

# Building an MCP Server Using Postgres

BRUCE MOMJIAN



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

*<https://momjian.us/presentations>*



*Creative Commons Attribution License*

*Last updated: April 2026*

# Background

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

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

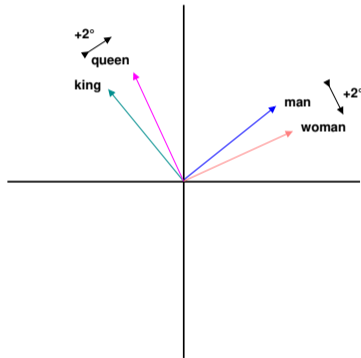
# Outline

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. Creating a Postgres MCP tool

# 1. Generative AI

King - Man + Woman = Queen

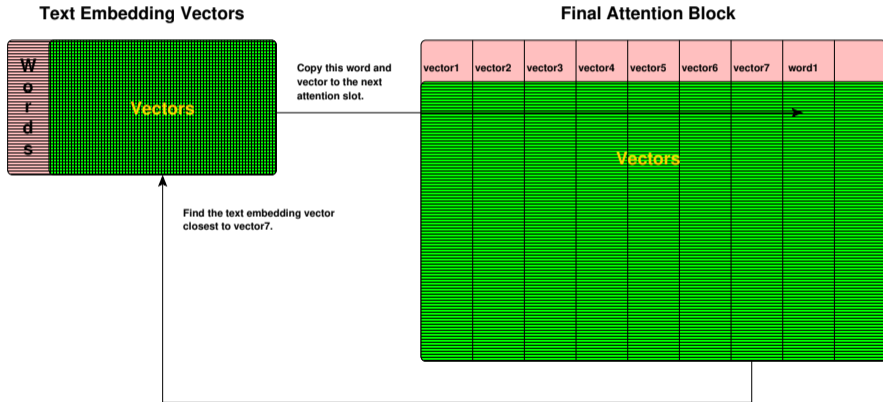
Woman - Man + King = Queen



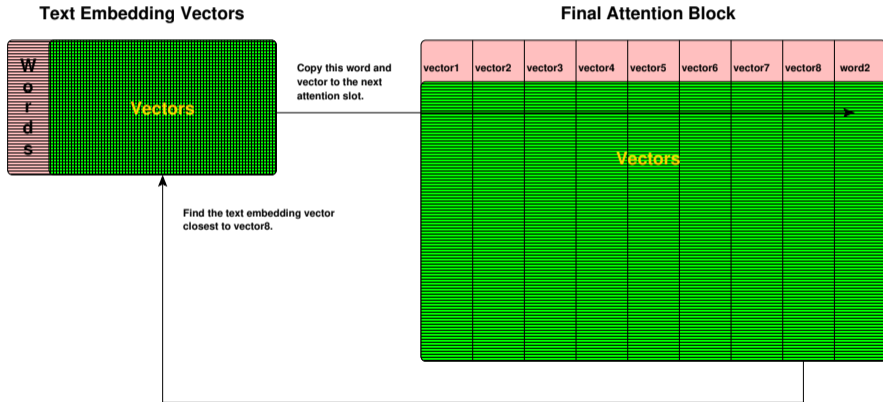
<https://momjian.us/main/presentations/ai.html#trenches>



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

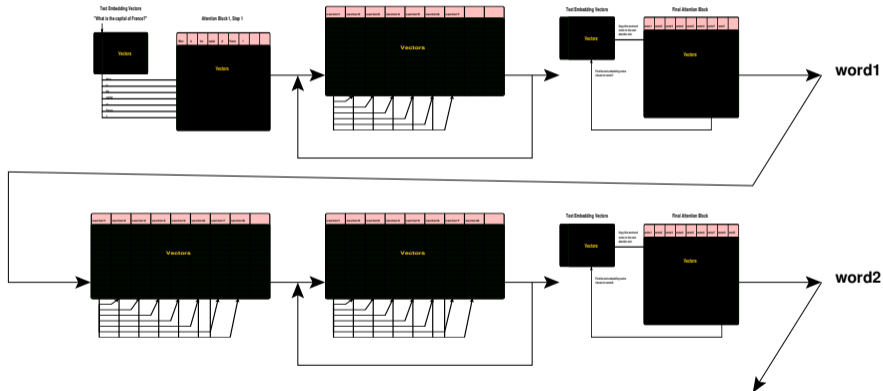


# After Running All Attention Blocks Again: Store and Output Second Word



# 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?'
```

```
...
```

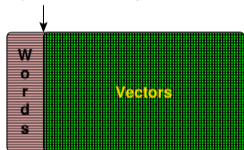
```
Payload:
```

```
-----
```

```
{  
  "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

Text Embedding Vectors  
RAG → "Briefly what is the capital of France?"



Briefly

what

is

the

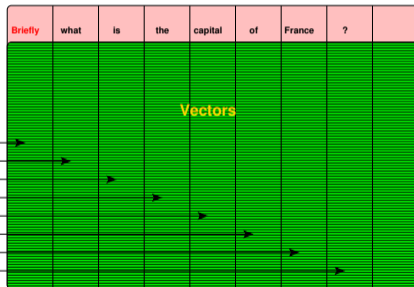
capital

of

France

?

Attention Block 1, Step 1



# 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) 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 and tools based on the appropriateness of the output
- Can trigger external tasks based on the output

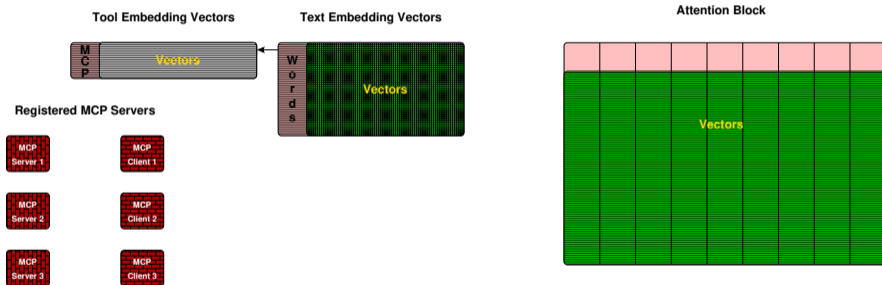
# How Model Context Protocol (MCP) Works

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.

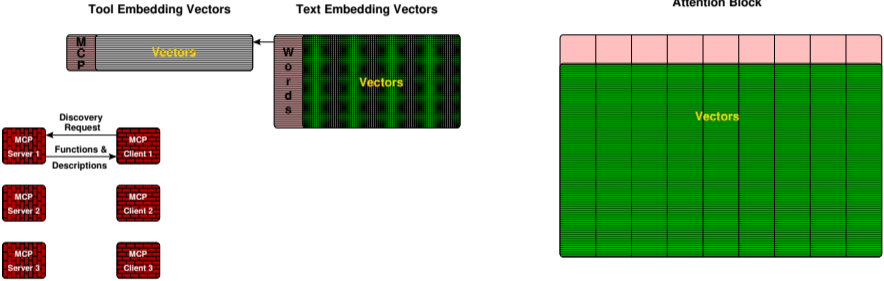
<https://netfoundry.io/ai/how-an-ai-agent-decides-to-call-mcp-tools/>

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

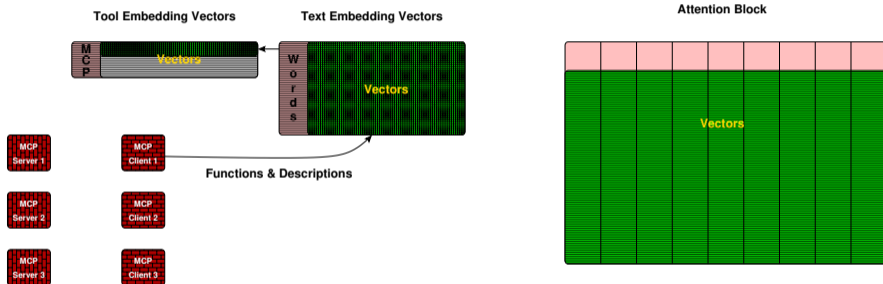
# Registered MCP Servers



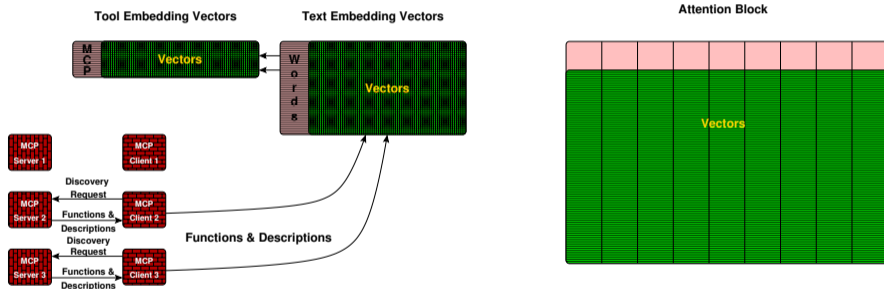
# Discover Request for First MCP Server



# Load First Tool Embedding Vector

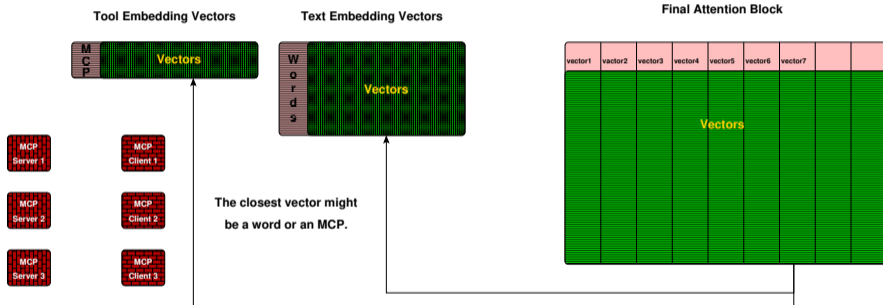


# Load Remaining Tool Embedding Vectors

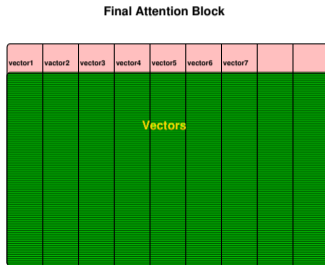
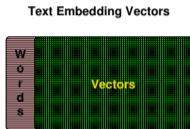
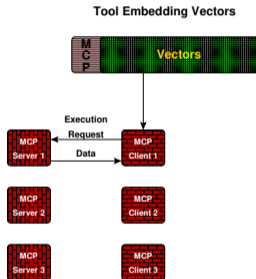




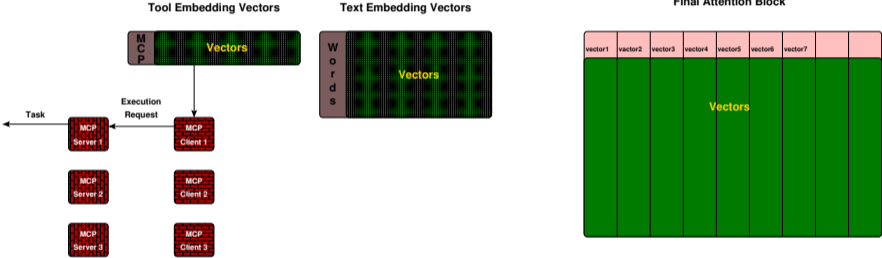
# A Word or an MCP Call?



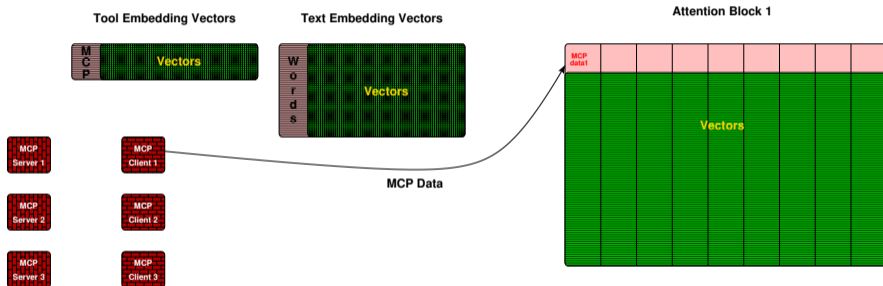
# Assume MCP Call



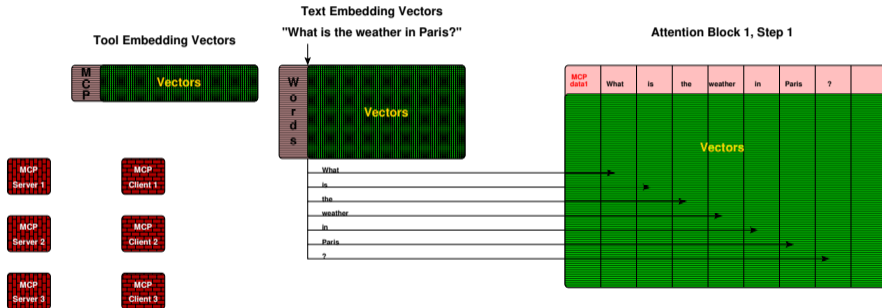
# MCP Could Perform an External Task



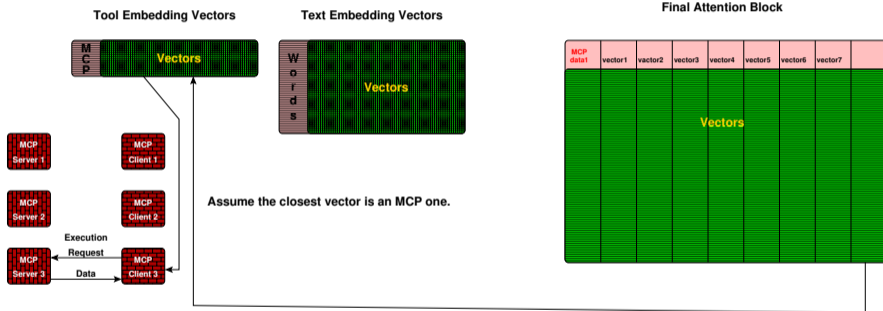
# MCP Data Added to System Prompt



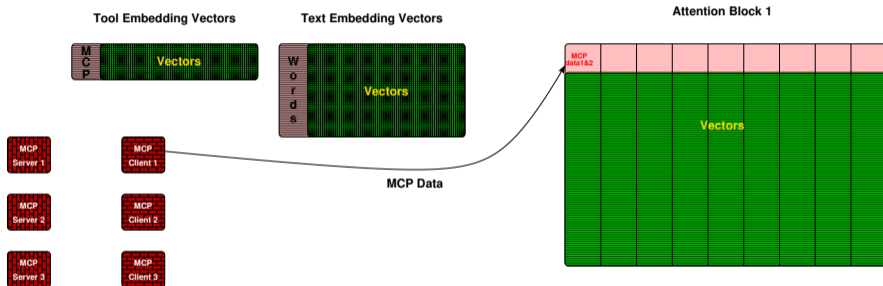
# Re-execute User Query With MCP Data



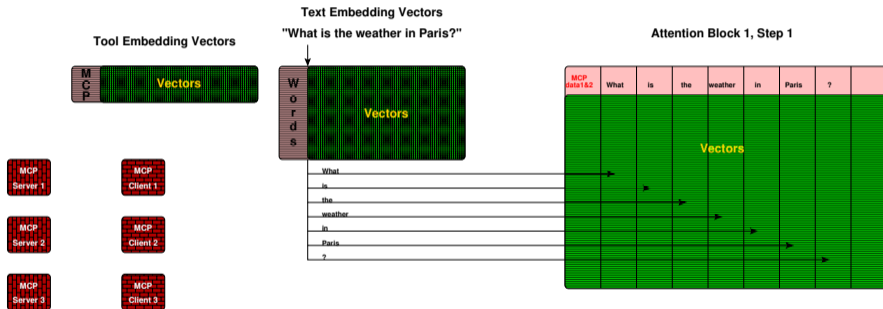
# Another MCP Call



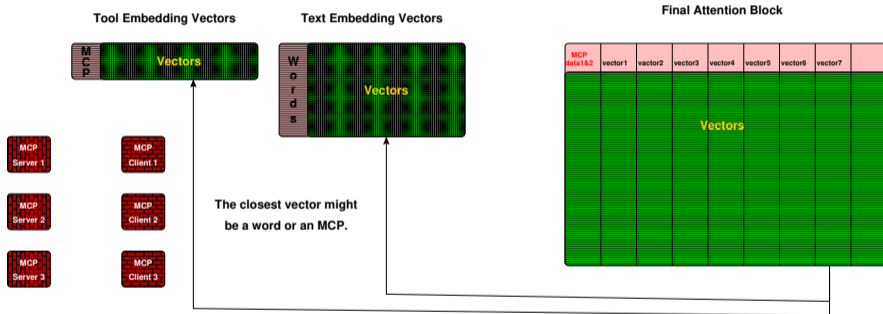
# MCP Data Added to System Prompt



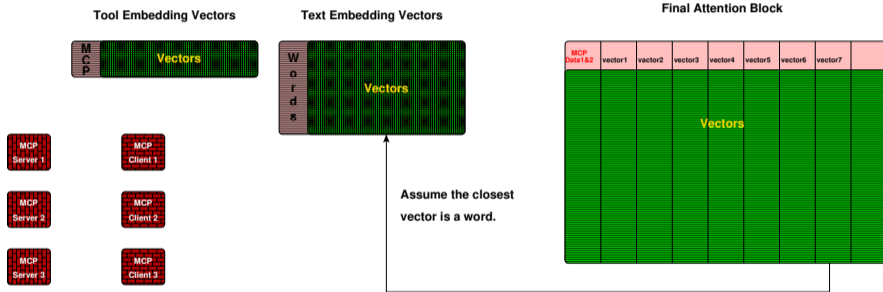
# Re-execute User Query With More MCP Data



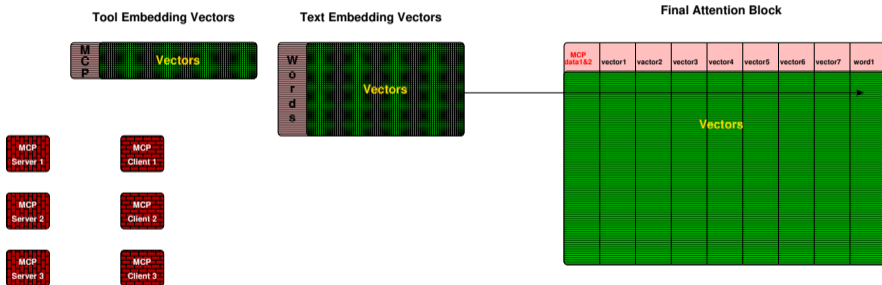
# A Word or an MCP Call?



# 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 Check

*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

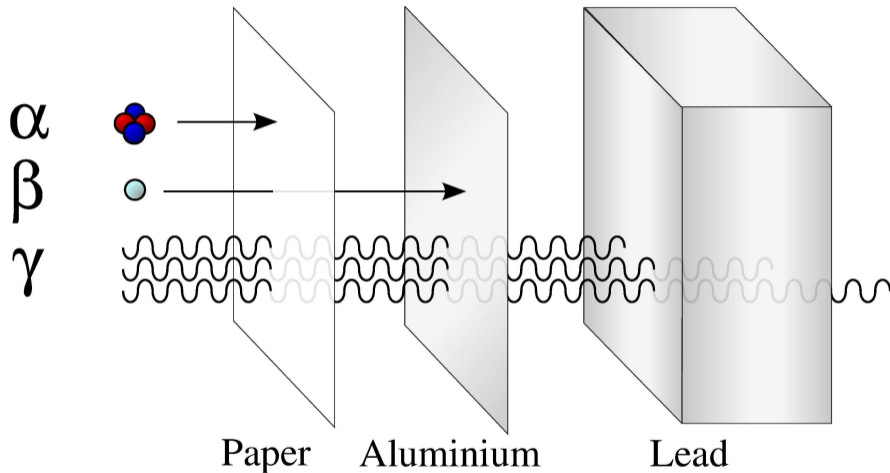


[https://www.gqelectronicsllc.com/comersus/store/comersus\\_viewItem.asp?idProduct=5859](https://www.gqelectronicsllc.com/comersus/store/comersus_viewItem.asp?idProduct=5859)

# Much Higher in Flight



# Detects Beta, Gamma, and X-Ray Radiation, but not Alpha



<https://en.wikipedia.org/wiki/Radiation>

# 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)

```
#!/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. Start the MCP server
4. Test the MCP server
5. 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

```
$ # create a Python virtual environment
$ mkdir -p ~/.venvs
$ python3 -m venv ~/.venvs/fastmcp

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

# Write a Python Script to Call the Geiger Counter Script

```
#!/var/lib/postgresql/.venvs/fastmcp/bin/python

import subprocess
from fastmcp import FastMCP

mcp = FastMCP("Geiger counter MCP server")

@mcp.tool
def geiger() -> int:
    """Return the radiation level (CPM) at 13 Roberts Road, Newtown Square, PA, USA"""
    return subprocess.check_output("/var/lib/postgresql/tmp/geiger", shell=True, text=True)

if __name__ == "__main__":
    mcp.run(transport="http", host="localhost", port=8090)
```

# Start the MCP Server



🌟 **FastMCP 3.0 is coming!**  
Pin `fastmcp < 3` in production, then upgrade when you're ready.

🚀 **Update available: 2.14.5**  
Run: `pip install --upgrade fastmcp`

```
[02/19/26 15:23:12] INFO: Starting MCP server 'Geiger counter MCP server' with transport 'http' on http://localhost:8090/mcp  
INFO: Started server process [302013]  
INFO: Waiting for application startup.  
INFO: Application startup complete.  
INFO: Uvicorn running on http://localhost:8090 (Press CTRL+C to quit)
```

server.py:2580

# Configure Apache as a Proxy

```
<VirtualHost *:443>
  ...
  ProxyPreserveHost On
  ProxyPass /mcp http://localhost:8090/mcp
  ProxyPassReverse /mcp http://localhost:8090/mcp
  ...
</VirtualHost>
```

# Test the Mcp Server: Get the mcp-session-id

```
$ curl --silent --show-headers \
  --request POST https://momjian.us/mcp \
  --header "Content-Type: application/json" \
  --header "Accept: text/event-stream, application/json" \
  --data \
  '{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "initialize",
    "params": {
      "protocolVersion": "2024-11-05",
      "capabilities": {},
      "clientInfo": {
        "name": "curl-client",
        "version": "1.0.0"
      }
    }
  }' |
# Output the session id so it can be easily set in the shell.
awk '$1 == "mcp-session-id:" {printf "export MCP_SESSION_ID='%s'\n", $2}'
export MCP_SESSION_ID=fcd1ac720122480388a80bbe003441ca
$ export MCP_SESSION_ID=fcd1ac720122480388a80bbe003441ca
```

# List MCP Tools

```
$ curl --request POST https://momjian.us/mcp \
  --header "Content-Type: application/json" \
  --header "Accept: text/event-stream, application/json" \
  --header "Mcp-Session-Id: $MCP_SESSION_ID" \
  --data \
  '{
    "jsonrpc": "2.0",
    "id": 2,
    "method": "tools/list"
  }'
event: message
data: {"jsonrpc":"2.0","id":2,"result":{"tools":[{"name":"geiger",
"description":"Return Geiger counter CPM reading at 13 Roberts Road, Newtown Square, PA, USA",
"inputSchema":{"properties":{},"type":"object"},
"outputSchema":{"properties":{"result":{"type":"string"}}, "required":["result"],
"type":"object", "x-fastmcp-wrap-result":true}, "_meta":{"_fastmcp":{"tags":[]}}}]}}
```

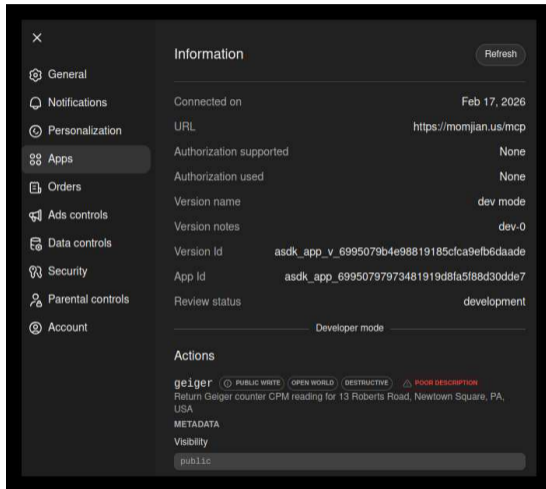
<https://portkey.ai/blog/mcp-message-types-complete-json-rpc-reference-guide/>

# Call MCP Tool

```
$ curl --request POST https://momjian.us/mcp \  
  --header "Content-Type: application/json" \  
  --header "Accept: text/event-stream, application/json" \  
  --header "Mcp-Session-Id: $MCP_SESSION_ID" \  
  --data \  
'{  
  "jsonrpc": "2.0",  
  "id": 2,  
  "method": "tools/call",  
  "params": {  
    "name": "geiger",  
    "arguments": {}  
  }  
'  
event: message  
data: {"jsonrpc":"2.0","id":2,"result":{"content":[{"type":"text","text":"14\n"}],  
"structuredContent":{"result":"14\n"},"isError":false}}
```

<https://portkey.ai/blog/mcp-message-types-complete-json-rpc-reference-guide/>

# Register the MCP Server With ChatGPT



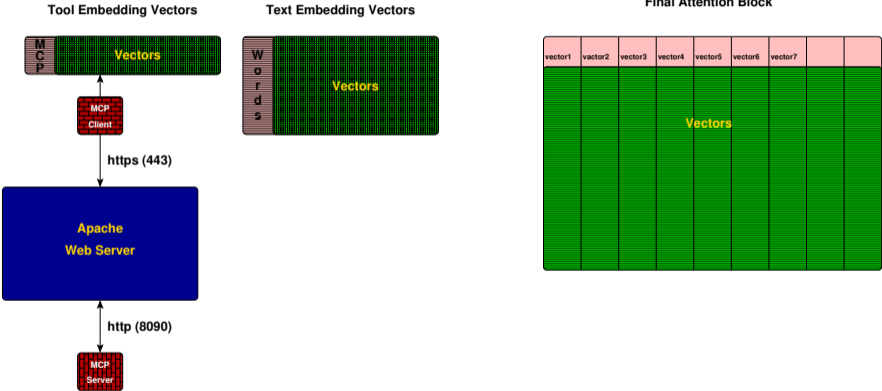
The screenshot displays the configuration page for an MCP server. On the left is a sidebar menu with options: General, Notifications, Personalization, Apps (selected), Orders, Ads controls, Data controls, Security, Parental controls, and Account. The main content area is titled 'Information' and includes a 'Refresh' button. It lists various server details:

- Connected on: Feb 17, 2026
- URL: <https://momjian.us/mcp>
- Authorization supported: None
- Authorization used: None
- Version name: dev mode
- Version notes: dev-0
- Version Id: asdk\_app\_v\_6995079b4e98819185cfca9efb6daade
- App Id: asdk\_app\_69950797973481919d8fa5f88d30dde7
- Review status: development

Below the information is a 'Developer mode' toggle, which is currently turned on. Under the 'Actions' section, there is an entry for an app named 'geiger'. This app has several status tags: PUBLIC WRITE, OPEN WORLD, DESTRUCTIVE, and POOR DESCRIPTION. Its description is 'Return Geiger counter CPM reading for 13 Roberts Road, Newtown Square, PA, USA'. Underneath, there is a 'METADATA' section with a 'Visibility' dropdown menu currently set to 'public'.

<https://gofastmcp.com/integrations/chatgpt>

# Architecture



OpenAI only communicates with MCP on port 443.

# Radiation Check

*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:*

...

# Radiation Check

*Use my custom app*

*Called Tool:*

*The live radiation reading at 13 Roberts Road, Newtown Square, PA is:*

*14 CPM (Counts Per Minute)*

*What That Means*

- *14 CPM is a normal background radiation level.*
- *Typical natural background radiation generally ranges from 5–25 CPM, depending on location and detector sensitivity.*
- *This level is considered safe and normal for outdoor ambient radiation.*

*If you'd like, I can convert this to approximate  $\mu\text{Sv/hr}$  or help interpret trends over time.*

# Custom App

*Briefly, why did I have to specify that my custom app should be used?*

*Because I don't automatically use custom or external apps unless you explicitly tell me to.*

*When you first asked, I defaulted to general/public sources. Saying "use my custom app" told me to access your specific Geiger tool instead of standard data sources.*

# Average Radiation

*Take five measurements and give me an average.*

*Called Tool:*

*Here are the five live measurements (CPM):*

1. 15
2. 16
3. 13
4. 15
5. 15

*Average Calculation*

$$(15+16+13+15+15) / 5 = 74 / 5 = 14.8$$

***Average Radiation Level: 14.8 CPM***

*This remains well within normal natural background radiation levels.*

*If you'd like, I can continue sampling and calculate a rolling average or standard deviation for trend analysis.*

## 7. Creating a Postgres MCP Tool



Job Adriaensz. Berckheyde, *The Baker*

# Create Database Schema for Pretzel Inventory Tracking

```
CREATE DATABASE mcp;
```

```
CREATE USER mcp;
```

```
\connect mcp
```

```
CREATE TABLE pretzel (  
    quantity INTEGER CHECK (quantity >= 0)  
);
```

```
INSERT INTO pretzel VALUES (0);
```

```
GRANT ALL ON TABLE pretzel TO mcp;
```

# Write an MCP Python Script

```
#!/var/lib/postgresql/.venvs/fastmcp/bin/python

import subprocess
from fastmcp import FastMCP

mcp = FastMCP("Pretzel inventory tracker MCP server")

@mcp.tool
def get_pretzel_inventory() -> int:
    """Return the number of unsold pretzels"""
    return subprocess.check_output(
        "psql --tuples-only -c 'SELECT quantity FROM pretzel;' -d mcp",
        shell=True, text=True)
```

# Write an MCP Python Script

```
@mcp.tool
def sold_one_pretzel() -> str:
    """Call this when a pretzel is sold;
       it reduces the number of sold pretzels by one;
       success returns 'UPDATE 1'"""
    return subprocess.check_output(
        "psql --tuples-only -c 'UPDATE pretzel SET quantity = quantity - 1;' -d mcp",
        shell=True, text=True)

@mcp.tool
def baked_6_pretzels() -> str:
    """Call this when a tray of 6 pretzels is baked and ready for sale;
       success returns 'UPDATE 1'"""
    return subprocess.check_output(
        "psql --tuples-only -c 'UPDATE pretzel SET quantity = quantity + 6;' -d mcp",
        shell=True, text=True)

if __name__ == "__main__":
    mcp.run(transport="http", host="localhost", port=8090)
```

# Inventory Report

*Using my custom app, how many pretzels are available for sale?*

*Called Tool:*

*Your app currently shows 0 pretzels available for sale.*

*Looks like it's time to bake another batch!*

# Pretzels Baked

*I just baked a tray of pretzels; please update the inventory*

*Called Tool:*

*Nice! Your tray of 6 pretzels has been added to inventory.*

*You now have 6 pretzels available for sale.*

## Two Pretzels Sold

*I sold two pretzels; please update the inventory*

*Called Tool:*

*Got it! I've recorded the sale of 2 pretzels.*

*You now have 4 pretzels remaining in inventory.*

## Four Pretzels Sold

*I sold four pretzels; please update the inventory*

*Called Tool:*

*All set!*

*I've recorded the sale of 4 more pretzels.*

*You now have 0 pretzels remaining in inventory. Time to bake another tray!*

# One Pretzel Sold

*I sold one pretzel; please update the inventory*

*Called Tool:*

*I attempted to update the inventory, but the system returned an error from the database:  
The sold\_one\_pretzel command failed with a database execution error.*

*A follow-up attempt to check inventory also failed due to a data validation issue (the system returned 0, but not in the expected integer format).*

*It appears there may be a database connectivity or schema issue (possibly the pretzel table or quantity field).*

*Please check:*

- That the database (mcp) is running*
- The pretzel table exists*
- The quantity column is valid and contains a numeric value*

*Once that's resolved, I can retry the inventory update immediately.*

# MCP Script Error

```
INFO: 52.173.123.14:0 - "GET /mcp HTTP/1.1" 200 OK
INFO: 52.173.123.9:0 - "POST /mcp HTTP/1.1" 202 Accepted
INFO: 52.173.123.7:0 - "POST /mcp HTTP/1.1" 200 OK
ERROR: new row for relation "pretzel" violates check constraint "pretzel_quantity_check"
DETAIL: Failing row contains (-1).
[02/19/26 12:26:21] Error calling tool 'sold_one_pretzel'
```

```
----- Traceback (most recent call last) -----
/var/lib/postgresql/.venvs/fastmcp/lib/python3.13/site-packages/fastmcp/tools/tool_manager.py:160 in call_tool
```

```
/var/lib/postgresql/.venvs/fastmcp/lib/python3.13/site-packages/fastmcp/tools/tool.py:381 in run
```

```
... 2 frames hidden ...
```

```
/usr/lib/python3.13/subprocess.py:472 in check_output
```

```
469 |         |         empty = b''
470 |         |         kwargs['input'] = empty
471 |         |
> 472 |         return run(*popenargs, stdout=PIPE, timeout=timeout, check=True,
473 |                    **kwargs).stdout
474 |         |
475 |         |
```

```
/usr/lib/python3.13/subprocess.py:577 in run
```

```
574 |         |         raise
575 |         |         retcode = process.poll()
576 |         |         if check and retcode:
> 577 |         |         raise CalledProcessError(retcode, process.args,
578 |                                         output=stdout, stderr=stderr)
579 |         |         return CompletedProcess(process.args, retcode, stdout, stderr)
580 |         |
```

```
CalledProcessError: Command 'psql --tuples-only -c 'UPDATE pretzel SET quantity = quantity - 1;' -d mcp' returned non-zero exit status 1.
```

# What's Missing

My simplistic example shows ChatGPT calling the appropriate function based on the function's description. I have not shown:

- MCP authentication
- Parameters to MCP functions
- Dynamic SQL with necessary security restrictions

<https://cardinalops.com/blog/mcp-defaults-hidden-dangers-of-remote-deployment/>

<https://www.pgedge.com/blog/lessons-learned-writing-an-mcp-server-for-postgresql>

<https://www.pulsemcp.com/servers?q=postgres>

# Conclusion



<https://momjian.us/presentations>

<https://www.flickr.com/photos/ndrwfgg/>