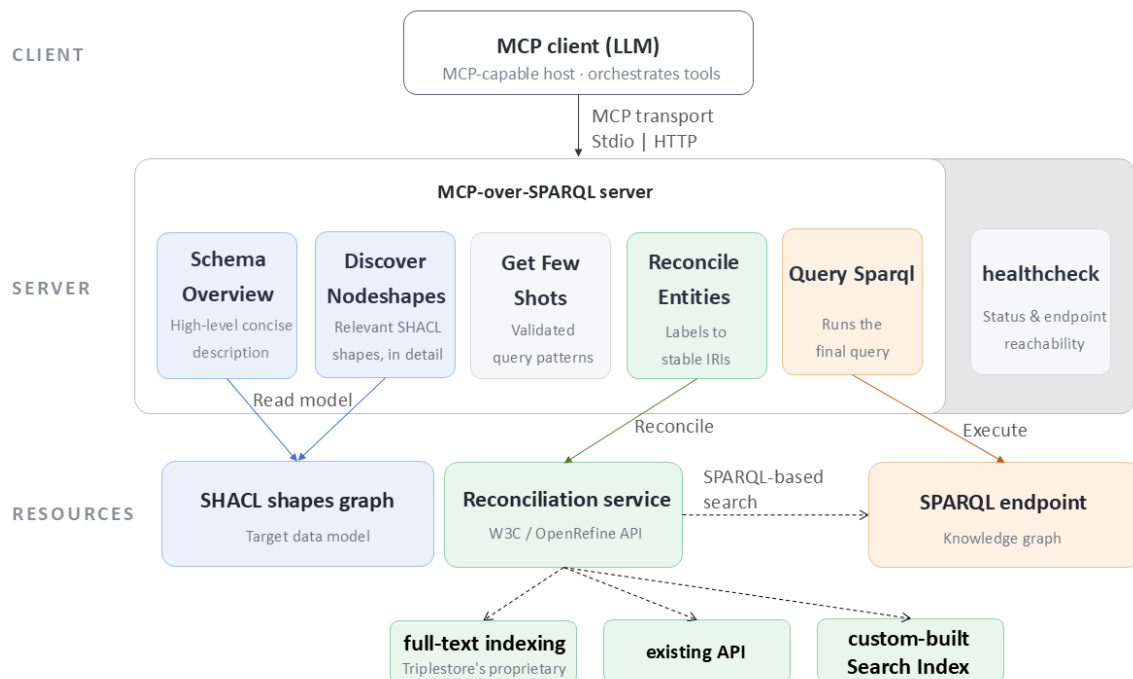


Figure 1: Architecture of MCP-over-SPARQL.

The *schema_overview* tool derives a high-level concise description of the graph and returns it as a Markdown listing of the classes it contains, along with some additional information about each of them. It is deliberately incomplete and acts as a high-level table of contents, not sufficient on its own to write a correct query. The schema overview also includes a concise description of expected query use-cases, indicating for each use-case the corresponding class to use and the expected joins to and from other classes. For example, the use-case ‘find the medicinal products that contain a given active substance’ enters the graph through the Substance class, reconciles the substance (such as ketoprofen) into its URI, and reaches the corresponding `med:SpecialitePharmaceutique` instances through the `med:estSubstanceActiveDans` relation.

The *discover_nodeshapes* tool returns detailed definitions of shapes. It reads the SHACL file and returns a detailed list of properties for each requested shape. For each property, it provides a label, a description, and details such as its path, datatype, and cardinality. Beyond structural constraints, each shape can also carry AI-oriented guidance in *sh:agentInstruction*, an annotation introduced in SHACL 1.2 [5]. This allows the data publisher to guide how a class or property should be used with “hints”, for example by indicating which identifier to prefer or how to interpret a relation.

The *get_few_shots* tool returns curated, validated examples for the graph, each pairing a natural language question with the corresponding SPARQL query.

The *reconcile_entities* tool relies on an underlying service that conforms to the Reconciliation Service API Specification [6]. It takes two input parameters, a label to search and the URI of the entity’s shape to look up, and returns a list of entities, each with a score. This service can be implemented, depending on the project, as a plain SPARQL-based search (inefficient and unsuitable for plain-text lookups), the triplestore’s proprietary full-text indexing, an existing API, or a custom-built search index filled with the labels of the graph’s entities.

Finally, the *query_sparql* tool takes a SPARQL query as input and returns the result in `application/sparql-results+json`.

The server can be accessed either locally or remotely. It can be configured per “project”, each corresponding to a specific knowledge graph, with its own SHACL model and SPARQL endpoint, and exposed as a dedicated set of MCP tools. A unique server deployment can thus expose multiple knowledge graphs through MCP.

3. Schema-Grounded Query Workflow

In a typical case, the AI client turns a natural-language question into a SPARQL query by following the multi-step workflow shown in Figure 2. The client first calls *schema_overview* to get a high-level map of the graph, then *get_few_shots* to read curated questions and query examples, then *discover_nodeshapes* to read the detailed properties of the relevant classes. When the question mentions named entities, it calls *reconcile_entities* to resolve their labels into valid URIs. It finally builds a SPARQL query from the discovered structure and the resolved URIs, and runs it with *query_sparql*, returning the results to the user.

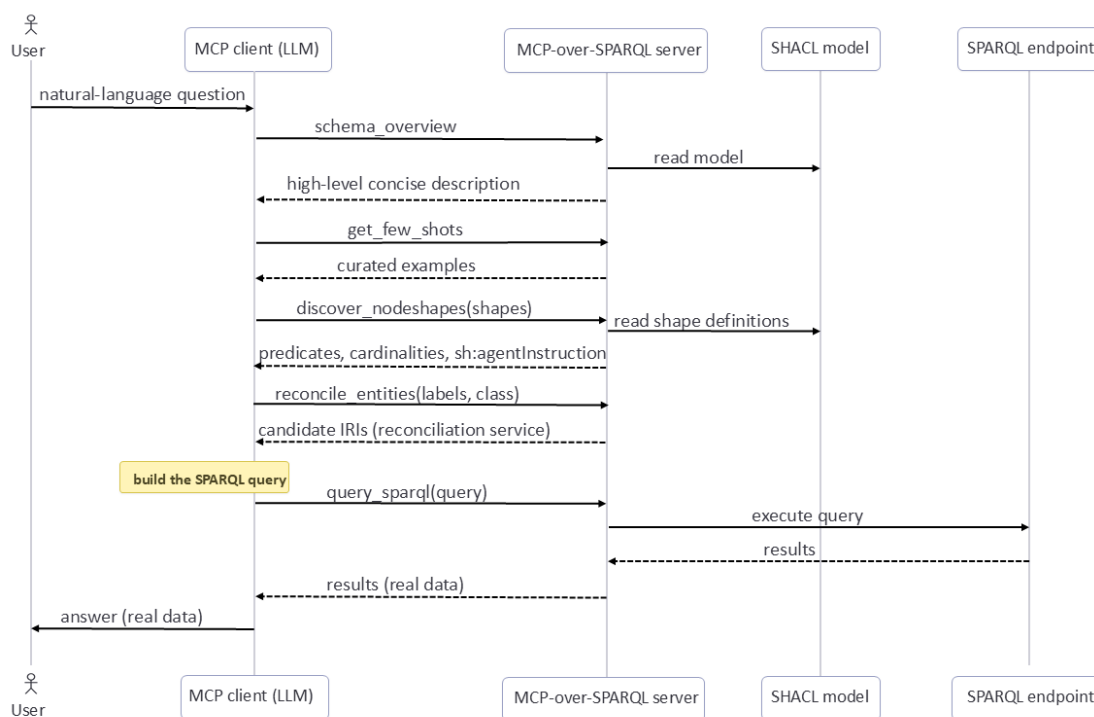


Figure 2: Lifecycle of a query processing: the MCP client orchestrates the server’s tools to turn a natural-language question into a query executed against the endpoint.

However, this order is not enforced. The tool descriptions recommend discovering the schema first and never guessing predicates or paths from prior knowledge, but the client remains free to call the tools in whatever order it needs, using only those relevant to the question.

4. Demonstration

An online prototype of MCP-over-SPARQL can be tested at <https://services.sparnatural.eu/mcp/ruim-mcp>. This is an MCP connector URL, meant to be registered in an MCP-capable AI client, not a web page to open in a browser. A 2-minute video [7] shows how to connect it to an AI client. The prototype exposes the structure of the Standardized Drug Nomenclature of France e-health agency (RUIM) [8], with its SHACL specification [9], and its online SPARQL endpoint [10]. The SHACL specification contains extensive documentation of each entity and properties. Although we can work with an algorithm to derive SHACL shapes automatically, in this case the SHACL shapes were manually designed in a spreadsheet, which is then converted into RDF.

Both examples below follow the workflow of Figure 2, although not every step is always used.

Using Mistral Vibe (Chat) with the query “What does bi profenid contains?” (“Que contient le bi profenid?”), the AI follows this workflow:

1. Calls *schema_overview* to read the overview of the graph structure. This step is sometimes skipped, because the allowed shape identifiers (the graph classes) are already exposed as an enum in the input schema of the *discover_nodeshapes* tool. Since the client sees these classes directly in the tool description, it may consider that reading the schema overview is unnecessary and call *discover_nodeshapes* right away.
2. Calls *discover_nodeshapes* with the shapes `med:SpecialitePharmaceutique`, `med:Element` and `med:Substance` (“medicinal product”, “manufactured item” and “substance”) to read their properties.
3. Calls *reconcile_entities* with “bi profenid” on `med:SpecialitePharmaceutique`, and retrieves its URI (`...SpecialitePharmaceutique_61165486`).
4. Generates and executes the following query, to retrieve the correct answer (“Ketoprofène”):

```
PREFIX med: <http://data.esante.gouv.fr/ansm/medicament/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?substance ?substance_label WHERE {
  <http://data.esante.gouv.fr/ansm/medicament/
    SpecialitePharmaceutique_61165486 >
    med:substanceActive ?substance .
  ?substance rdfs:label ?substance_label .
  FILTER (LANG(?substance_label) = 'fr')
}
LIMIT 20
```

Using Claude Sonnet 4.6, low effort, with the request “What are the generics of perindopril?” (“Quels sont les génériques du périndopril?”), the AI follows this sequence:

1. Calls *get_few_shots* to read example queries.
2. Calls *discover_nodeshapes* with the shapes `med:GroupeGenerique` (“generic drug grouping”) and `med:SpecialitePharmaceutique` (“medicinal product”) to read their properties.
3. Calls *reconcile_entities* with label “périndopril” and type `med:SpecialitePharmaceutique` — no result as “périndopril” is actually a Substance and not a medicinal product.
4. Calls *reconcile_entities* again, this time with type `med:Substance` — it returns 4 Substances.
5. Then produces and executes a (correct) SPARQL query containing the 4 Substances URI, the drug groupings and the marketing authorization holder of each drug (although not requested).

Note how the AI was able to overcome two ambiguities with the original query: 1/ it asks for generic drugs but using the name of a substance and not a product; 2/ the name of the substance is itself ambiguous. Note also how the query is correct at the first iteration.

Some tests also show the limits of the approach: sometimes the AI does not use the MCP server but uses a web search, or its own knowledge; sometimes it does not use the *reconcile_entities* tool but rather tries to lookup the entities directly in SPARQL (we tried to mitigate that with *sh:agentInstruction* hints); sometimes the answer is correct but requires multiple SPARQL queries in order to retrieve a good result.

5. Related Work

Several MCP servers exposing SPARQL access have appeared recently, and a growing ecosystem is now catalogued in public registries (PulseMCP) [11]. Comunica MCP SPARQL [12] puts public and private knowledge graphs behind a single SPARQL-based interface, covering endpoints and other decentralized

sources. It focuses on query execution, exposing only two query-running tools, and does not describe the schema, so the client infers the structure itself, which works on widely known graphs (DBpedia, Wikidata) but degrades on less common ones. We address that gap by grounding the interaction in a SHACL specification and exposing dedicated tools (*schema_overview*, *discover_nodeshapes*, curated examples via *get_few_shots*, and URI reconciliation), so the client discovers the real structure instead of guessing it. TogoMCP [13], in the life-science domain, follows a close workflow, separating entity resolution from schema-guided SPARQL generation, and its ablation study confirms that schema context is the decisive component. It differs from ours on three points. It describes each database with a MIE file, a YAML document, whereas we rely on SHACL, a W3C Recommendation the publisher may already maintain as documentation, with agent-facing guidance carried by the shapes themselves through *sh:agentInstruction* rather than by a separate file. For entity resolution it calls resource-specific REST APIs, while our *reconcile_entities* tool relies on the standardized Reconciliation Service API, so any conforming service plugs in. Finally, TogoMCP is a curated deployment over one portal, while our server is configured per project, each pairing its own SHACL model with its own SPARQL endpoint, so any publisher can expose its own graph, in any domain.

6. Conclusion and Future Work

The MCP-over-SPARQL approach enables a SHACL-grounded connection of a SPARQL-accessible knowledge graph from an AI. It can be configured for different knowledge graphs. It is hoped that even small LLM models would be capable of understanding the structure of the graph and querying it in SPARQL.

This approach leads to a hybrid AI tool (combination of generative AI & knowledge graph) about French drug model developed by the French digital health agency. The next phase involves evaluating this proof of concept through a benchmark of drug-related questions designed by pharmacists. The respective impact of generative AI and knowledge will be blindly evaluated: one pharmacist and two physicians will evaluate the hybrid AI vs. the generative AI alone.

7. Ethics and Environmental Footprint

It must be recalled that the AI systems used are responsible for an increase in the demand for electricity, water, and rare materials. Moreover, the providers of these systems do not disclose the content of their models, the source of the data used, or the work done in poor countries to make it usable.

Acknowledgments

This work was partially funded by (a) the PARTAGES project (n° DOS0245197) under the French National call from BPI France (France 2030 digital commons for generative AI).

Declaration on Generative AI

During the preparation of this work, Claude Opus was used in order to: Improve Writing Style. The authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

- [1] S. Harris, A. Seaborne, SPARQL 1.1 Query Language, <https://www.w3.org/TR/sparql11-query/> (2013).

- [2] Anthropic, Introducing the Model Context Protocol, <https://www.anthropic.com/news/model-context-protocol> (2024).
- [3] Sparna, MCP-over-SPARQL, <https://github.com/sparna-git/sparnatural-services> (2026).
- [4] H. Knublauch, D. Kontokostas, Shapes Constraint Language (SHACL), <https://www.w3.org/TR/shacl/> (2017).
- [5] W3C Data Shapes Working Group, SHACL 1.2 Core, <https://www.w3.org/TR/shacl12-core/> (2026).
- [6] W3C Entity Reconciliation Community Group, Reconciliation Service API, Version 0.2, <https://www.w3.org/community/reports/reconciliation/CG-FINAL-specs-0.2-20230410/> (2023).
- [7] 2-min video of MCP-over-SPARQL prototype, <https://www.youtube.com/watch?v=67il9X9f0Kc> (2026).
- [8] Development of a standardized drug nomenclature in France, *Stud Health Technol Inform.* 327, 198–202 (2025). doi:10.3233/SHTI250301.
- [9] SHACL specification of the RUIM used for the MCP-over-SPARQL prototype, <https://github.com/sparna-git/referentiel-medicaments/blob/main/shacl/RUIM-SHACL-instances.ttl> (2026).
- [10] RUIM SPARQL endpoint, <https://smt.esante.gouv.fr/sparql/> (2026).
- [11] PulseMCP, SPARQL MCP servers directory, <https://www.pulsemcp.com/servers?q=sparql> (2026).
- [12] R. Taelman, J. De Smet, Comunica MCP SPARQL: Improving the Accuracy of AI Agents using SPARQL-based Access to Decentralized Knowledge Graphs, in: 23rd Extended Semantic Web Conference (ESWC), Posters and Demos (2026). <https://rubensworks.github.io/article-eswc2026-demo-comunica-mcp/>
- [13] A. R. Kinjo, Y. Yamamoto, S. Bustamante-Larriet, J. E. Labra-Gayo, T. Fujisawa, TogoMCP: Natural Language Querying of Life-Science Knowledge Graphs via Schema-Guided LLMs and the Model Context Protocol, *bioRxiv* (2026). doi:10.64898/2026.03.19.713030.