

Using the Gremlin API for a Graph Data Model



Leonard Lobel
CTO, SLEEK TECHNOLOGIES
lennilobel.wordpress.com



Cosmos DB Graph Database

Graph container

Horizontal partitioning, provisioned throughput, global distribution, indexing

Vertex and Edge objects

Entities and relationships
Both can hold arbitrary key-value pairs

Create/Retrieve/Update/Delete

GraphSON and Gremlin

Apache TinkerPop

<http://tinkerpop.apache.org>

Focus on relationships

Chain relationship queries with Gremlin



Graph Database Scenarios

Complex relationships

Many “many-to-many” relationships

Excessive JOINs

Analyze interconnected data and relationships

Typical graph applications

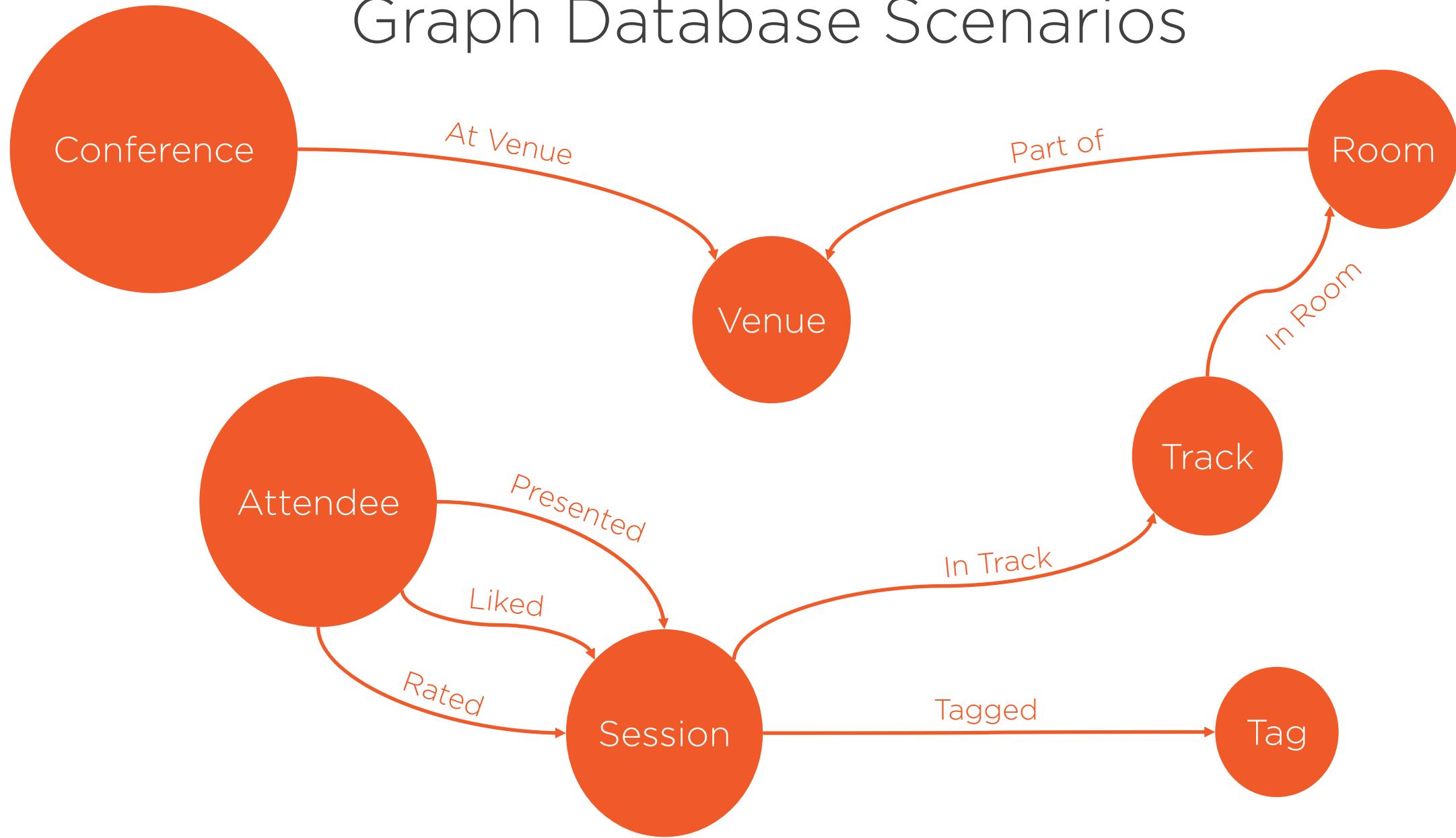
Social networks
Recommendation engines
Knowledge graphs
Many more...



Graph Database Scenarios



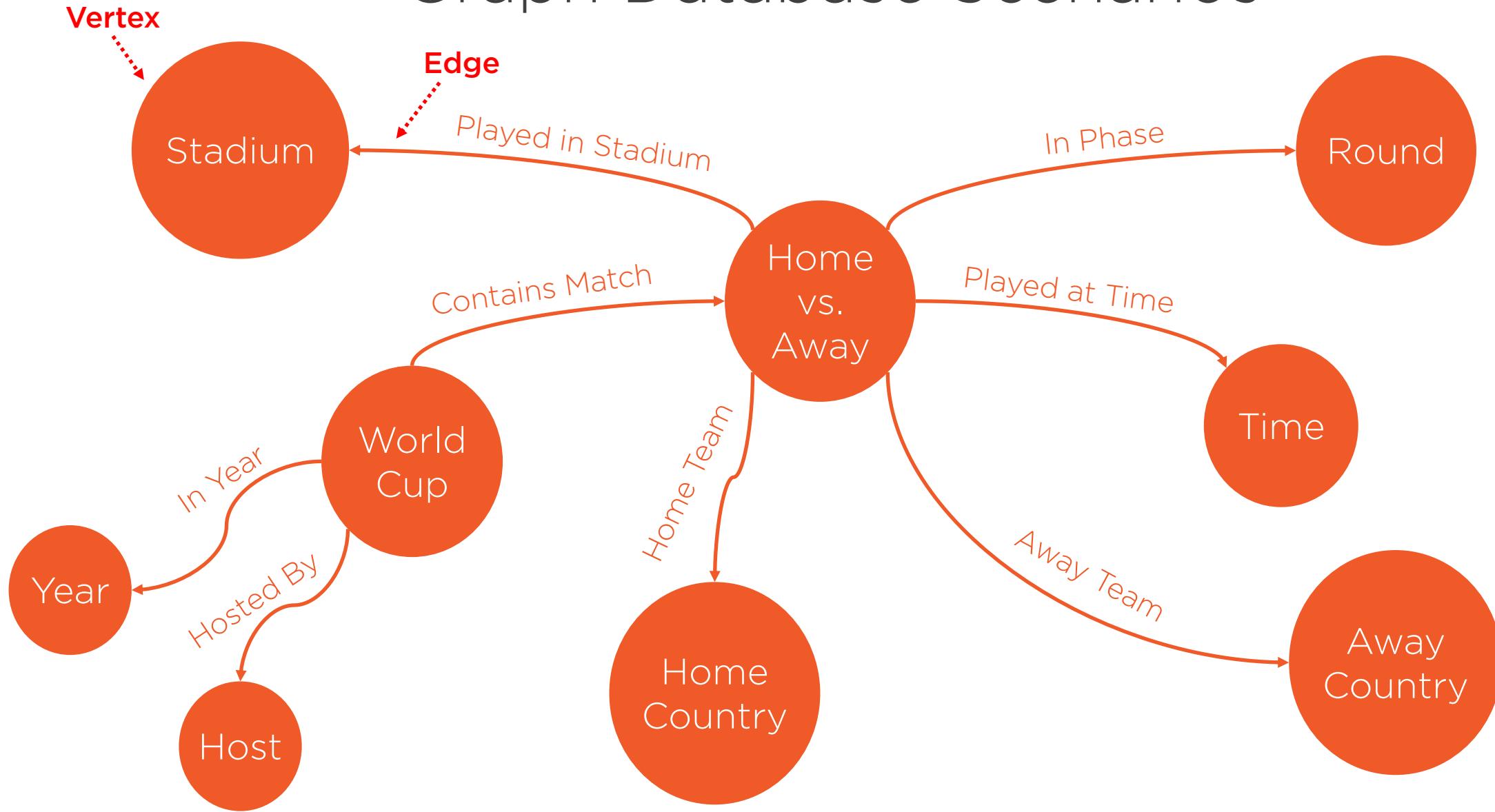
Graph Database Scenarios



Graph Database Scenarios



Graph Database Scenarios



Vertices and Edges

Vertex and Edge properties

id (within partition key)

label (type)

Additional arbitrary properties (including the partition key)

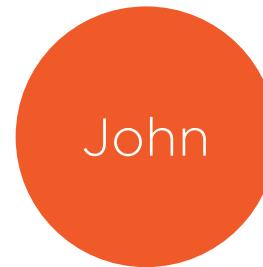
Additional Edge properties

Cardinality
(in-and-out vertices)

Create two edges for bi-directionality



Vertices and Edges



Vertex

label: person
id: John
age: 25
likes: pizza



Vertex

label: company
id: Acme
founded: 2001
location: NY

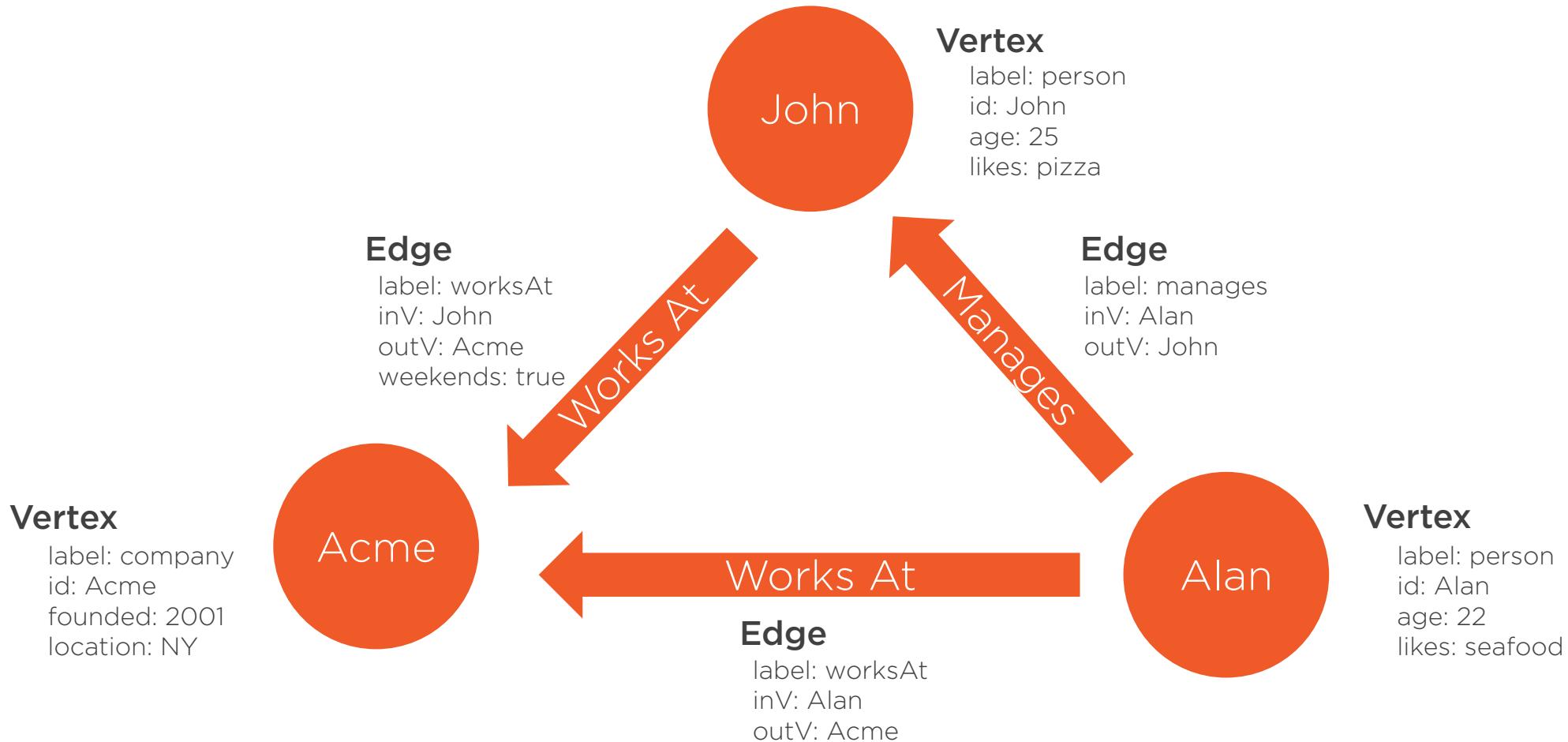


Vertex

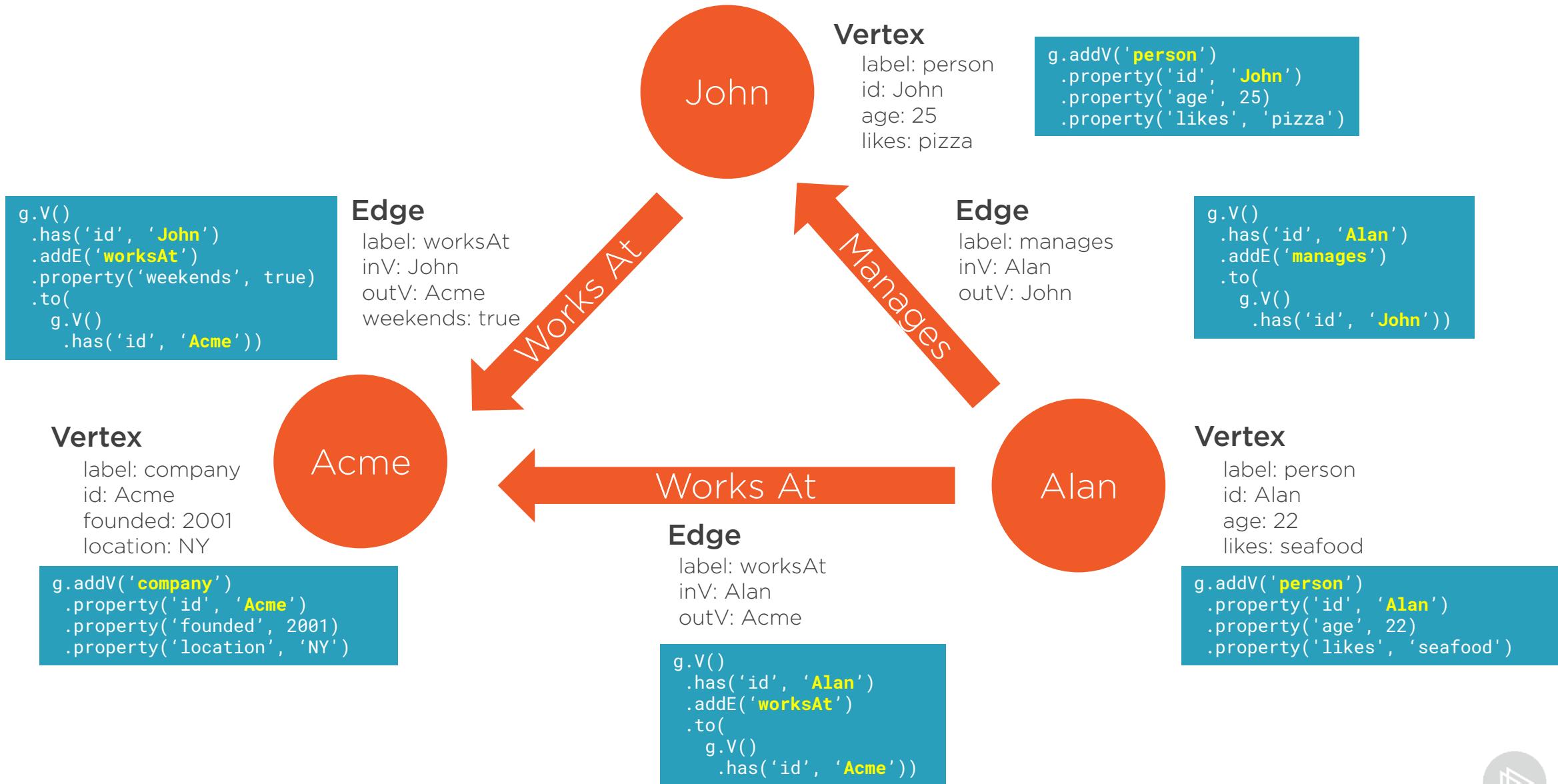
label: person
id: Alan
age: 22
likes: seafood



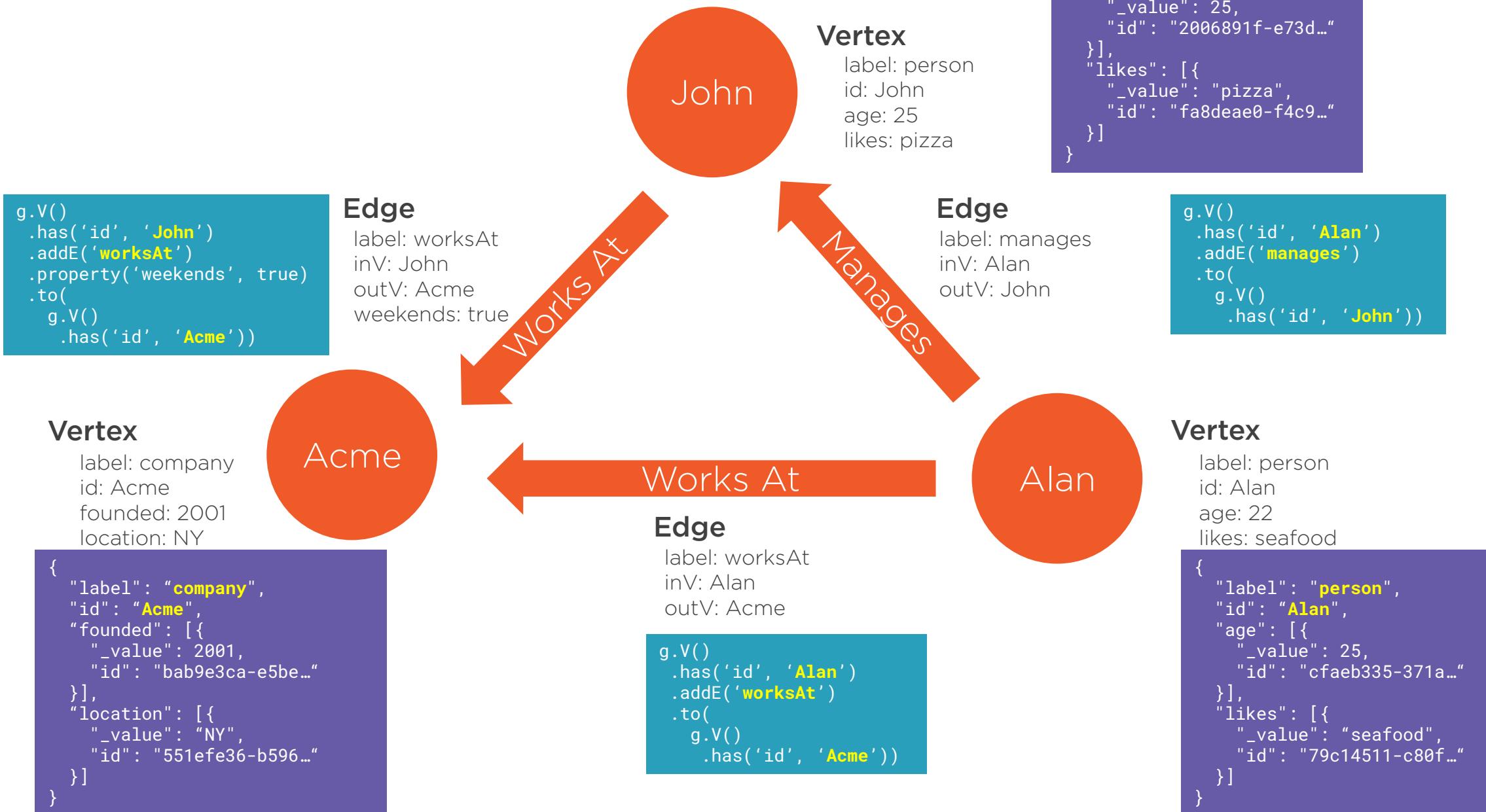
Vertices and Edges



Populating the Graph



Populating the Graph



Populating the Graph

```
{  
  "id": "c837bf96-fc2d...",  
  "_vertexId": "John",  
  "_vertexLabel": "person",  
  "label": "worksAt",  
  "_sink": "Acme",  
  "_sinkLabel": "company",  
  "weekends": true,  
  "_isEdge": true  
}
```

Vertex

label: company
id: Acme
founded: 2001
location: NY

```
{  
  "label": "company",  
  "id": "Acme",  
  "founded": [{  
    "_value": 2001,  
    "id": "bab9e3ca-e5be..."  
  }],  
  "location": [{  
    "_value": "NY",  
    "id": "551efe36-b596..."  
  }]  
}
```

Edge

label: worksAt
inV: John
outV: Acme
weekends: true



Vertex

label: person
id: John
age: 25
likes: pizza

```
{  
  "label": "person",  
  "id": "John",  
  "age": [{  
    "_value": 25,  
    "id": "2006891f-e73d..."  
}],  
  "likes": [{  
    "_value": "pizza",  
    "id": "fa8deae0-f4c9..."  
}]  
}
```

Edge

label: manages
inV: Alan
outV: John



```
{  
  "id": "59609f8f-c925...",  
  "_vertexId": "Alan",  
  "_vertexLabel": "person",  
  "label": "manages",  
  "_sink": "John",  
  "_sinkLabel": "person",  
  "_isEdge": true  
}
```

Edge

label: worksAt
inV: Alan
outV: Acme

```
{  
  "id": "fb3c578f-9ef7...",  
  "_vertexId": "Alan",  
  "_vertexLabel": "person",  
  "label": "worksAt",  
  "_sink": "Acme",  
  "_sinkLabel": "company",  
  "_isEdge": true  
}
```

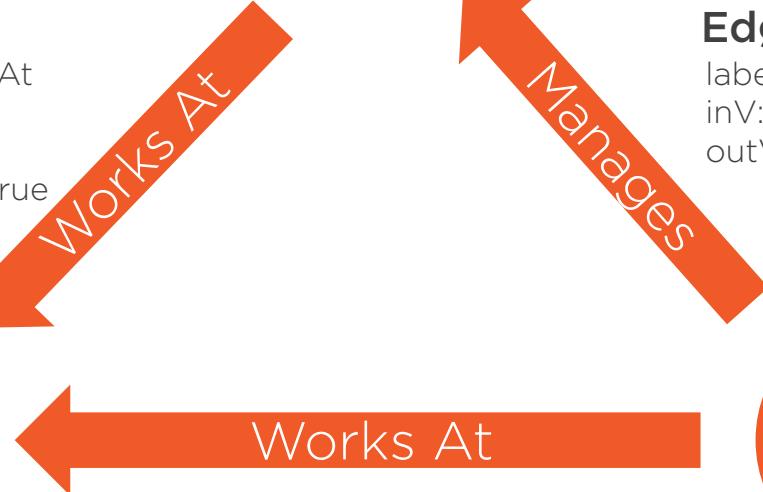
Edge

label: worksAt
inV: Alan
outV: Acme

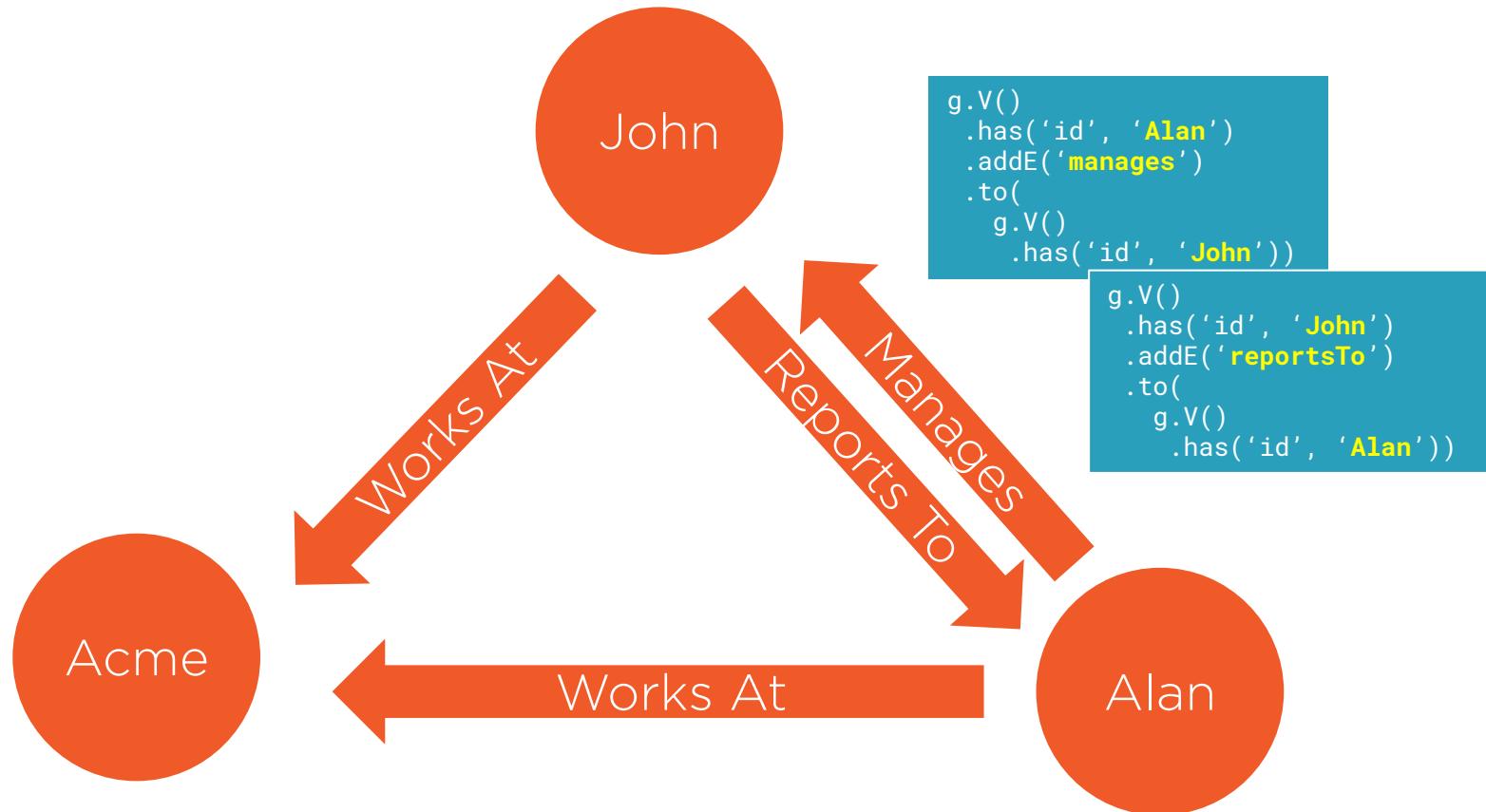
Vertex

label: person
id: Alan
age: 22
likes: seafood

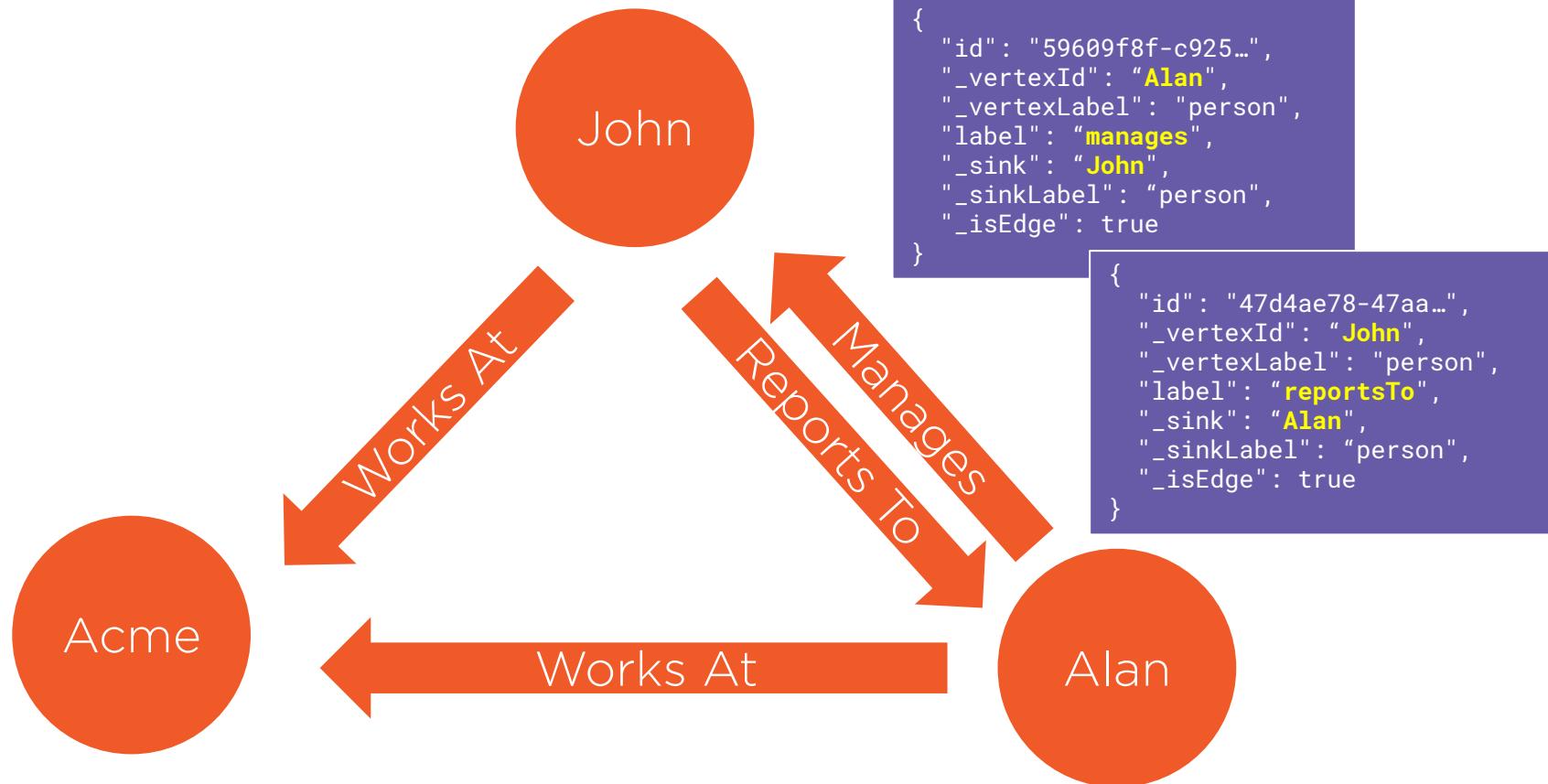
```
{  
  "label": "person",  
  "id": "Alan",  
  "age": [{  
    "_value": 25,  
    "id": "cfaeb335-371a..."  
}],  
  "likes": [{  
    "_value": "seafood",  
    "id": "79c14511-c80f..."  
}]  
}
```



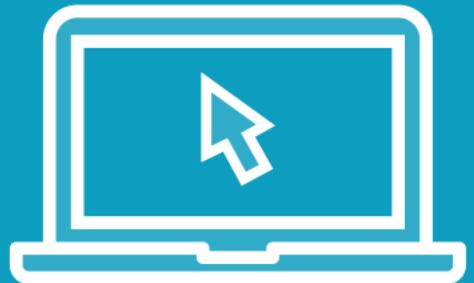
Bi-directional Relationships



Bi-Directional Relationships



Demo



Creating a simple graph



Writing Gremlin Queries

Functional language

Chain multiple steps together

Define Vertices and Edges

`.addV('label')`

`.addE('label')`

`.property('key', 'value')`

Query on filters and relationships

`.V('label')`

`.out('label')`

`.has('property', condition)`

Cosmos DB Gremlin support

<http://aka.ms/gremlin-support>



Azure Cosmos DB Gremlin

https://docs.microsoft.com/en-us/azure/cosmos-db/gremlin-support#gremlin-steps

Gremlin steps

Filter

- > Samples
- > Concepts
- < How To Guides
- < Develop
 - > SQL API
 - > MongoDB API
- < Graph API
 - Gremlin support
 - > Table API
 - Change feed
 - Geospatial
 - Indexing
 - Connected Service in Visual Studio
- > Manage
- > Integrate
- > Reference
- > Resources

Download PDF

step	Description	TinkerPop 3.2 Documentation	Notes
addE	Adds an edge between two vertices	addE step	
addV	Adds a vertex to the graph	addV step	
and	Ensures that all the traversals return a value	and step	
as	A step modulator to assign a variable to the output of a step	as step	
by	A step modulator used with group and order	by step	
coalesce	Returns the first traversal that returns a result	coalesce step	
constant	Returns a constant value. Used with coalesce	constant step	
count	Returns the count from the traversal	count step	

Feedback

Edit

Share

Theme

Light

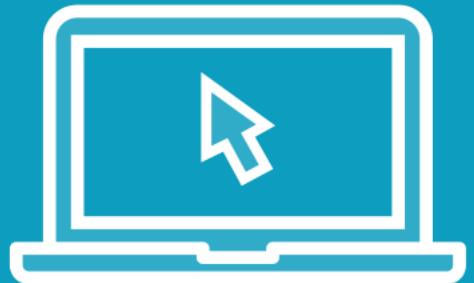
In this article

- Gremlin by example
- Gremlin features
- Gremlin wire format: GraphSON
- Gremlin partitioning
- Gremlin steps
- Next Steps

Is this page helpful? X

YES NO

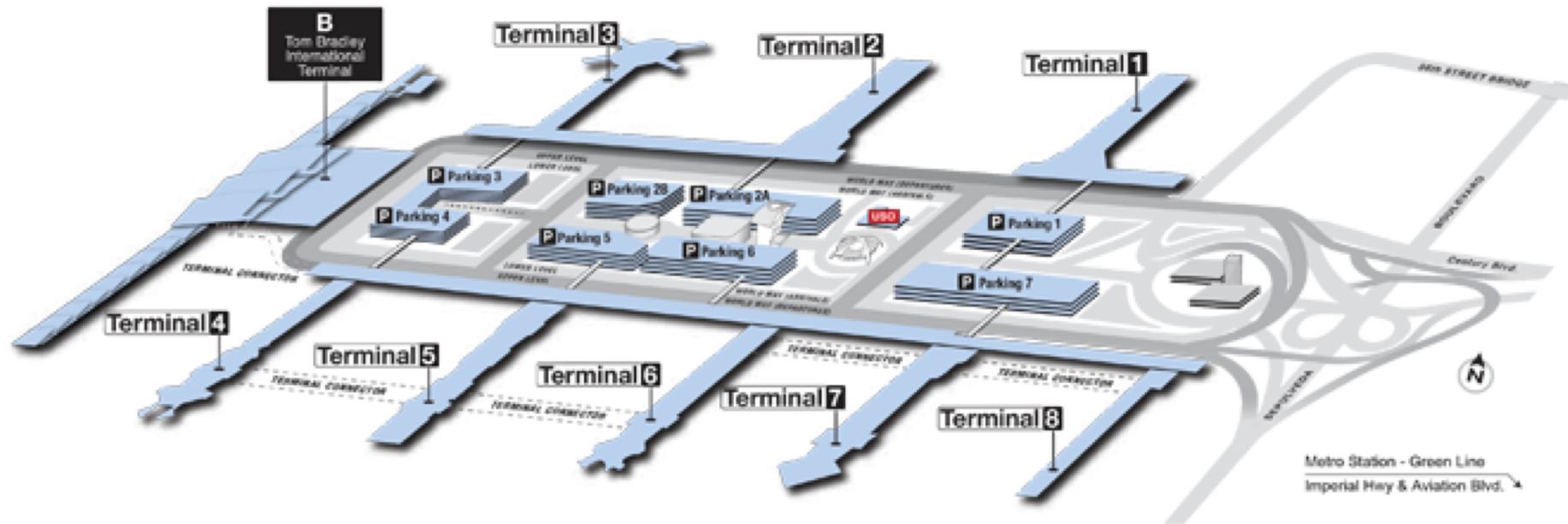
Demo



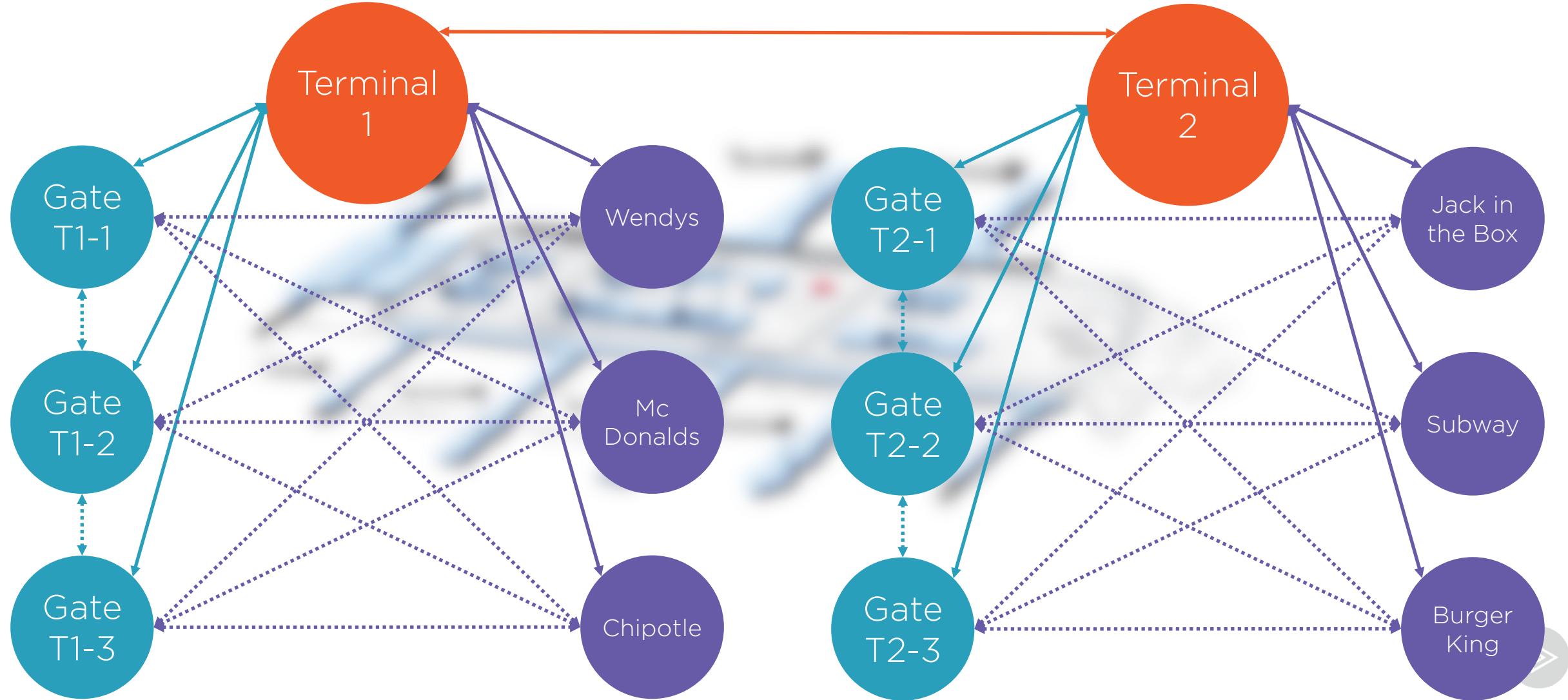
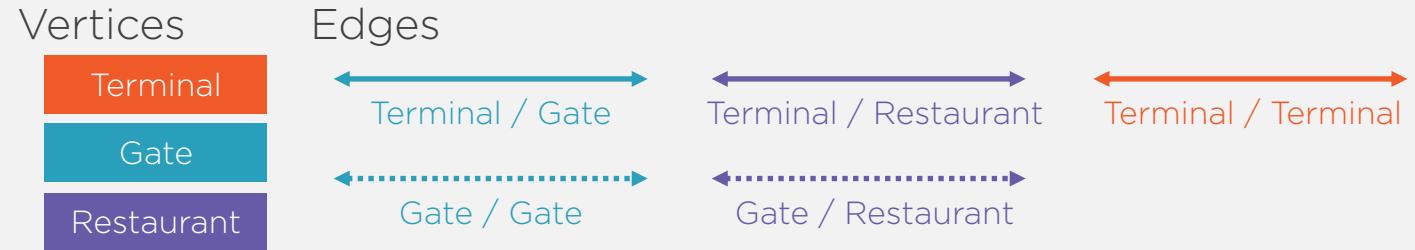
Busy world traveler



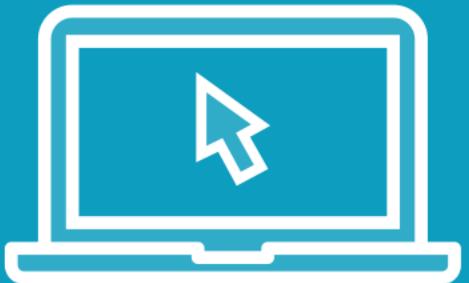
Demo: Busy World Traveler



Demo: Busy World Traveler



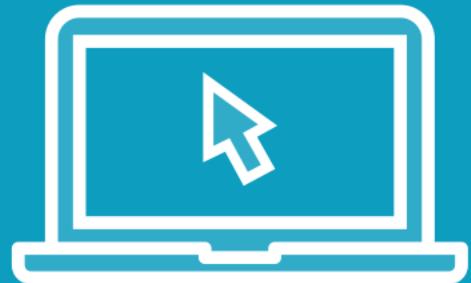
Demo



Populating the airport graph



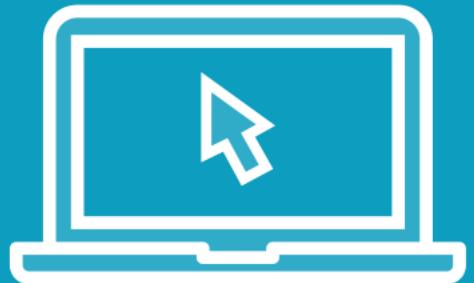
Demo



Querying the airport graph



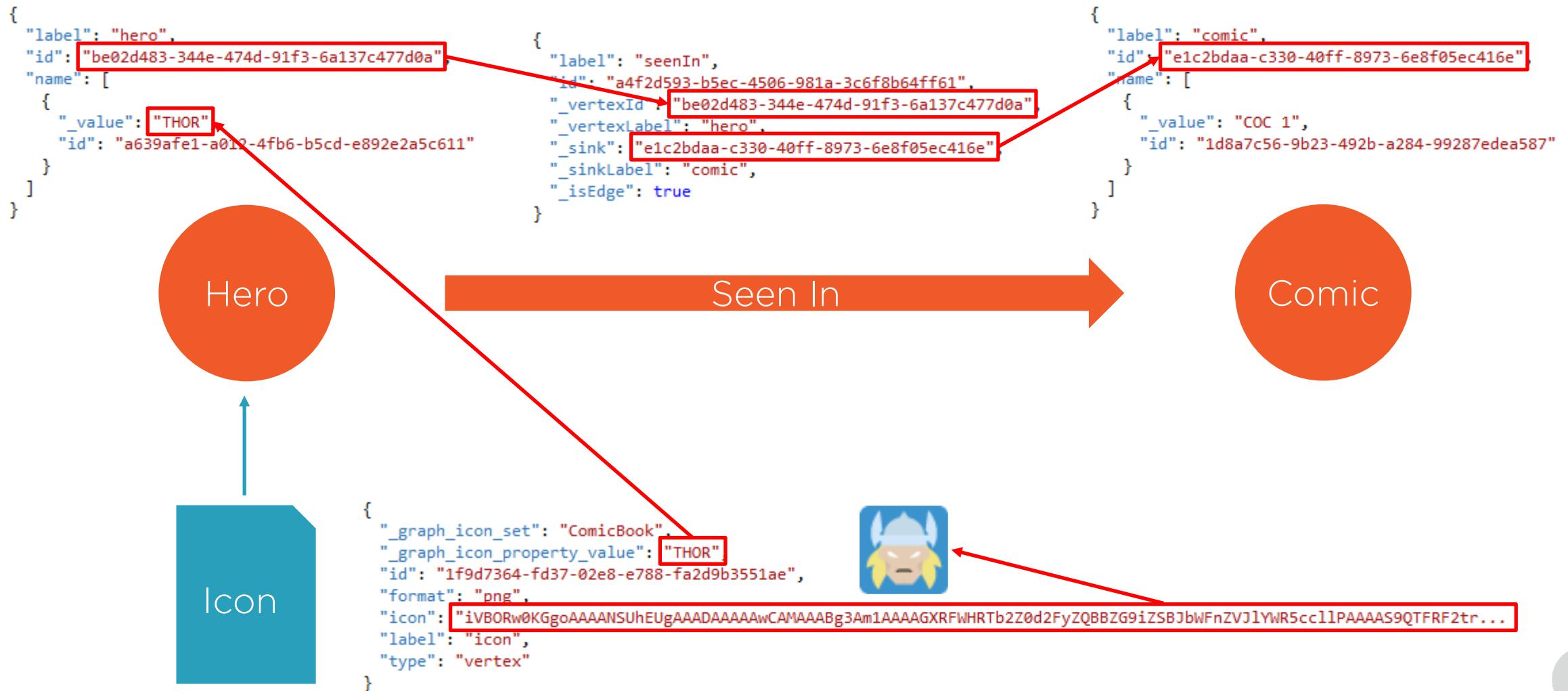
Demo



Multi-model comic book catalog



Multi-Model Comic Book Catalog



Summary



Graph database

- Vertices and edges

Typical scenarios

- Many complex relationships
- Analyze interconnected data

Just another Cosmos DB data model

- Horizontal partitioning
- Provisioned throughput
- Global distribution
- Indexing policies

Apache TinkerPop

- GraphSON
- Gremlin

Mixing Gremlin and SQL APIs



Learning Azure Cosmos DB

Thank You!

