Microsoft

Leveraging AI capabilities in your modern data workload

What problems are we trying to solve with AI?

✓*Smarter searching* on your existing data

✓Bring in other documents/text for *centralized vector searching*

✓Provide *building blocks for* Intelligent assistants, RAG, AI Agents

✓Take advantage of AI in a *secure and scalable* fashion

✓*Overcome complexity* by using the familiar SQL language

Security

Familiarity

Reliability

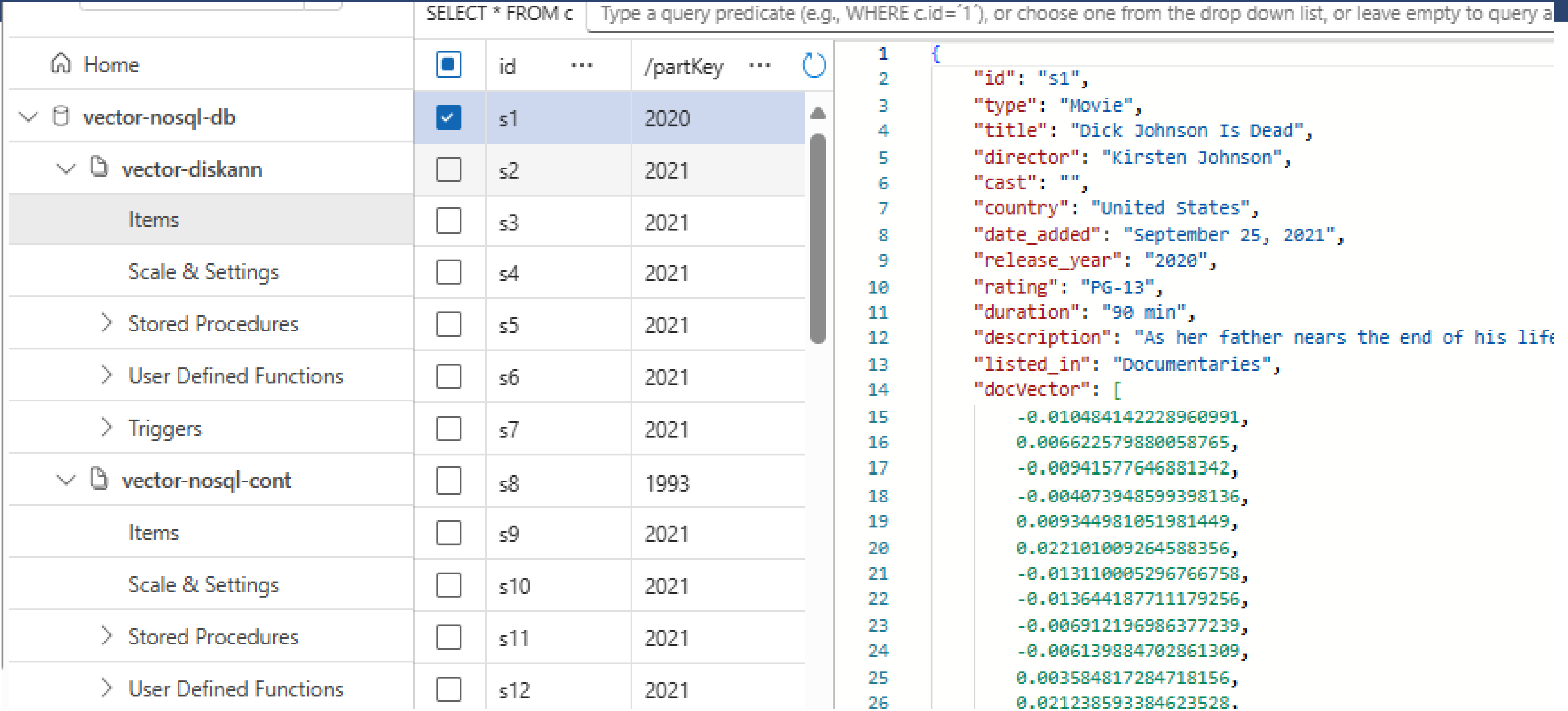
Integrated

Scalability

Microsoft

Vector Search in Azure Cosmos DB

Flexible schema



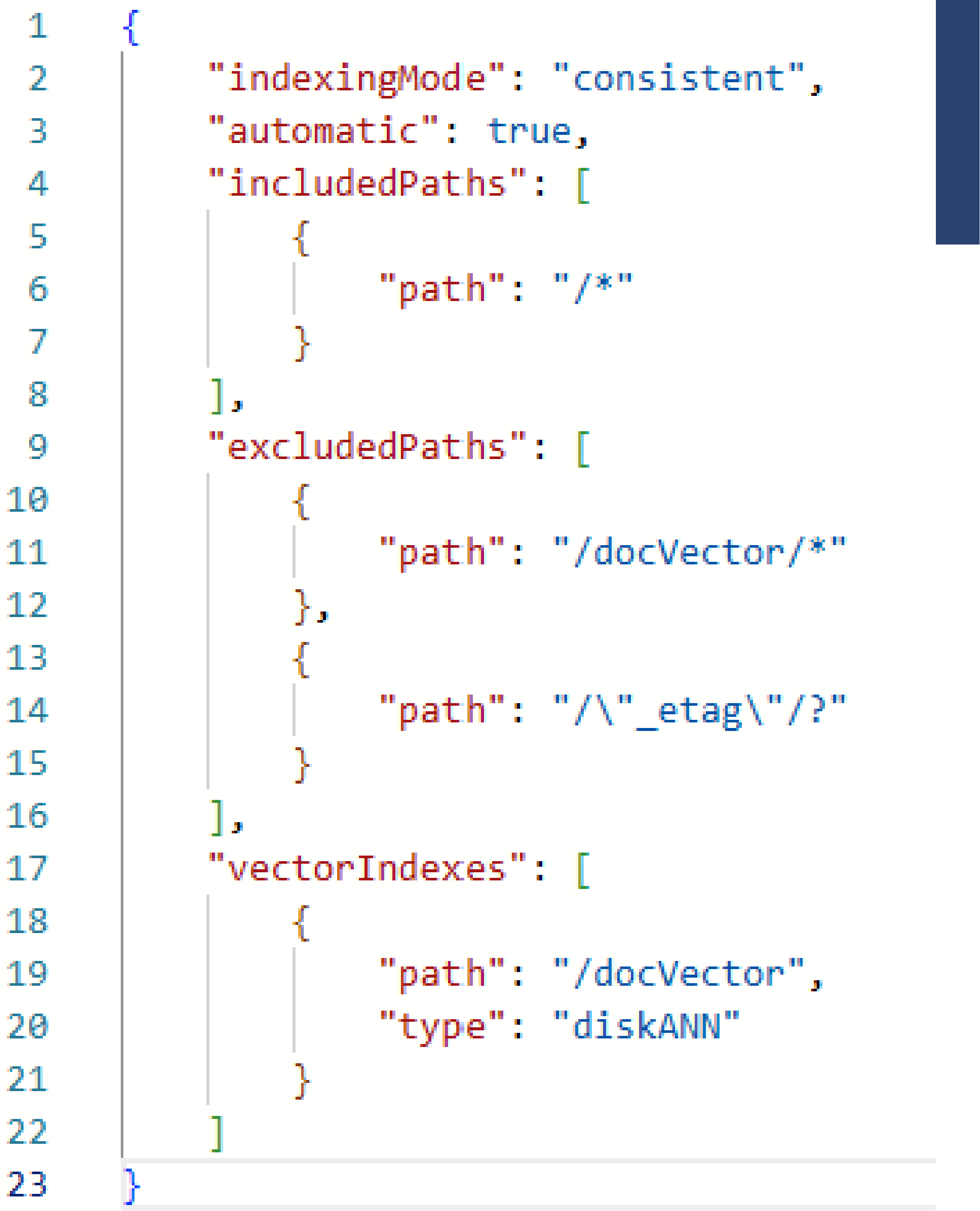
SELECT * FROM c Type a query predicate (e.g., WHERE c.id=´1´), or choose one from the drop down list, or leave empty to query a

■	id	/partKey
☑	s1	2020
□	s2	2021
□	s3	2021
□	s4	2021
□	s5	2021
□	s6	2021
□	s7	2021
□	s8	1993
□	s9	2021
□	s10	2021
□	s11	2021
□	s12	2021

```
{
    "id": "s1",
    "type": "Movie",
    "title": "Dick Johnson Is Dead",
    "director": "Kirsten Johnson",
    "cast": "",
    "country": "United States",
    "date_added": "September 25, 2021",
    "release_year": "2020",
    "rating": "PG-13",
    "duration": "90 min",
    "description": "As her father nears the end of his life
    "listed_in": "Documentaries",
    "docVector": [
        -0.010484142228960991,
        0.006622579880058765,
        -0.00941577646881342,
        -0.004073948599398136,
        0.009344981051981449,
        0.022101009264588356,
        -0.013110005296766758,
        -0.013644187711179256,
        -0.006912196986377239,
        -0.006139884702861309,
        0.003584817284718156,
        0.021238593384623528,
```

Indexing

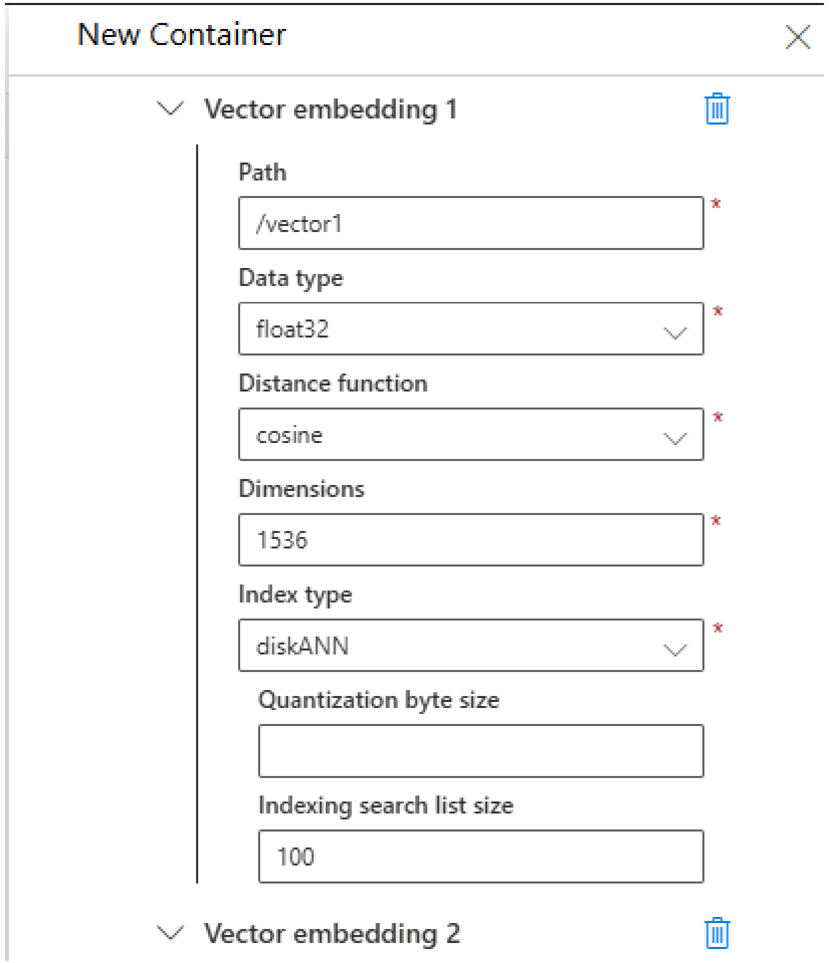
- Create a DiskANN **index on the document embedding path**
- **Add the vector path to the “excludedPaths”** section of the indexing policy (to avoid indexing as a regular array path)
- Skip vector indexing if type and dimensions differ from container vector policy
- **DiskANN requires at least 1000 documents**, otherwise will search all documents
- **Best practice**: exclude everything and selective add properties to be indexed



```
{
    "indexingMode": "consistent",
    "automatic": true,
    "includedPaths": [
        {
            "path": "/*"
        }
    ],
    "excludedPaths": [
        {
            "path": "/docVector/*"
        },
        {
            "path": "/\"_etag\"/?"
        }
    ],
    "vectorIndexes": [
        {
            "path": "/docVector",
            "type": "diskANN"
        }
    ]
}
```

Indexing

- More than one vector policy/index is supported



New Container

Vector embedding 1

Path *
/vector1

Data type *
float32

Distance function *
cosine

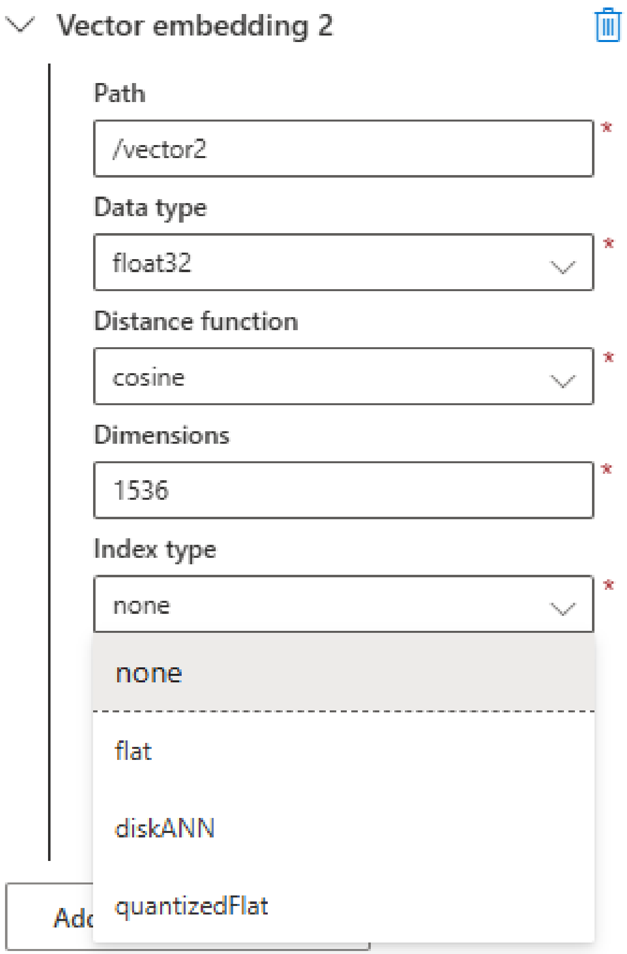
Dimensions *
1536

Index type *
diskANN

Quantization byte size

Indexing search list size
100

Vector embedding 2



Vector embedding 2

Path *
/vector2

Data type *
float32

Distance function *
cosine

Dimensions *
1536

Index type *
none

Ingestion

- https://devblogs.microsoft.com/cosmosdb/azure-cosmos-db-vector-search-with-diskann-part-1-full-space-search/

- 20 RUs for a small document with a 768-dimensional vector
- 30 RUs for a small document with a 1,536-dimensional vector
- 50 RUs for a small document with a 3,072-dimensional vector
- Document size and regular indexing policy will add to the base cost

Latency

Scenario	P50 Latency (ms)	P95 Latency (ms)	Avg Latency (ms)	Recall@k
	k=10/50	k=10/50	k=10/50	k=10/50
100k vectors	25/95	29/108	25/95	92.47/98.03
1M vectors	34/130	51/164	39/133	89.2/95.41
35M vectors	108/544	166/879	112/569	90.73/95.67

Query cost (RU)

Scenario	P50 RU Cost	P95 RU Cost	Avg RU Cost
	k=10/50	k=10/50	k=10/50
100k vectors	36/159	39/169	36/159
1M vectors	38/170	42/179	45/182
35M vectors	282/1,249	300/1,298	282/1,245

Cost estimation by provision mode/workload

Monthly cost (in USD) for Azure Cosmos DB for NoSQL in the East US 2 region

Scenario	Manual (100% 10 QPS)	Manual (100% 100 QPS)	Autoscale (50% 1 QPS, 50% 10 QPS)	Autoscale (100% 10QPS)	Autoscale (50% 10 QPS, 50% 100 QPS)	Autoscale (100% 100 QPS)
100k vectors	$22.46	$224.64	$18.53	$33.70	$185.33	$336.96
1M vectors	$24.19	$241.92	$19.96	$36.29	$199.58	$362.88
35M vectors	$172.80	$1,728.00	$142.56	$259.20	$1,425.60	$2,592.00

Vector Search in Azure PostgreSQL

Microsoft

Vector Search in Azure SQL

Finding the Best Focaccia Bread in town

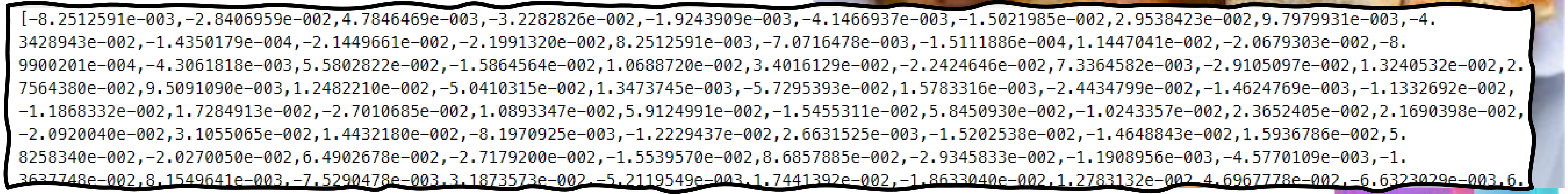
How AI and SQL can help you?

Focaccia is “similar” to

- Pizza
- Flatbread
- Bread

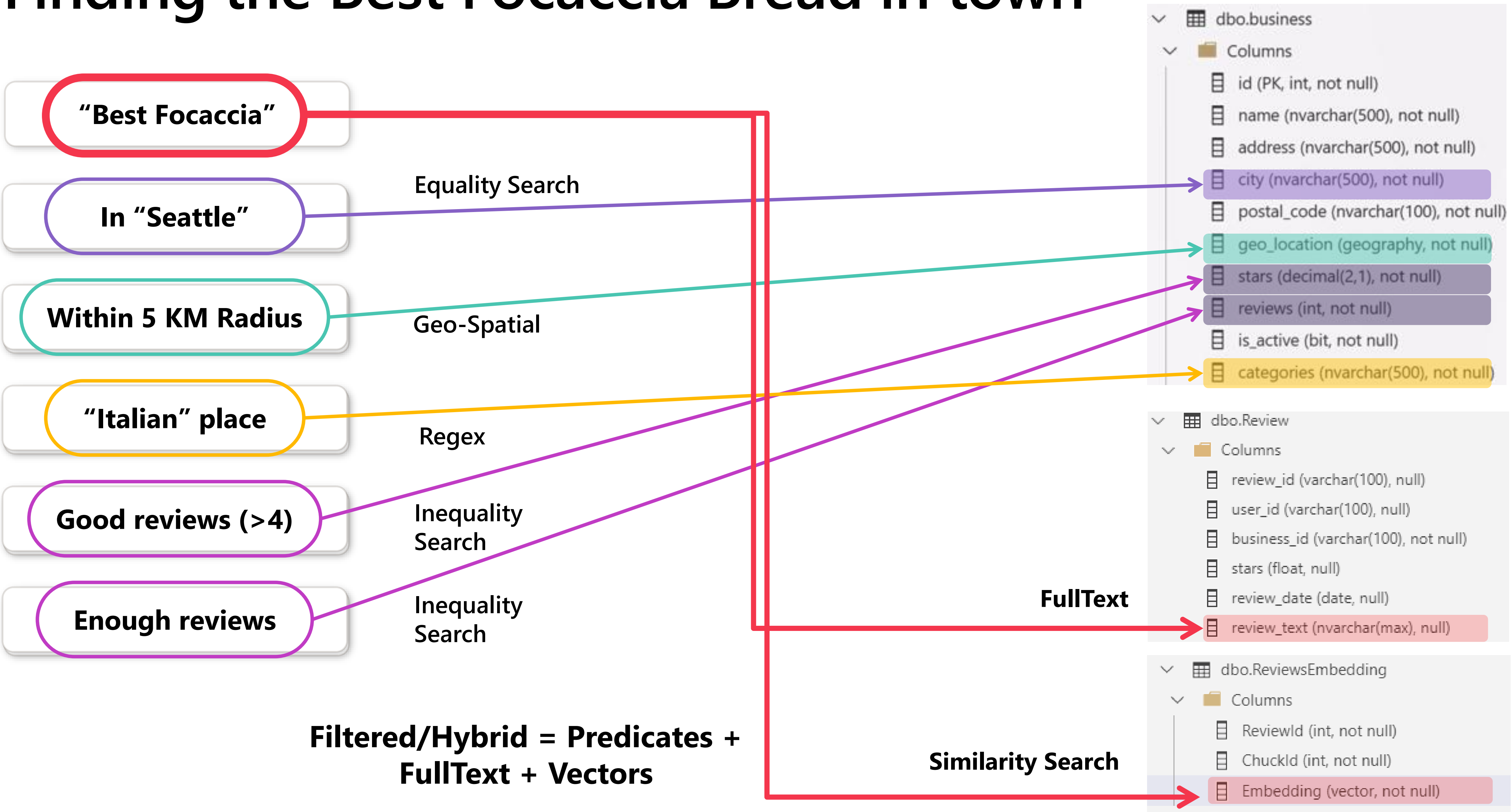
What about misspellings?

Focaccia’s embedding:



```
[-8.2512591e-003,-2.8406959e-002,4.7846469e-003,-3.2282826e-002,-1.9243909e-003,-4.1466937e-003,-1.5021985e-002,2.9538423e-002,9.7979931e-003,-4.
3428943e-002,-1.4350179e-004,-2.1449661e-002,-2.1991320e-002,8.2512591e-003,-7.0716478e-003,-1.5111886e-004,1.1447041e-002,-2.0679303e-002,-8.
9900201e-004,-4.3061818e-003,5.5802822e-002,-1.5864564e-002,1.0688720e-002,3.4016129e-002,-2.2424646e-002,7.3364582e-003,-2.9105097e-002,1.3240532e-002,2.
7564380e-002,9.5091090e-003,1.2482210e-002,-5.0410315e-002,1.3473745e-003,-5.7295393e-002,1.5783316e-003,-2.4434799e-002,-1.4624769e-003,-1.1332692e-002,
-1.1868332e-002,1.7284913e-002,-2.7010685e-002,1.0893347e-002,5.9124991e-002,-1.5455311e-002,5.8450930e-002,-1.0243357e-002,2.3652405e-002,2.1690398e-002,
-2.0920040e-002,3.1055065e-002,1.4432180e-002,-8.1970925e-003,-1.2229437e-002,2.6631525e-003,-1.5202538e-002,-1.4648843e-002,1.5936786e-002,5.
8258340e-002,-2.0270050e-002,6.4902678e-002,-2.7179200e-002,-1.5539570e-002,8.6857885e-002,-2.9345833e-002,-1.1908956e-003,-4.5770109e-003,-1.
3637748e-002,8.1549641e-003,-7.5290478e-003,3.1873573e-002,-5.2119549e-003,1.7441392e-002,-1.8633040e-002,1.2783132e-002,4.6967778e-002,-6.6323029e-003,6.
```

Finding the Best Focaccia Bread in town



“Best Focaccia”

In “Seattle”

Within 5 KM Radius

“Italian” place

Good reviews (>4)

Enough reviews

Equality Search

Geo-Spatial

Regex

Inequality Search

Inequality Search

FullText

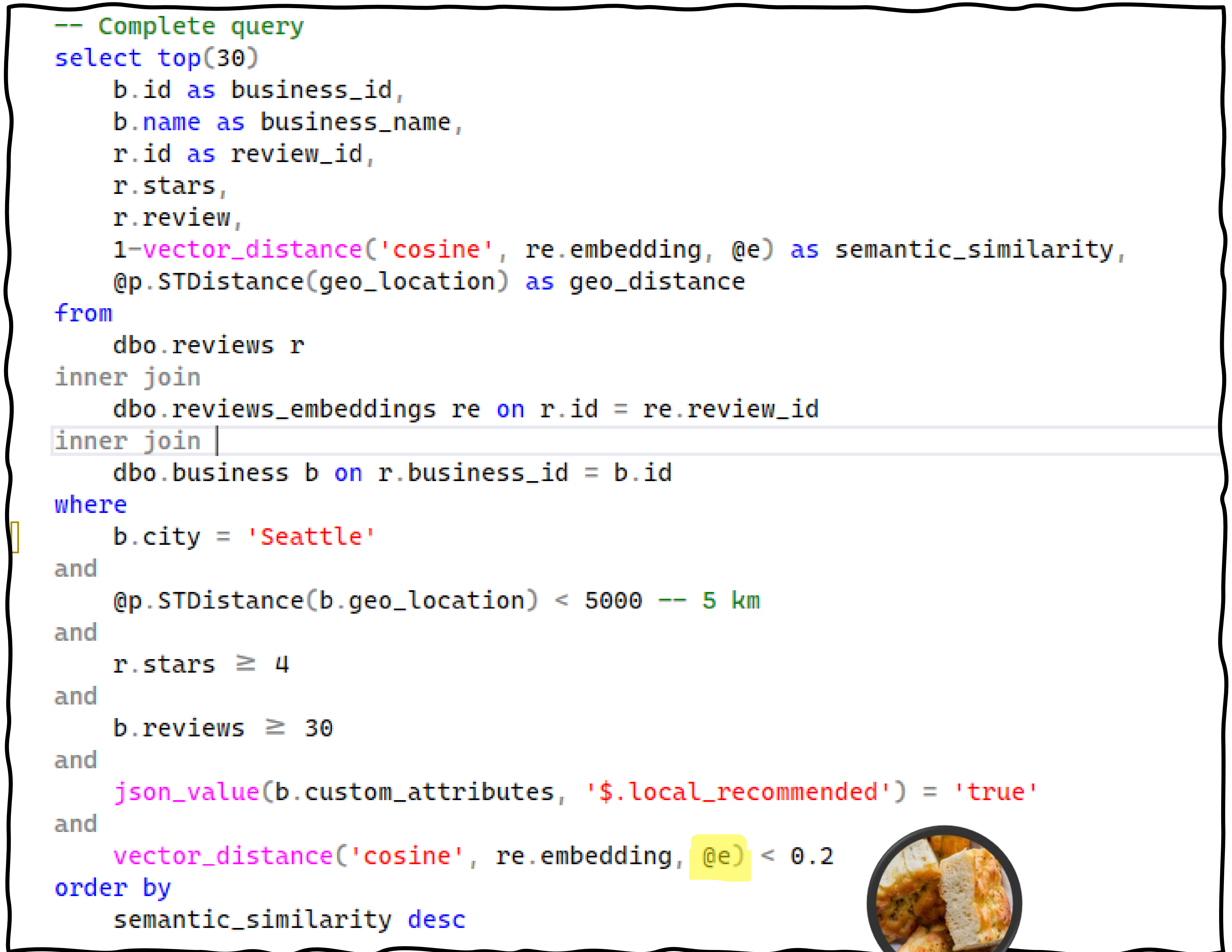
Similarity Search

dbo.business
- Columns
  - id (PK, int, not null)
  - name (nvarchar(500), not null)
  - address (nvarchar(500), not null)
  - city (nvarchar(500), not null)
  - postal_code (nvarchar(100), not null)
  - geo_location (geography, not null)
  - stars (decimal(2,1), not null)
  - reviews (int, not null)
  - is_active (bit, not null)
  - categories (nvarchar(500), not null)

dbo.Review
- Columns
  - review_id (varchar(100), null)
  - user_id (varchar(100), null)
  - business_id (varchar(100), not null)
  - stars (float, null)
  - review_date (date, null)
  - review_text (nvarchar(max), null)

dbo.ReviewsEmbedding
- Columns
  - ReviewId (int, not null)
  - ChuckId (int, not null)
  - Embedding (vector, not null)

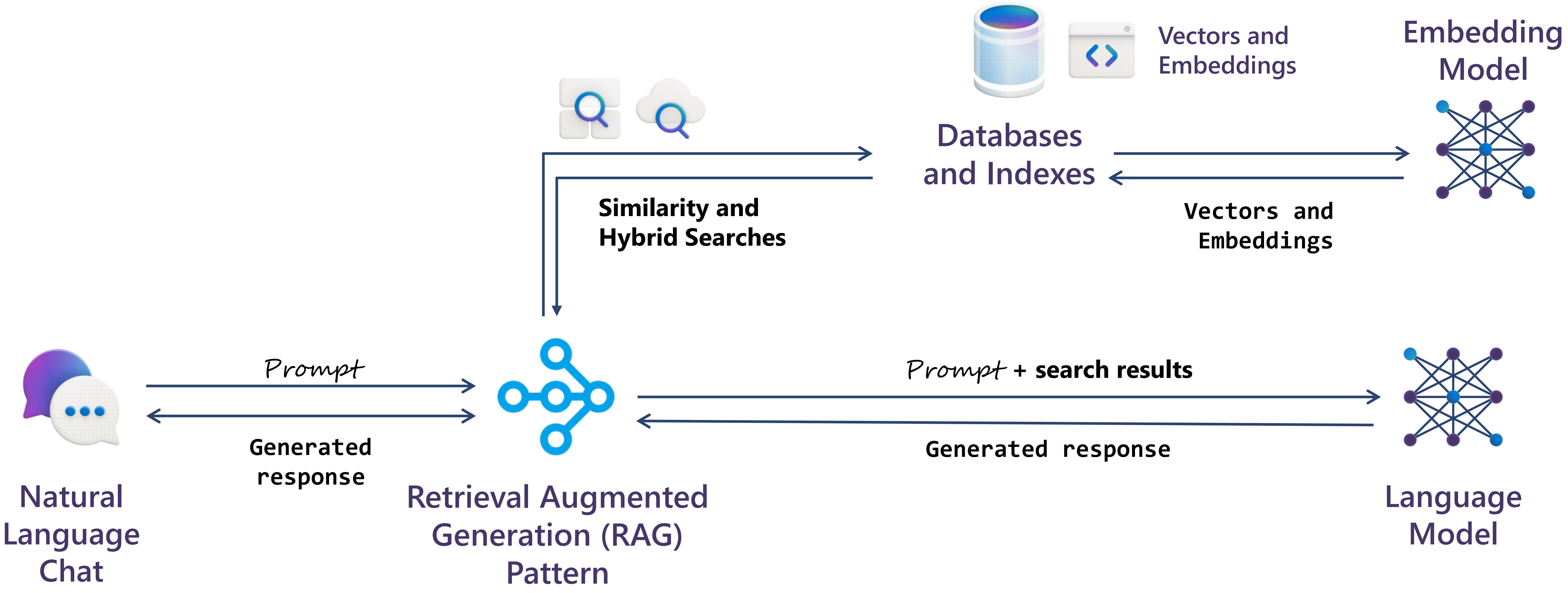
Filtered/Hybrid = Predicates + FullText + Vectors



```
-- Complete query
select top(30)
    b.id as business_id,
    b.name as business_name,
    r.id as review_id,
    r.stars,
    r.review,
    1-vector_distance('cosine', re.embedding, @e) as semantic_similarity,
    @p.STDistance(geo_location) as geo_distance
from
    dbo.reviews r
inner join
    dbo.reviews_embeddings re on r.id = re.review_id
inner join
    dbo.business b on r.business_id = b.id
where
    b.city = 'Seattle'
and
    @p.STDistance(b.geo_location) < 5000 -- 5 km
and
    r.stars ≥ 4
and
    b.reviews ≥ 30
and
    json_value(b.custom_attributes, '$.local_recommended') = 'true'
and
    vector_distance('cosine', re.embedding, @e) < 0.2
order by
    semantic_similarity desc
```

RAG Pattern Explained

Retrieval Augmented Generation



https://learn.microsoft.com/en-us/azure/cosmos-db/gen-ai/rag-chatbot (Cosmos DB)
https://github.com/Azure-Samples/rag-postgres-openai-python (PostgreSQL)
https://github.com/Azure-Samples/azure-sql-db-chatbot (Azure SQL)

RAG Limitations

- RAG is fantastic when it comes to find *similar* content given a somehow approximate request.
- *Is it effective for all scenarios?*
- *What about very precise and direct request?*

  For example: "Show me all the products in category VideoGames updated in the last 3 days"

  Natural Language to SQL is *needed*

  Navigate relationships can lead to better relevance in top results

RAG Limitations

“Show me all the products in category VideoGames updated in the last 3 days”

```
SELECT
     *
FROM
     dbo.products p
INNER JOIN
     dbo.categories c ON p.category_id = c.id
WHERE
     c.Name = ‘videogame’
AND
     p.last_updated >= DATEADD(day, -3, SYSDATETIME())
```

AI Agents and Agentic RAG

Now things are a bit more complex, and the process is more suitable for *agents*

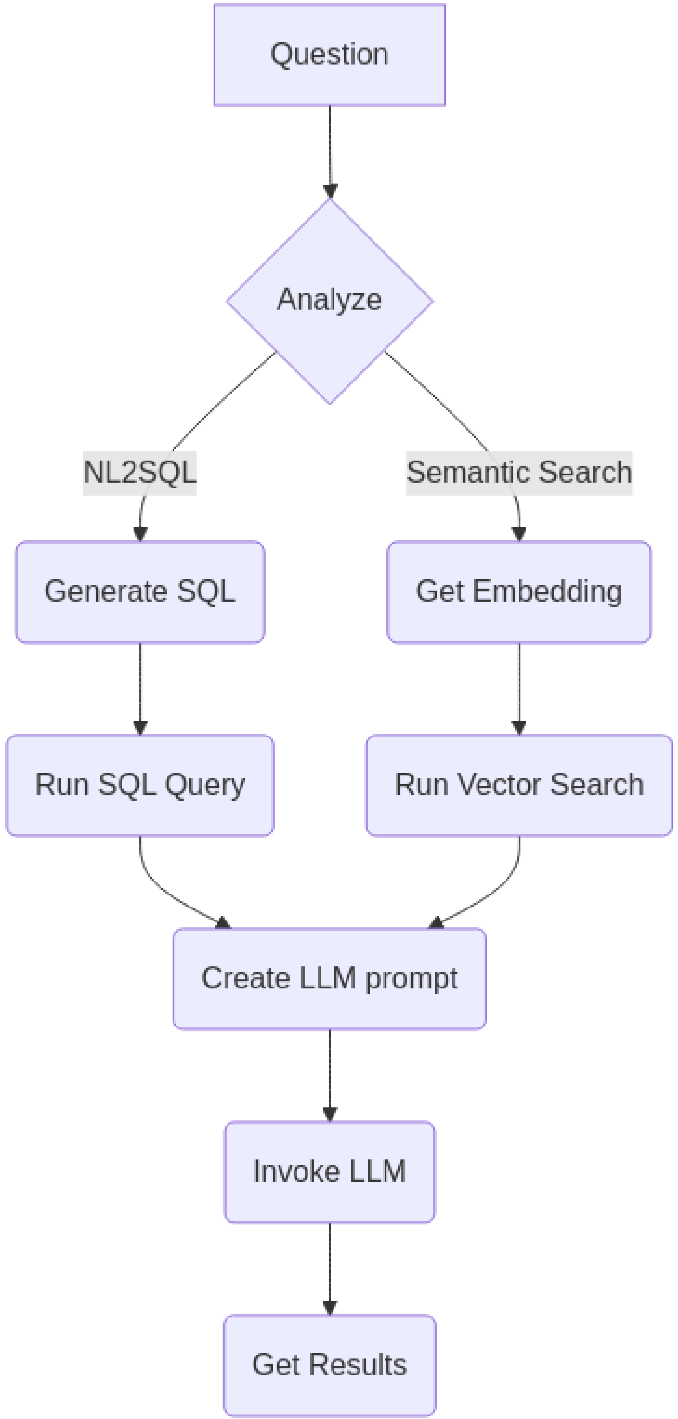
Orchestrator agent: understand if the questions needs NL2SQL or not

If yes, a specialized database agent (who has been informed of the database schema) will

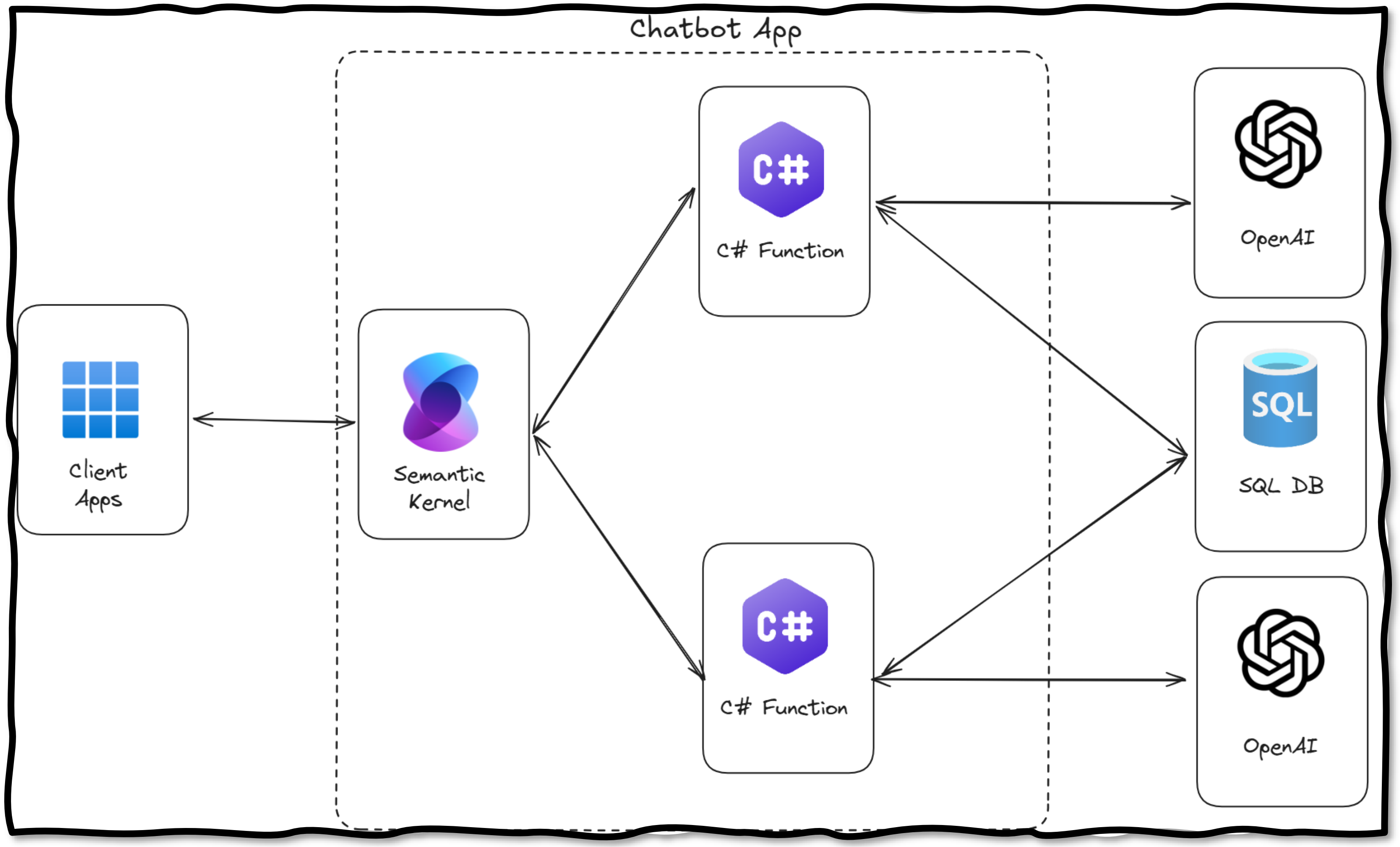
- generate and execute the query
- Or just run the semantic search as usual

Use the output of the previous agent to build the LLM prompt

Invoke LLM and get results



End-to-End Application (Agents with C#)

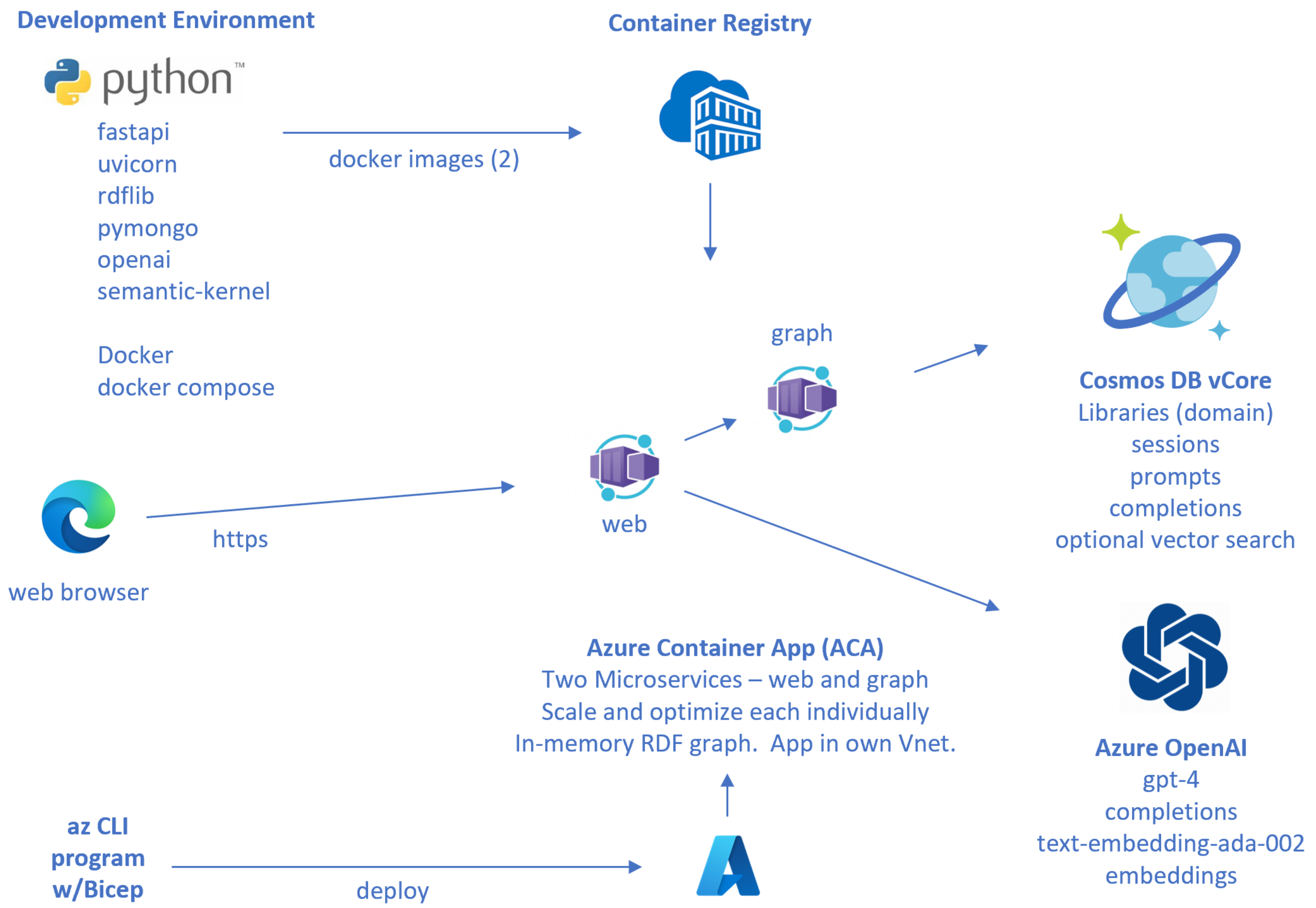


https://github.com/Azure-Samples/azure-sql-db-chat-sk

AI knowledge graphs | Microsoft Learn

https://aka.ms/graphrag (MSR)

CosmosAIGraph Architecture



Development Environment

python

fastapi
uvicorn
rdflib
pymongo
openai
semantic-kernel

Docker
docker compose

docker images (2)

Container Registry

graph

web

https

web browser

Cosmos DB vCore
Libraries (domain)
sessions
prompts
completions
optional vector search

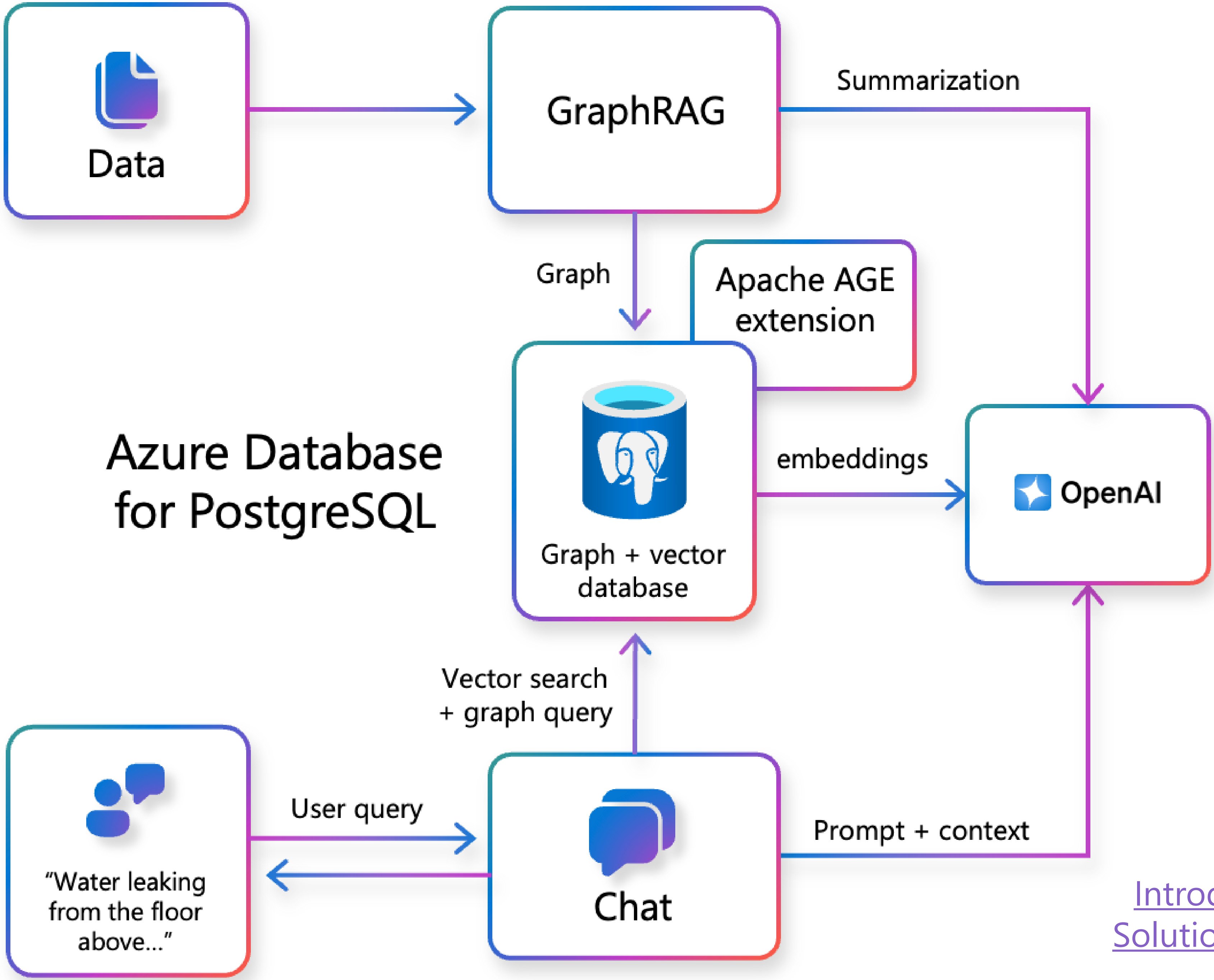
Azure Container App (ACA)
Two Microservices – web and graph
Scale and optimize each individually
In-memory RDF graph. App in own Vnet.

az CLI
program
w/Bicep

deploy

Azure OpenAI
gpt-4
completions
text-embedding-ada-002
embeddings

https://github.com/cjoakim/CosmosAIGraph



Data

GraphRAG

Summarization

Graph

Apache AGE extension

Azure Database for PostgreSQL

Graph + vector database

embeddings

OpenAI

Vector search + graph query

User query

Chat

Prompt + context

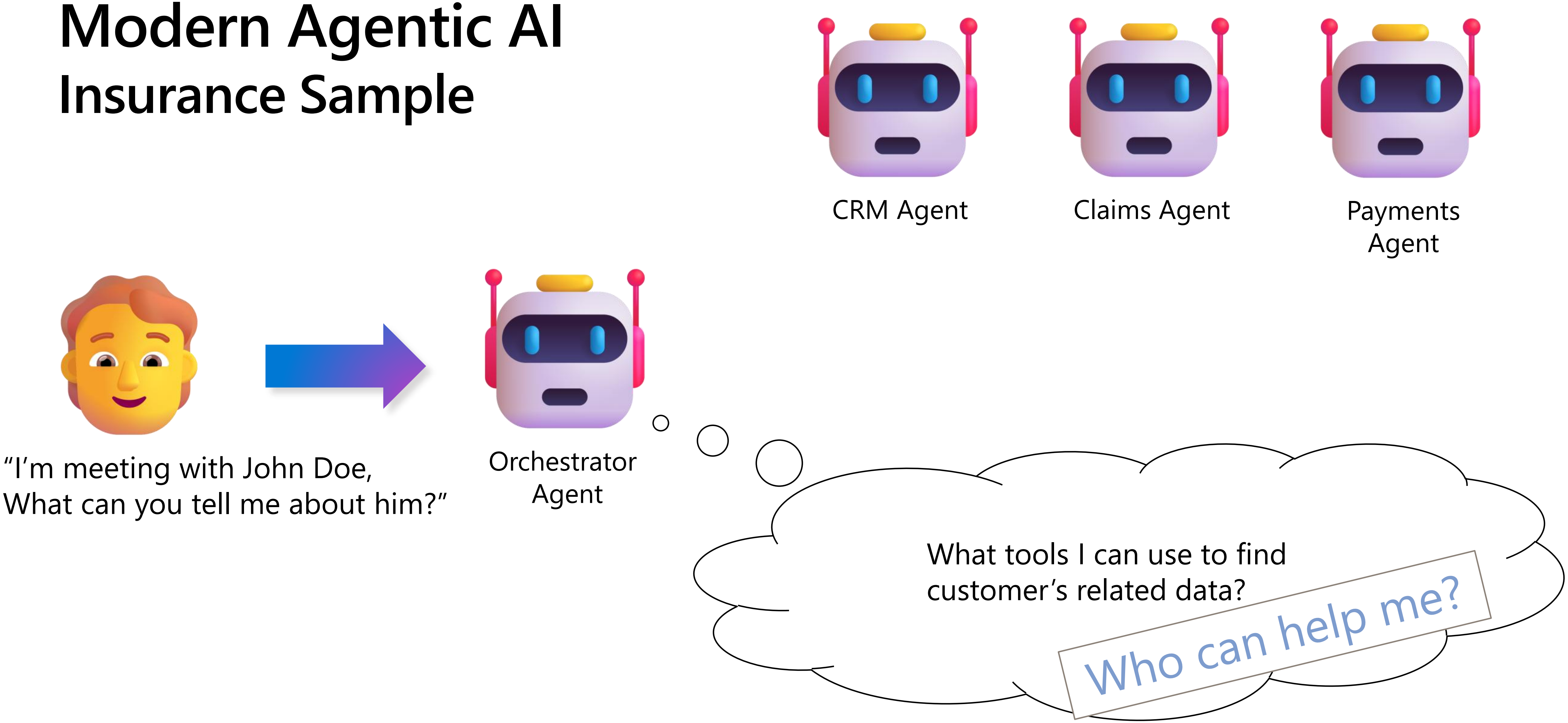
"Water leaking from the floor above..."

Introducing the GraphRAG Solution for Azure Database for PostgreSQL

Microsoft

AI and Agentic Agents

Modern Agentic AI
Insurance Sample



CRM Agent

Claims Agent

Payments Agent

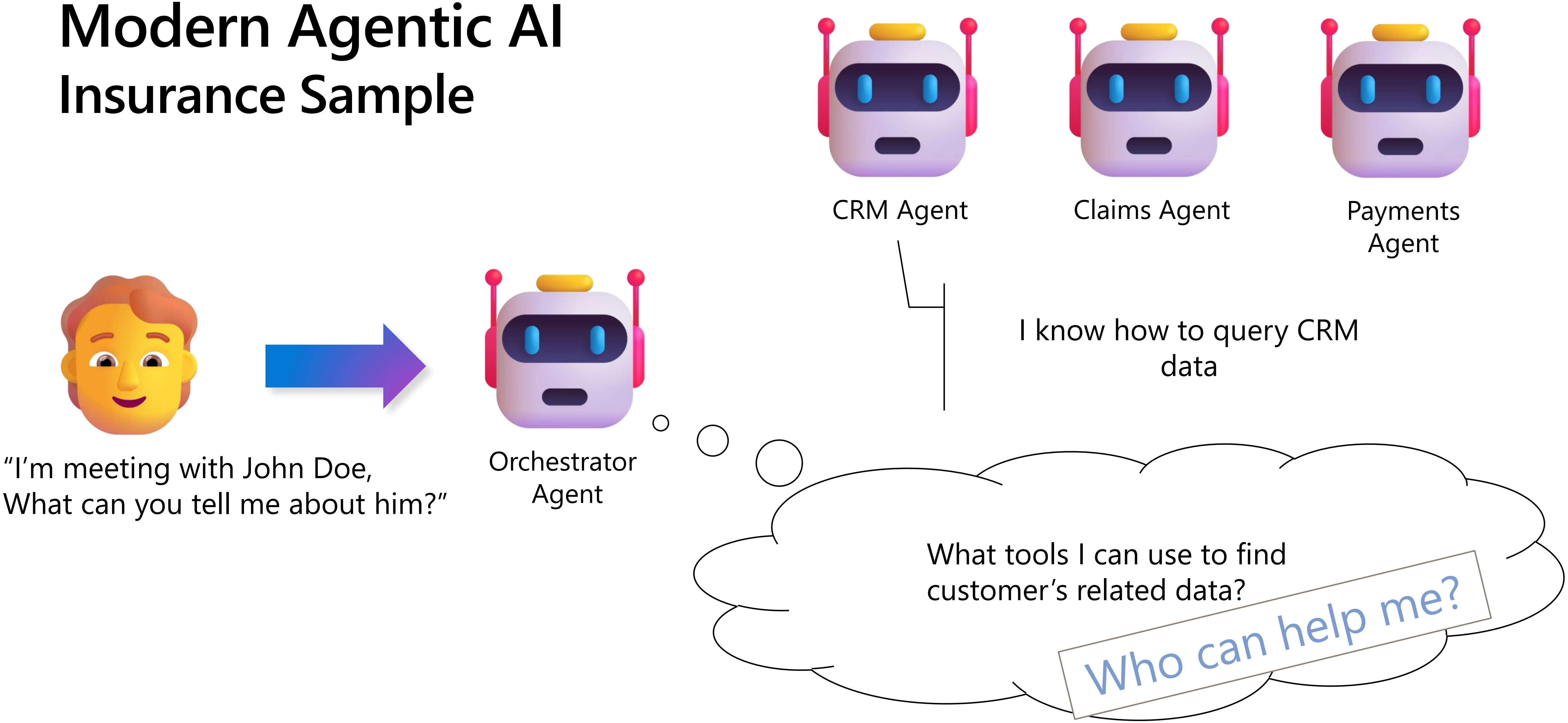
"I'm meeting with John Doe,
What can you tell me about him?"

Orchestrator Agent

What tools I can use to find customer's related data?

Who can help me?

Modern Agentic AI
Insurance Sample



CRM Agent

Claims Agent

Payments Agent

I know how to query CRM data

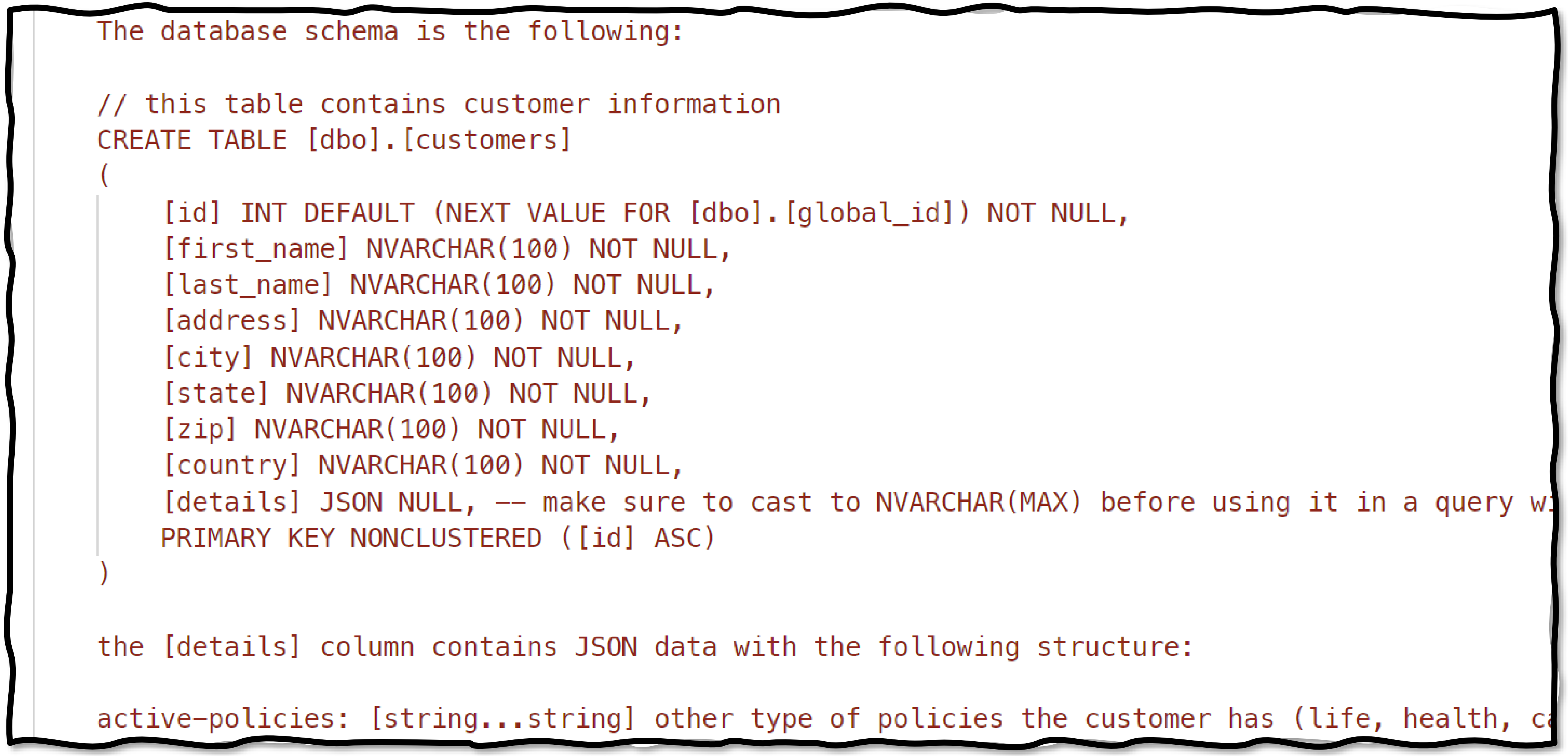
“I’m meeting with John Doe,
What can you tell me about him?”

Orchestrator Agent

What tools I can use to find customer’s related data?

Who can help me?

Specialized Agent has an “expert-level” view of the data



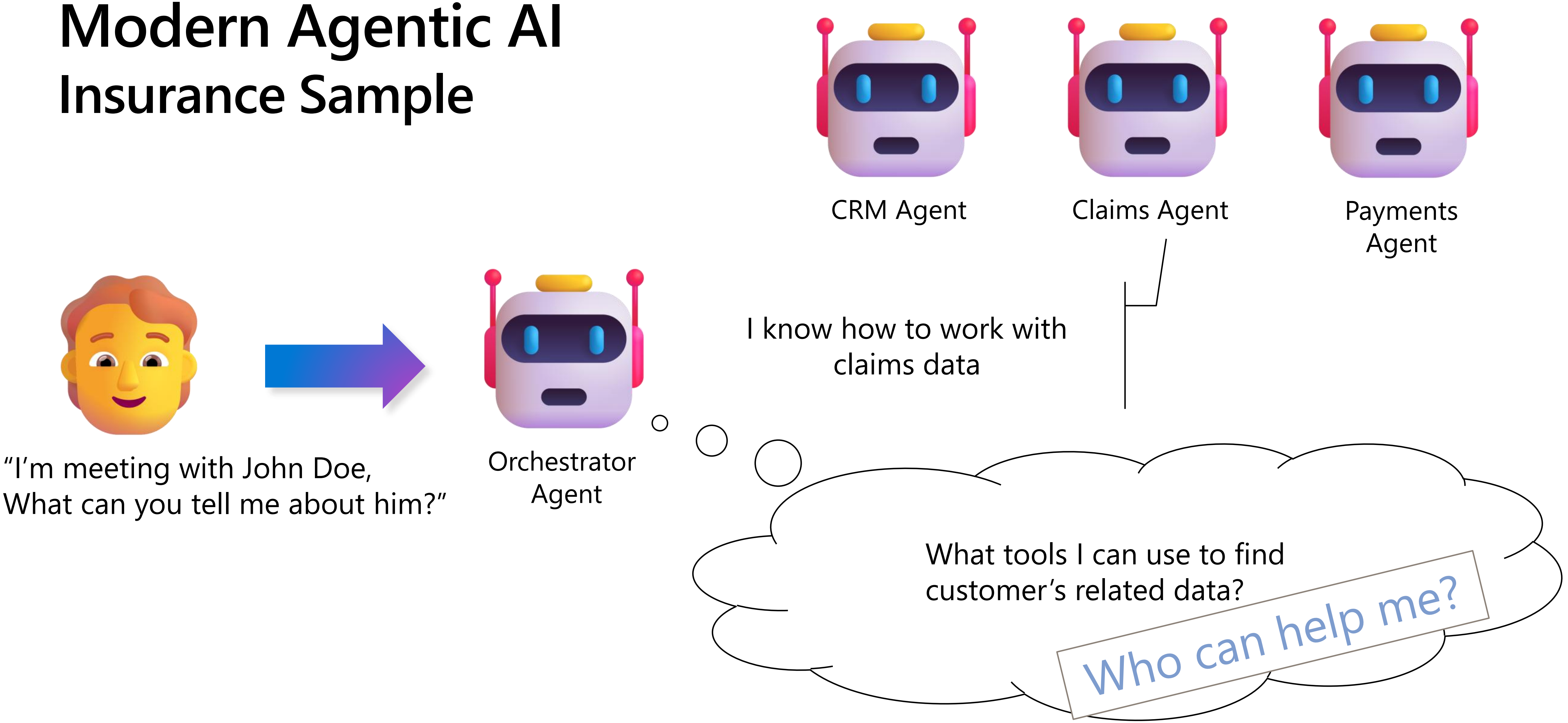
```
The database schema is the following:

// this table contains customer information
CREATE TABLE [dbo].[customers]
(
    [id] INT DEFAULT (NEXT VALUE FOR [dbo].[global_id]) NOT NULL,
    [first_name] NVARCHAR(100) NOT NULL,
    [last_name] NVARCHAR(100) NOT NULL,
    [address] NVARCHAR(100) NOT NULL,
    [city] NVARCHAR(100) NOT NULL,
    [state] NVARCHAR(100) NOT NULL,
    [zip] NVARCHAR(100) NOT NULL,
    [country] NVARCHAR(100) NOT NULL,
    [details] JSON NULL, -- make sure to cast to NVARCHAR(MAX) before using it in a query wi
    PRIMARY KEY NONCLUSTERED ([id] ASC)
)

the [details] column contains JSON data with the following structure:

active-policies: [string...string] other type of policies the customer has (life, health, ca
```

Modern Agentic AI
Insurance Sample



CRM Agent

Claims Agent

Payments Agent

I know how to work with claims data

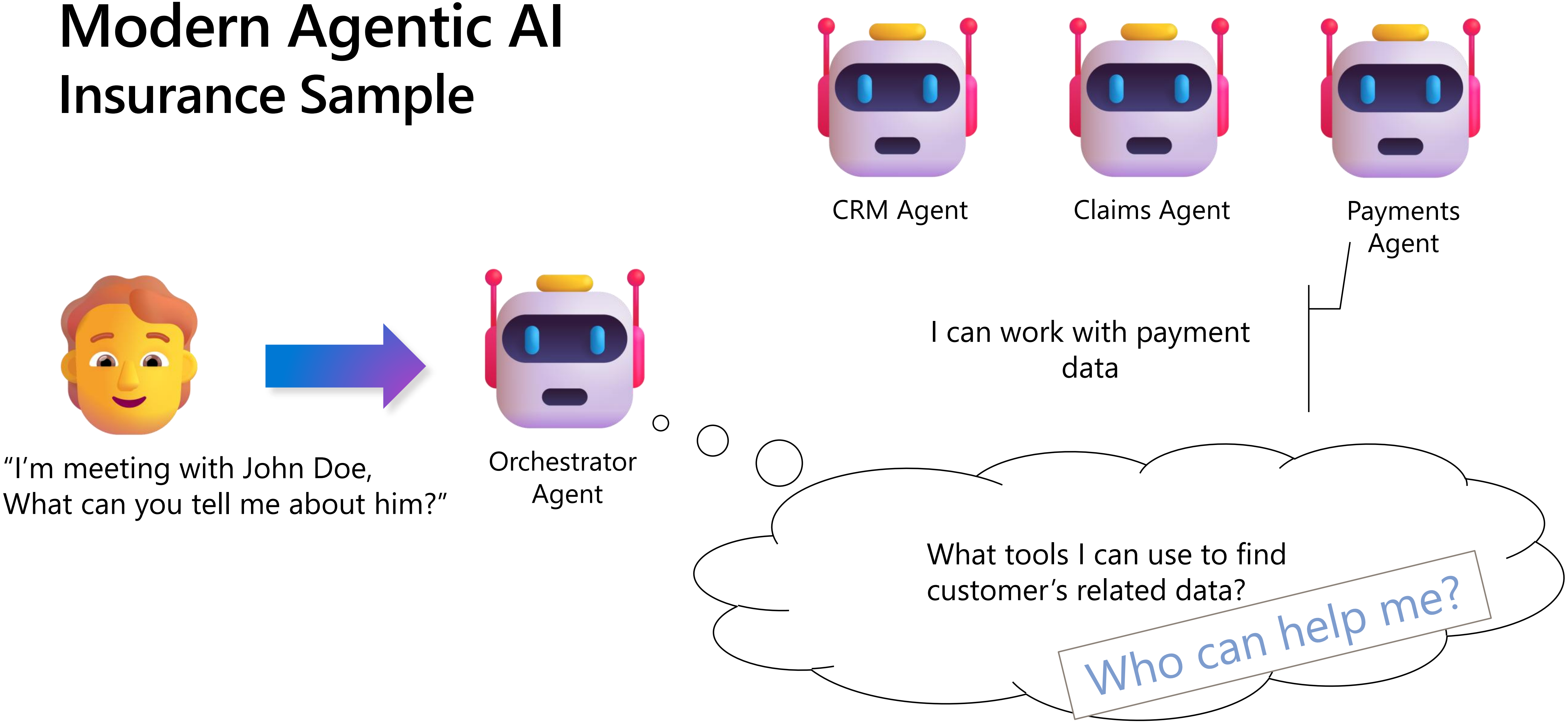
“I’m meeting with John Doe,
What can you tell me about him?”

Orchestrator Agent

What tools I can use to find customer’s related data?

Who can help me?

Modern Agentic AI
Insurance Sample



CRM Agent

Claims Agent

Payments Agent

I can work with payment data

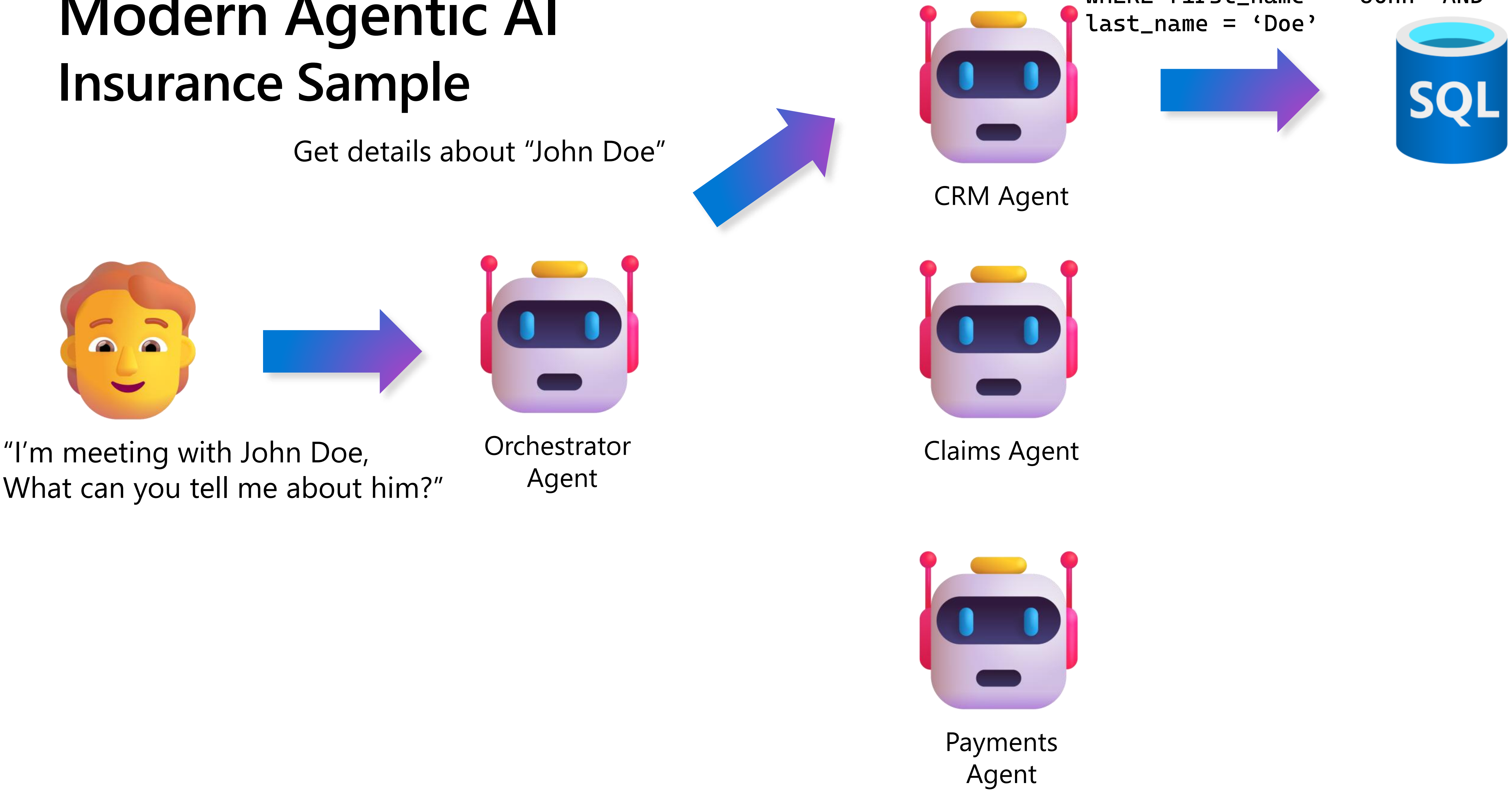
“I’m meeting with John Doe,
What can you tell me about him?”

Orchestrator Agent

What tools I can use to find customer’s related data?

Who can help me?

Modern Agentic AI Insurance Sample



```
SELECT * FROM pass.customers
WHERE first_name = ‘John’ AND
last_name = ‘Doe’
```

Get details about “John Doe”

“I’m meeting with John Doe,
What can you tell me about him?”

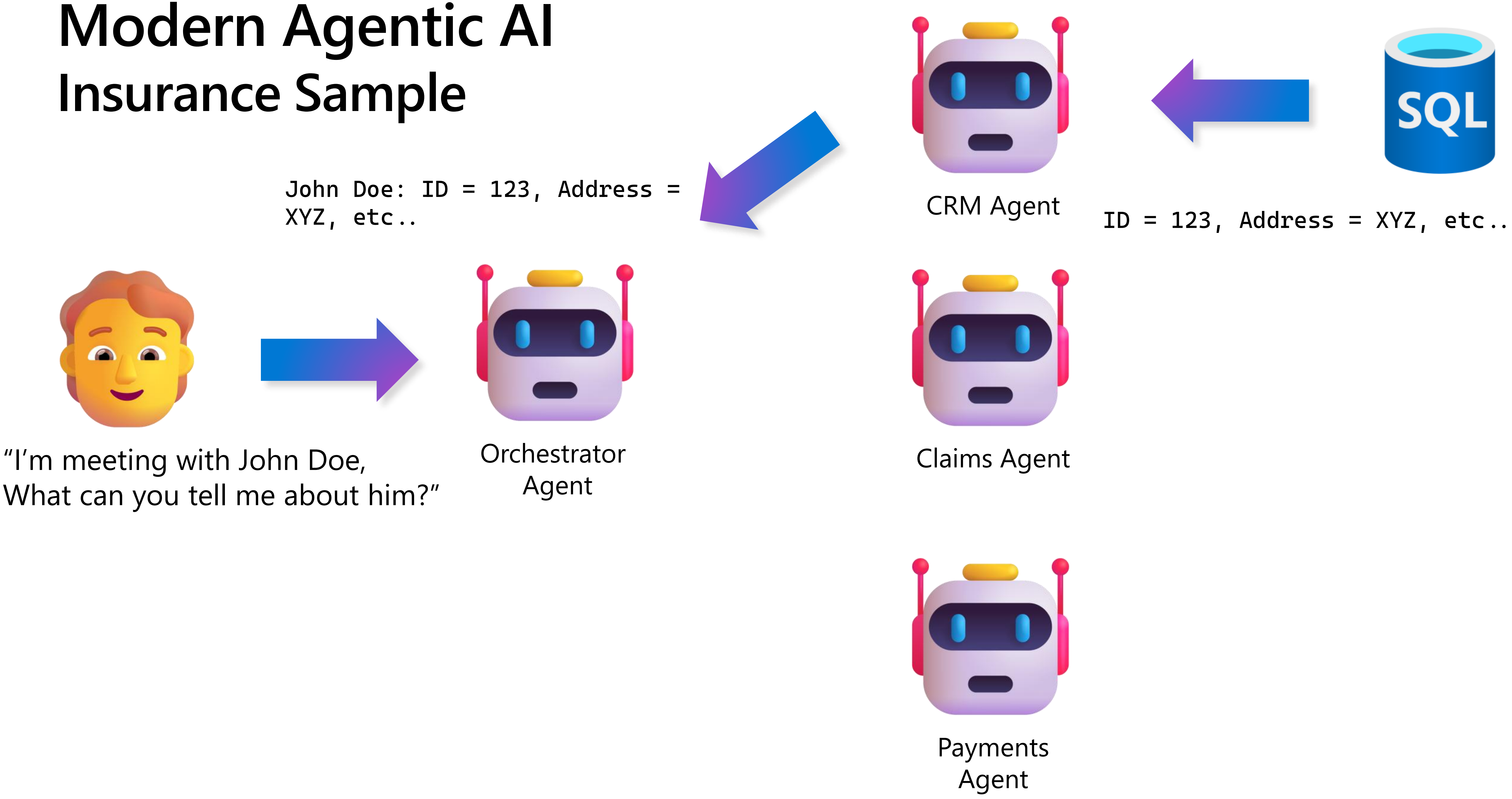
Orchestrator
Agent

CRM Agent

Claims Agent

Payments
Agent

Modern Agentic AI
Insurance Sample



```
John Doe: ID = 123, Address =
XYZ, etc..
```

CRM Agent

```
ID = 123, Address = XYZ, etc..
```

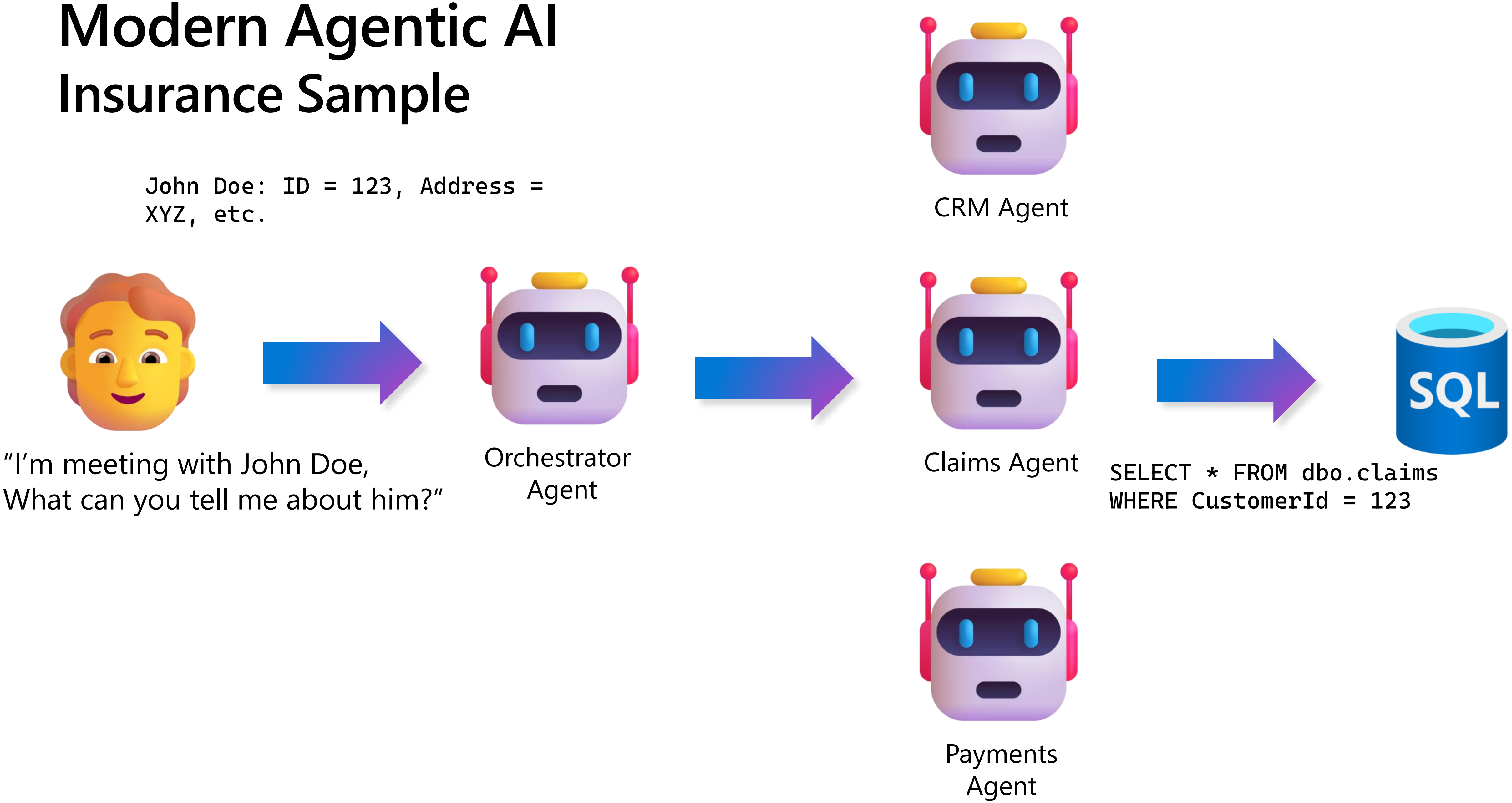
“I’m meeting with John Doe,
What can you tell me about him?”

Orchestrator Agent

Claims Agent

Payments Agent

Modern Agentic AI
Insurance Sample



```
John Doe: ID = 123, Address =
XYZ, etc.
```

“I’m meeting with John Doe,
What can you tell me about him?”

Orchestrator
Agent

CRM Agent

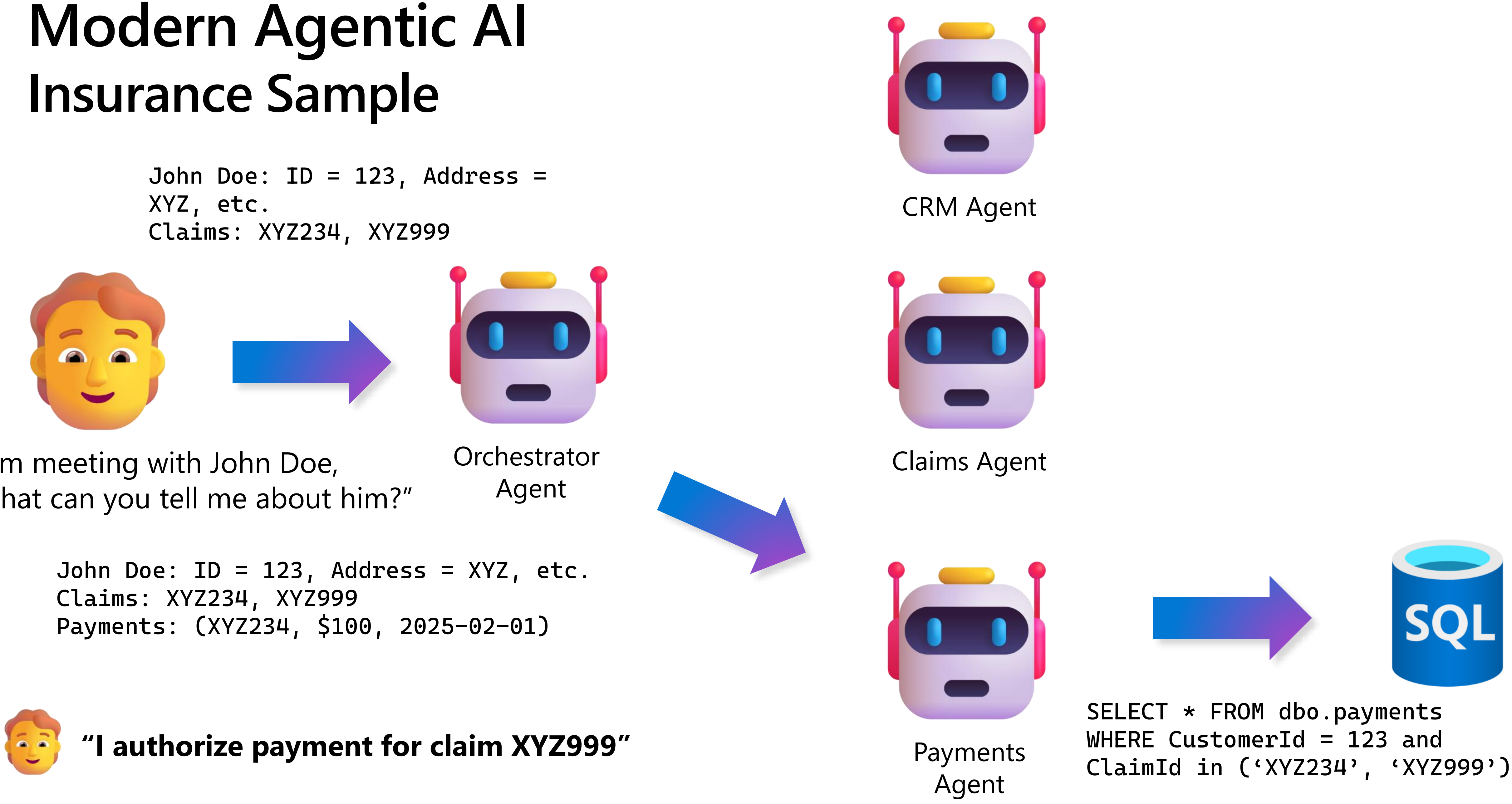
Claims Agent

```
SELECT * FROM dbo.claims
WHERE CustomerId = 123
```

SQL

Payments
Agent

Modern Agentic AI
Insurance Sample



```
John Doe: ID = 123, Address =
XYZ, etc.
Claims: XYZ234, XYZ999
```

"I'm meeting with John Doe,
What can you tell me about him?"

Orchestrator
Agent

CRM Agent

Claims Agent

Payments
Agent

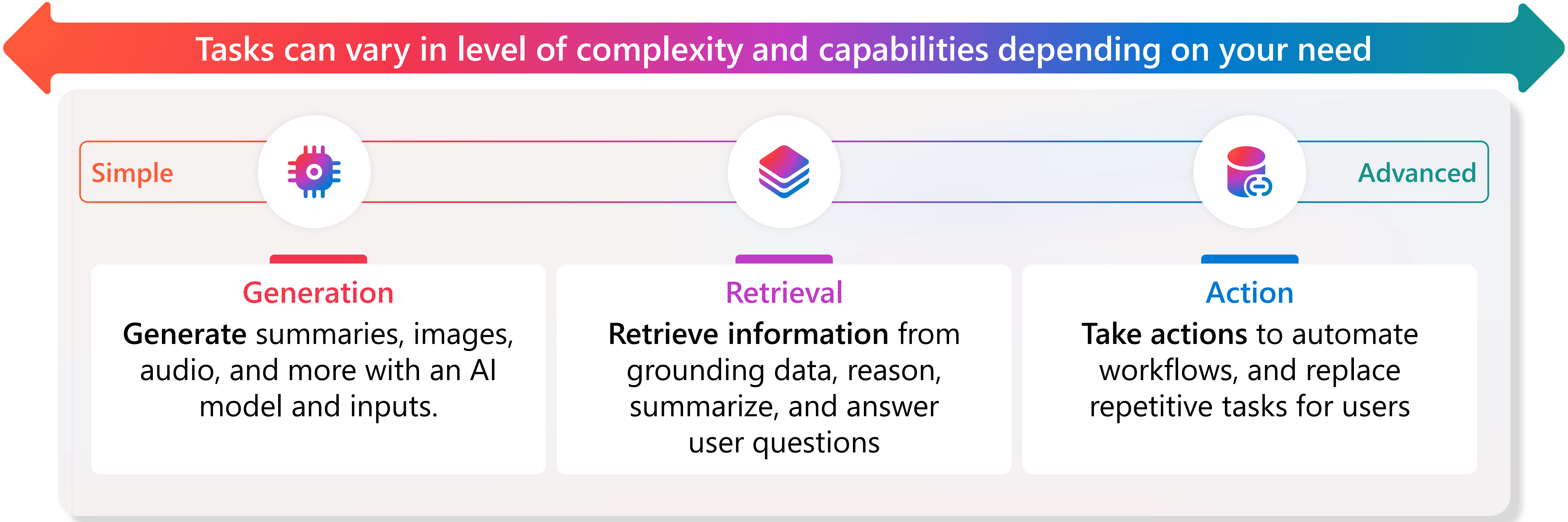
```
John Doe: ID = 123, Address = XYZ, etc.
Claims: XYZ234, XYZ999
Payments: (XYZ234, $100, 2025-02-01)
```

```
SELECT * FROM dbo.payments
WHERE CustomerId = 123 and
ClaimId in ('XYZ234', 'XYZ999')
```

"I authorize payment for claim XYZ999"

What are agents?

AI designed to perform a task



Tasks can vary in level of complexity and capabilities depending on your need

Simple

Advanced

Generation

Generate summaries, images, audio, and more with an AI model and inputs.

Retrieval

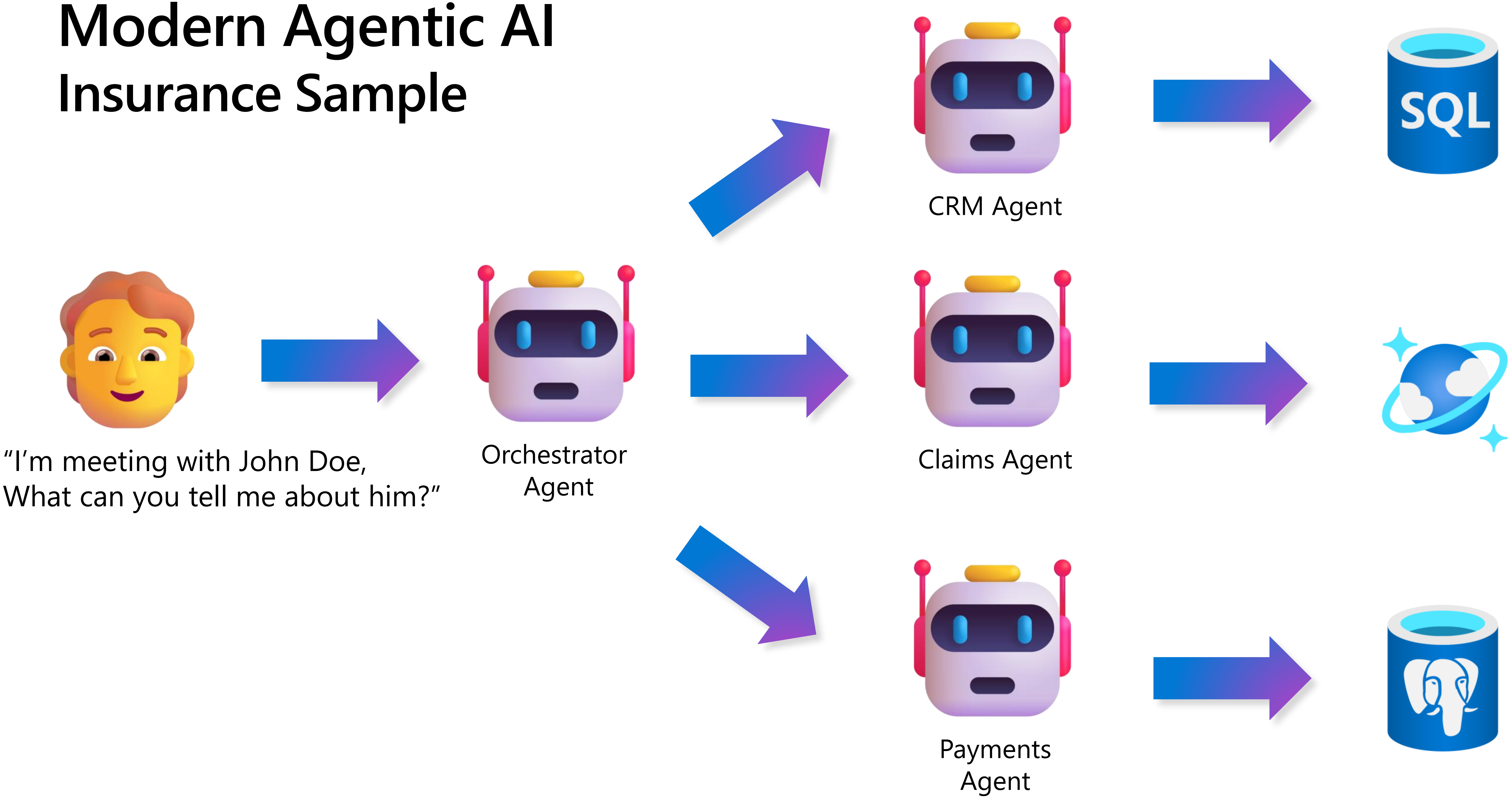
Retrieve information from grounding data, reason, summarize, and answer user questions

Action

Take actions to automate workflows, and replace repetitive tasks for users

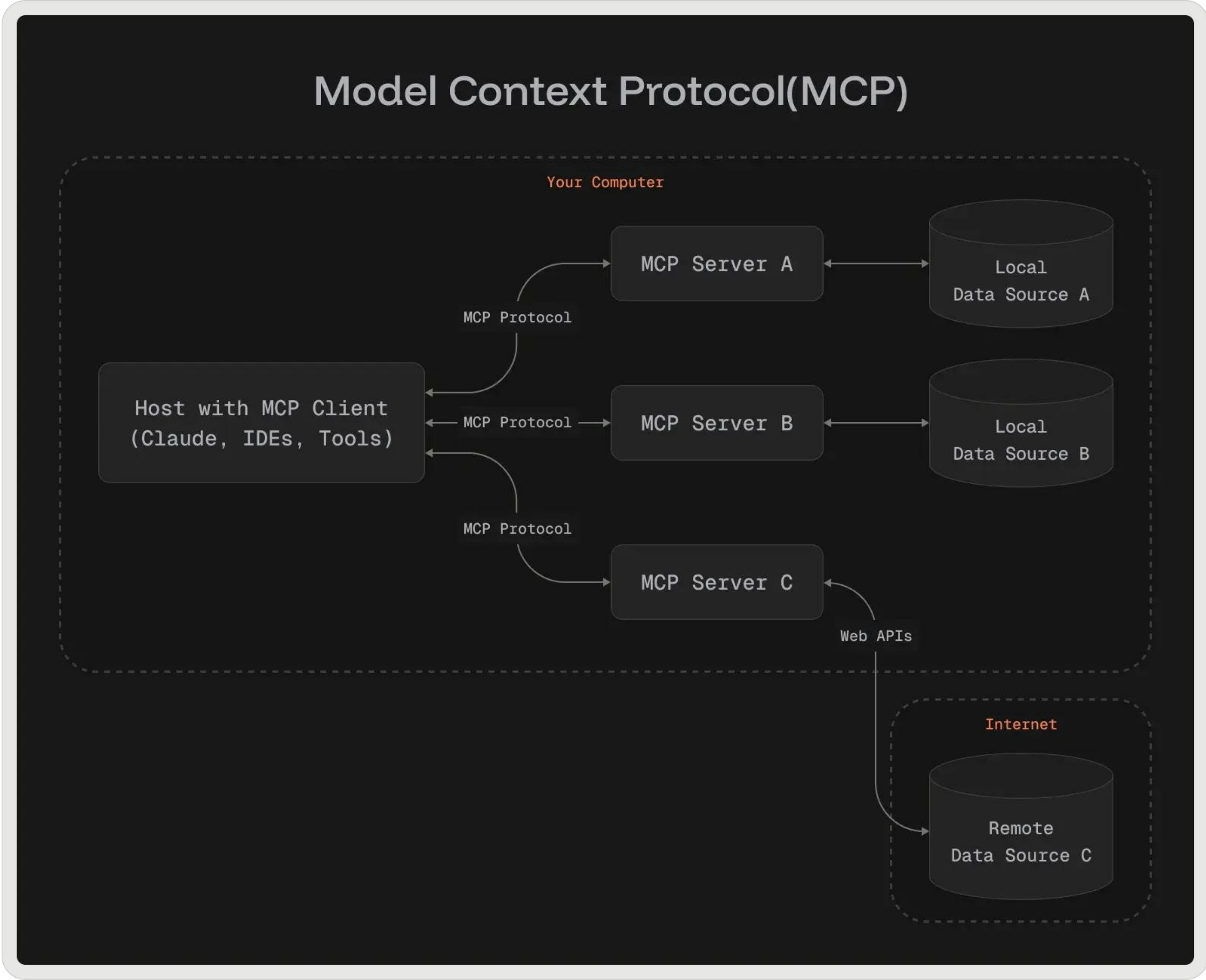
Microsoft

Modern Agentic AI
Insurance Sample



MCP – Model Context Protocol

The Model Context Protocol (MCP) is an **open standard** designed to **bridge AI applications** with external systems and data sources, ensuring secure interactions. It simplifies **how models interact** with various tools and databases through a single, secure interface, enabling them to **invoke functions and fetch data** efficiently



Official Microsoft MCP Tools

- Official GitHub MCP Server [Demo]
- Azure SDK MCP Server [Demo]
- Azure AI Agent Service [Demo] [GitHub] [Blog]
- Playwright MCP Server [GitHub]
- Azure CosmosDB MCP Server [GitHub]
- Foundry Windows Actions Tool [Demo]

References – Azure Cosmos DB

- https://github.com/Azure-Samples/chat-with-your-data-solution-accelerator
- https://azurecosmosdb.github.io/gallery/
- https://github.com/Azure/document-vector-pipeline
- https://github.com/Azure-Samples/rag-postgres-openai-python/
- https://github.com/Azure-Samples/azure-postgres-pgvector-python

- https://devblogs.microsoft.com/cosmosdb/azure-cosmos-db-vector-search-with-diskann-part-1-full-space-search/
- https://devblogs.microsoft.com/cosmosdb/sharded-diskann-focused-vector-search-for-better-performance-and-lower-cost/
- https://devblogs.microsoft.com/cosmosdb/new-vector-search-full-text-search-and-hybrid-search-features-in-azure-cosmos-db-for-nosql/

References – Azure SQL

- Demos from Fabric Conference workshop
- Semantic Kernel / Agentic RAG demo
- Vectors demo repo
- Chat with your data
- SQL Database Vector Search Sample
- Migrate and Modernize

References – Azure PostgreSQL

- Introducing the GraphRAG Solution for Azure Database for PostgreSQL
- https://github.com/Azure-Samples/rag-postgres-openai-python
- Introducing the Semantic Ranking Solution for Azure Database for PostgreSQL
- Build AI Apps with Azure Database for PostgreSQL
- https://github.com/mcaps-microsoft/csa-cto

Thank you!

- dandy.weyn@microsoft.com
- luciano.moreira@microsoft.com
- mehasodhi@microsoft.com
- prmadi@microsoft.com