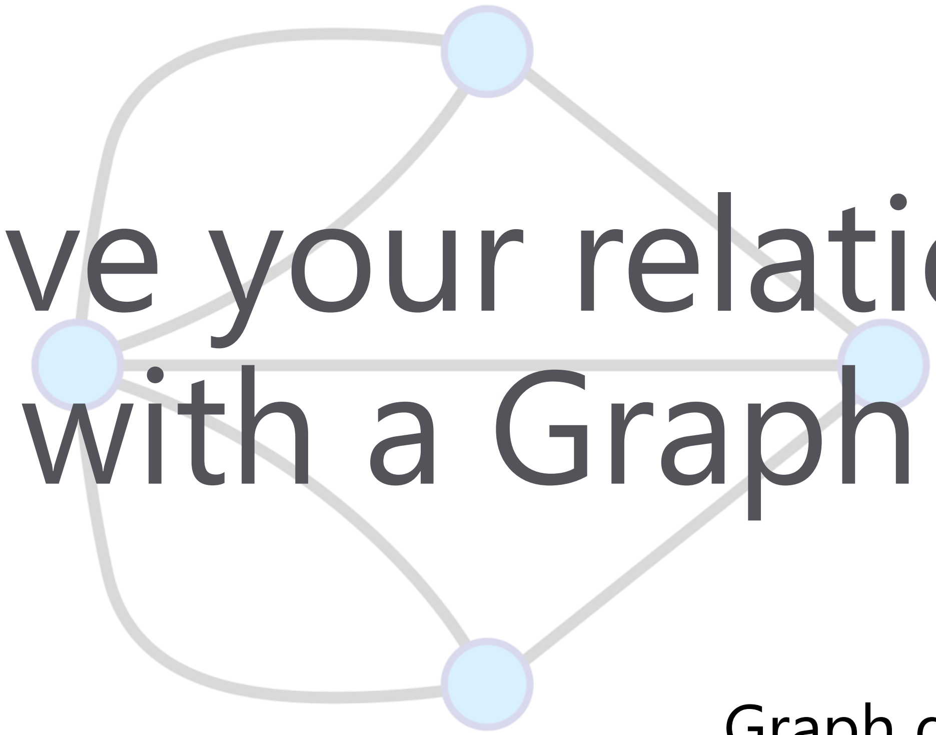
Save your relation with a Graph



Graph databases in .NET

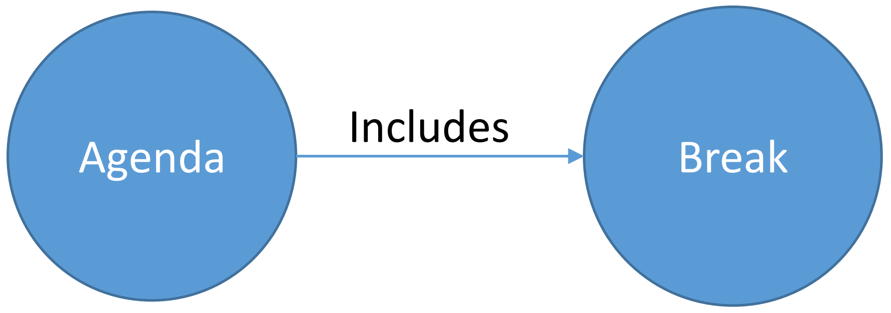
Daniël te Winkel
@daniel_tewinkel
@BetabitNL

Agenda

- Introduction
- Use cases
- Tools and first code

  break

- Modelling
- Graph languages
- CRUD
- Querying

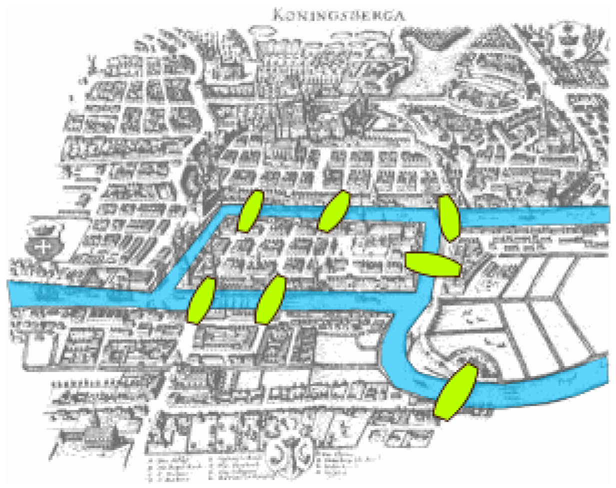


Introduction

Graph theory

First paper by Leonhard Euler (1707-1783)

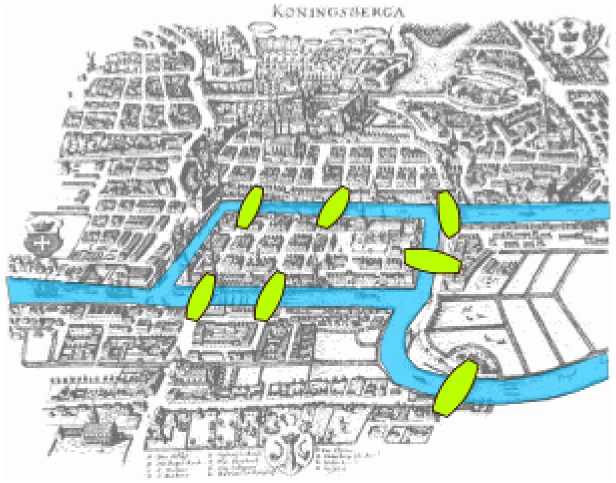
Seven Bridges of Königsberg, published in 1736

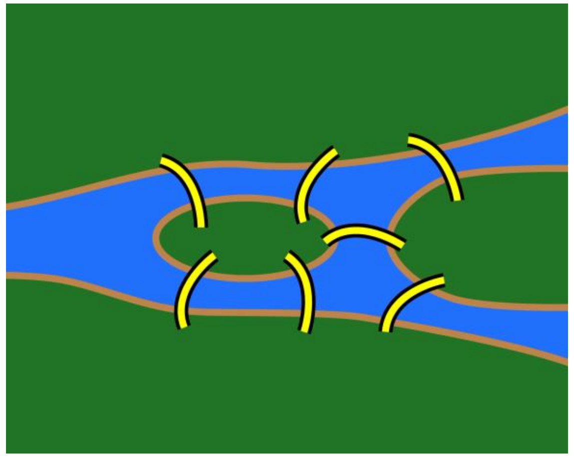


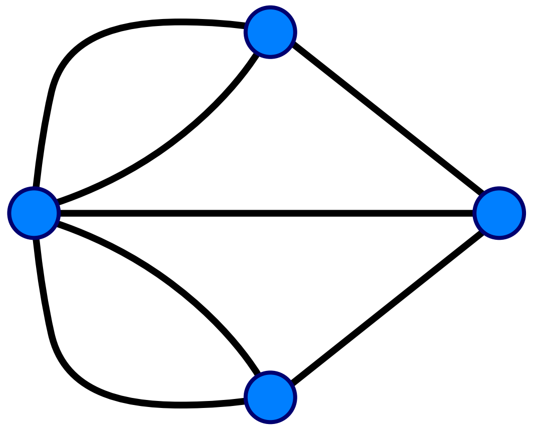
betabit

Seven Bridges of Königsberg

Devise a walk through the city that would cross each of those bridges once and only once.



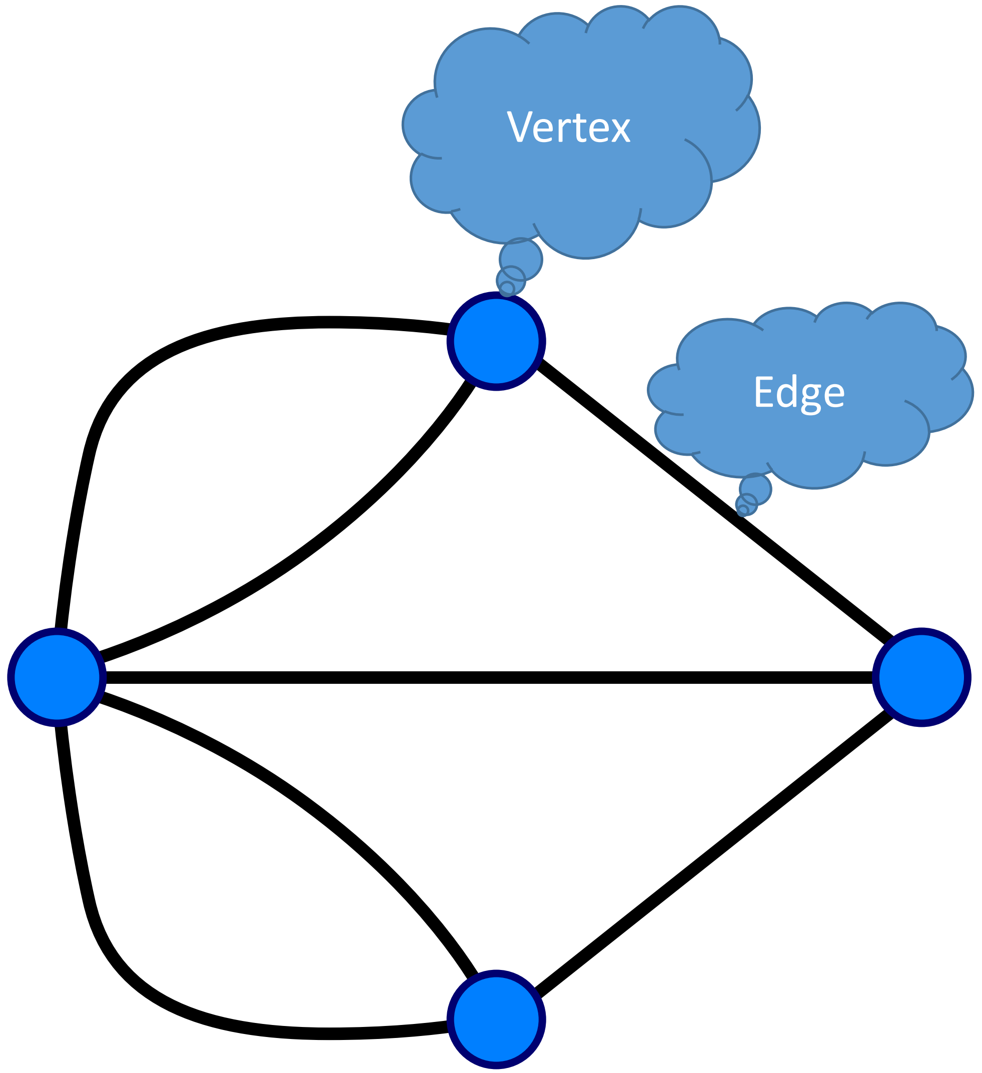




Vertices and Edges

Or nodes and relationships…

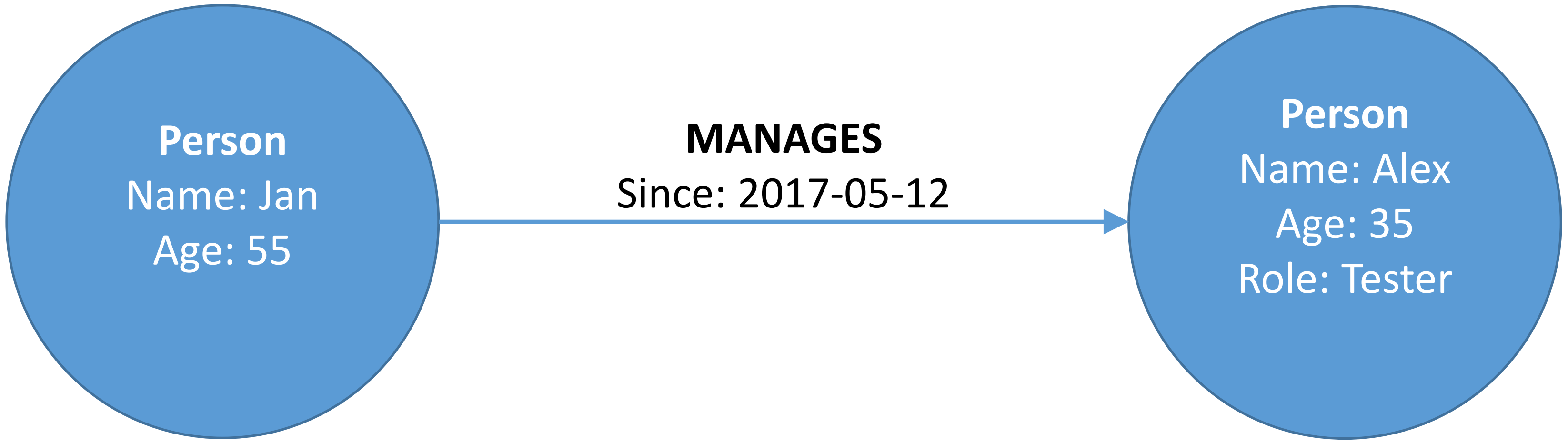
Or nodes and edges…



betabit {…}

Directed Property Graph

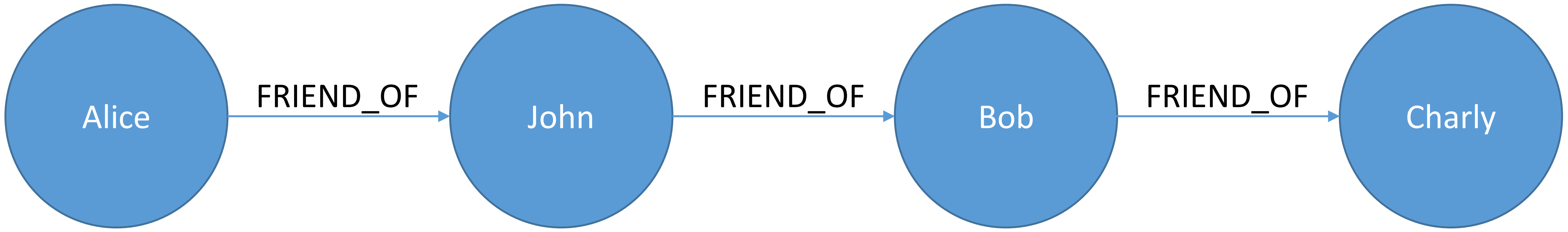
Nodes and relationships can have labels.
Nodes and relationships can have properties.



Graphs are schemaless (NoSQL)

It’s all about the paths

A path is a sequence of nodes connected by relations.



Graph databases

- **Cosmos DB** (Tinkerpop)
- **Neo4j**
- TinkerPop
- Microsoft SQL server 2017
- AllegroGraph
- OrientDB
- Many more…

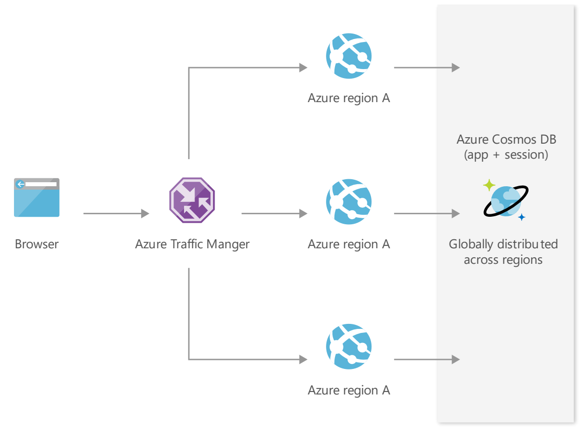
Graph languages

- **Gremlin** (Cosmos DB / Tinkerpop)
- **Cypher** (Neo4j)
- GraphQL
- SPARQL (Allegrograph)
- A few more…

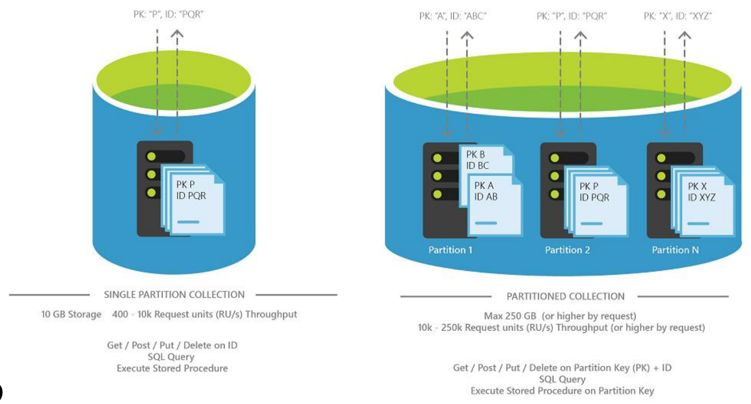
Cosmos DB / Gremlin

Microsoft's multi-model database in Azure

- Globally distributed
- Massive scale
- Guaranteed low latency
- Very high availablity
- Five consistency levels
- Graph model based on Tinkerpop
- RUs / second (~ read of 1KB document)



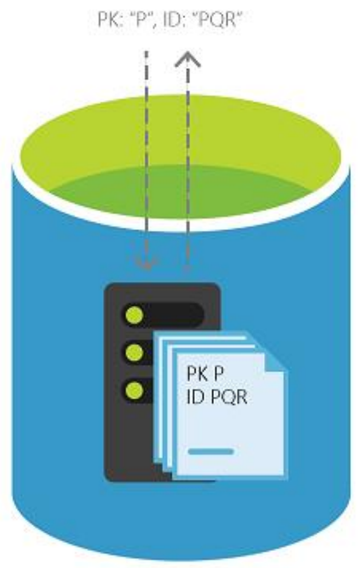
Cosmos DB - Partitioning


PK: "P", ID: "PQR"
PK P
ID PQR
SINGLE PARTITION COLLECTION
10 GB Storage 400 - 10k Request units (RU/s) Throughput
Get / Post / Put / Delete on ID
SQL Query
Execute Stored Procedure
PK: "A", ID: "ABC"
PK: "P", ID: "PQR"
PK: "X", ID: "XYZ"
PK B
ID BC
PK A
ID AB
PK P
ID PQR
PK X
ID XYZ
Partition 1
Partition 2
Partition N
PARTITIONED COLLECTION
Max 250 GB (or higher by request)
10k - 250k Request units (RU/s) Throughput (or higher by request)
Get / Post / Put / Delete on Partition Key (PK) + ID
SQL Query
Execute Stored Procedure on Partition Key

- Partition key optional if collection < 10GB
- Ids for Vertices and Edges must be unique per partition
- Partition key property must be present in each vertex
- Edges are stored in same partition as their out vertex
  - `john.addE('knows').to(mary)`

- PartitionKey property?
- Choose partition key to best query from a single partition?

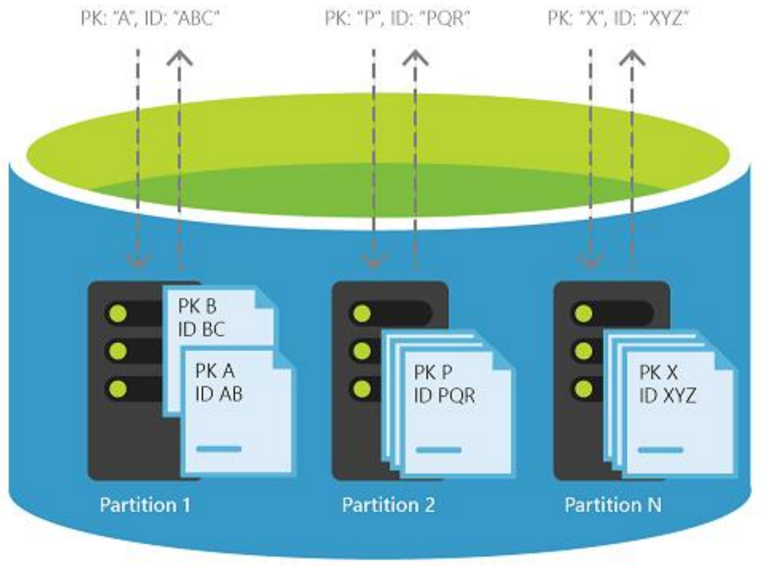
Cosmos DB - Partitioning



SINGLE PARTITION COLLECTION

10 GB Storage 400 - 10k Request units (RU/s) Throughput

Get / Post / Put / Delete on ID
SQL Query
Execute Stored Procedure



PARTITIONED COLLECTION

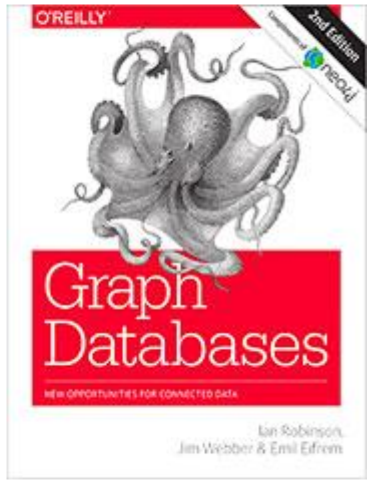
Max 250 GB (or higher by request)
10k - 250k Request units (RU/s) Throughput (or higher by request)

Get / Post / Put / Delete on Partition Key (PK) + ID
SQL Query
Execute Stored Procedure on Partition Key

betabit

Neo4j / Cypher

- (One of) the oldest players in the field
- Native graph storage and processing
- Huge community
- Easy to learn
- Free (e-)book ☺
- Enterprise grade graph database
- Java, C#, Python, Javascript, Ruby, and more


O'REILLY
2nd Edition
neo4j
Graph Databases
NEW OPPORTUNITIES FOR CONNECTED DATA
Ian Robinson, Jim Webber & Emil Eifrem

What about the competition?

Why not use SQL?

- Joining tables for relation can become slow and cumbersome.

Or another NoSQL solution?

- In general, they don’t support relations at all.

They may or may not be a better option for some use cases but not for…

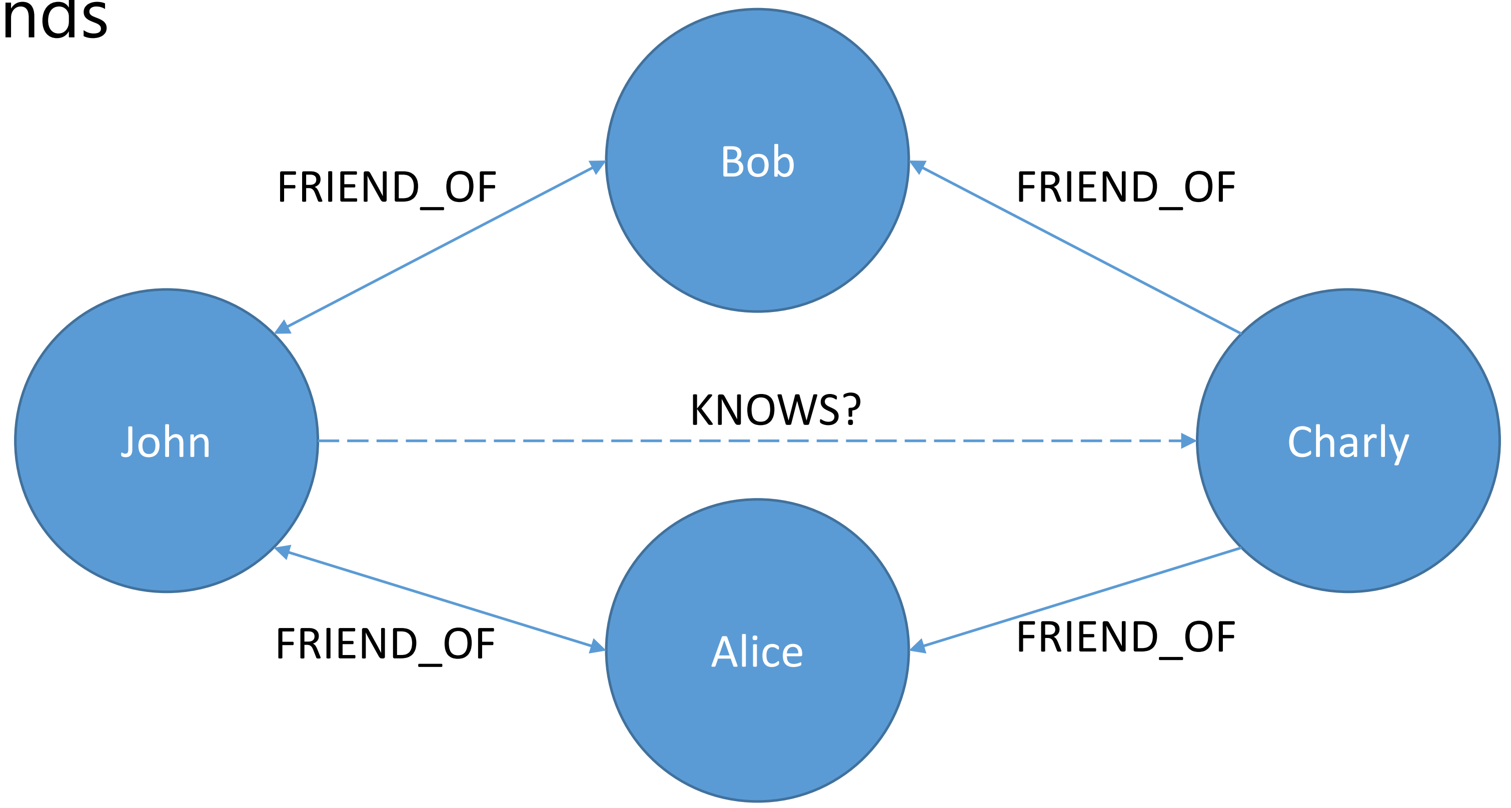
Use Cases

Use cases for Graph databases

- Social data
- Recommendations
- Fraud detection
- Geospatial
- Authorization
- Analytics

Social data

Common friends
Influencers



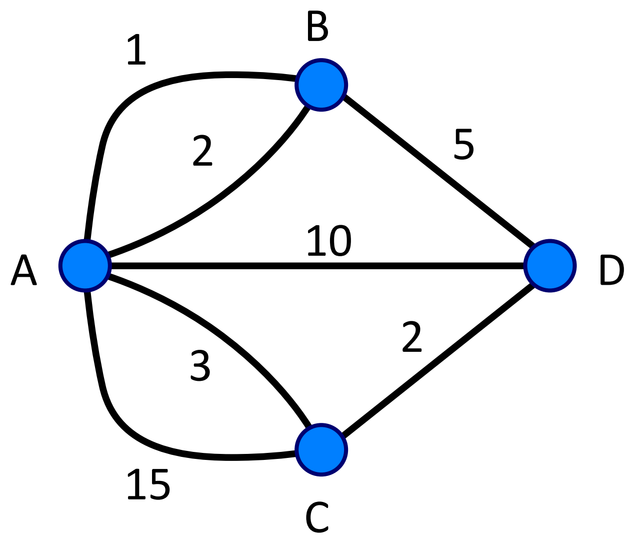
Six degrees of separation?

Geospacial

Eulers problem

Route calculations

- Shortest
- Cheapest

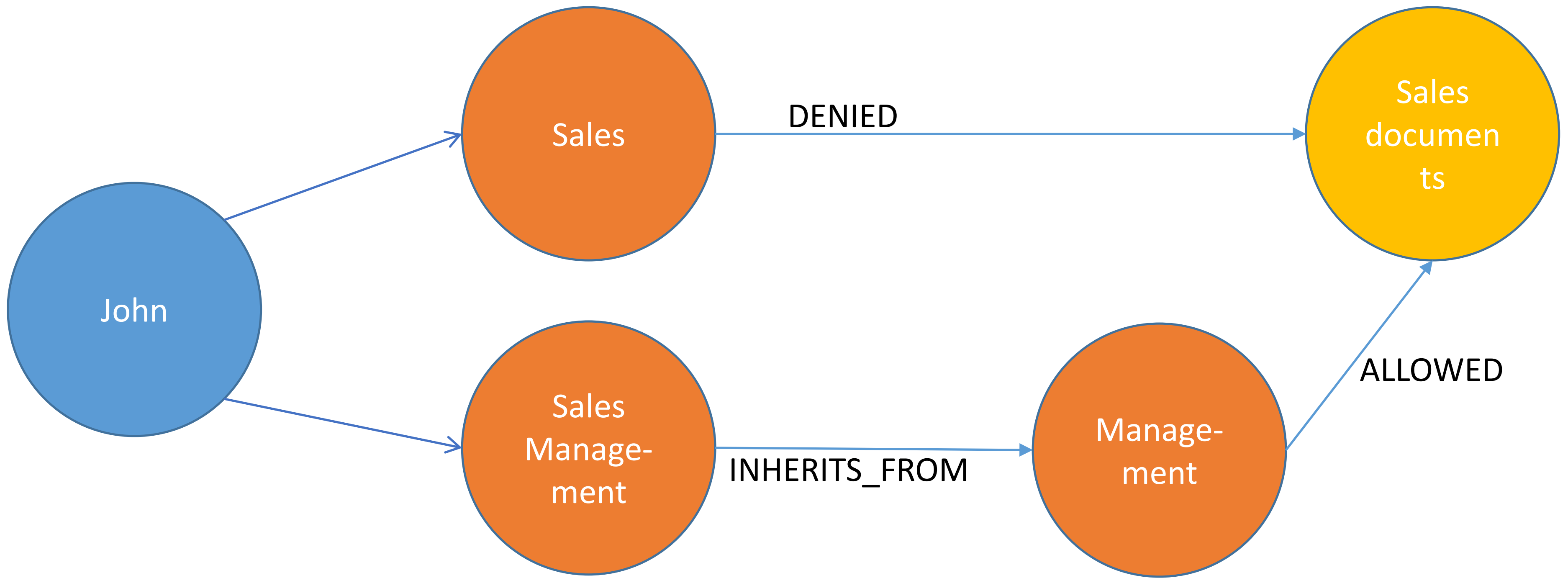


```
MATCH p = (a:Place {name: "A"})-[:CONNECTS_TO*]->(n:Place {name: "D"})
RETURN p AS shortestPath, reduce(cost=0, r in relationships(p) | cost + r.Cost) AS totalCost
ORDER BY totalCost ASC
LIMIT 1
```

betabit

Authorization

Store and verify fine-grained access control

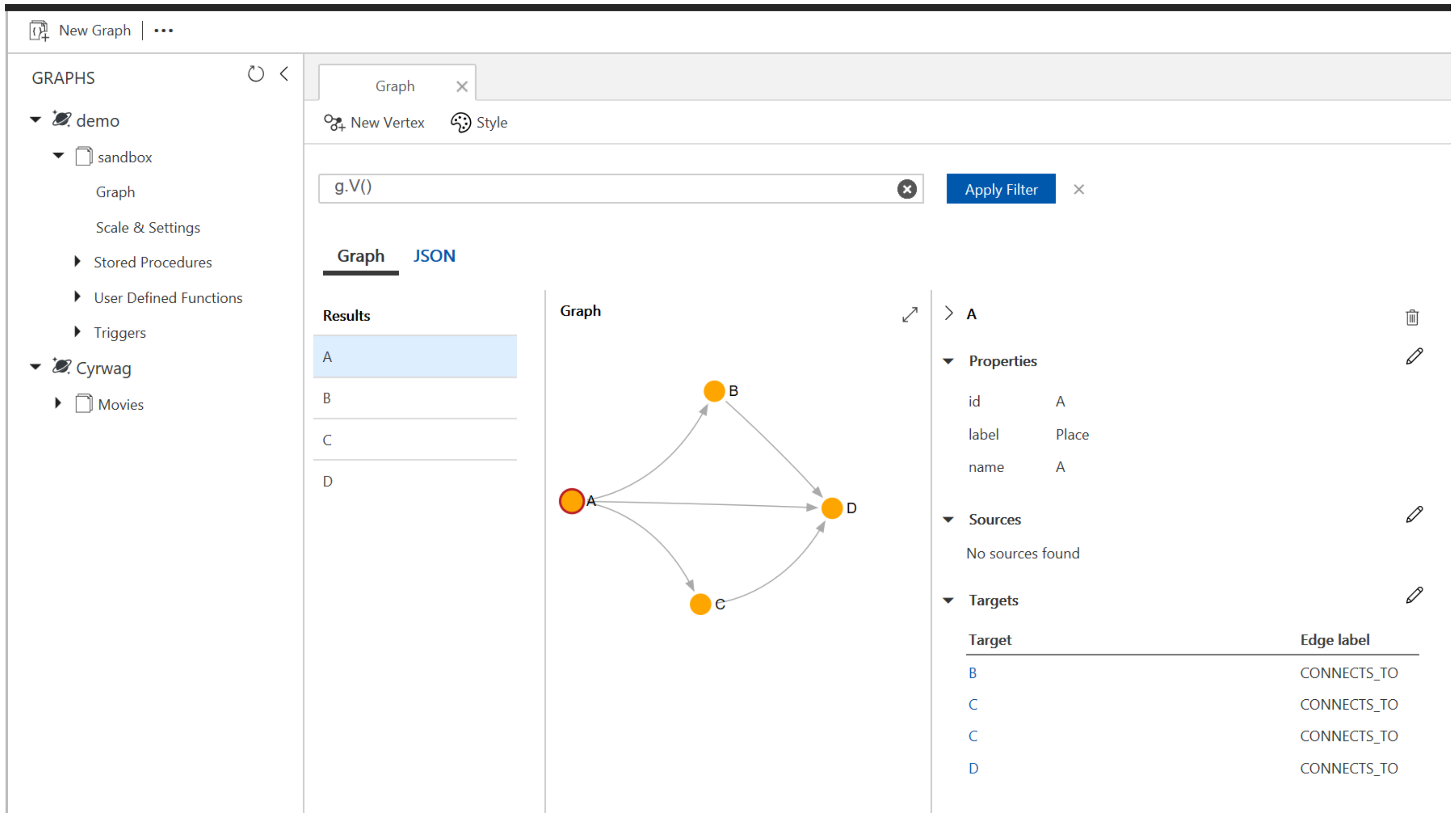


betabit

Tools and first code

Cosmos DB

Azure Portal



New Graph | ...

GRAPHS

- demo
  - sandbox
    - Graph
    - Scale & Settings
    - Stored Procedures
    - User Defined Functions
    - Triggers
- Cyrwag
  - Movies

Graph

New Vertex Style

```
g.V()
```

Apply Filter

Graph JSON

Results

- A
- B
- C
- D

Graph

A

Properties

id	A
label	Place
name	A

Sources

No sources found

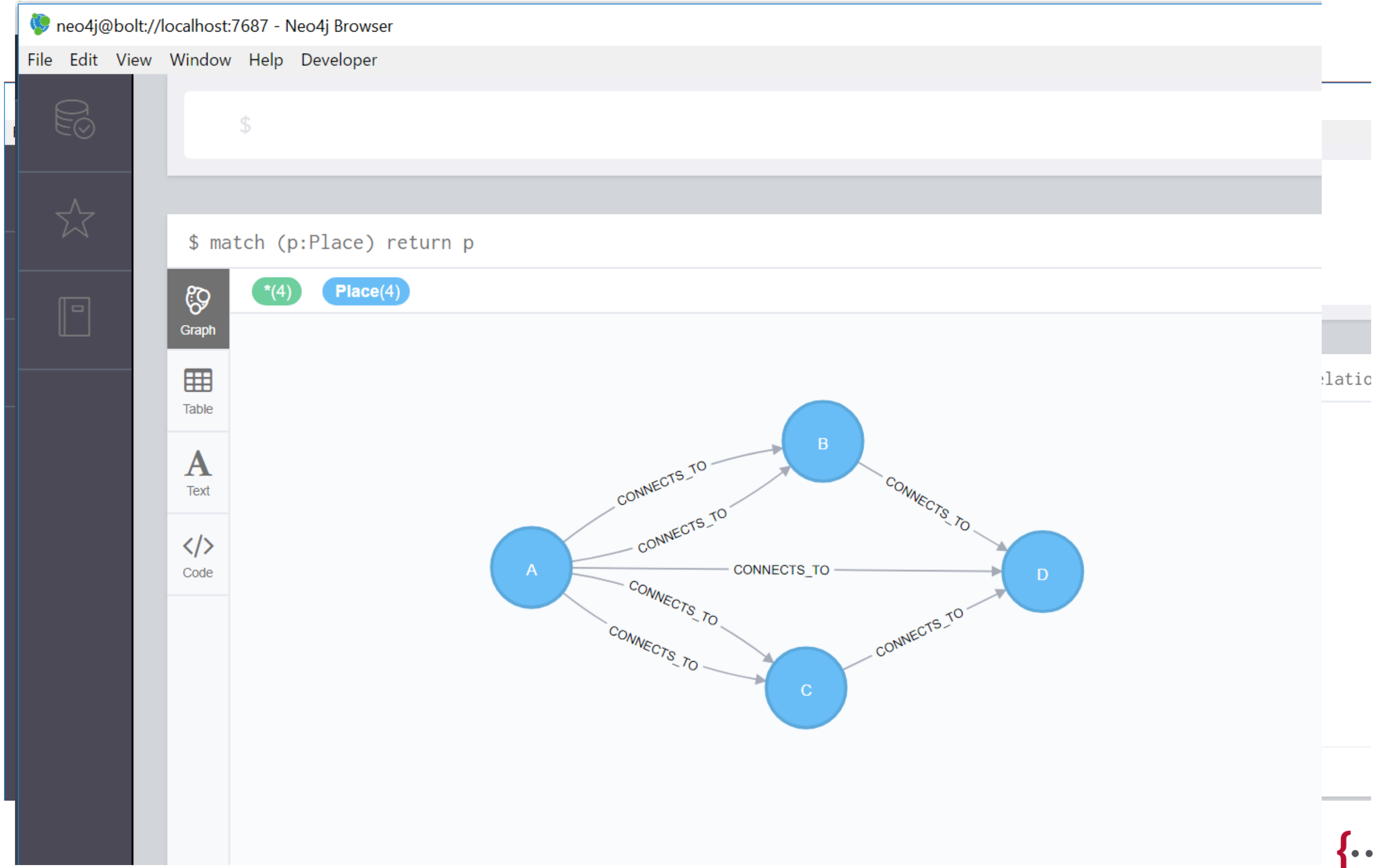
Targets

Target	Edge label
B	CONNECTS_TO
C	CONNECTS_TO
C	CONNECTS_TO
D	CONNECTS_TO

Neo4j

Desktop

Browser



```
$ match (p:Place) return p
```

betabit

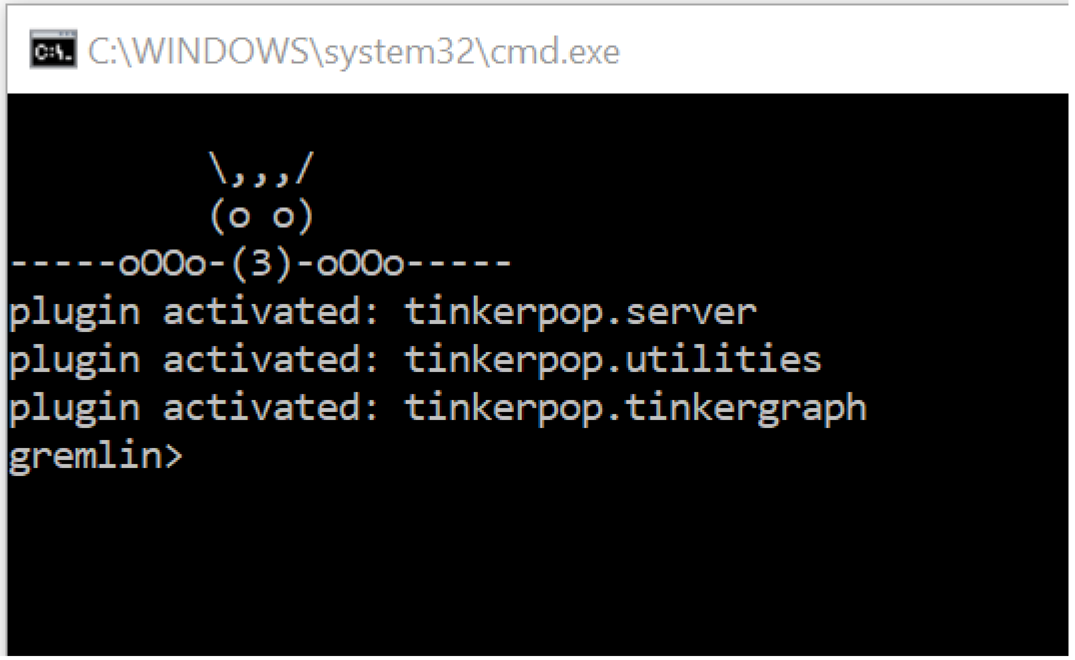
Apache Tinkerpop

Run as local Server

- Not the same as Cosmos DB!

Run local client

- Tekst based
- Can connect to Cosmos DB



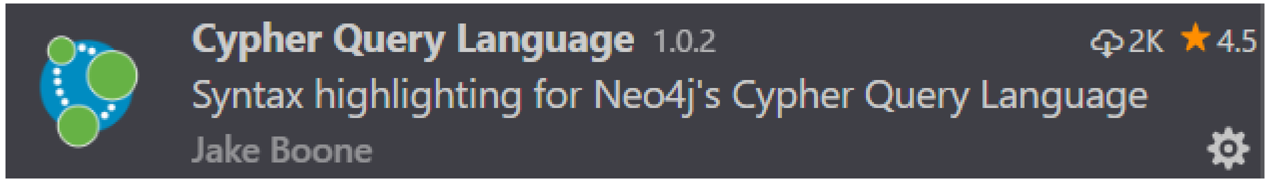
```
         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

betabit

Visual Studio Code

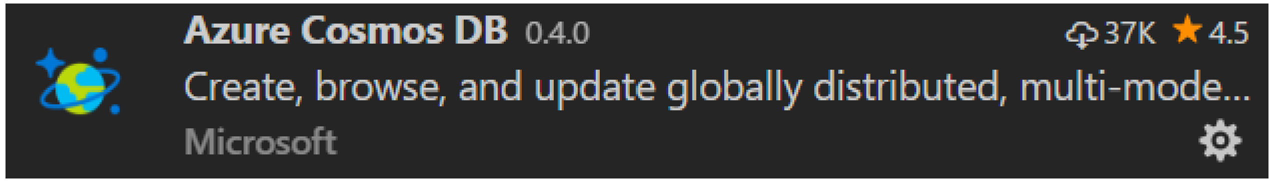
Useful plugins:

- Cypher Query Language



Cypher Query Language 1.0.2 2K ★ 4.5
Syntax highlighting for Neo4j's Cypher Query Language
Jake Boone

- Azure Cosmos DB

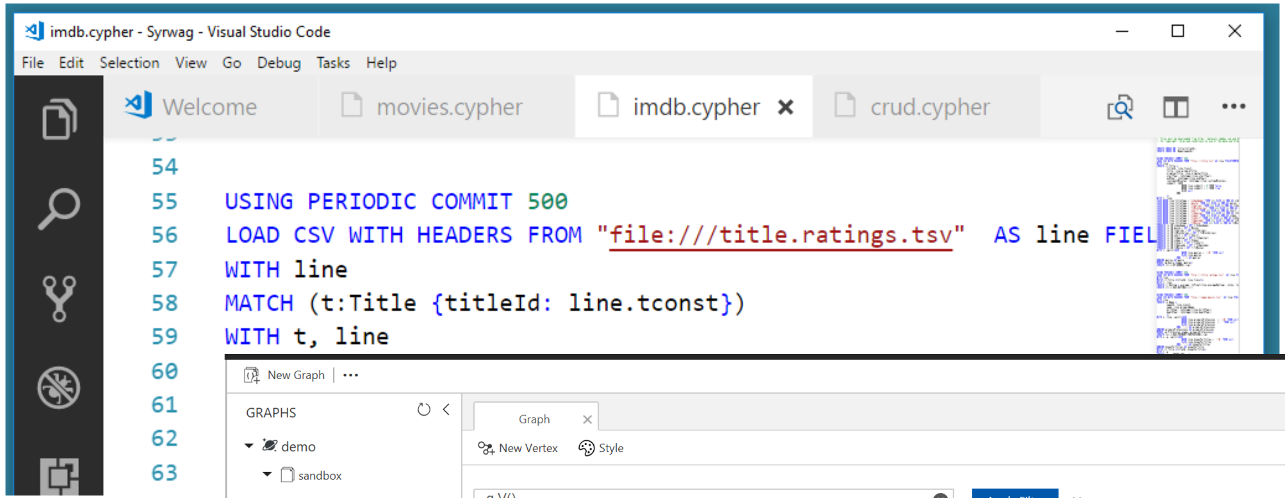


Azure Cosmos DB 0.4.0 37K ★ 4.5
Create, browse, and update globally distributed, multi-mode...
Microsoft

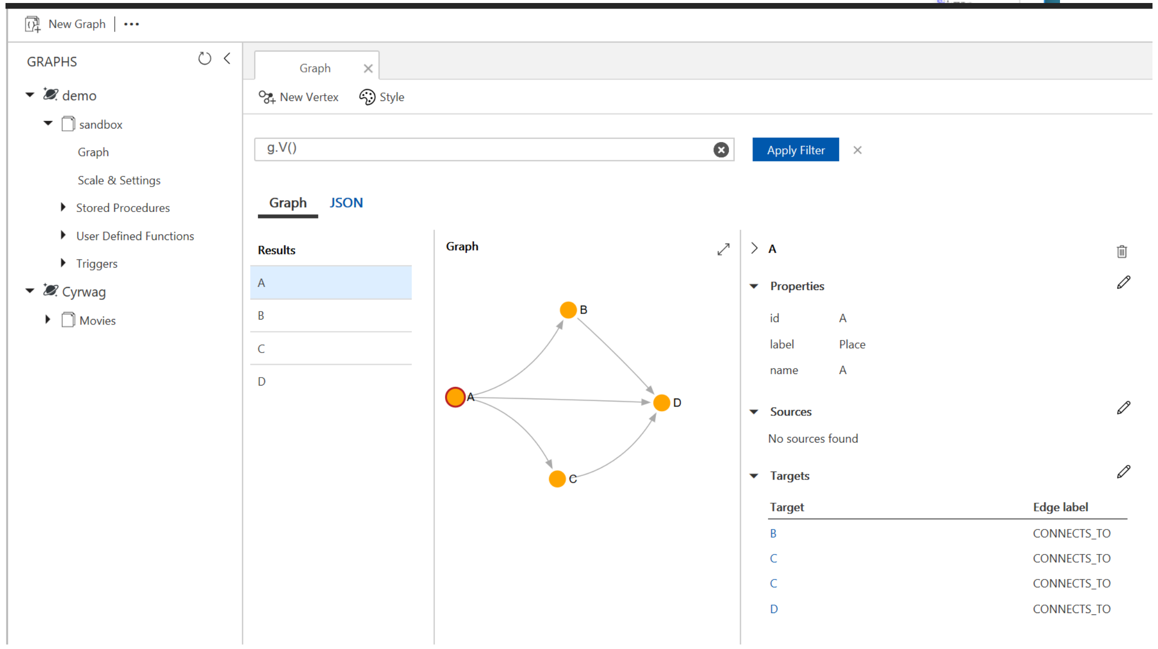
{...}
betabit

Tools demo

And a bit of (query) code


imdb.cypher - Syrwag - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
Welcome
movies.cypher
imdb.cypher
crud.cypher

```
USING PERIODIC COMMIT 500
LOAD CSV WITH HEADERS FROM "file:///title.ratings.tsv"  AS line FIEL
WITH line
MATCH (t:Title {titleId: line.tconst})
WITH t, line
```

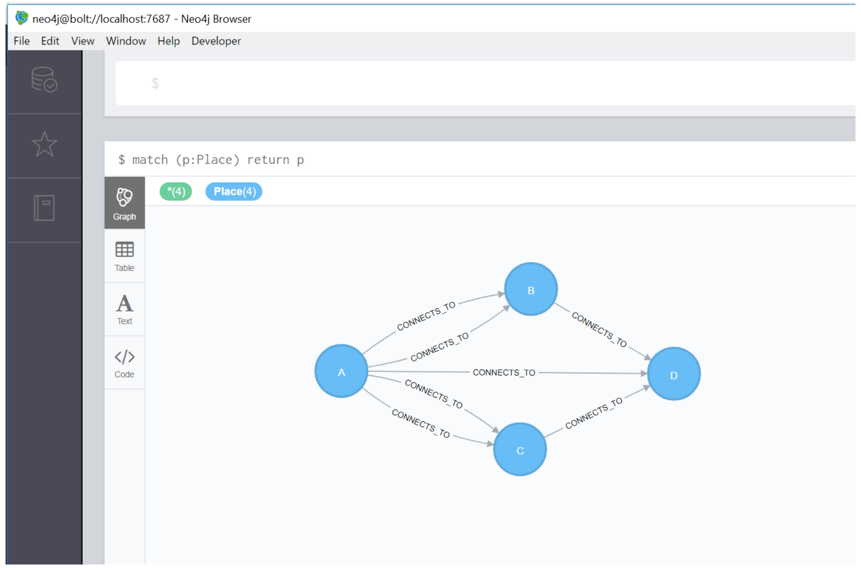


```
g.V()
```

id	A
label	Place
name	A

No sources found

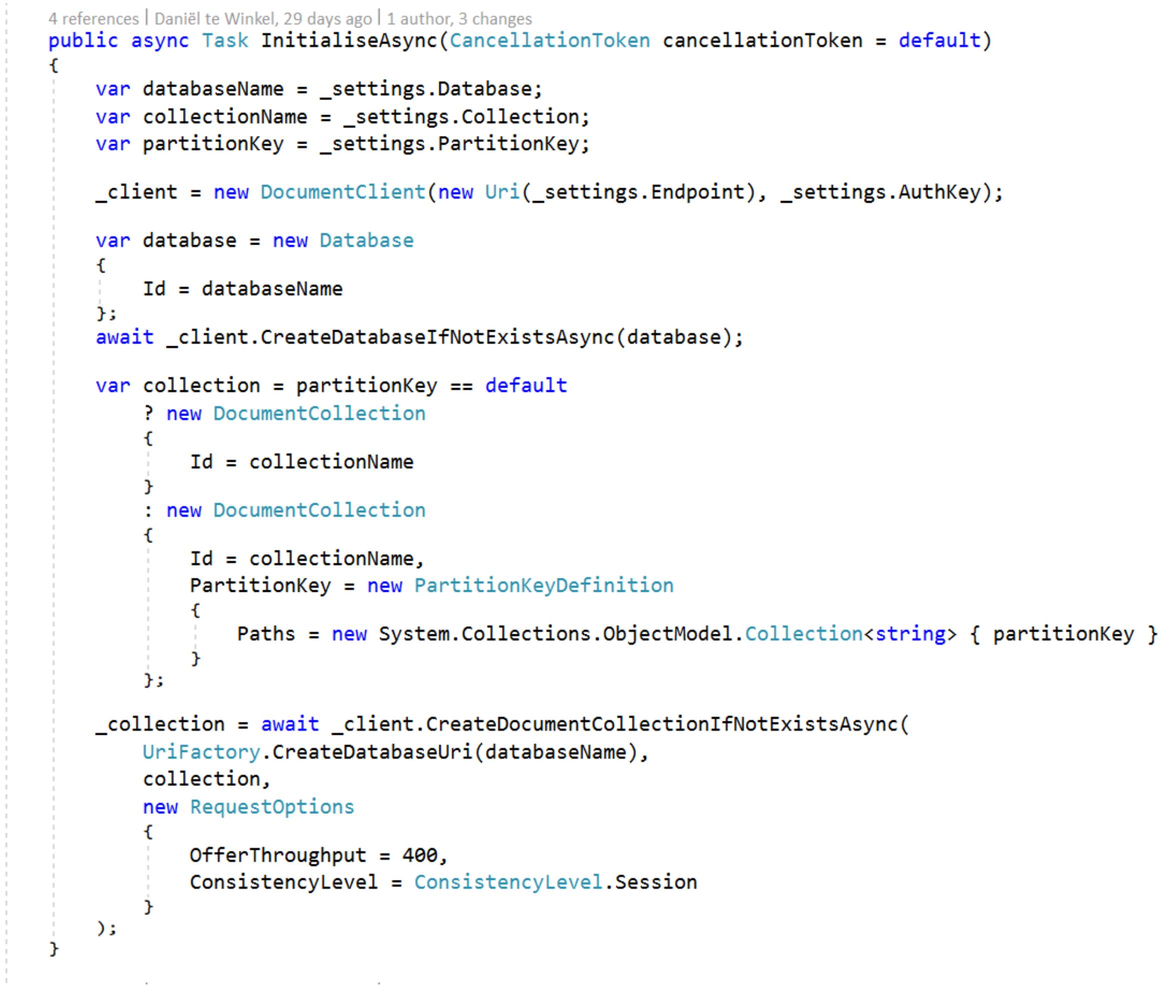
Target	Edge label
B	CONNECTS_TO
C	CONNECTS_TO
C	CONNECTS_TO
D	CONNECTS_TO



```
$ match (p:Place) return p
```

betabit

Code demo – Connecting to the database



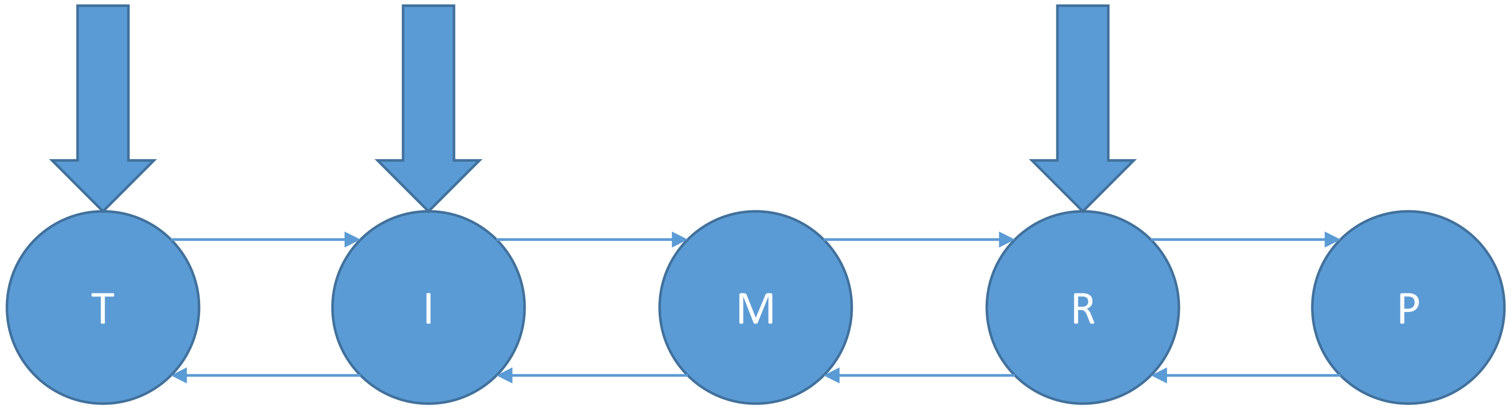
4 references | Daniël te Winkel, 29 days ago | 1 author, 3 changes

```
public async Task InitialiseAsync(CancellationToken cancellationToken = default)
{
    var databaseName = _settings.Database;
    var collectionName = _settings.Collection;
    var partitionKey = _settings.PartitionKey;

    _client = new DocumentClient(new Uri(_settings.Endpoint), _settings.AuthKey);

    var database = new Database
    {
        Id = databaseName
    };
    await _client.CreateDatabaseIfNotExistsAsync(database);

    var collection = partitionKey == default
        ? new DocumentCollection
        {
            Id = collectionName
        }
        : new DocumentCollection
        {
            Id = collectionName,
            PartitionKey = new PartitionKeyDefinition
            {
                Paths = new System.Collections.ObjectModel.Collection<string> { partitionKey }
            }
        };

    _collection = await _client.CreateDocumentCollectionIfNotExistsAsync(
        UriFactory.CreateDatabaseUri(databaseName),
        collection,
        new RequestOptions
        {
            OfferThroughput = 400,
            ConsistencyLevel = ConsistencyLevel.Session
        }
    );
}
```

Break

(Daniël)-[Needs]->(Coffee)

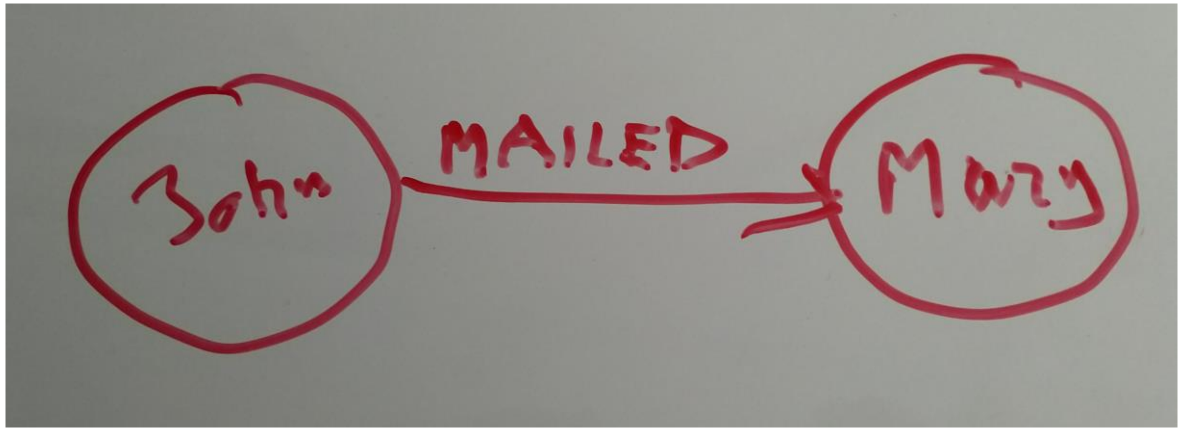
Some magic?



Modelling

Start with a whiteboard!

- Graphs are natural to draw; just circles and arrows.
- Use real examples
- Don't try to put too much detail in



Test in a database

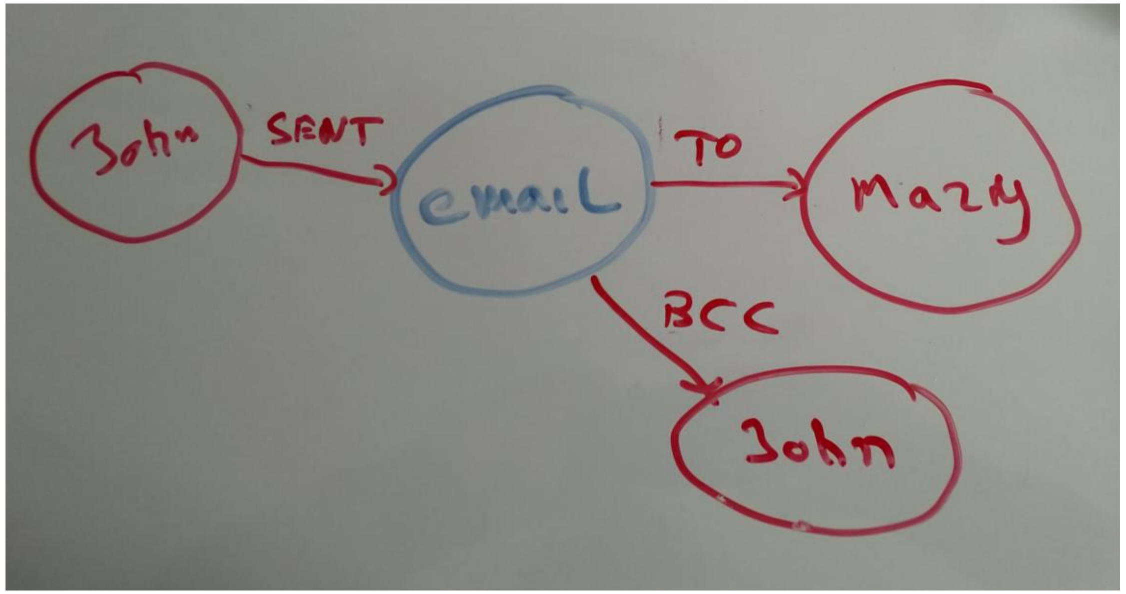
Enter the examples in a database.
Test with queries that the business requires.

```
john = g.addV('Person').property('name', 'John')
mary = g.addV('Person').property('name', 'Mary')
john.addE('MAILED').to(mary)
```

Step by step

Next increment:

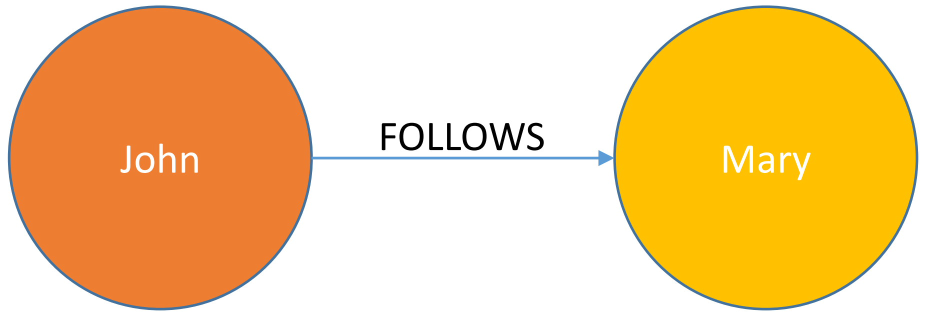
- On whiteboard
- In database
- Test queries



What is what

Nodes should describe **things**

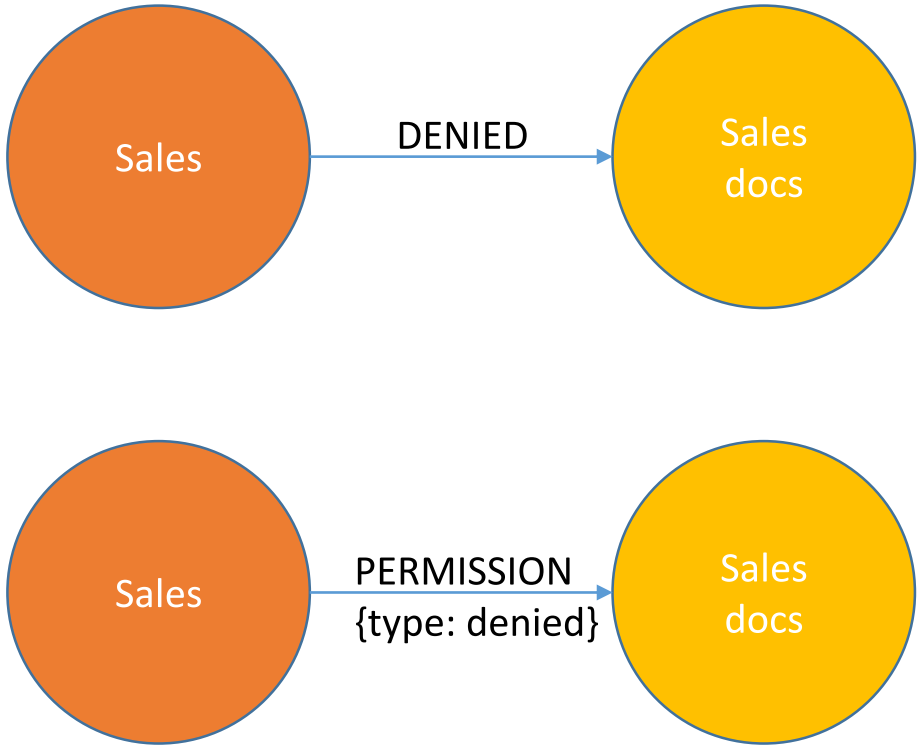
Relations should describe the **relation** between the things



betabit

Labels or properties

Using specialised relationships will be faster to query



Graph languages

Gremlin

- Developed by Apache

- Functional language
- Multiple values possible for a Vertex property
- Many Dialects
  - Gremlin-Java8, Gremlin-Groovy, Gremlin-Python, e.t.c.

Cypher

- Created by Neo4j
- Open sourced as openCypher

- Graph query language
- Declarative pattern matching
- Similarities to SQL

(Neo4j)-[created]->(Cypher)

My first query

Return all vertices.

```
Cypher:  match (n) return n
Gremlin: g.V()
```

Return all vertices with a given name.

```
Cypher:  match (n {name: “john”}) return n
Gremlin: g.V().has(‘name’, ‘john’)
```

betabit

CRUD

CRUD

- Some Gremlin
- A bit more Cypher
- And some C#

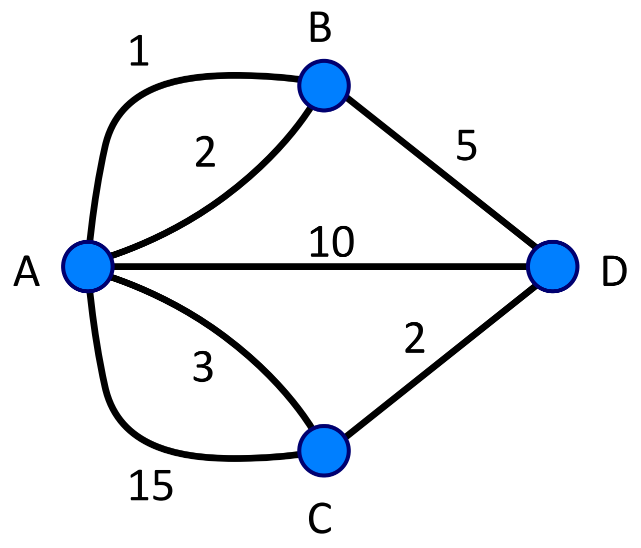
```
// Extend the model
MATCH (B:Place {name: "B"})
MATCH (C:Place {name: "C"})
MATCH (D:Place {name: "D"})
CREATE (D)-[:CONNECTS_TO {Cost: 5}]->(B)
CREATE (D)-[:CONNECTS_TO {Cost: 2}]->(C)
```

```
query = query
    .Create($"(m{pos}:Movie {{entity{pos}}})")
    .WithParam($"entity{pos}", movieEntity);
```

Querying

Querying

```
(Daniël)-[Shows]->(Some queries)
```



The end?

Hands-on session?

If you are interested in a hands-on session to explore the (IMDB?) Graph database technology, please respond to the feedback mail

Questions?

betabit{...}

Conclusions

Graph databases are great at querying **connected** data.

Graph databases are a great tool to have in your data toolbox!

betabit

More information

Links to the presentation and the code will be shared at:

https://www.betabit.nl/nl/kennis/save-your-relation-with-a-graph

betabit

Thank you for your time and attention