Building a Knowledge Graph Assistant: Combining Tavily and Neo4j MCP Servers with Claude

D Dustin Pulver Follow 7 min read · Jan 29, 2025

Imagine having an AI assistant that not only searches the web in real-time but also automatically organizes the information into a interactive knowledge graph. By combining the Tavily MCP server for web search with the Neo4j MCP server for graph database operations, we can create a powerful system where Claude functions similarly to a ReACT (Reason-Act) agent with structured constraints, intelligently processing information and creating visual knowledge representations through a defined set of tools and actions.

What is Model Context Protocol (MCP)?

The Model Context Protocol, developed by Anthropic, is an open standard that enables AI systems to interact seamlessly with various data sources and

tools. Think of it as a universal translator that allows AI models like Claude to communicate with external services securely and effectively.

What is Tavily

Tavily is a search engine, specifically designed for AI agents and tailored for RAG purposes. Through the Tavily Search API, AI developers can effortlessly integrate their applications with realtime online information. Tavily's primary mission is to provide factual and reliable information from trusted sources, enhancing the accuracy and reliability of AI generated content and reasoning.

What is Neo4j

Neo4j is a leading graph database platform designed to store, manage, and analyze interconnected data. By representing information as nodes and relationships, it excels in applications like knowledge graphs, network analysis, and recommendation systems, enabling efficient exploration of complex data structures.

Understanding the Architecture

Our setup consists of three main components:

1. **Tavily MCP Server:** Provides real-time web search and content extraction capabilities
2. **Neo4j MCP Server:** Handles graph database operations and visualization
3. **Claude as an Agent:** Orchestrates the flow of information between these services and creates interactive visualizations

This architecture allows Claude to:

- Search the web for relevant information using Tavily
- Extract and process the gathered data
- Store and organize information in Neo4j graphs

Setting Up Tavily MCP

The setup process involves two main steps: installing prerequisites and configuring the Claude desktop app. Let's break it down.

Prerequisites

Before we begin, make sure you have:

- A Tavily API key
- Claude Desktop installed
- Git (optional, depending on installation method)
- Neo4j Desktop installed

you will also need Node.js version 20 or higher and uvx to properly execute the MCP servers.

How to install uvx on Windows and macOS

`uvx` is a command provided by the `uv` tool, a fast Python package and project manager, needed to execute the Tavily MCP server. Here's a quick guide to installing `uv` on both Windows and macOS.

Installation on macOS

Using Homebrew — make sure to have Homebrew installed

```
brew install uv
```

Installation on Windows

Using WinGet — make sure to have WinGet installed.

```
winget install -id astral-sh.uv -e
```

How to install Node.js on Windows and macOS

Node.js gives you access to the Node Package Manager (`npm`) and the `npx` command-line tool, needed to execute the Neo4j MCP server.

Installation on macOS

Using Homebrew — make sure to have Homebrew installed

```
brew install nodejs
```

Installation on Windows

Using WinGet — make sure to have WinGet installed.

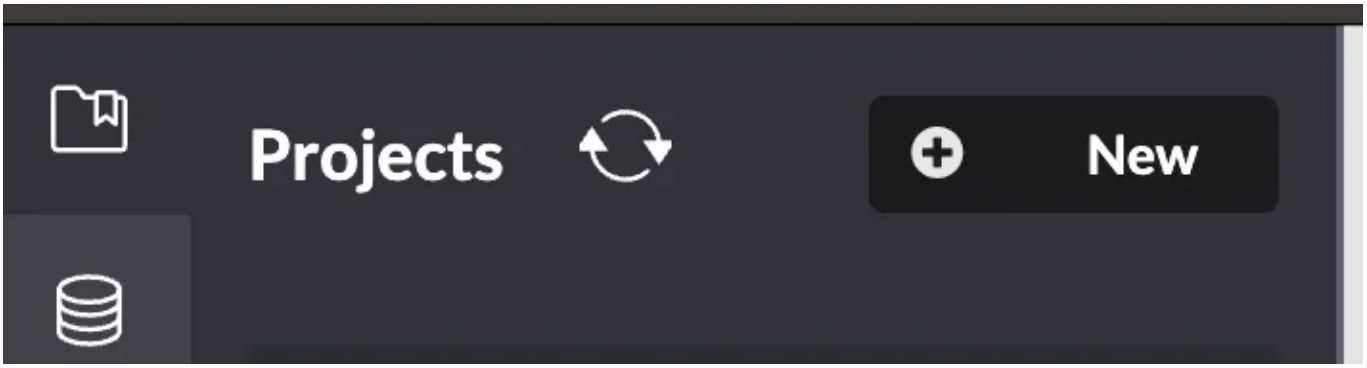
```
winget install -id OpenJS.NodeJS -e
```

Neo4j Desktop Setup

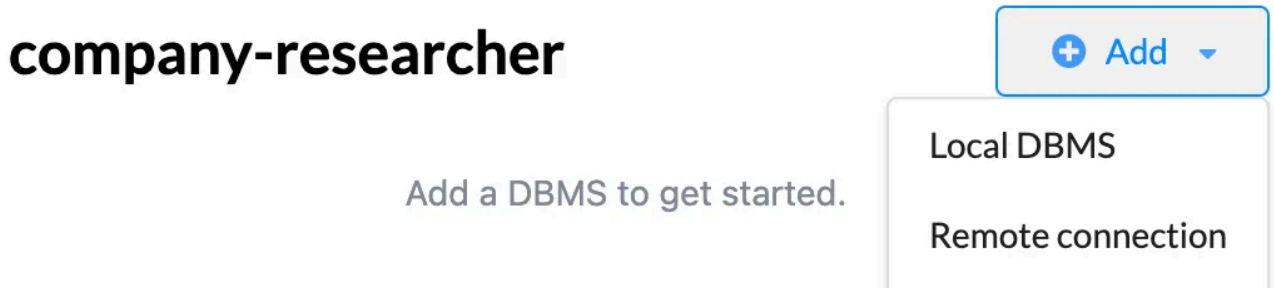
If you don't already have Neo4j desktop installed you can download it here

Once installed follow these steps to create an empty graph for claude to interact with

1. click the `new` button in the top left of the screen and select `Create project`



2. Give your project a name, click the `add` button and select `Loacl DBMS`

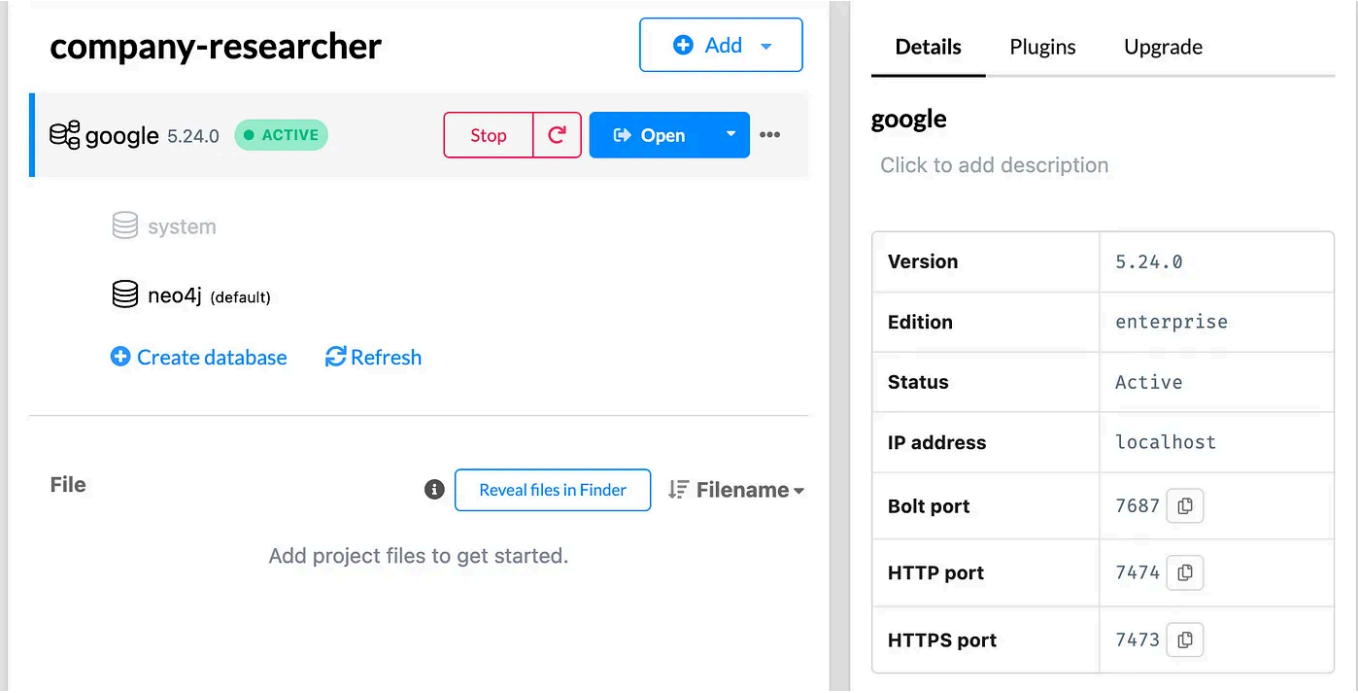


3. Give the graph a name and a password (you will need this in a few steps)

4. Click the `Create` button to create the graph

5. Click the `Start` button and wait for your DBMS to become active

6. Click on the active DBMS to the right of the `Stop` button to see `Details`, take node of the `Bolt port` (you will need this in a few steps)



MCP Quick Setup

(Note: if you run into trouble with this setup you can refer to the tavily-mcp GitHub and the mcp-neo4J GitHub to install locally with git)

Configuring Claude Desktop

Now comes the crucial part: telling Claude Desktop how to communicate with the Tavily MCP server. The process varies slightly depending on your operating system.

For macOS users:

```
touch "$HOME/Library/Application Support/Claude/claude_desktop_config.json"
open -e "$HOME/Library/Application Support/Claude/claude_desktop_config.json"
```

For Windows users:

```
%APPDATA%\Claude\claude_desktop_config.json
```

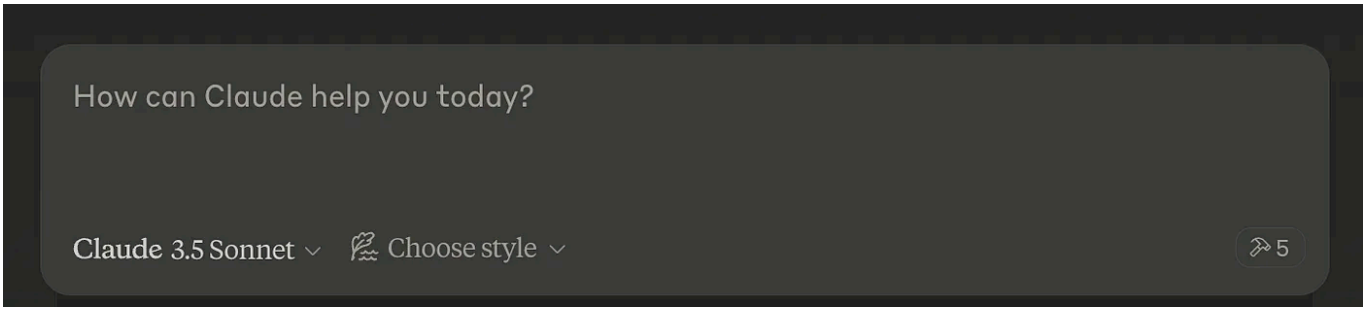
Add the following configuration:

- replace `your-api-key-here` with your actual Tavily API key
- If you don't have an API key, you can sign up for a free account here
- replace `port` with the the `Bolt port` from step 6 of the Neo4j Desktop Setup section
- replace `your-password` with the password you set in step 3 of the Neo4j Desktop Setup section

```
{
  "mcpServers": {
    "tavily-mcp": {
      "command": "npx",
      "args": ["-y", "tavily-mcp@0.1.2"],
      "env": {
        "TAVILY_API_KEY": "your-api-key-here"
      }
    },
    "neo4J": {
      "command": "uvx",
      "args": ["mcp-neo4j-cypher",
               "--db-url", "bolt://localhost:port",
               "--password","your-password"]
    }
  }
}
```

Using Tavily MCP with Claude

After configuring the `claude_desktop_config.json`, **restart the Claude desktop app.** Once you reopen the desktop app you'll notice a new **hammer icon** in the bottom left corner, indicating that MCP tools are available.



Here's where the magic begins!

Demo

In this demo, we will showcase Claude functioning like a ReACT agent powered by Tavily, conducting in-depth research on Google executives and leveraging Neo4j to visually present the research findings. Claude responses to a single prompt executing a 6-step process in one go with no back-and-forth prompting needed.

We ask Claude to execute the following;

1. Find the top 4 C-Level executives at Google
2. Run a search for each one of the executives to find their LinkedIn urls, repeat until all executives have a Linkedin url
3. Create a table of each executive and their LinkedIn URL
4. tavily-extract all their LinkedIn URLs

5. With the results from tavily-extract in part 4 generate a Neo4j graph with all of the executives found.

Use the following:

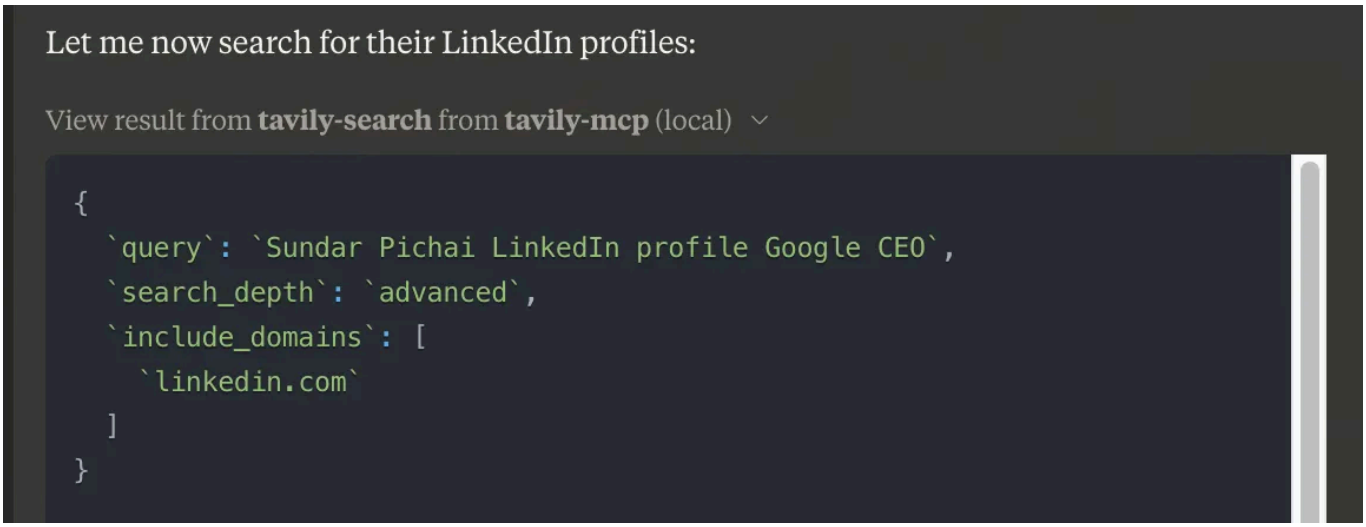
- objects: [person, university, company]
- relationships : [works at, previously worked at, studied at]

6. write a detailed report on your findings about the team, and mention the dominant university

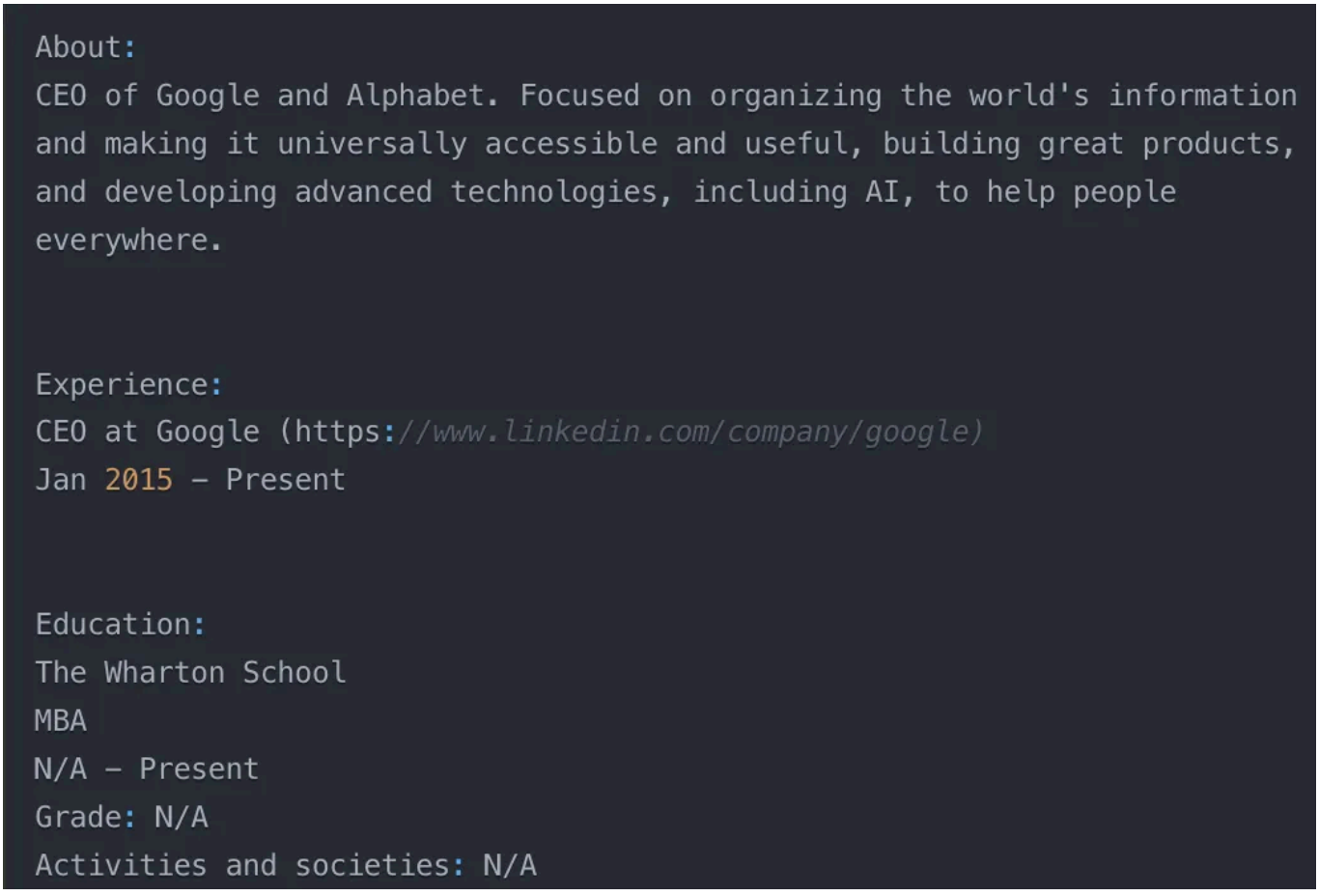
Snippets from the demo :

Claude autonomously created a Tavily Search query, strategically setting parameters to optimize results. To retrieve LinkedIn profiles of Google executives, it configured the query with `include_domains` restricted to

`linkedin.com`, ensuring that all returned results originated exclusively from LinkedIn.



Results returned from tavily-search as seem by Claude:



Given the results from Tavily Search and Extract Claude autonomously creates a Neo4j cypher query that creates relationships in the Neo4j graph connecting Thomas Kurian to Google (WORKS_AT), Oracle and McKinsey & Company (PREVIOUSLY_WORKED_AT), and Stanford and Princeton Universities (STUDIED_AT).

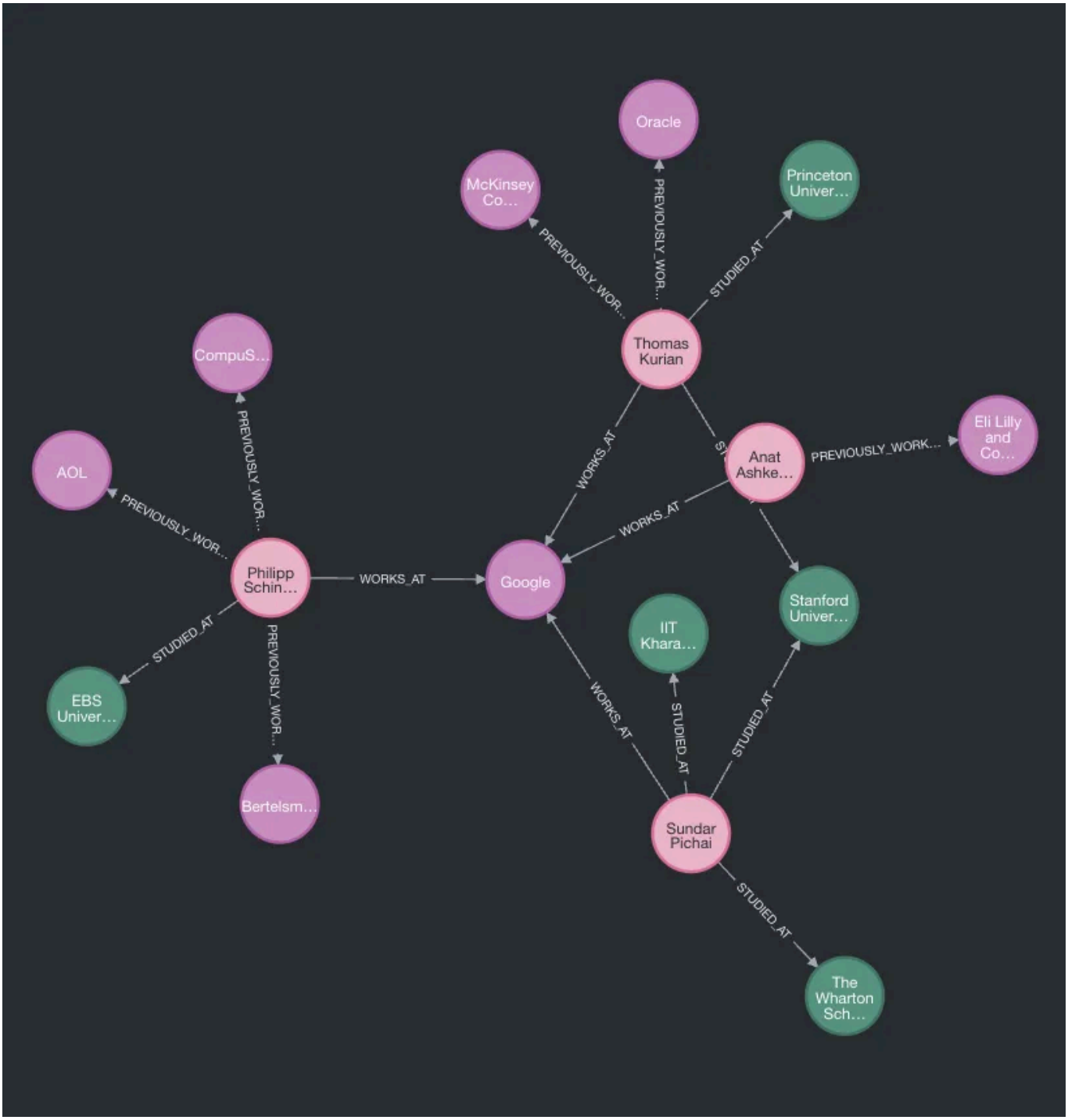
Claude generated Neo4j cypher query:

View result from **write-neo4j-cypher**

```
{
  `query`: `
MATCH (thomas:Person {name: 'Thomas Kurian'}),
      (google:Company {name: 'Google'}),
      (oracle:Company {name: 'Oracle'}),
      (mckinsey:Company {name: 'McKinsey & Company'}),
      (stanford:University {name: 'Stanford University'}),
      (princeton:University {name: 'Princeton University'})
CREATE (thomas)-[:WORKS_AT]->(google)
CREATE (thomas)-[:PREVIOUSLY_WORKED_AT]->(oracle)
CREATE (thomas)-[:PREVIOUSLY_WORKED_AT]->(mckinsey)
CREATE (thomas)-[:STUDIED_AT]->(stanford)
CREATE (thomas)-[:STUDIED_AT]->(princeton)
`
}

[{'_contains_updates': True, 'relationships_created': 5}]
```

The final graph generated as seen in the demo:



Looking Forward

The combination of Claude Desktop and Tavily MCP represents a significant step forward in AI assistance. By combining Neo4j's powerful graph visualization capabilities with Tavily's sophisticated search and extraction abilities, we're seeing a glimpse of how AI assistants will become increasingly capable research and analysis partners.

This integration opens up exciting possibilities for:

- Conducting comprehensive research with real-time web access
- Creating dynamic knowledge graphs from search results
- Visualizing complex relationships between information
- Building interactive research tools
- Automating data collection and organization

The ability to have Claude act like a ReACT agent that can seamlessly coordinate between these tools creates a powerful system for both research and visualization. Whether you're:

- Conducting academic or market research
- Staying up-to-date with industry developments
- Analyzing complex relationships in data
- Creating interactive knowledge bases
- Visualizing information networks

This setup provides a sophisticated toolkit that enhances Claude's already impressive capabilities. As these technologies continue to evolve, we can expect even more powerful integrations and capabilities to emerge.

For Troubleshooting Tips

visit tavily-mcp

Tavily Neo4j Claude

D

Written by Dustin Pulver

13 followers · 4 following

Follow

Responses (1)

Joe Whittle

What are your thoughts?

Neo4j
Jan 30

Hi Dustin, thanks for the great article! If you have any feedback for the MCP neo4j servers, please let us know (or open an GH issue).

2 Reply

More from Dustin Pulver

Dustin Pulver

Connect Your Coding Assistant to the Web: Integrating Tavily MCP with Cline in VS Code

Unleash the full potential of AI-assisted coding with real-time web access

Mar 19

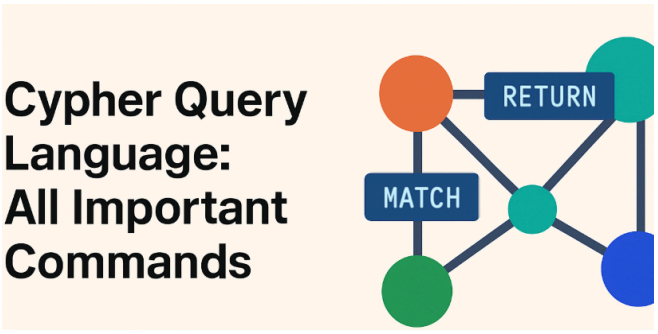
See all from Dustin Pulver

Recommended from Medium

Dr. Manoj Kumar Yadav

Ontology and Knowledge Representation in the Context of ...

Ontology as a concept is rarely discussed in the context of Large Language Models...



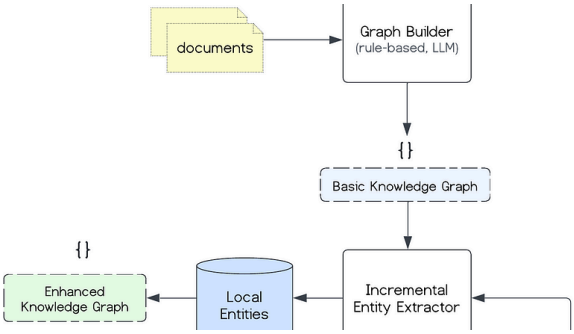
Melike Demirdağ

Mastering Cypher: A Deep Dive into the Query Language for Graph...

From social networks to recommendation engines, graphs are everywhere. Let's explor...

Jan 9 1

May 19 5



Kamlesh Singh

Server-Sent Events (SSE) vs. WebSockets: Why SSE Might Be...

Server-Sent Events is a server push technology enabling a client to receive...

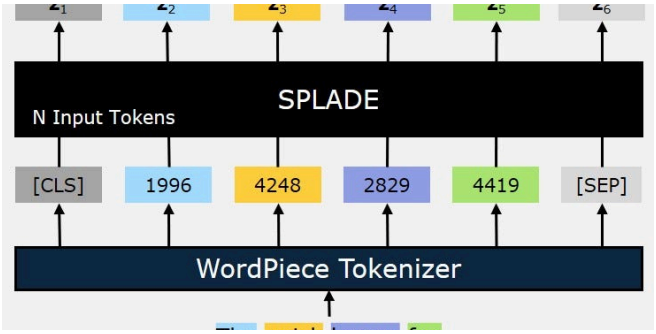
Feb 12

Ngoc

Part 2A: Implementing a Graph Builder to Extract a Basic...

In this post, I'll show you how to construct a basic document-centric knowledge graph...

Jan 22 80



Micheal Lanham

Orchestrating AI Agents with Behavior Trees: The Missing Piec...

Orchestrating AI Agents with Behavior Trees: The Missing Piece in Agent Architectures

Mar 15 65

In AI Advances by Dr. Leon Eversberg

How to Use SPLADE for Better LLM RAG Retrieval

Building an advanced RAG pipeline with sparse encoder retrieval models

5d ago 539 3