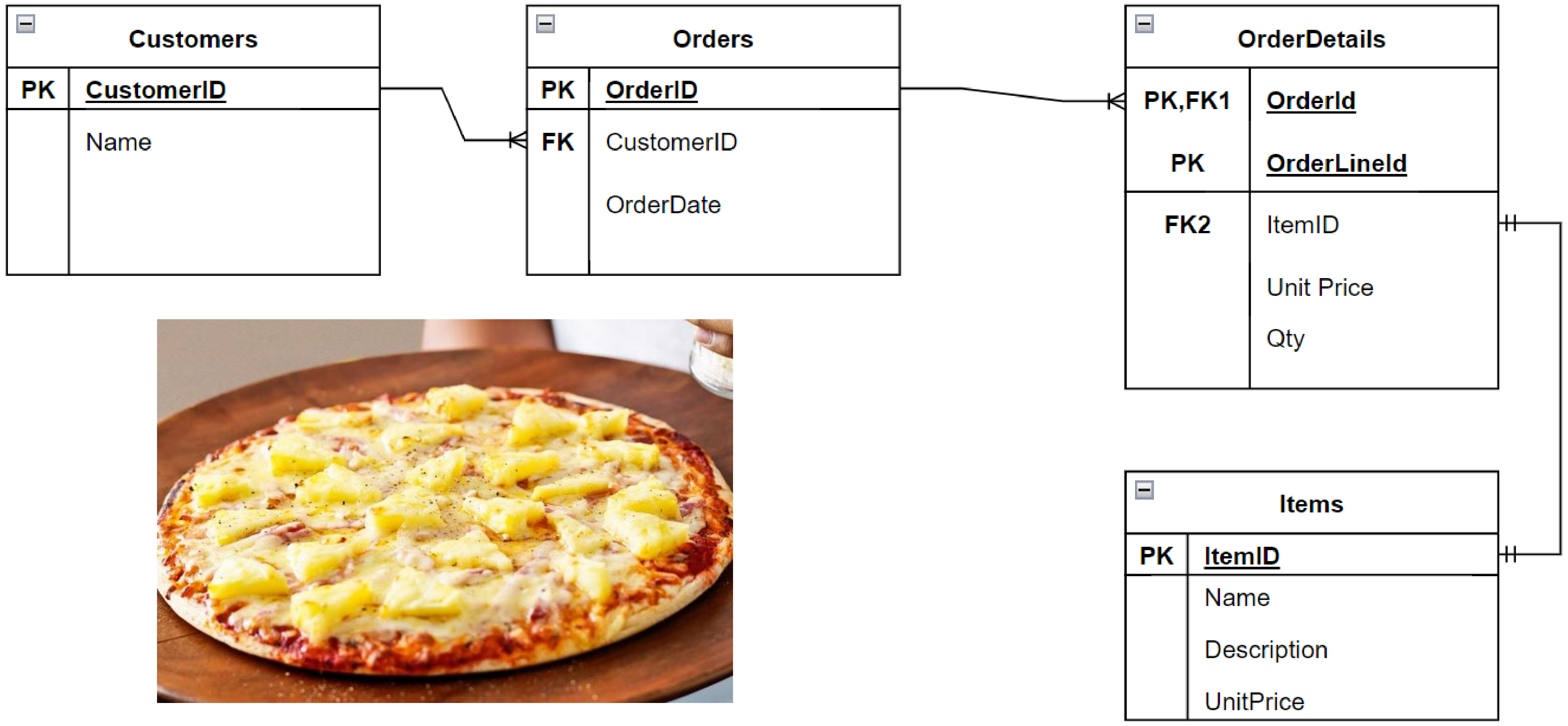
Data Modeling for Azure Cosmos DB for the Relational Mind

Sarah "sadukie" Dutkiewicz

Microsoft MVP, Developer Technologies

Relational Data: Sadukie's Pizza Store



How are you using this data?

General questions

- ❑How is this data being used?
- ❑What fields are queried together?
- ❑How often does the data change?
- ❑Is this a read-heavy data set?
- ❑Is this a write-heavy data set?

Modeling for a Document Database

Data Modeling for Document Stores

- Schema-less
- How is data being used?
  - Reads?
  - Writes?
- Denormalization
- Embedding vs referencing data
- Partition considerations
- Microsoft Docs - Modeling data in Azure Cosmos DB: https://docs.microsoft.com/en-us/azure/cosmos-db/sql/modeling-data
- Azure Cosmos DB Essentials Season 2 – Schema Design Strategy: https://www.youtube.com/watch?v=bKDaL-GRSAM

Denormalize / Embed

- Contained relationships
- One-to-few relationships
- Infrequent changes
- Queried together
- Data has bounds

```
{
    "CustomerName": "Matt",
    "OrderId": 3,
    "OrderDate": "2022-03-01T16:50:11",
    "OrderDetails": [
        {
            "OrderLineId": 1,
            "ItemName": "Cheese Pizza",
            "Qty": 4,
            "UnitPrice": 10.99
        }
    ]
}
```

Normalize / Reference

- One-to-many relationships
- Many-to-many relationships
- Frequent changes
- Data has no bounds

- Manually managed – no foreign keys/constraints

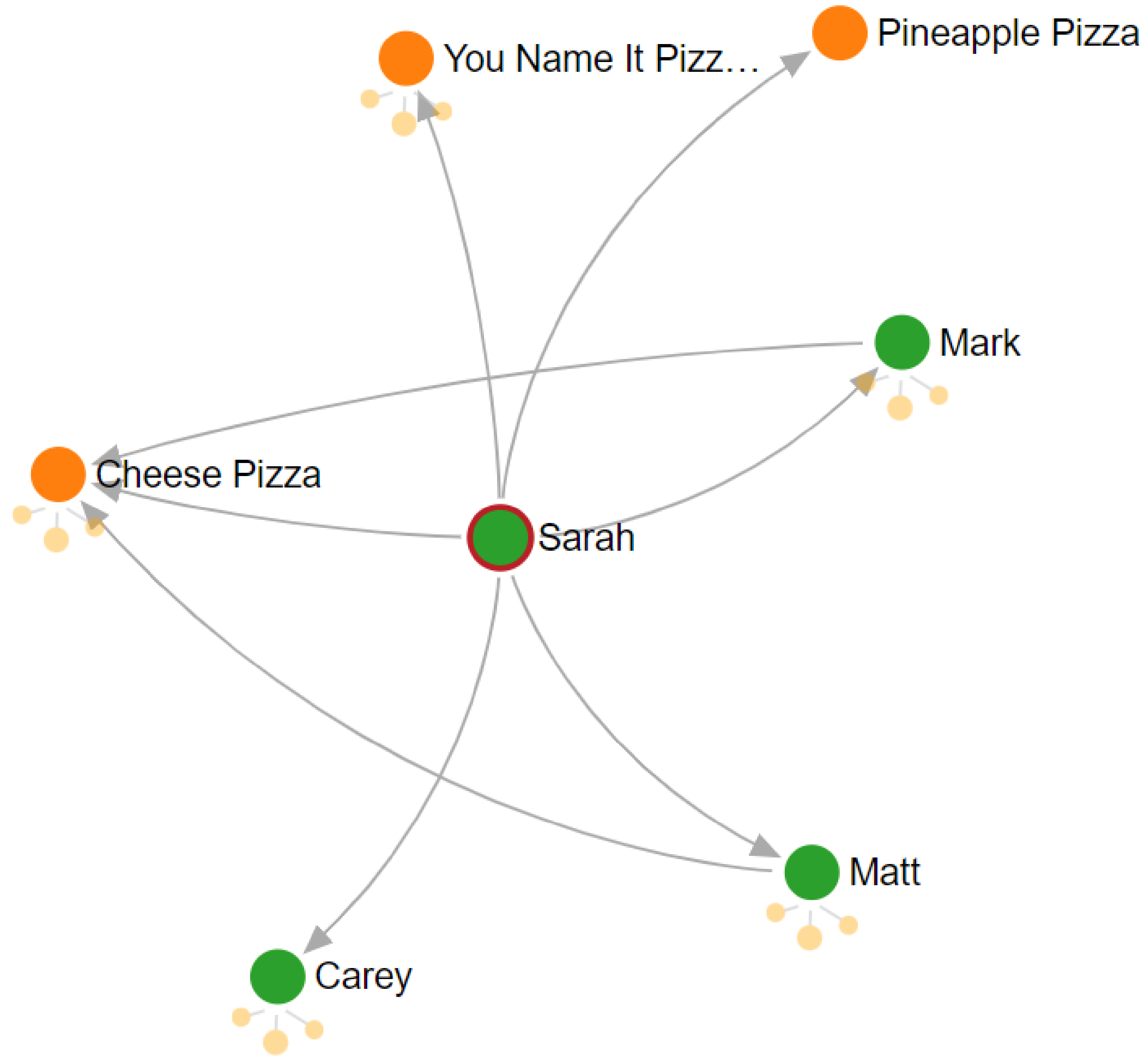
- Requires more round trips – no JOINs in the relational SQL sense
- Typically better for **write-heavy loads**

How are you using this data?

Demo: Document Store for Order History – Azure Cosmos DB SQL API

Reference:
Microsoft Docs – Azure Cosmos DB PDF query cheat sheets:
https://docs.microsoft.com/en-us/azure/cosmos-db/sql/query-cheat-sheet

Modeling for a Graph Database


You Name It Pizz…
Pineapple Pizza
Mark
Cheese Pizza
Sarah
Matt
Carey

People who bought pineapple pizza also bought…

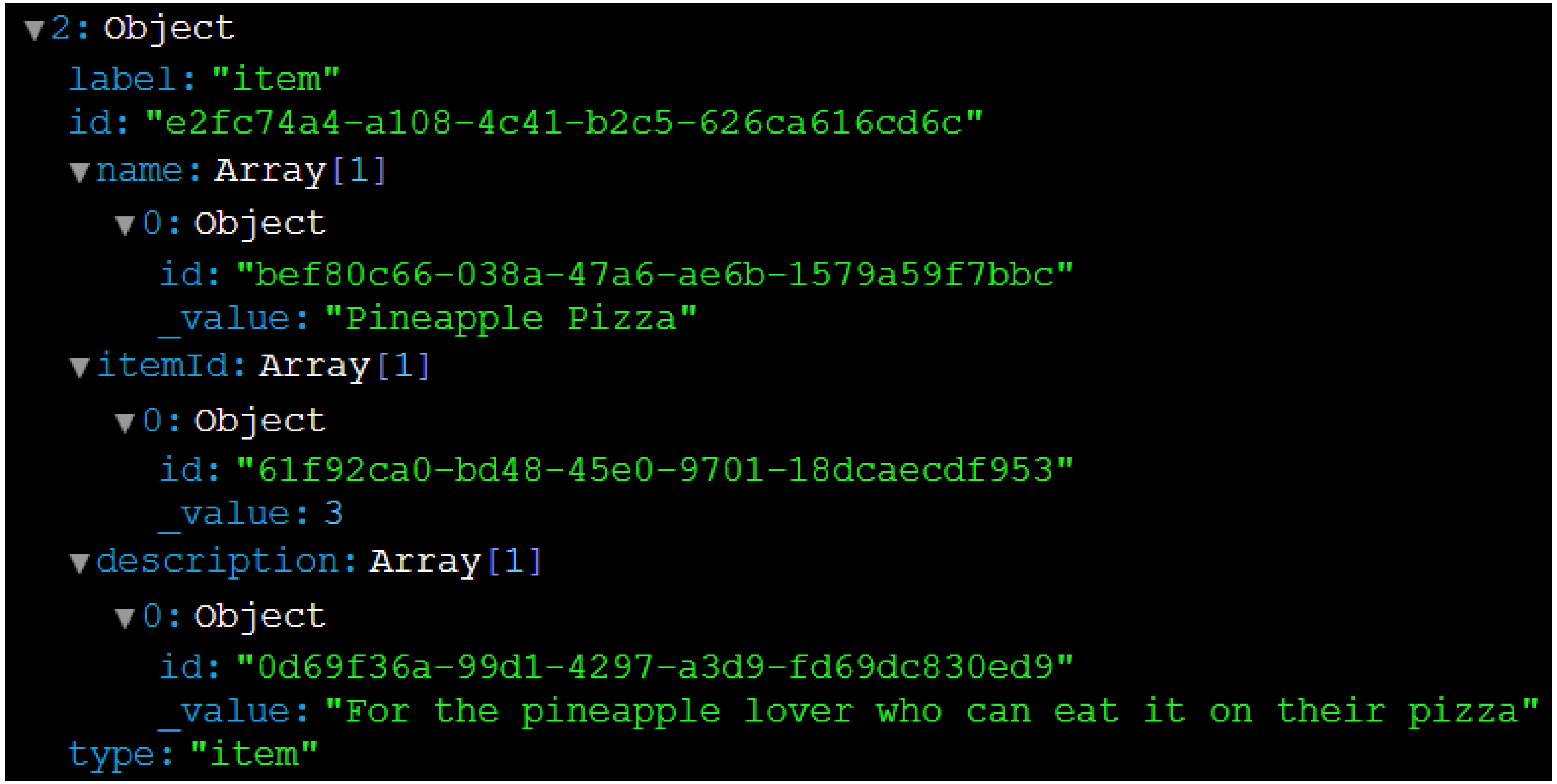
- g.V()
  ```
  .has('name','Pineapple Pizza’)
  .inE('bought’)
  .outV()
  .outE('bought’)
  .inV()
  .hasLabel('item’)
  ```
- .executionprofile()

Graph Database

- Azure Cosmos DB Gremlin API
- Using Gremlin syntax
- Use while exploring **relationships** (edges) between **things (**vertices**)**
- Collection of vertices and edges is known as a **graph**

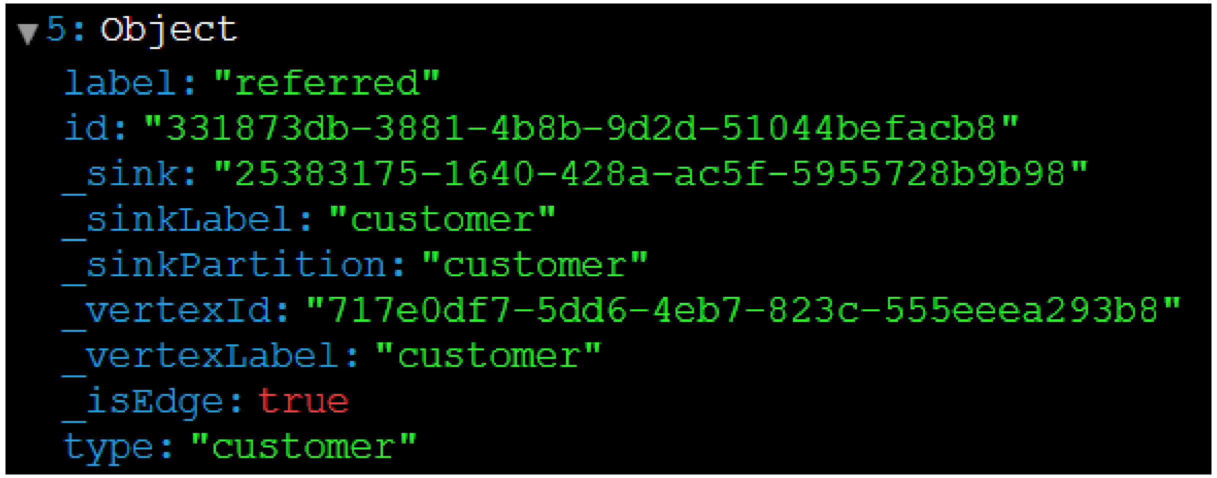
- Common uses:
  - Recommendation engine
  - Fraud detection
  - Managing IT networks – licensing tracking, asset management
  - Software dependency graphs

JSON under the covers - Vertex



```
▼2: Object
  label: "item"
  id: "e2fc74a4-a108-4c41-b2c5-626ca616cd6c"
 ▼name: Array[1]
   ▼0: Object
     id: "bef80c66-038a-47a6-ae6b-1579a59f7bbc"
     _value: "Pineapple Pizza"
 ▼itemId: Array[1]
   ▼0: Object
     id: "61f92ca0-bd48-45e0-9701-18dcaecdf953"
     _value: 3
 ▼description: Array[1]
   ▼0: Object
     id: "0d69f36a-99d1-4297-a3d9-fd69dc830ed9"
     _value: "For the pineapple lover who can eat it on their pizza"
  type: "item"
```

JSON under the covers - Edge



```
▼5: Object
   label: "referred"
   id: "331873db-3881-4b8b-9d2d-51044befacb8"
   _sink: "25383175-1640-428a-ac5f-5955728b9b98"
   _sinkLabel: "customer"
   _sinkPartition: "customer"
   _vertexId: "717e0df7-5dd6-4eb7-823c-555eeea293b8"
   _vertexLabel: "customer"
   _isEdge: true
   type: "customer"
```

Graph Database Modeling Considerations

- Identify entities within a single query
  - Entities become vertex objects of a graph.
- Determine direction of relationships
  - Azure Cosmos DB Gremlin API is directional – **in** and **out**
  - Vertices are stored with **out** edges
  - **in** is cross-partition – so if used frequently, add edges in both directions
- Understand **read and write requirements**
  - Helps with graph strategies
- Queries can be evaluated with `.executionprofile()`

Graphing at Scale

- Azure Cosmos DB horizontal scaling at work
- Partition key is used for balancing
    - Required for Gremlin drivers 3.x
    - Required for Azure portal
- Vertices stored based on partition key
- Edges are stored with their source vertices

- Microsoft Docs – Data partitioning in Azure Cosmos DB GremlinAPI: https://docs.microsoft.com/en-us/azure/cosmos-db/graph/graph-partitioning

Graph Strategies

- Vertex-based properties
  - Reduces redundancy – common relational mindset
  - Adds:
    - Model complexity
    - Increased latency
    - Query complexity
    - Computation cost
- Property-embedded vertices
  - Allows properties tied to the entity
  - Key-values
  - Adds redundancy across nodes
  - Reduces model complexity
  - Simpler queries
  - Cost-efficient traversals

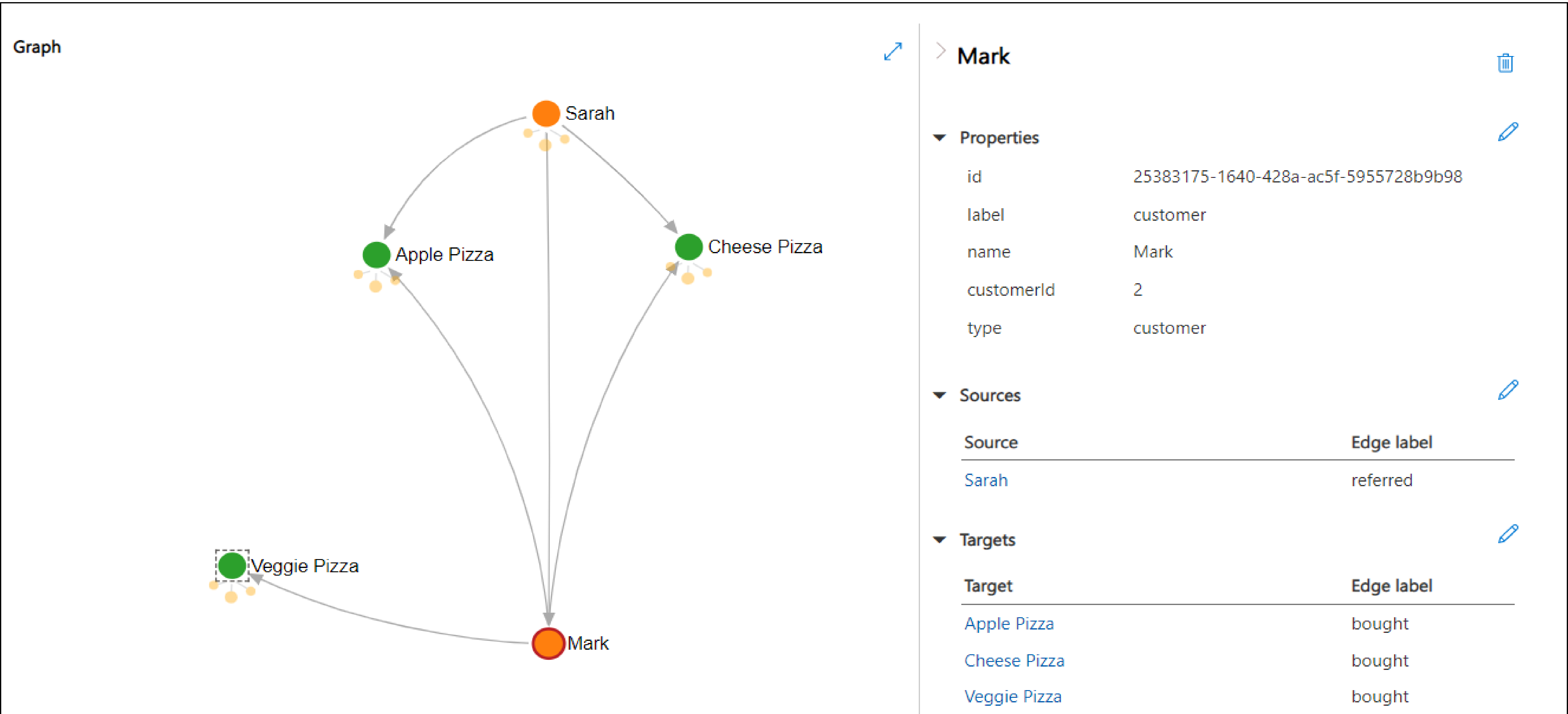
- Microsoft Docs – Graph data modeling for Azure Cosmos DB Gremlin API: https://docs.microsoft.com/en-us/azure/cosmos-db/graph/graph-modeling

How are you using this data?

Demo: Graph for Recommendation Engine – Azure Cosmos DB Gremlin API

Reference:
Microsoft Docs – Azure Cosmos DB PDF query cheat sheets:
https://docs.microsoft.com/en-us/azure/cosmos-db/sql/query-cheat-sheet

Demo – Recommendation Engine



Mark

Properties

id	25383175-1640-428a-ac5f-5955728b9b98
label	customer
name	Mark
customerId	2
type	customer

Sources

Source	Edge label
Sarah	referred

Targets

Target	Edge label
Apple Pizza	bought
Cheese Pizza	bought
Veggie Pizza	bought

Resources

Azure Cosmos DB Resources for Data Modeling

- Modeling data in Azure Cosmos DB - Azure Cosmos DB | Microsoft Docs
- Schema Design Strategy | Azure Cosmos DB Essentials Season 2
- Azure Cosmos DB PDF query cheat sheets | Microsoft Docs
- Data partitioning in Azure Cosmos DB Gremlin API | Microsoft Docs
- Graph data modeling for Azure Cosmos DB Gremlin API | Microsoft Docs

Contact Info

- Twitter: @sadukie
- LinkedIn: https://www.linkedin.com/in/sadukie