Gradus Vol 12, No 2 (2025)
ISSN 2064-8014

GENERATIVE AI-BASED DATA ANALYSIS USING MODEL CONTEXT PROTOCOL

Kálmán Bolla [1*], *Márk Kovács* [2]

[1] Department of Information Technology, GAMF Faculty of Engineering and Computer Science, John von Neumann University, Hungary, https://orcid.org/ 0000-0002-4666-0990
[2] Department of Information Technology, GAMF Faculty of Engineering and Computer Science, John von Neumann University, Hungary, https://orcid.org/0009-0005-0615-9754
https://doi.org/10.47833/2025.2.CSC.004

Keywords:
Artificial Intelligence
Generative AI
Large Language Model
Model Context Protocol
Data Analysis

Article history:
Received 30 Szeptember 2025
Revised 7 November 2025
Accepted 10 November 2025

Abstract
In this paper we present a system that can perform data analysis and update operations on a database using Generative AI. The high-level architecture of the system and a detailed description of some of its elements will be presented, and a possible practical solution demonstrating this concept in operation will be published. Finally, pre-generated prompts will be used to verify the accuracy of the system's operation.

1 Introduction

In this article, it is assumed that we want to analyse a set of data stored in a fictitious company dating back many years and extract valuable information for the company. Data analysis is of paramount importance for any company that stores large amounts of raw data in its databases over several years. Processing and analysing raw data can yield many business, strategic and operational benefits, helping companies to make more accurate and faster decisions. In this article, we show how the implementation of data analysis using generative artificial intelligence is possible without the user needing to be familiar with the operation of the specific database system and its query language.

Generative artificial intelligence has become a subject of considerable scientific interest in recent years, with the emergence can be used to solve more complex problems than ever before. In addition to generating general text, images, sound and video, large language models are now capable of generating e.g. SQL scripts, which allows integration with a relational database. This capability of LLMs can be exploited to generate queries to be interpreted by the relational database from a user-specified query.

The integration of a relational database and LLM is challenging to be implemented directly, but this problem can be overcome by using the Model Context Protocol (MCP) announced in November 2024. MCP aims to unify the communication between AI models and external data sources.

In this article, we first present the theoretical background necessary for understanding, outline the high-level implementation of our solution, and then the concrete implementation. Finally, the validation of the resulting system and the results obtained are to be established.

2 Theoretical background

This chapter presents the theoretical knowledge essential to understand how the planned system works.

* Corresponding author.
E-mail address: bolla.kalman@nje.hu

2.1 Generative AI

Generative Artificial Intelligence (Gen AI) is one of the most dynamically evolving branches of Artificial Intelligence, with the primary goal of generating content. This content is not only be text-based, but can also be generated using images, video or sound.

The real breakthrough in the evolution of large language models, was caused by the presentation of transformer architecture in Vaswani et al.'s paper [1], creating the basis of today's modern LLMs. The transformer revolutionized natural language processing by enabling models to handle long-range correlations and efficient parallel processing. The next important achievement presented by Devlin and co-authors is the BERT model [2], which can produce deep bidirectional language representations from unlabeled text. BERT significantly outperforms the original transformer architecture on general language understanding tasks such as question answering and text interpretation. Brown and his co-authors presented a paper [3] on GPT-3, which has now worked with 175 billion parameters. GPT-3's architecture retains the basic elements of the transformer, but it is the large scaling and autoregressive design that make it revolutionary, especially in creative tasks. At the time, it was the largest language model ever to demonstrate that increasing the number of parameters radically improves the efficiency of learning from a few examples.

The survey studies [4] [5] published in the last few years provide a comprehensive picture of the development of LLMs, their architecture, training methods, applications and future research directions.

Modern LLMs are not only capable of generating natural text, but can also be used to generate code in various programming languages, generate complex SQL statements, etc. In this article, we will exploit this feature of LLMs, which is to generate correct and complex SQL statements from natural language queries.

2.2 Model context protocol

The Model Context Protocol (MCP) [6] is an open standard and protocol introduced by Anthropic in November 2024. Its purpose is to unify the communication between AI applications (e.g. LLM) and external data sources and tools. It will allow to create a client-server architecture between the AI system and the data source covered by the MCP, where AI will act as client and MCP as server. The single interface provided by MCP is designed to simplify communication between AI systems. In MCP, the concepts of tool, prompt and resource are core components of the protocol, which play different roles in the interaction between AI models and external systems. A tool is an executable function or operation provided by MCP servers. Prompts are predefined templates, managed by the user that help the AI model to make optimal use of tools or resources. Resource is a data source or content made available by the MCP servers for clients to read.

3 High-level system architecture

As mentioned in the introduction of the theoretical background, MCP is a new standard that helps AI agents to access different data sources such as databases, APIs, etc. When using MCP, three important participants can be distinguished: host, client and server. The host computer will run the client (even more than one, e.g. several chatbots) and these clients will connect to one or more MCP servers via MCP protocol. The MCP server will be responsible for integration with the data source, where the latter can be e.g. a relational database, web server, locally stored files, etc. The data provided by the MCP server will eventually be passed to Gen AI, which can produce the most accurate answer possible by augmenting the original question posed by the user with the information in the data source.

The figure below shows the interaction between Gen AI and MCP server.

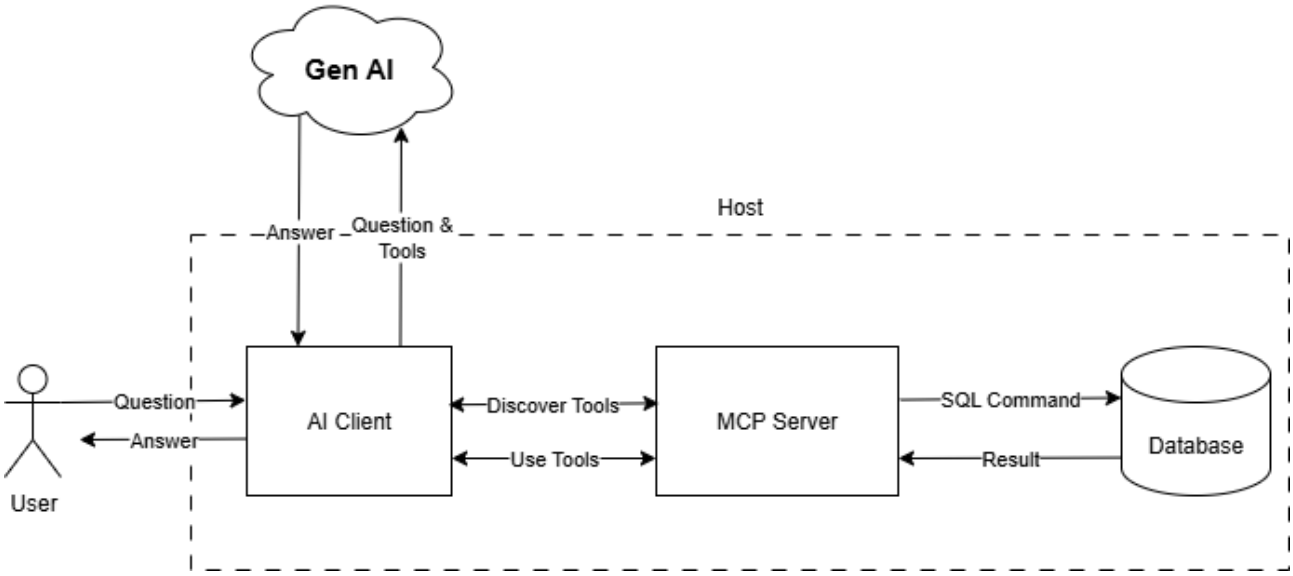

Figure 1. MCP architecture: The relational database contains the data required for queries, which the MCP server accesses using a technical user. The tools recommended by the MCP server are automatically discovered and used by Generic AI. Processing and generation are performed by LLM running in the cloud.

As mentioned earlier, the MCP protocol consists of three parts: tool, prompt and resource. In this article, we assume that using only the tool is sufficient for the MCP server design, and that expected results can be obtained by implementing only two generic tools. The first tool is responsible for querying the structure of the tables in the database, and the second one is able to handle and execute all SQL statements coming from Gen AI.

4 Implementation

This chapter describes the elements used for the specific implementation that creates the system described in the previous chapter.

4.1 Database

The database included in the system will be provided by the Northwind sample database published by Microsoft. Northwind models the operations of a fictional company, Northwind Trades, which is engaged in the trade of exotic food products. The example database actually stores data from a typical ERP system, such as customers, suppliers, products and categories, customer orders, suppliers, employees, etc. Order data is stored at roughly two-year intervals, from 1996 to 1998.

4.2 Gen AI

For the implementation, we chose Claude AI's advanced generative artificial intelligence (AI) chatbot, developed by Anthropic. It aims at providing natural, human-like text and multimodal (text, image, voice) interactions. We want to take advantage of two main features, one is the large context window and the other is the customizable integration options. The former allows for larger amounts of content to be handled, the integration options include support for MCP server. At the time of writing this paper, we were using Claude 4 Sonnet model, which allowed us to generate fast and detailed responses.

4.3 AI Client

The AI client in the system is Claude Desktop, which is also a desktop application developed by Anthropic, allowing us to directly use the Claude AI models running in the cloud. Besides being able to have a text dialogue with the user, it can control the host computer. For us, the Claude desktop is important because of its customisability and its integration with hosted systems. We can

register up to several MCP servers in the configuration file (*claude_desktop_config.json*) that comes with the application, so we can supplement the general knowledge provided by Claude with data from our own dataset.

4.4 MCP server

Our own MCP server was implemented in dotnet core and the *ModelContextProtocol* Nuget package was used for the implementation. Two tools were defined, which in practice means one function per level:

- *GetDatabaseSchema*
- *ExecuteQuery*

GetDatabaseSchema is a tool without input parameters that connects to the database and retrieves its meta-information. Output is table data returned in json format with attribute names, attribute types.

The *ExecuteQuery* tool is responsible for general SQL command execution, where the input to the function is an SQL command, which can be a query or a data modification. Successful operation execution returns the result of the statement also in json format to the caller.

It is worth noting that this solution can lead to data loss or loss of the entire data set in case of inappropriate permissions. Taking this into account, it is advisable to connect the MCP server to the relational database with the most restricted technical user possible, in order to prevent e.g. schema deletion or modification. In addition, we can limit the deletion of records, depending on the usage, we can also allow queries only, without changing the tool, but limiting the privileges of the technical user of the database. Further security considerations such as transaction boundaries, text to SQL generation and execution need to be examined, but these issues will not be discussed in this article.

5 Validation and results

Simple and complex queries and modifications have been defined to check the functioning of the system. By simple operation execution, we mean that a single SQL command is generated in response to a user query. For complex queries, it is assumed that several SQL commands are required to compile the correct answer. It is important to note that token usage, response times, and context window limits were not examined during testing.

The prompts we selected and used to check the correctness of the system are:

5.1 Prompt 1

Can you access the Northwind database? What tables are in the database?

The first question enquired whether Claude AI could reach the relational database on the host machine through the MCP server. If so, query what tables and columns are in the database.

Generative AI successfully located the *GetDatabaseSchema* tool and displayed the tables.

5.2 Prompt 2

Query the id, name and unit price of the products. Display the result in a table.

We want to query products by selecting three attributes from the products table and display the result in a table.

In this case, since the database schema was already known, we simply used the *ExecuteQuery* tool and retrieved the necessary data from the database. The result of the query was displayed in an HTML table, along with basic descriptive statistics based on the resulting dataset.

5.3 Prompt 3

Find out how much revenue the company had each year and summarise the results in a markdown table.

Based on the orders recorded in the database, we would like to produce statistics broken down by year.

Claude has correctly identified that tables of orders and order items will be required for the query and has correctly generated a grouping operation. In addition, he ran two more SQL queries (three in total) to better understand the data set, and found that there was no ordering data for a full year in 1996 and 1998.

5.4 Prompt 4

What products are in short supply and should you order more?

For products, we store in the database how many items are in stock and how many are needed. From these two pieces of information, it is possible to determine whether a product needs to be ordered.

During the question execution location identified which two attributes of the products table should be considered. It generated a piece of SQL query and sorted the result into two groups of urgent and high priority on its own.

5.5 Prompt 5

How many orders delivered by Speedy Express per year?

We filter the orders for a specific supplier and calculate the number of orders.

The first SQL query determined the ID of the supplier company, then the next query grouped the number of orders by year based on the ID and calculated the number of orders shipped by the company in question in each year.

5.6 Prompt 6

What is the average monthly revenue generated by each employee and what is the variation?

First, a SQL join was generated, where employee, order and order items were queried by employee on a monthly basis. It then calculated descriptive statistics for all employees by executing another SQL command.

5.7 Prompt 7

Which customers have reduced their ordering frequency in the last 6 months?

In the first generated query, the earliest and latest order dates were calculated, this was necessary to be able to interpret the last 6 months of text. Based on the last order date, it could now query customers who had no orders in the specified interval or who had reduced their order volume by a large amount (minimum 50%).

5.8 Prompt 8

Which country has the highest customer churn rate?

This is a very similar exercise to the previous query, except that you need to group customers by country. The calculated results are displayed in descending order by dropout value.

5.9 Prompt 9

Estimate the customer lifetime value (CLTV) for the customer with ID SAVEA.

In this task, we want to define a CLTV value for a specific customer (SAVEA). The CLTV, or customer lifetime value, is a business metric that shows how much revenue a particular customer generates for a business over the lifetime of the customer relationship. In other words, it expresses how much money a customer is expected to spend with the company during the time he or she remains a customer.

The first query generated collects all the information needed to calculate the CLTV, such as the number of orders placed by the customer, order dates, total revenue. Then, based on these orders, it calculated the order frequency and wrote two more queries about the customer's orders. Finally, it displayed the calculated values, created a short summary about the customer and calculated the expected revenue from the customer for 3 and 5 years.

5.10 Prompt 10

A customer with AROUT ID would like to order 10 pieces of Chai and 5 pieces of Aniseed Syrup. It should be delivered to 6000 Kecskemét Izsáki út 10 by Speedy Express. Please place the order with the employee ID of Robert King.

As a last prompt, we implemented an employee's order taking, where the products and delivery details are to be entered into the database according to the instruction.

The SQL statements generated for the implementation queried Robert King's employee ID, the ID of the two products mentioned in the prompt, and the ID of the delivery company Speedy Express. It then entered the order and the order items into the database, and finally ran a query to verify that the order had been successfully recorded.

5.11 Summary of the results

The questions formulated above were handled correctly by Claude Sonnet 4 using model MCP tools, answering each question based on the data in the database on the host machine. Depending on the query, one or more SQL commands were generated, these commands were checked one by one and the database data content was also checked.

No problems have occurred concerning the response times to the questions or with the context window being too large.

6 Conclusion

In our article, we presented a system based on Generative AI to perform data analysis from a locally hosted database. The system is powered by the MCP protocol presented by Anthropic, which greatly simplifies the communication between the participants. Between Generative AI and the database, we created a custom MCP server by defining two tools. We checked the system's operation with predefined prompts, where we concluded that the system generated appropriate answers to the questions asked, supplemented with information from the database.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, “Attention Is All You Need,” 31st Conference on Neural Information Processing Systems (NIPS), Advances in Neural Information Processing Systems, Vol. 30., 2017, doi: 10.48550/arXiv.1706.03762

[2] J. Devlin, M. Chang, K. Lee, K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *Proceedings of NAACL-HLT 2019 (Minneapolis, Minnesota)*, pp. 4171–4186, 2019, doi: 10.18653/v1/N19-1423

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam et. al., Language Models are Few-Shot Learners, *Proceedings of NAACL-HLT* 2020, 2020, arXiv:2005.14165

[4] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang et. al., A Survey of Large Language Models, arXiv preprint, 2023, arXiv:2303.18223

[5] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, J. Gao, Large Language Models: A Survey, arXiv preprint, 2024, doi: 10.48550/arXiv.2402.06196

[6] Introducing the Model Context Protocol, https://www.anthropic.com/news/model-context-protocol, 2024 november 25