

# Postgres as an MCP Server

BRUCE MOMJIAN



This presentation covers how Postgres can service model context protocol (MCP) workloads.

<https://momjian.us/presentations>



*Creative Commons Attribution License*

*Last updated: January 2026*

# Background

My previous presentation, *Databases in the AI Trenches*, covered

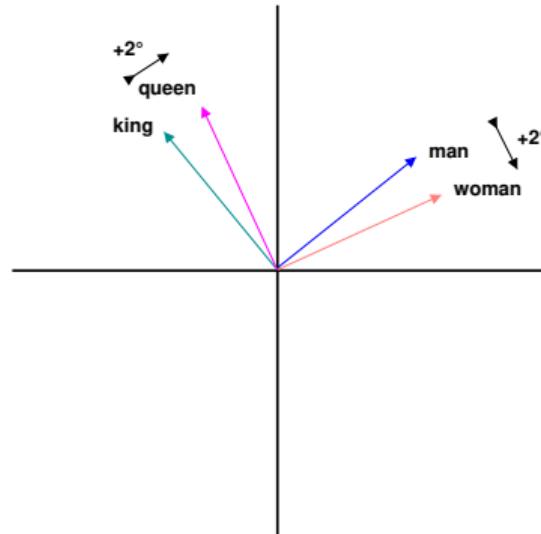
- Hyper-dimmensional vectors
- Text embeddings
- Semantic/vector search
- Generative AI
- RAG
- Deployment

# Outline

1. Generative AI
2. LLM limitations
3. Retrieval-augmented generation (RAG)
4. Model context protocol (MCP)
5. Using a ChatGPT pre-installed MCP server
6. Creating an MCP server
7. Using a custom MCP server with OpenAI
8. Using a Postgres MCP server to read data
9. Using a Postgres MCP server to write data

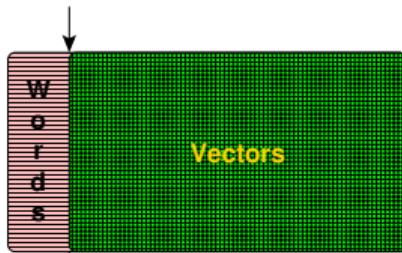
# 1. Generative AI

$$\text{King} - \text{Man} + \text{Woman} = \text{Queen}$$
$$\text{Woman} - \text{Man} + \text{King} = \text{Queen}$$



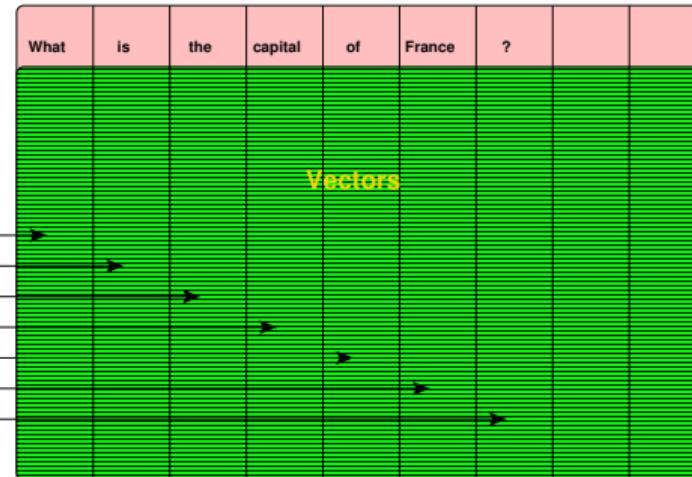
# Populate Initial Attention Block

**Text Embedding Vectors**  
"What is the capital of France?"

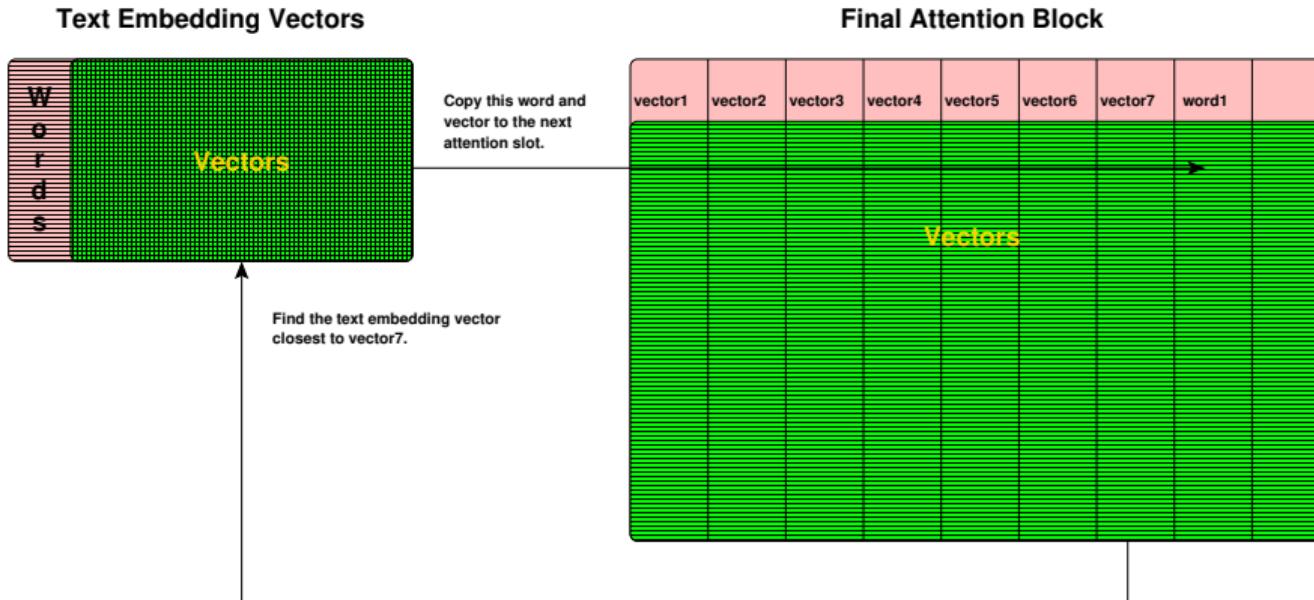


What  
is  
the  
capital  
of  
France  
?

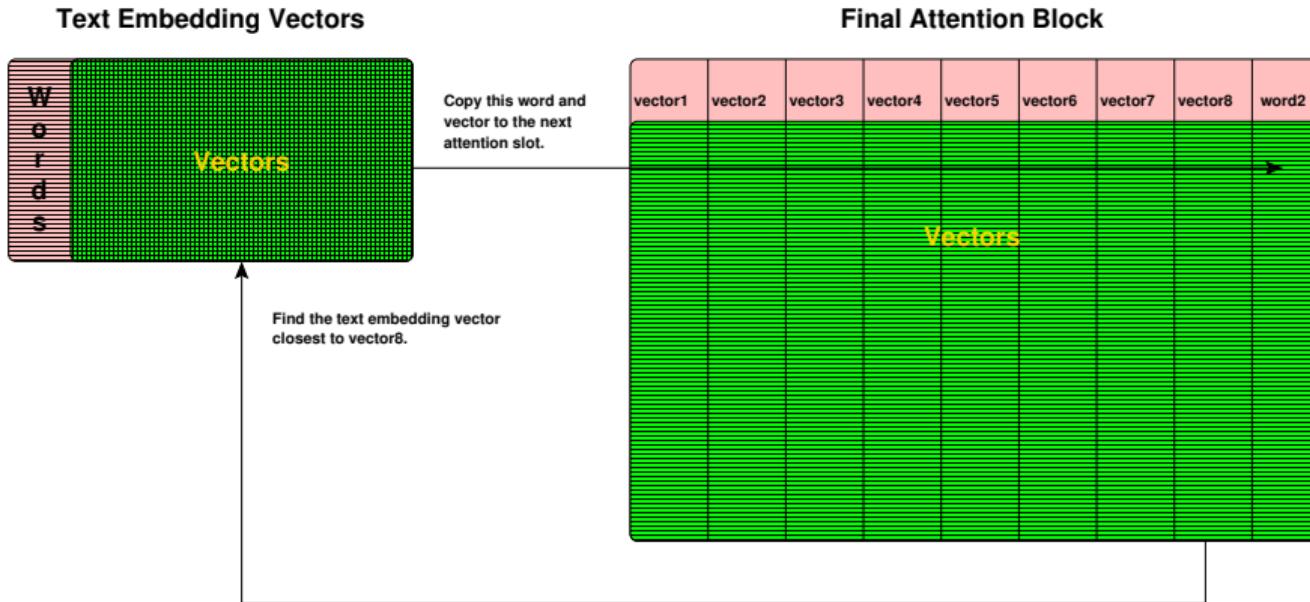
**Attention Block 1, Step 1**



# After Running All Attention Blocks: Store and Output First Word

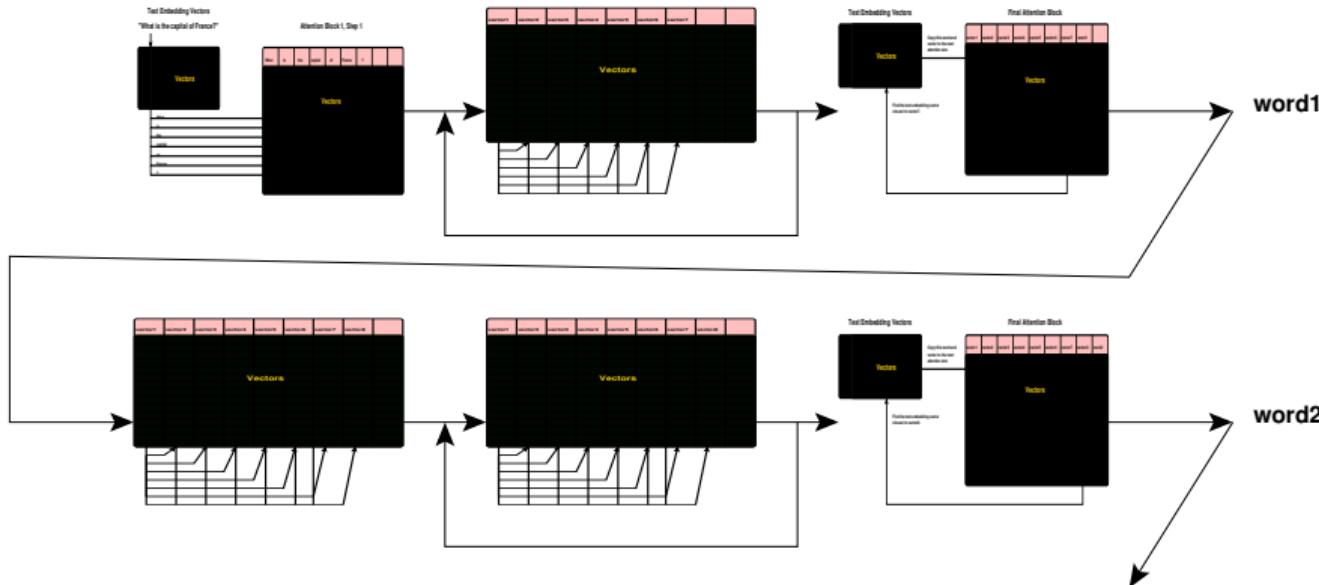


## After Running All Attention Blocks Again: Store and Output Second Word



# Transformer Flow

## Attention Blocks



## 2. LLM Limitations

Large language models (LLMs) are good at generating text, but have limitations:

- Trained on massive amounts of text
- Updating trained vectors with new text is complex
- Results are imprecise

# Where LLMs Struggle

- Discrete data (unclear text relationship)
- Dynamic data (data that frequently changes or changed after the model was trained)
- Accuracy (generative AI output is statistically sampled)

### 3. RAG: System vs. User Prompts

Generative AI Prompt	
System	User
context	user request
restrictions	
personality	

<https://blog.promptlayer.com/system-prompt-vs-user-prompt-a-comprehensive-guide-for-ai-prompts/>

# System Prompt Additions with Retrieval-Augmented Generation (RAG)

```
$ rag -n 'briefly' 'Where is Paris?'
```

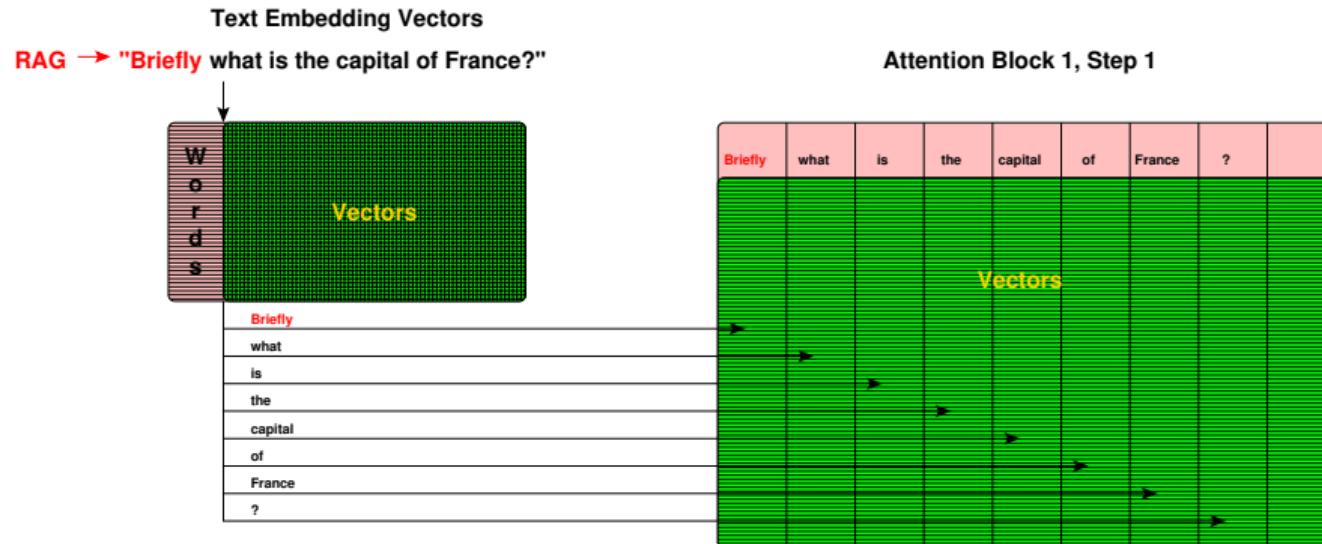
...

Payload:

-----

```
{  
  "model": "gpt-4o-mini",  
  "stream": true,  
  "temperature": 0.5,  
  "max_tokens": 8192,  
  "messages": [  
    {  
      "role": "system"  
      "content": "briefly",  
    },  
    {  
      "role": "user",  
      "content": "Where is Paris?"  
    }  
  ]  
}
```

# RAG in the Transformer



# Problems with RAG

RAG text is added to the “system” section of the generative AI prompt, but it has several limitations:

- Can only do what the AI application programmers intended
- Can’t adjust the system prompt content based on the appropriateness of the output
- Can’t perform external tasks

<https://labelstud.io/blog/seven-ways-your-rag-system-could-be-failing-and-how-to-fix-them/>

## 4. Model Context Protocol (MCP)

Model context protocol (MCP) is a protocol that allows language models to easily retrieve such **external data**, and perform **external tasks** (agentic AI), e.g.:

- What was last month's **total revenue**? (discrete data)
- What is the **current weather** in Colombo, Sri Lanka? (dynamic data)
- What is the **price** per kilogram of Earl Grey tea? (must be accurate)
- **Send email** to retailers about price changes (action)
- **Book a flight** from Philadelphia to Colombo, Sri Lanka (action)

<https://thenewstack.io/why-the-model-context-protocol-won/>  
<https://towardsdatascience.com/clear-intro-to-mcp/>

## Advantages of MCP vs. RAG

- Uses an industry-standard communication protocol for generative AI applications
  - MCP can be added dynamically, without programmer involvement
- Can iteratively call multiple MCP servers based on the appropriateness of the output
- Can trigger external tasks based on the output

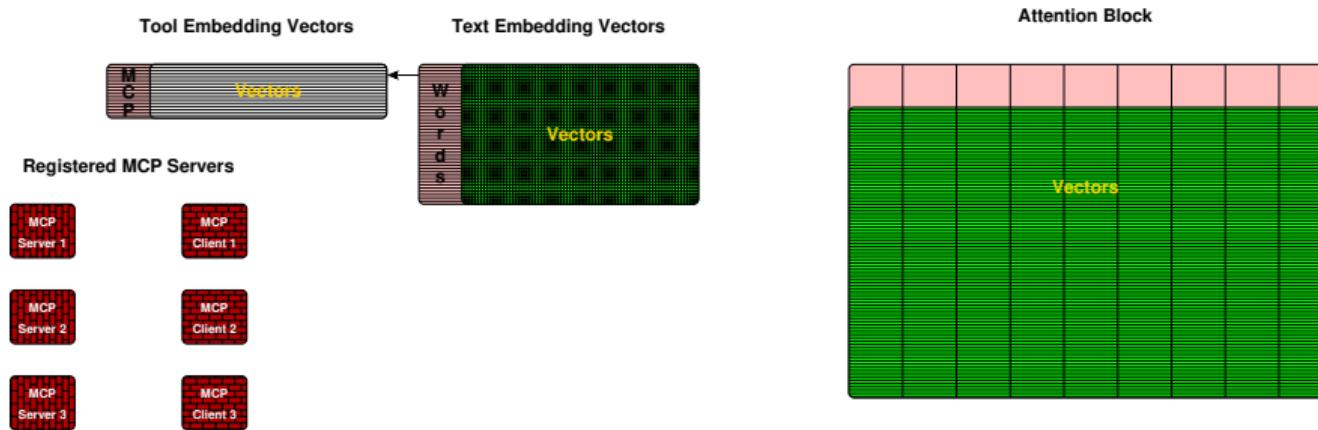
# How Model Context Protocol (MCP) Works

Instead of having programmers decide what data to put in the system prompt, LLMs are used to choose among registered MCP servers, and the LLM output can also trigger external actions via MCP.

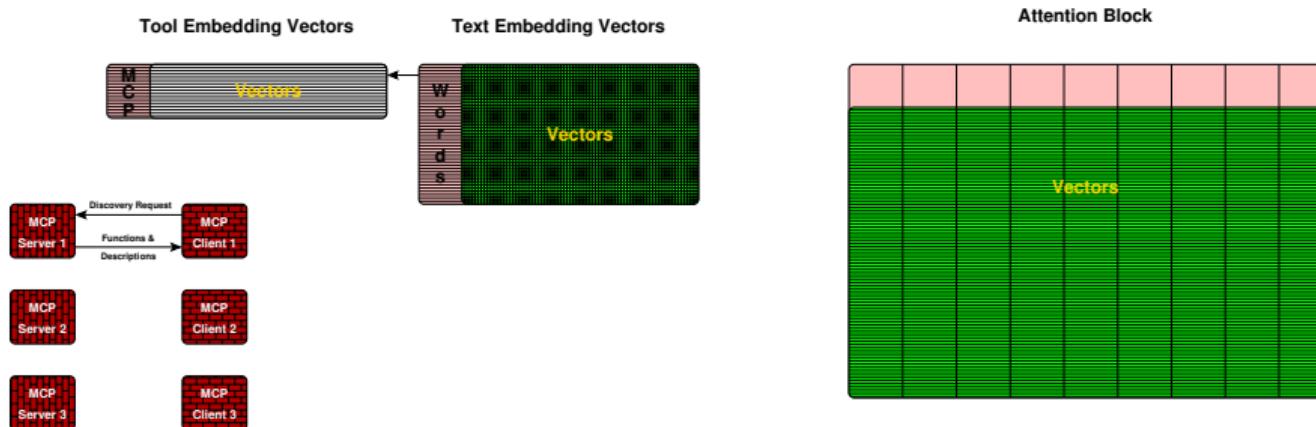
<https://netfoundry.io/ai/how-an-ai-agent-decides-to-call-mcp-tools/>

<https://quickchat.ai/post/mcp-explained>  
<https://www.descope.com/learn/post/mcp>

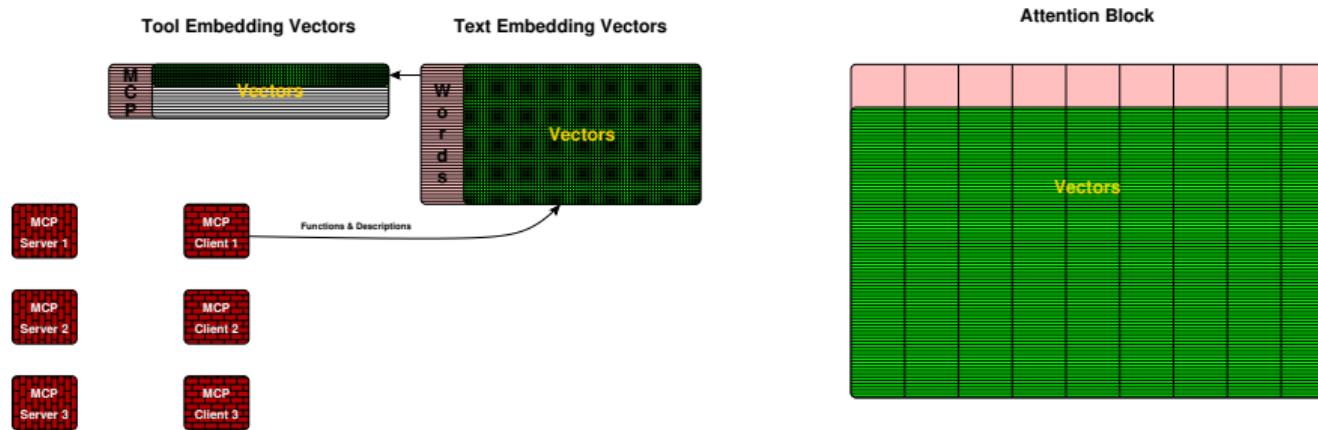
# Registered MCP Servers



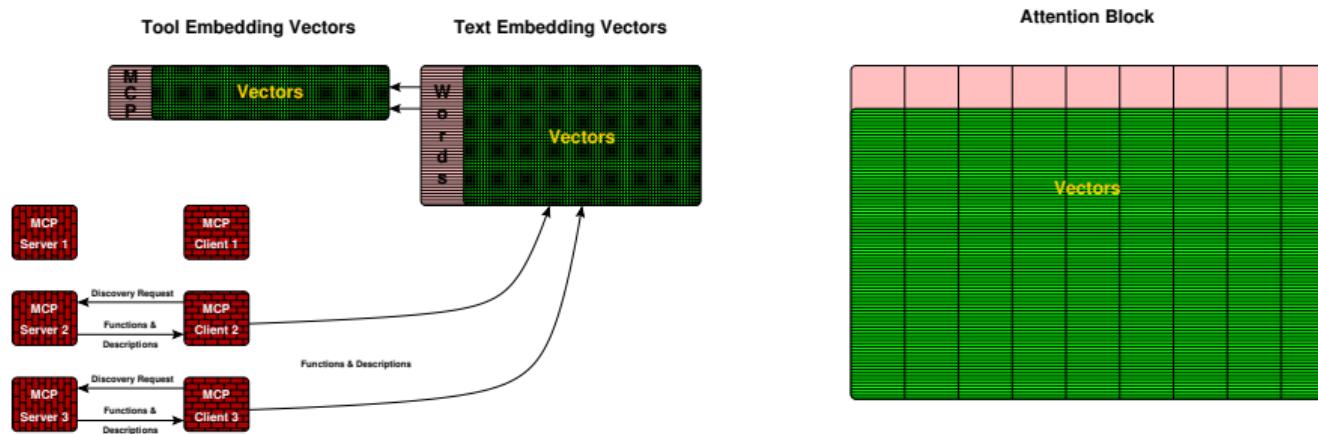
# Discover Request for First MCP Server



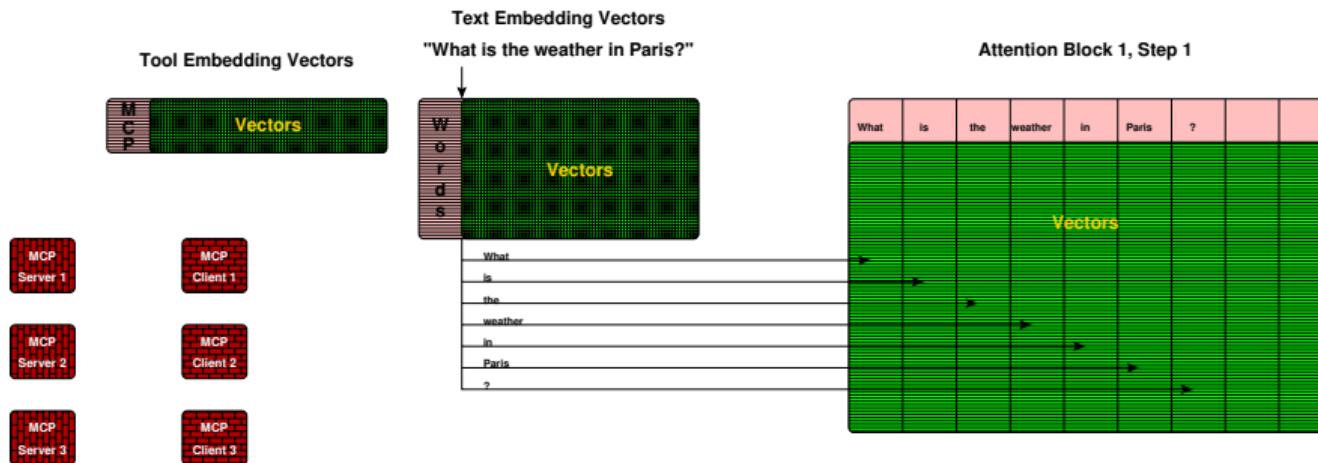
# Load First Tool Embedding Vector



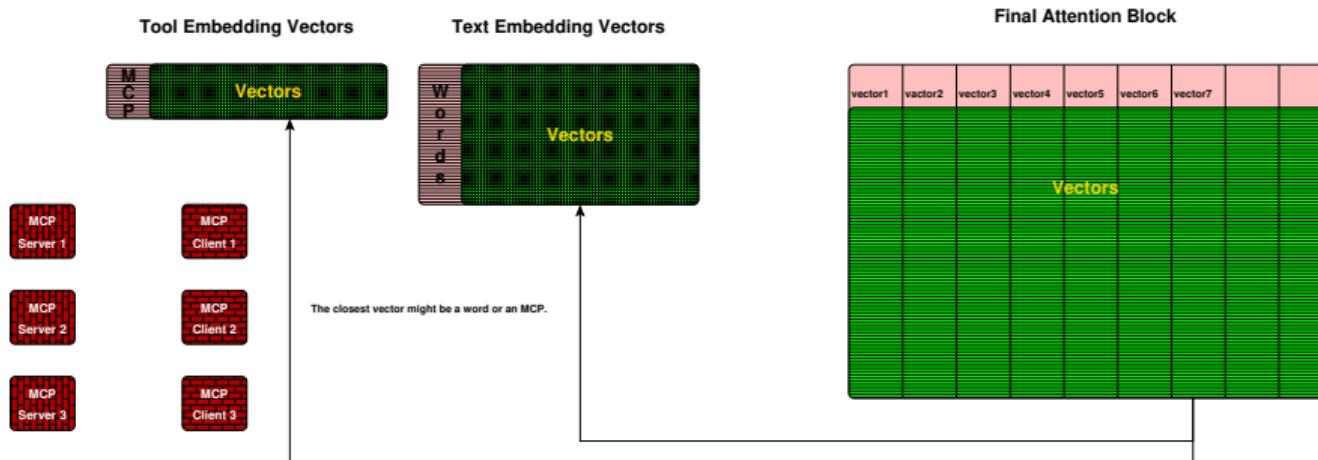
# Load Remaining Tool Embedding Vectors



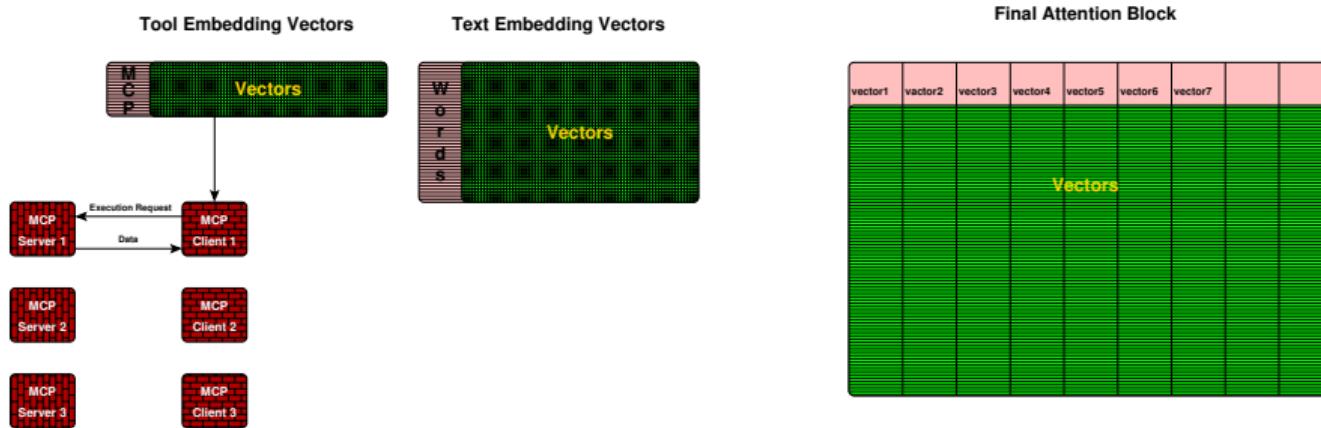
# Execute User Query



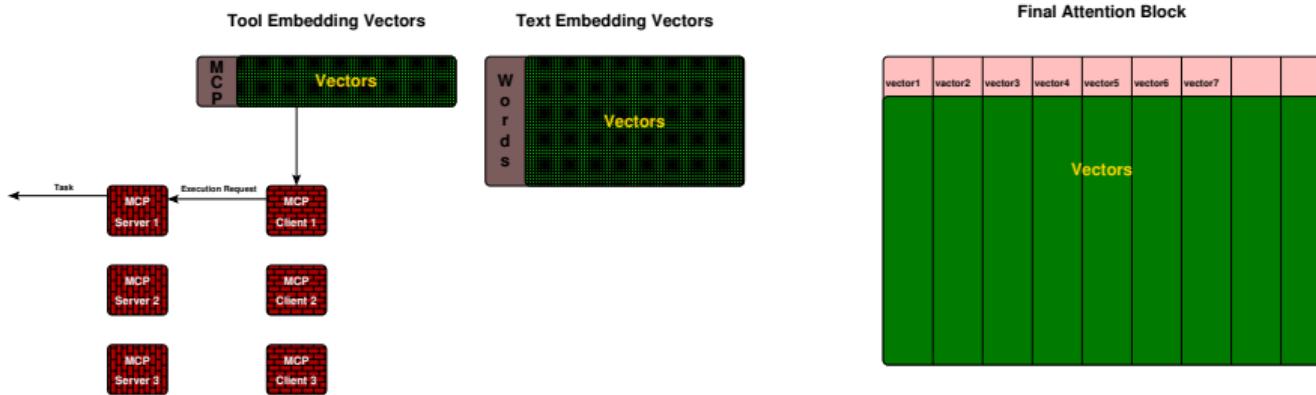
# A Word or an MCP Call?



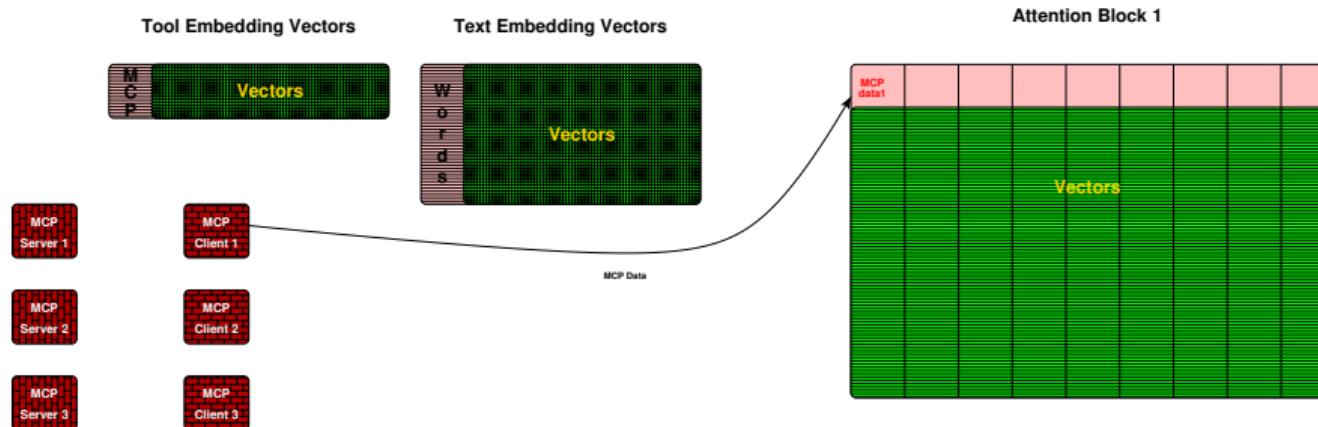
# Assume MCP Call



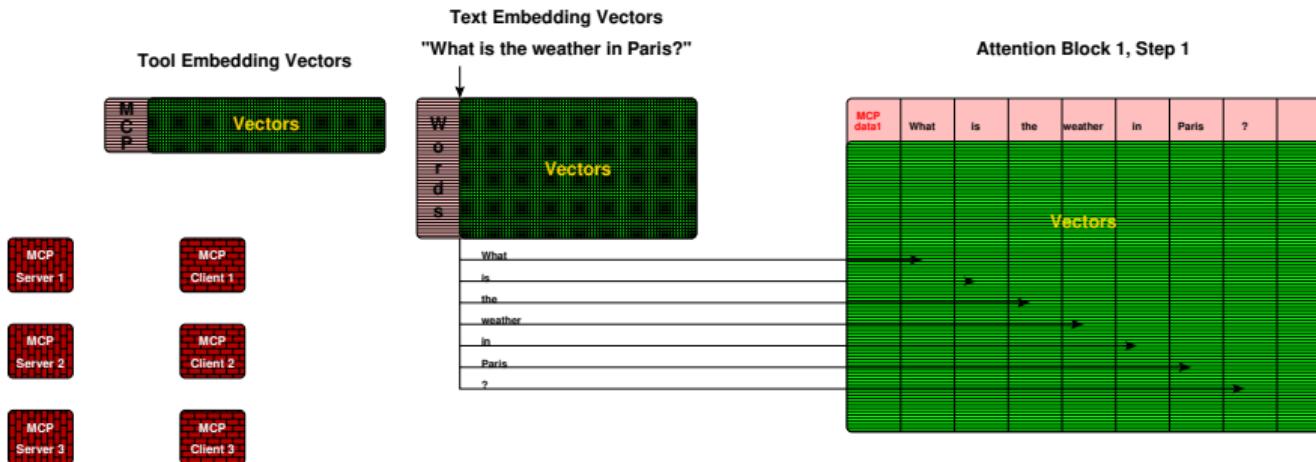
# MCP Could Perform an External Task



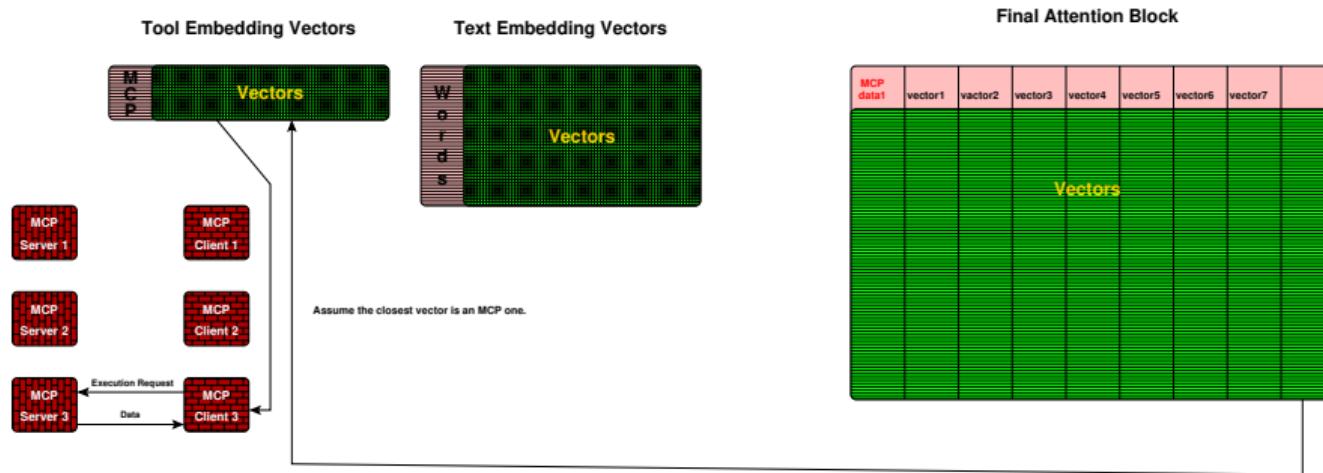
# MCP Data Added to System Prompt



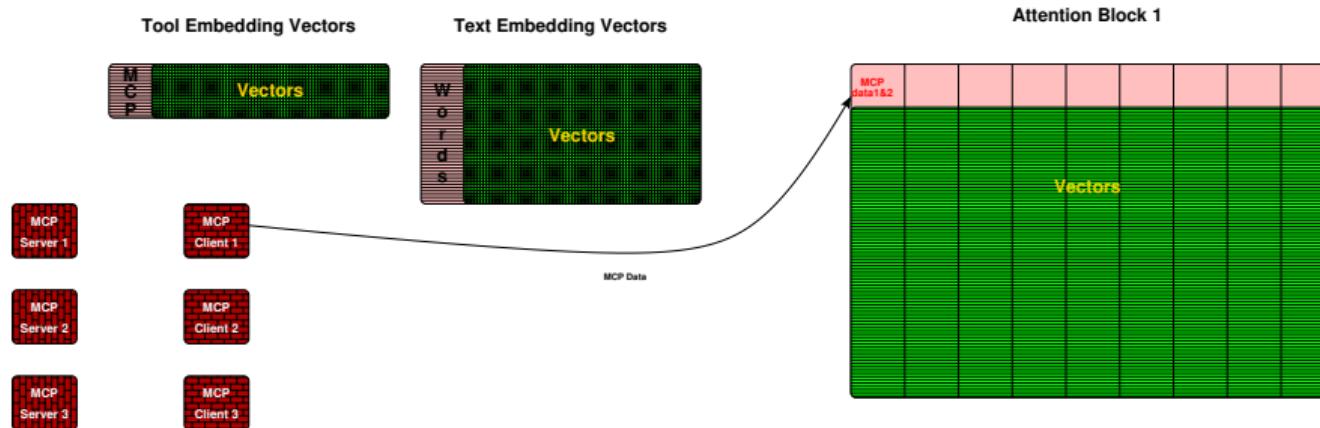
# Re-execute User Query With MCP Data



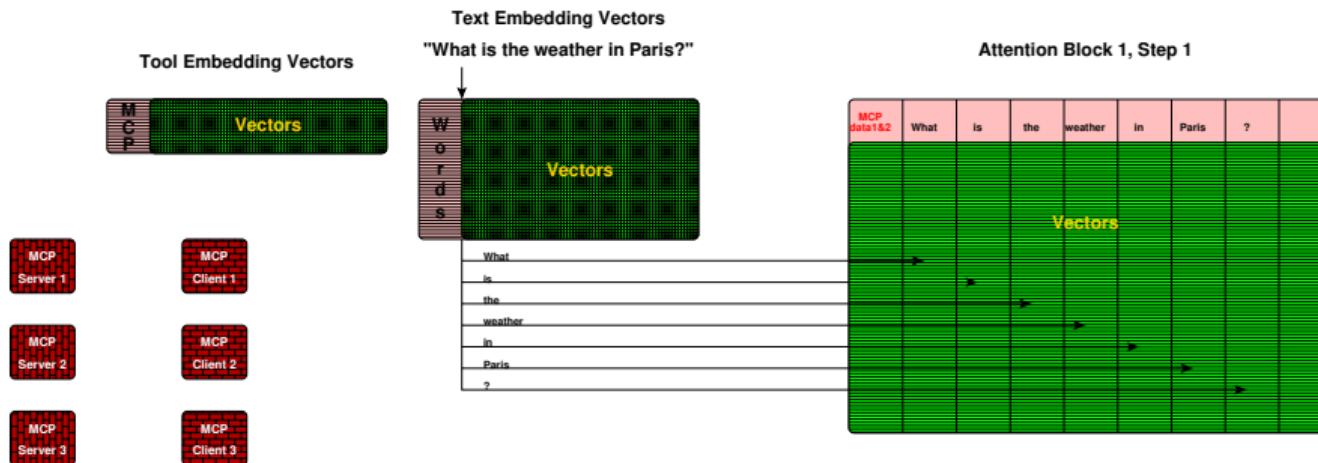
# Another MCP Call



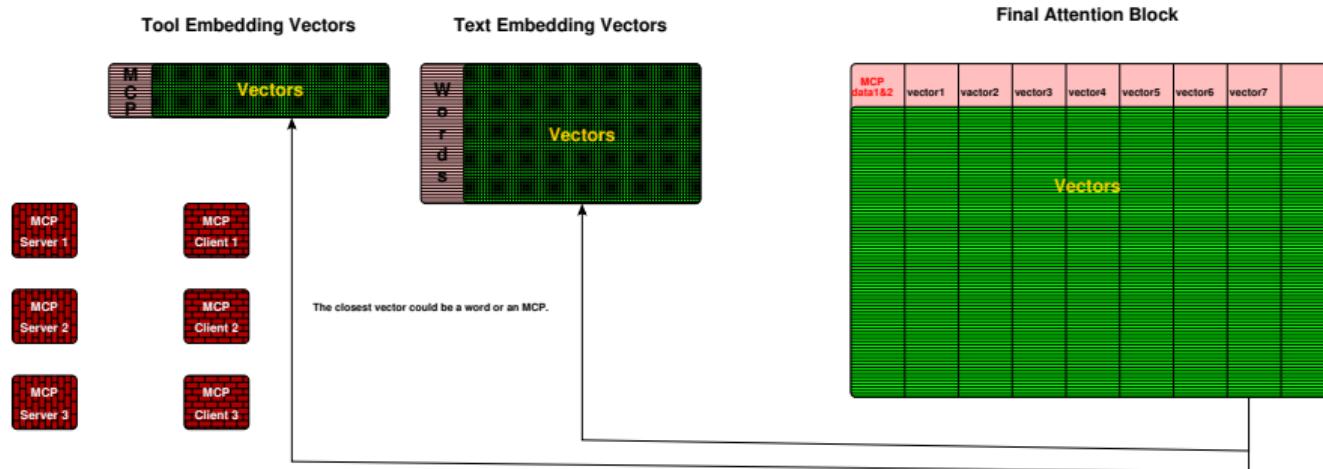
# MCP Data Added to System Prompt



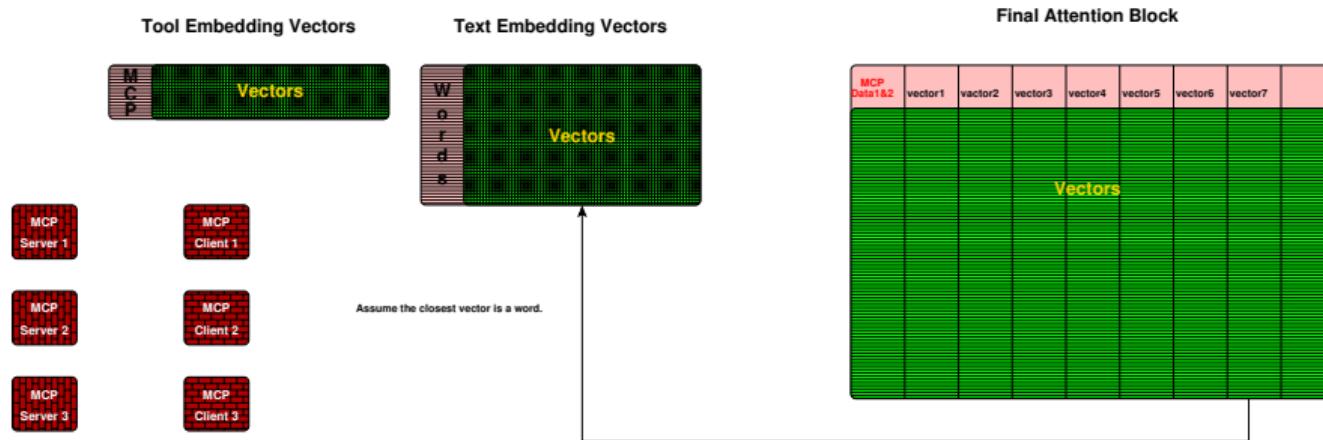
# Re-execute User Query With More MCP Data



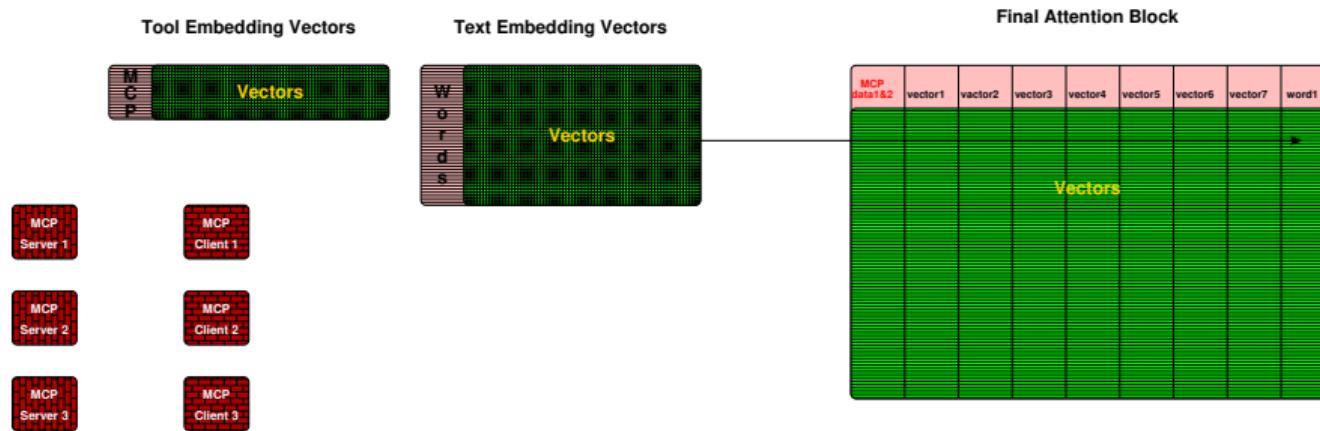
# A Word or an MCP Call?



# Assume a Word



# Output the First Word



## 5. Using a ChatGPT (v5.2) Pre-Installed MCP Server

*What is the current weather in Newtown Square, Pennsylvania, USA?*

*Here's the current weather and forecast for Newtown Square, Pennsylvania (USA):*

*Today:*

*Mostly sunny with a high near 39 °F (4 °C).*

*Light west wind around 5 mph.*

*Tonight, a chance of snow after midnight, increasing clouds, low around 29-32 °F.*

*...*

# What MCP Tool?

*What MCP tool did you just use?*

*I used the web MCP tool.*

*Specifically, I used `web.search` to look up current, time-sensitive weather information for Newtown Square, PA. Weather is considered fresh, real-time data, so I can't rely on my training alone and must query the web to get accurate conditions and forecasts.*

*In short:*

- *MCP tool: `web`*
- *Function: `search`*
- *Why: Weather changes constantly and must be fetched live*

*If you're curious, the system requires the web tool for things like:*

- *weather*
- *current news*
- *live prices or schedules*
- *anything location- or time-dependent*

...

# Other MCP Tools

*Briefly, what MCP tools do you have access to?*

*web – live web info (weather, news, current facts)*

*python – internal computation*

*python\_user\_visible – visible Python (tables, charts, files)*

*genui – utility widgets (weather, currency, units, time)*

*canmore – canvas docs & code*

*automations – reminders & scheduled tasks*

*image\_gen – image generation/editing*

*bio – long-term memory*

*api\_tool – special internal tools*

*container – sandbox execution*

*user\_settings – UI preferences*

# External Actions

*Can you use MCP to perform any external tasks?*

*Short answer: No — not on my own.*

*I cannot initiate or control external systems (servers, APIs, databases, devices, jobs) unless: they are explicitly exposed as an MCP tool, and the platform allows me to invoke them.*

*The MCP tools I have are sandboxed and permissioned.*

...

## 6. Creating an MCP Server: Radiation

*What is the radiation level at 13 Roberts Road, Newtown Square, PA USA?*

*I cannot provide a specific real-time radiation measurement (e.g., gamma dose rate or radon concentration) at a precise street address like 13 Roberts Road, Newtown Square, PA, USA — that level of environmental monitoring isn't publicly published for individual properties or exact GPS points. Instead, here's the best available broader context on radiation where that address is located:*

...

# GQ Electronics GMC-800 Geiger Counter Radiation Detector

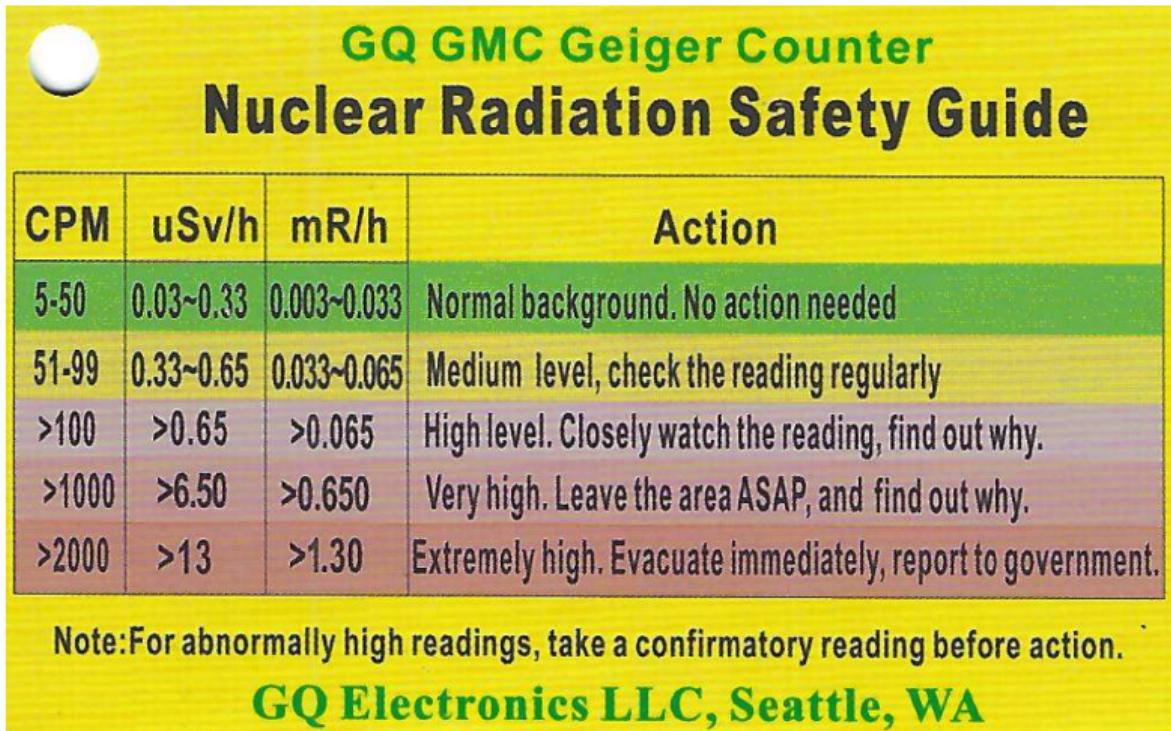


[https://www.gqelectronicsllc.com/comersus/store/comersus\\_viewItem.asp?idProduct=5859](https://www.gqelectronicsllc.com/comersus/store/comersus_viewItem.asp?idProduct=5859)

# Much Higher in Flight



# Radiation Safety Card



**GQ GMC Geiger Counter**  
**Nuclear Radiation Safety Guide**

CPM	uSv/h	mR/h	Action
5-50	0.03~0.33	0.003~0.033	Normal background. No action needed
51-99	0.33~0.65	0.033~0.065	Medium level, check the reading regularly
>100	>0.65	>0.065	High level. Closely watch the reading, find out why.
>1000	>6.50	>0.650	Very high. Leave the area ASAP, and find out why.
>2000	>13	>1.30	Extremely high. Evacuate immediately, report to government.

Note: For abnormally high readings, take a confirmatory reading before action.

**GQ Electronics LLC, Seattle, WA**



Geiger  
counter CPM is  
45,  $\geq 40$

# Retrieve Current Radiation Reading

## Returns CPM (Counts per Minute)

jj

```
#!/bin/bash

stty -F "$GEIGER_TTY" raw -echo "$GEIGER_SPEED"

perl << 'END'
    my $device = "$ENV{GEIGER_TTY}";
    open(my $geiger, '+<', $device) or
        die("cannot open $device");
    print({$geiger} '<GETCPM>');
    # Newer models use a 4-byte CPM for the GMC-500+, not 2
    die("cannot read four bytes from $device")
        if (read($geiger, my $byte4, 4) != 4);
    print unpack('N', $byte4) . "\n";
END
```

<http://www.gqelectronicsllc.com/download/GQ-RFC1801.txt>

# Create an MCP Server

1. Install fastmcp
2. Write a Python script to call the Geiger counter script
3. Test the MCP server
4. Register the MCP server with ChatGPT

<https://www.freecodecamp.org/news/how-to-build-your-own-mcp-server-with-python/>  
<https://gofastmcp.com/getting-started/welcome>

## Install fastmcp

```
$ ~/venvs/fastmcp/bin/python -m pip install fastmcp
```

# Write Geiger Counter Python Script

*What is the total of all invoices?*

To calculate the total of all invoices, we need to sum the amounts for each order:

1. Order 735: Amount = 21
2. Order 4423: Amount = 13
3. Order 1334: Amount = 5

Now, adding them together:

$$\text{Total} = 21 + 13 + 5 = 39$$

The total of all invoices is 39.

## Data Can Be Passed via RAG

*What is the average invoice amount?*

To calculate the average invoice amount, we first need to sum the amounts from the provided orders and then divide by the number of orders.

Here are the amounts from the orders:

- Order 735: 21
- Order 4423: 13
- Order 1334: 5

Now, let's calculate the total amount:

$$\text{Total amount} = 21 + 13 + 5 = 39$$

Next, we count the number of orders, which is 3.

Now, we can calculate the average:

$$\text{Average invoice amount} = \text{Total amount} / \text{Number of orders} = 39 / 3 = 13$$

So, the average invoice amount is 13.

# Complex Calculations Are Possible

*Today is April 15, 2024. With 30-day payment expected, which invoices are overdue?*  
To determine which invoices are overdue as of April 15, 2024, we need to check the invoice dates and see if they are more than 30 days old from the current date.

1. \*\*Invoice for gears\*\*:

- Inv Date: April 12, 2024
- Payment Due Date: May 12, 2024 (April 12 + 30 days)
- Status: Not overdue

2. \*\*Invoice for pencils\*\*:

- Inv Date: February 5, 2024
- Payment Due Date: March 6, 2024 (February 5 + 30 days)
- Status: Overdue

## Complex Calculations Are Possible

3. \*\*Invoice for glue\*\*:

- Inv Date: March 25, 2024
- Payment Due Date: April 24, 2024 (March 25 + 30 days)
- Status: Not overdue

Based on this analysis, the only overdue invoice is for \*\*pencils\*\* (order\_id 4423).

# Complex Calculations Are Possible

*What is the total of invoices that are not pencils?*

To calculate the total of invoices that are not for pencils, we will look at the provided orders and sum the amounts for each product except for pencils.

Here are the relevant orders:

1. Gears: 21
2. Glue: 5

Now, we add these amounts together:

$$21 \text{ (gears)} + 5 \text{ (glue)} = 26$$

Therefore, the total of invoices that are not for pencils is **26**.

## Complex Calculations Are Possible

*Today is April 15, 2024. What is the next expected order date?*

Since today's date is April 15, 2024, and the most recent order date in the provided data is April 12, 2024, we can estimate the next expected order date by looking at the pattern of previous orders. However, there is not enough information in the provided data to determine a specific frequency or pattern of orders. If we assume that orders are placed regularly, we can consider the time intervals between the previous orders:

1. Order on February 5, 2024
2. Order on March 25, 2024 (48 days after the previous order)
3. Order on April 12, 2024 (18 days after the previous order)

Given the decreasing intervals, it's challenging to predict a specific date. However, if we were to assume a similar timeframe, we could expect the next order to occur sometime in the following weeks. A rough estimate might suggest the next order could be expected around late April to early May 2024, but without a clear pattern, it remains uncertain.

# SQL Queries Can Be Generated

*What SQL query will generate the latest invoice date?*

To generate the latest invoice date from a table (let's assume the table is named 'invoices'), you can use the following SQL query:

```
```sql
SELECT MAX(inv_date) AS latest_invoice_date
FROM invoices;
```
```

This query uses the 'MAX()' function to find the most recent date in the 'inv\_date' column, returning it as 'latest\_invoice\_date'. Make sure to replace 'invoices' with the actual name of your table if it's different.

# Data Can Be Taken from a Database

```
CREATE TABLE invoice (
    order_id INTEGER PRIMARY KEY,
    product TEXT NOT NULL,
    amount NUMERIC(10,2),
    inv_date DATE
);

INSERT INTO invoice VALUES
    (735, 'gears', 21, '2024-04-12'),
    (4423, 'pencils', 13, '2024-02-05'),
    (1334, 'glue', 5, '2024-03-25');

$ # This invoice list can be customized by customer.
$ rag "$(psql --tuples-only -c ' \
    SELECT to_json(invoice.*) \
    FROM invoice \
    WHERE cust_id = 12;' ai)" \
> 'How many invoices are there?'
There are three invoices.
```

## 8. Model Context Protocol (MCP)

Text embeddings are good at encoding word relationships, but poor at reporting discrete, dynamic, and always-accurate data. Model context protocol (MCP) is a protocol that allows language models to easily retrieve such **external data**, and perform **external tasks** (agentic AI), e.g.:

- What was last month's **total revenue**?
- What is the **current weather** in Colombo, Sri Lanka?
- What is the **price** per kilogram of Earl Grey tea?
- **Send email** to retailers about price changes
- **Book a flight** from Philadelphia to Colombo, Sri Lanka

<https://thenewstack.io/why-the-model-context-protocol-won/>  
<https://towardsdatascience.com/clear-intro-to-mcp/>

# Customer Resource Handler

```
this.server.setRequestHandler(ListResourcesRequestSchema, async () => {
  return {
    resources: [
      {
        uri: "weather://current",
        mimeType: "application/json",
        name: "Current Weather",
        description: "Retrieve/select customer information"
      }
    ]
  };
});
```

The resource handler uses freeform text to describe its capabilities.

<https://dev.to/glaucia86/building-a-production-ready-weather-mcp-server-with-clean-architecture-redis-cache-and-solid-32cp>

<https://modelcontextprotocol.info/docs/concepts/resources/>  
<https://towardsdatascience.com/tools-for-your-l1m-a-deep-dive-into-mcp/>

## 9. Deployment

overview.eps

# Relational Databases for Semantic/Vector Search & Generative AI

| Section | AI Feature                           | Details                               | DB Appropriateness |
|---------|--------------------------------------|---------------------------------------|--------------------|
| 3       | text embedding vector training       | batched changes, mostly static        | poor               |
| 4       | semantic/vector search               | search existing database contents     | good               |
| 5 & 6   | transformers & generative AI         | billions of comparisons               | poor               |
| 7       | Retrieval-augmented generation (RAG) | add details to user queries           | good               |
| 7       | language queries on retrieved data   | supplied as JSON                      | good               |
| 7       | generate SQL queries                 | natural language to SQL               | unknown*           |
| 8       | Model context protocol (MCP)         | external data & actions for AI models | good               |
|         | data analysis                        | regression and time series            | good               |

Relational databases continue to be appropriate for discriminative AI. AI tools can also help with database migration, e.g., Oracle to Postgres.

[https://momjian.us/main/blogs/pgblog/2018.html#November\\_28\\_2018](https://momjian.us/main/blogs/pgblog/2018.html#November_28_2018)

\* [https://www.reddit.com/r/SQL/comments/127zawr/who\\_here\\_is\\_using\\_chatgpt\\_to\\_help\\_with\\_sql\\_code/](https://www.reddit.com/r/SQL/comments/127zawr/who_here_is_using_chatgpt_to_help_with_sql_code/)

# Deployment Options

As you have seen, there are several options for using generative AI:

- Cloud service model, e.g., ChatGPT
- Self-managed model
  - only publicly trained, e.g., Meta's Llama
    - publicly trained with private fine-tuning<sup>pre</sup>, e.g., domain-specific chat assistant
  - locally trained public and private data sets, e.g., structured-wikipedia
  - locally trained with only private data
  - private training requires an open data model
    - Hugging Face offers many pretrained models and public data sets
    - most OpenAI models are not open<sup>\*</sup>
- The above options can be augmented with local data, i.e., RAG; data derived from
  - personal preferences
  - relational data and supplied as JSON, e.g., PostgreSQL
  - text retrieved either via full text/phrase search or semantic/vector search

<https://zapier.com/blog/hugging-face/>

<https://www.infoworld.com/article/2338922/5-easy-ways-to-run-an-llm-locally.html>

<https://www.nature.com/articles/d41586-024-02998-y>

\* <https://analyticsindiamag.com/ai-mysteries/6-open-source-models-from-openai/>

# Postgres AI Solutions

- pgvector, already covered
- Tembo's pg\_vectorize
- EDB's AI extension *aidb* (Pipelines)
- Timescale's AI extension *pgai*
- PostgresML's AI toolkit
- TensorChord's VectorChord
- AI toolkits from cloud vendors
- Lantern Cloud
- Workik, PopSQL, and SQL AI SQL query generators
- Postgres.AI's PostgreSQL chatbot
- DBtune for server parameter tuning

<https://github.com/ftisiot/postgresql-ai-projects>

## 10. Conclusion

- Pre-computer philosophy
- 1950's Turing test
- 1980's Expert systems
- 1970's - 1990's AI winter
- 2010's Robotics
- 2013 word2vec by Google
- 2017 Attention blocks by Google
- 2022 ChatGPT for generating text, DALL-E for generating images

<https://www.youtube.com/watch?v=uocYQH0cWTs>

# How Did Google Miss the Boat?

- Groundbreaking research by Google in the 2010's to support language translation
- Products focused on revenue-generating activities like web search and advertising
- Worked on AI also to support device control
- Did not focus on aggregating knowledge across web pages like ChatGPT, or did not wish to risk existing revenue streams

[https://www.reddit.com/r/MLQuestions/comments/18wc52b/why\\_isnt\\_google\\_ahead\\_of\\_the\\_competition\\_when\\_it/](https://www.reddit.com/r/MLQuestions/comments/18wc52b/why_isnt_google_ahead_of_the_competition_when_it/)

# The Future: Revolutionary Vision

*A year ago, if you had said to me in our lifetime will we have capabilities like we have today now with ChatGPT4 ... if you explain the kinds of things that ChatGPT4 can do I probably would have said ... I don't know if we will have those capabilities in our lifetime — and now we have it today — so the speed at which this is moving is staggering. — Jon Krohn, June 12, 2023*

<https://www.youtube.com/watch?v=Ku9PM26Cc2c&t=2h12m57s>  
<https://neurosciencenews.com/ai-existential-threat-27543/>

# The Future: Incremental Vision

*We are used to the idea that people or entities that can express themselves, or manipulate language, are smart — but that's not true. You can manipulate language and not be smart, and that's basically what LLMs (large language models) are demonstrating. — Yann LeCun, October 11, 2024*

[https://www.linkedin.com/posts/yann-lecun\\_an-article-in-the-wall-street-journal-in-activity-7250915579228827648-WWA2/](https://www.linkedin.com/posts/yann-lecun_an-article-in-the-wall-street-journal-in-activity-7250915579228827648-WWA2/)

<https://www.msn.com/en-in/money/news/why-were-unlikely-to-get-artificial-general-intelligence-anytime-soon/ar-AA1EWy4y>

<https://www.theatlantic.com/culture/archive/2025/06/artificial-intelligence-illiteracy/683021/>

# Conclusion

24017301626\_14e1fa2565\_c.jpg



<https://momjian.us/presentations>

<https://www.flickr.com/photos/davep-uk/>