OMCP: Model Context Protocol Servers for the OMOP Common Data Model

Shihao Shenzhang[1], Niko Möller-Grell[1], Zhangshu Joshua Jiang[6], Richard Dobson[1,2,3,4,5], Vishnu V Chandrabalan[6,7]

1. **Department of Biostatistics and Health Informatics, Institute of Psychiatry, Psychology and Neuroscience (IoPPN), King's College London, London, UK**
2. **Institute for Health Informatics, University College London, London, UK**
3. **NIHR Biomedical Research Centre, University College London Hospitals National Health Service Foundation Trust, London, UK**
4. **Health Data Research UK London, University College London, London, UK**
5. **NIHR Biomedical Research Centre, South London and Maudsley National Health Service Foundation Trust and King's College London, London, UK**
6. **Lancashire and South Cumbria Secure Data Environment**
7. **Lancaster University**

Background

The Observational Medical Outcomes Partnership (OMOP) Common Data Model (CDM) has become a global standard for real-world evidence (RWE), harmonising electronic health record (EHR) data into a relational structure and mapping terms to standardised ontologies.[1] Maintained by the OHDSI collaborative, OMOP underpins initiatives such as NHS England's Secure Data Environment (SDE) and the UK's first RWE network, HERON, led by Oxford University and HDRUK.

Recent advances in large language models (LLMs) like GPT-4o and Claude Sonnet 3.7 create new opportunities for healthcare, enabling natural language prompts to be translated into SQL or analytics code.[2] Lightweight, locally deployed LLMs further reduce computational cost and address governance concerns by keeping data within institutional boundaries.

Anthropic's Model Context Protocol (MCP), introduced in 2024, extends LLMs with access to contextual tools and agentic workflows, allowing multi-step reasoning and orchestration of toolchains—features particularly valuable for healthcare research.[3]

Yet generating insights from OMOP databases still demands SQL and R expertise, and no privacy-preserving LLM tools currently target RWE generation. To address this gap, we present OMCP[4], a suite of MCP servers integrating the OMOP CDM. OMCP enables real-time natural language–to-database operations (e.g., dynamic queries, on-demand analytics), empowering researchers without SQL expertise while preserving advanced flexibility. This abstract introduces OMCP-SQL, the first OMOP-specific MCP server, comprising a Semantic Parser, Python, Data Validator, and R Sandbox, all connected through the OMCP-A2A framework for collaborative agentic workflows.

Methods

OMCP-SQL was developed using multiple open-source Python libraries and tested against a synthetic OMOP database derived from the publicly available Synthea dataset. While the MCP protocol offers flexibility in connecting language models to external data sources, it carries risks related to security, misuse, and interoperability. To mitigate these, we are implementing strict query-safety controls (blocking destructive operations), adopting transparent logging for traceability through Arize Phoenix, and encouraging deployment in secure, local environments rather than over public networks.

As illustrated in Figure 1, SQLGlot was used for syntactic and semantic validation of LLM-generated queries against user-configurable criteria.[6] This allowed for both fine-grained control over what queries were sent to the database engine for execution and the raising of applicable "LLM-friendly" exceptions when the SQL generated by LLMs did not pass validation. Syntactic validation prevents poor queries from being submitted to the data warehouse, potentially reducing the execution cost, especially for cloud-hosted systems. Semantic validation enables the filtering of queries to avoid destructive actions (e.g., INSERT, TRUNCATE) from being sent to the data warehouse, thereby reducing the risk of misconfigured access controls. Semantic validation also allows the user to configure which tables and columns the LLM may access.

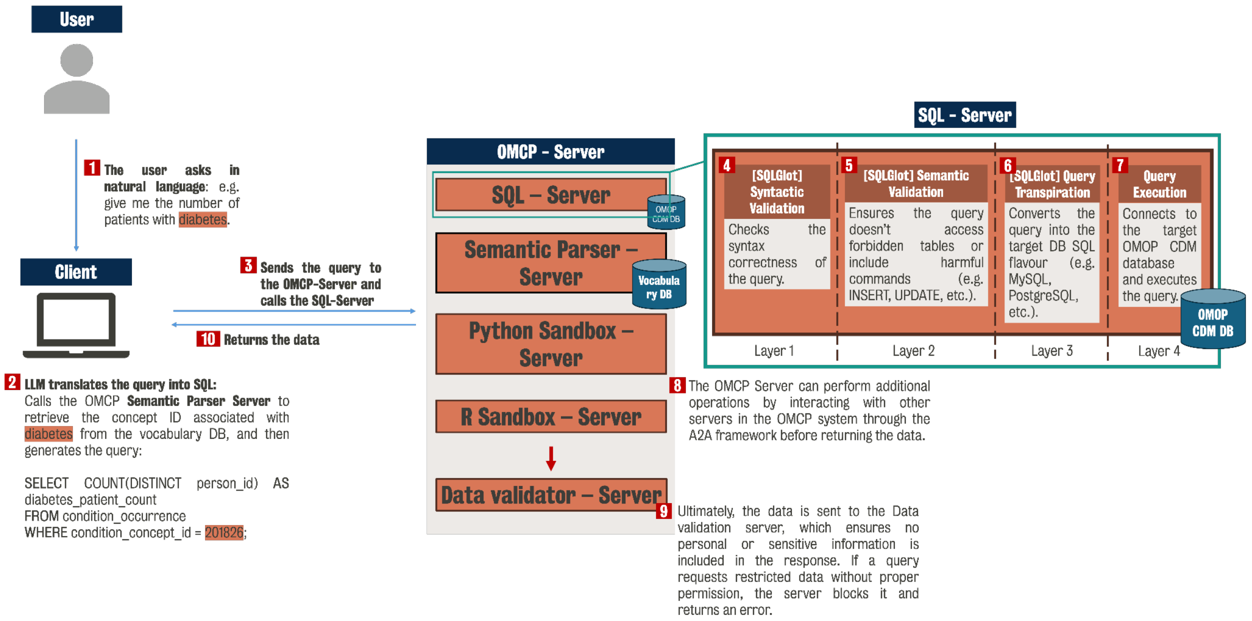

User
1 The user asks in natural language: e.g. give me the number of patients with diabetes.
Client
3 Sends the query to the OMCP-Server and calls the SQL-Server
10 Returns the data
2 LLM translates the query into SQL: Calls the OMCP Semantic Parser Server to retrieve the concept ID associated with diabetes from the vocabulary DB, and then generates the query:
SELECT COUNT(DISTINCT person_id) AS diabetes_patient_count
FROM condition_occurrence
WHERE condition_concept_id = 201826;
OMCP - Server
SQL – Server
OMOP CDM DB
Semantic Parser – Server
Vocabulary DB
Python Sandbox – Server
R Sandbox – Server
Data validator – Server
SQL - Server
4 [SQLGlot] Syntactic Validation
Checks the syntax correctness of the query.
5 [SQLGlot] Semantic Validation
Ensures the query doesn't access forbidden tables or include harmful commands (e.g. INSERT, UPDATE, etc.).
6 [SQLGlot] Query Transpiration
Converts the query into the target DB SQL flavour (e.g. MySQL, PostgreSQL, etc.).
7 Query Execution
Connects to the target OMOP CDM database and executes the query.
OMOP CDM DB
Layer 1
Layer 2
Layer 3
Layer 4
8 The OMCP Server can perform additional operations by interacting with other servers in the OMCP system through the A2A framework before returning the data.
9 Ultimately, the data is sent to the Data validation server, which ensures no personal or sensitive information is included in the response. If a query requests restricted data without proper permission, the server blocks it and returns an error.

Figure 1: OMCP server information flow diagram, with a zoom-in on the SQL-Tool. Multiple layers of security checks are performed using SQLGlot and Ibis before the query is executed against the database.

To support diverse OMOP deployments, SQLGlot is combined with the Ibis framework for SQL transpilation and dataframe semantics, enabling compatibility with multiple database backends.[7 8]

Finally, a Data Validation Tool ensures privacy and access control at the response level. It verifies that OMCP outputs do not include sensitive or personally identifiable information. Where applicable, it anonymises and generalises results; otherwise, it returns an error to prevent disclosure.

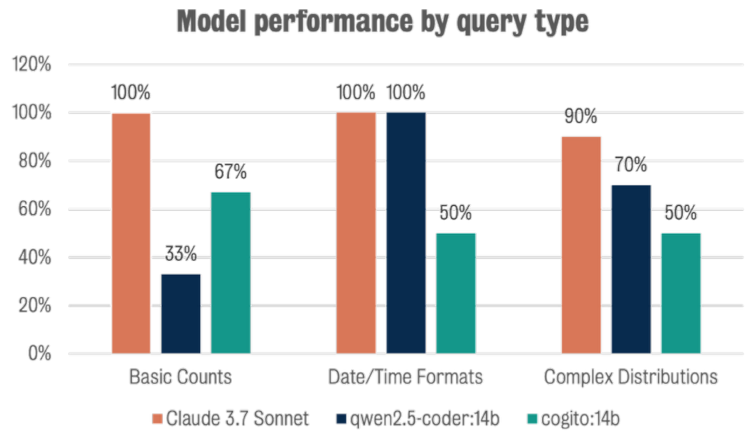
Results

OMCP-SQL was evaluated using both local large language models (LLMs) and Anthropic's Claude Sonnet 3.7. Local testing was performed with Qwen2.5-Coder:14B (Alibaba) and Cogito:14B, both served through Ollama v0.6.8, with Oterm[9] and LibreChat[10] acting as user-facing interfaces. For testing Anthropic models, we used Claude Desktop.

A subset of 15 query descriptions from the ACHILLES suite (Automated Characterisation of Health

Information at Large-scale Longitudinal Evidence Systems) was used to generate natural language prompts aimed at eliciting SQL queries for clinically relevant analyses. To evaluate system robustness, adversarial prompts were also crafted to induce destructive operations (e.g., INSERT, DELETE) or access restricted tables (e.g., METADATA).

System performance was evaluated based on the LLMs' ability to translate prompts into safe and valid SQL queries. Table 1 presents the results of this evaluation. Among the models tested, Claude Sonnet achieved the highest success rate at 93.3%, with an average of 4.73 attempts per query. The number of attempts reflects how often the system queried the database for context before generating a final, valid output.



Basic Counts: e.g. number of patients with at least 1 Dx and 1 Rx.
Date/Time Formats: e.g. number of persons by death month.
Complex Distributions: e.g. number of persons by year of birth.

Table 1: Model comparison with different types of prompts from ACHILLES provided to the model to evaluate its performance.

Conclusion

Our work on OMCP demonstrates the potential of large language models to bridge the accessibility gap between clinicians, researchers, and the OMOP Common Data Model. By combining the semantic SQL parsing capabilities of SQLGlot with the contextual reasoning of LLMs, we developed a system that enables natural language interaction with OMOP. Rather than building a domain-specific model, OMCP uses general-purpose LLMs via the standardised MCP protocol. This approach allows for the OHDSI community to integrate agentic frameworks, such as OMCP-A2A, with locally deployed LLMs, thereby reducing the risk of data leakage to commercial API endpoints.

Evaluation results indicate that local LLMs underperformed not simply because of model size, but due to limited adaptation to schema-grounded NL-to-SQL tasks, difficulties handling OMOP-specific logic, and the stricter enforcement of safety constraints. These models often produced hallucinations, unsafe operations, or invalid joins, further affected by quantisation and shallow repair loops. Addressing these limitations will likely require schema-aware prompting, domain-specific exemplars, and constrained decoding, rather than relying on scaling alone. OMCP thus offers both a practical tool for safe natural language querying and a framework for assessing model performance in health data contexts.

References

1. Ohdsi. The book of OHDSI Observational Health Data Sciences and Informatics. San Bernardino, Ca Ohdsi; 2019.
2. Yu P, Xu H, Hu X, Deng C. Leveraging Generative AI and Large Language Models: A Comprehensive Roadmap for Healthcare Integration. Healthcare [Internet]. 2023 Jan 1;11(20):2776. Available from: https://www.mdpi.com/2227-9032/11/20/2776
3. Introduction - Model Context Protocol [Internet]. Modelcontextprotocol.io. Model Context Protocol; 2025. Available from: https://modelcontextprotocol.io/introduction
4. MCP Server Directory: 4230+ updated daily | PulseMCP [Internet]. PulseMCP. 2025 [cited 2025 May 9]. Available from: https://www.pulsemcp.com/servers
5. A2A Project. A2A: Agent-to-Agent Protocol [Internet]. 2024 [cited 2025 Jun 29]. Available from: https://a2aproject.github.io/A2A/latest/
6. vvcb. OMOP MCP Server [Internet]. Github.io. 2025 [cited 2025 May 9]. Available from: https://fastomop.github.io/omcp/
7. sqlglot API documentation [Internet]. Sqlglot.com. 2023 [cited 2025 May 9]. Available from: https://sqlglot.com/sqlglot.html
8. Ibis [Internet]. Ibis. 2025 [cited 2025 May 9]. Available from: https://ibis-project.org/
9. oterm - oterm [Internet]. Github.io. 2025 [cited 2025 May 9]. Available from: https://ggozad.github.io/oterm/
10. LibreChat [Internet]. Librechat.ai. 2025. Available from: https://www.librechat.ai/