Azure Cosmos DB

45389618

Basic Characteristics

- Microsoft: „A globally distributed, massively scalable, multi-model database service“
- Multi-model: NoSQL (SQL) API, MongoDB API, PostgreSQL API, Cassandra API, **Gremlin (Graph) API**, Azure Table API
  - How? JSON! And RESTful API
  - Different data models (layers) are just specific JSON Schema within JSON SQL
- Globally distributed: Multi-Region -> many datacenters, Ultra-fast data replication, Regional failover
- Massively scalable: Partitioning using chosen Partition Key
  - All Items in a partition are co-located for speed
  - CosmosDB automatically scales out partitions

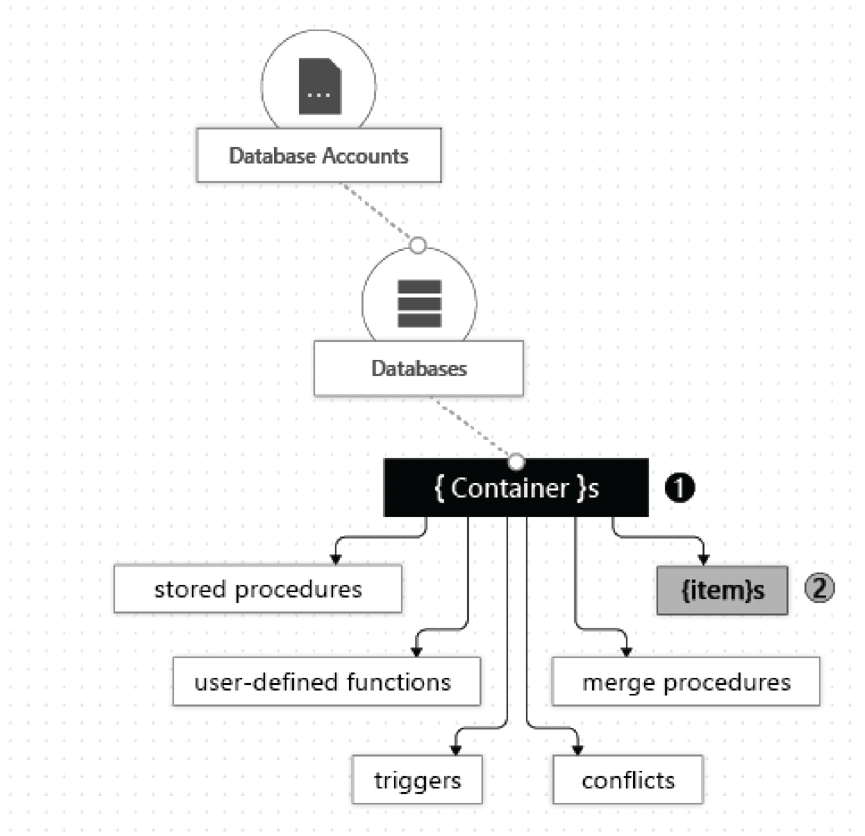
Installation & Setup

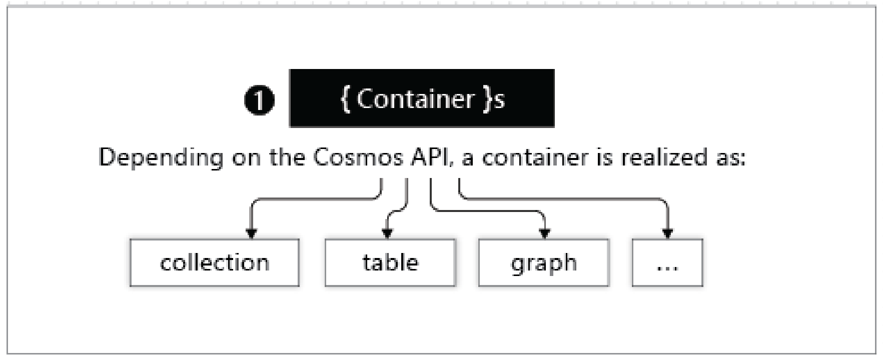
- Requirements:
  - Microsoft (Azure) personal account
  - Free Trial: https://cosmos.azure.com/try/
- Choose „Azure Cosmos DB for Apache Gremlin“ Graph API
- Check out tutorial: https://learn.microsoft.com/en-us/azure/cosmos-db/gremlin/quickstart-console
  - Database ID: ndbi040-project
  - Graph ID: Webcrawler
  - Partition Key: /recordId
- Optional: Download G.V() – Gremlin IDE & Visualisation tool for Cosmos DB

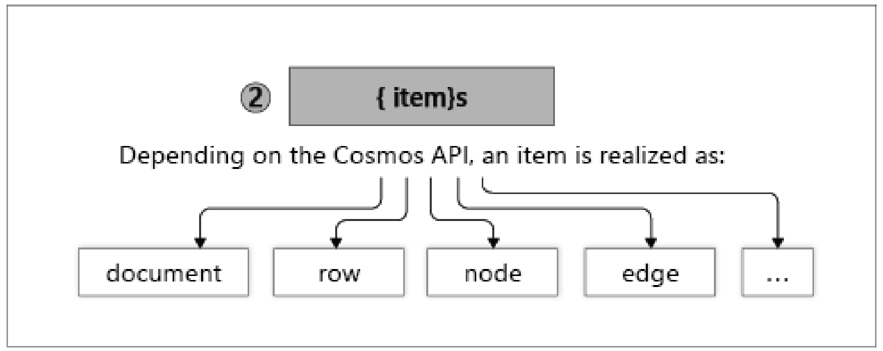
Data Models – API Layers

- NoSQL API - Database of Collections of Documents
  - Native SQL queries over JSON documents
  - New features priority
  - Recommended way to use Cosmos DB for new projects
- MongoDB API
  - BSON Document format
  - Simple migration to Cosmos DB by just changing a connection string
- PostgreSQL API
  - Single-Node or Multi-Node using Citus open source
- Apache Cassandra API – Column-oriented schema
- **Apache Gremlin API - Graph API (showcased in this presentation)**
- Table (Key-Value) API - Database of Tables, with Rows
- https://learn.microsoft.com/en-US/azure/cosmos-db/choose-api

Cosmos DB Entities







https://learn.microsoft.com/en-us/azure/cosmos-db/resource-model

Apache Gremlin

- Apache TinkerPop Standard
- Vertices „G.V()“ & Edges „G.E()“

```
g.V('id').out('rel').has('attr')
```

Labeled Property Graphs

Gremlin

$G = (V, E)$

- Vertices (Nodes) & Edges (Relationships)
- Nodes & Relationships can contain Properties = Mutable Key/Value pairs

Property	Data Type	Details
ID	String	Unique identifier. Auto-generated if not provided
Label	String	Entity type identifier. Query filtering clause.
Partition Key	Integer, String	Horizontal data distribution key. Query filtering clause.

Luis

vertex
id: Luis
label: Person
properties:
- age: 27

edge
id: edgeId
label: WORKS_AT
properties:
- distance: 10miles

Cosmos DB Gremlin API

- Only fully-managed horizontally-scalable PaaS graph database service.
- Graph engine based on Apache Tinkerpop standard
  - Adapted to a horizontally-scalable distributed storage system
  - Extended with Cosmos DB specific features: resource throttling, retry logic, and distributed runtime diagnostics.
- Wire-protocol compatibility with OSS connectors and libraries.
- Flat data, modelled within SQL API
  - Can use SQL API to add additional data to graph collection (e.g. _graph_icon_)
- Not all Apache Tinkerpop Gremlin functionalities are supported: https://docs.microsoft.com/en-us/azure/cosmos-db/gremlin-support

Apache TinkerPop Compatibility

Platform	Source	Getting Started	Version
.NET	Gremlin.NET on GitHub	Create Graph using .NET	3.4.0-RC2
Java	Gremlin JavaDoc	Create Graph using Java	3.2.0+
Node.js	Gremlin-JavaScript on GitHub	Create Graph using Node.js	3.3.4+
Python	Gremlin-Python on GitHub	Create Graph using Python	3.2.7
PHP	Gremlin-PHP on GitHub	Create Graph using PHP	3.1.0
Gremlin console	TinkerPop docs	Create Graph using Gremlin Console	3.2.0 +

Cosmos DB supported Gremlin steps

- CRUD
- Aggregation
- Computation
- Projection
- Cosmos DB specific steps
  - Partitioning Strategy
  - Execution Profile

DML Operations

- Create vertices and add properties

```
g.addV(<label>).property([...])
```

- Create edges with to() or from()

```
g.V(<id>).addE(<label>).to(g.V(<id>))
```

- Add/modify properties for vertices or edges

```
g.V(<id>).property(<key>, <value>)
```

Point-Lookups

- Vertex point-lookup by ID

```
g.V(<id>)
```

- Edges point-lookup by ID

```
g.E(<id>)
```

- Vertex point-lookup by ID and partition key

```
g.V([<pk>,<id>])
```

Filtering

- Property filtering using .has()

```
g.V().has('name', 'Luis')
```

- Traversal condition filtering using .where()

```
g.V(<id>).outE(<id>).
where(inV().has('distance', '3'))
```

- Comparison parameters

```
g.V().has('distance', gt(300))
```

Filtering

- Result set filtering using .range() or .limit()

```
g.V().limit(100)

g.V().range(1, 100)
```

Traversal

- Adjacency exploration .out()

```
g.V(<id>).out().has('name', 'Luis')
```

- The .repeat() function with .until() predicate.

```
g.V(<id>).repeat(out()).
until(has('name', 'Luis'))
```

- The .repeat() function with .times() predicate.

```
g.V(<id>).repeat(out()).times(8))
```

Aggregation

- Terminal step used for aggregation: .sum(), .avg(), .mean(), .count(), .min(), .max(), …

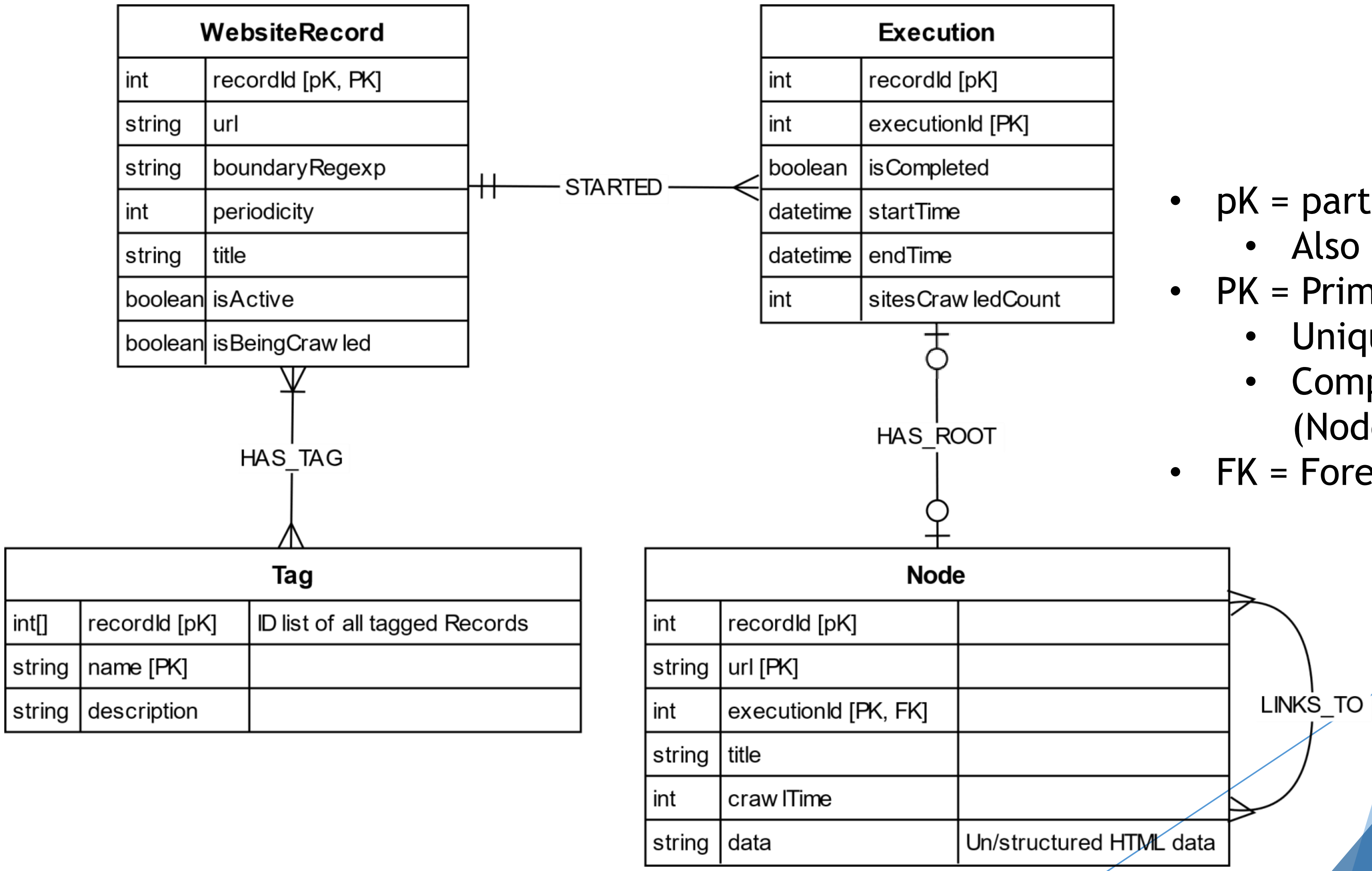
```
g.V(<id>).sum() | .avg() | group().by() ...
```

Demo: Webcrawler

Webcrawler Data Domain using Cosmos DB for Apache Gremlin API

Demo Repository: https://github.com/corovcam/NDBI040-CosmosDB-Presentation

ER Schema



WebsiteRecord	
int	recordId [pK, PK]
string	url
string	boundaryRegexp
int	periodicity
string	title
boolean	isActive
boolean	isBeingCraw led

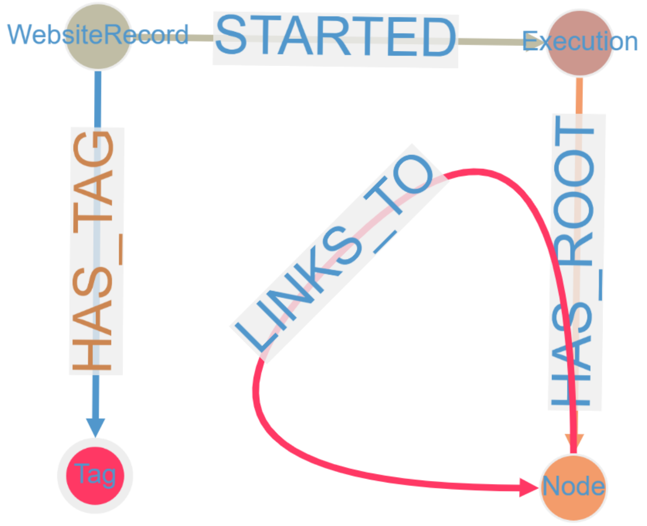
Execution	
int	recordId [pK]
int	executionId [PK]
boolean	isCompleted
datetime	startTime
datetime	endTime
int	sitesCraw ledCount

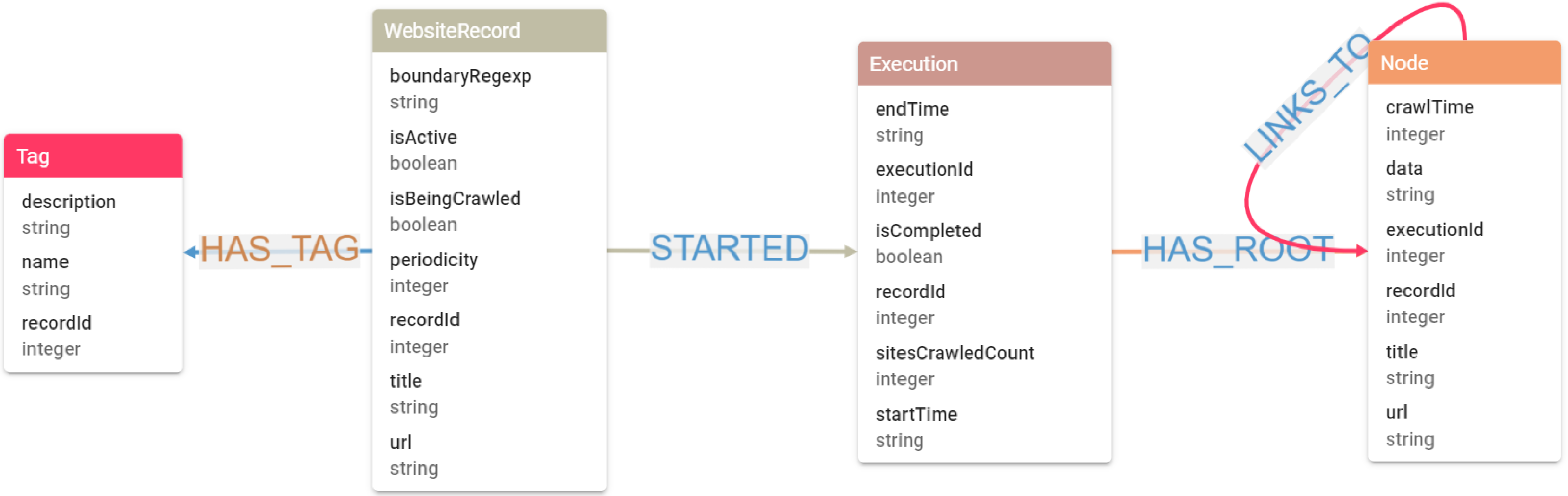
Tag		
int[]	recordId [pK]	ID list of all tagged Records
string	name [PK]	
string	description	

Node		
int	recordId [pK]	
string	url [PK]	
int	executionId [PK, FK]	
string	title	
int	craw lTime	
string	data	Un/structured HTML data

- pK = partitionKey
  - Also a FK
- PK = Primary Key
  - Unique
  - Composite (Node)
- FK = Foreign Key

Logical (Database) Schema





Tag

description	string
name	string
recordId	integer

WebsiteRecord

boundaryRegexp	string
isActive	boolean
isBeingCrawled	boolean
periodicity	integer
recordId	integer
title	string
url	string

Execution

endTime	string
executionId	integer
isCompleted	boolean
recordId	integer
sitesCrawledCount	integer
startTime	string

Node

crawlTime	integer
data	string
executionId	integer
recordId	integer
title	string
url	string

Indexing & Primary/Secondary Keys

- https://learn.microsoft.com/en-us/azure/cosmos-db/index-overview
- „By default, Azure Cosmos DB automatically indexes every property for all items in your container without having to define any schema or configure secondary indexes.“
- For our Webcrawler demo only 1 Composite Secondary Key is defined:
  - Node: [url, executionId]
- Most of Node retrieval operations will use combination of [url, executionId] because that uniquely identifies each Node

Indexing Policy

- Simple JSON configuration using: Azure Web Interface -> Graph -> Settings

```
{
    "indexingMode": "consistent",
    "automatic": true,
    "includedPaths": [
        {
            "path": "/*"
        }
    ],
    "compositeIndexes": [
        [
            {
                "path": "/url",
                "order": "ascending"
            },
            {
                "path": "/executionId",
                "order": "ascending"
            }
        ],
```

```
        [
            {
                "path": "/url",
                "order": "descending"
            },
            {
                "path": "/executionId",
                "order": "descending"
            }
        ]
    ]
}
```

Sample Data

- Check out **scripts/data.groovy**: https://github.com/corovcam/NDBI040-CosmosDB-Presentation
- Sample data is partially generated by ChatGPT and transformed into Gremlin operations

1. Query: Shortest Path

- Find the shortest path between "https://amazon.com/prime" and "https://amazon.com/prime/video/63d895bb-dac9-4d32-8e80-37783fe220c9/season/1/episode/7"

```
g.V().
  has("Node", "url", "https://amazon.com/prime").
  repeat(both().simplePath()).
    until(
      has(
        "url",
        "https://amazon.com/prime/video/63d895bb-dac9-4d32-8e80-
37783fe220c9/season/1/episode/7")).
  path().by(project('title', 'url').by('title').by('url')).
  limit(1)
```

1. Query Result (JSON)

```
...[
        {
            "title": "Amazon Prime",
            "url": "https://amazon.com/prime"
        },
        {
            "title": "Amazon Prime Video",
            "url": "https://amazon.com/prime/video"
        },
        {
            "title": "Penguins of Madagascar - Season 1",
            "url": "https://amazon.com/prime/video/63d895bb-dac9-4d32-8e80-
37783fe220c9/season/1"
        },
        {
            "title": "Penguins of Madagascar - Season 1 - Episode 7",
            "url": "https://amazon.com/prime/video/63d895bb-dac9-4d32-8e80-
37783fe220c9/season/1/episode/7"
        }
]...
```

2. Query: Degree Centrality

- Determine Degree Centrality (In/Out Edge Count) for each Node in the graph and order by degree value descending

```
g.V().
  hasLabel("Node").
  project("title", "url",
"degree").by("title").by("url").by(bothE().count()).
  order().by(select("degree"), decr)
```

2. Query Result (JSON)

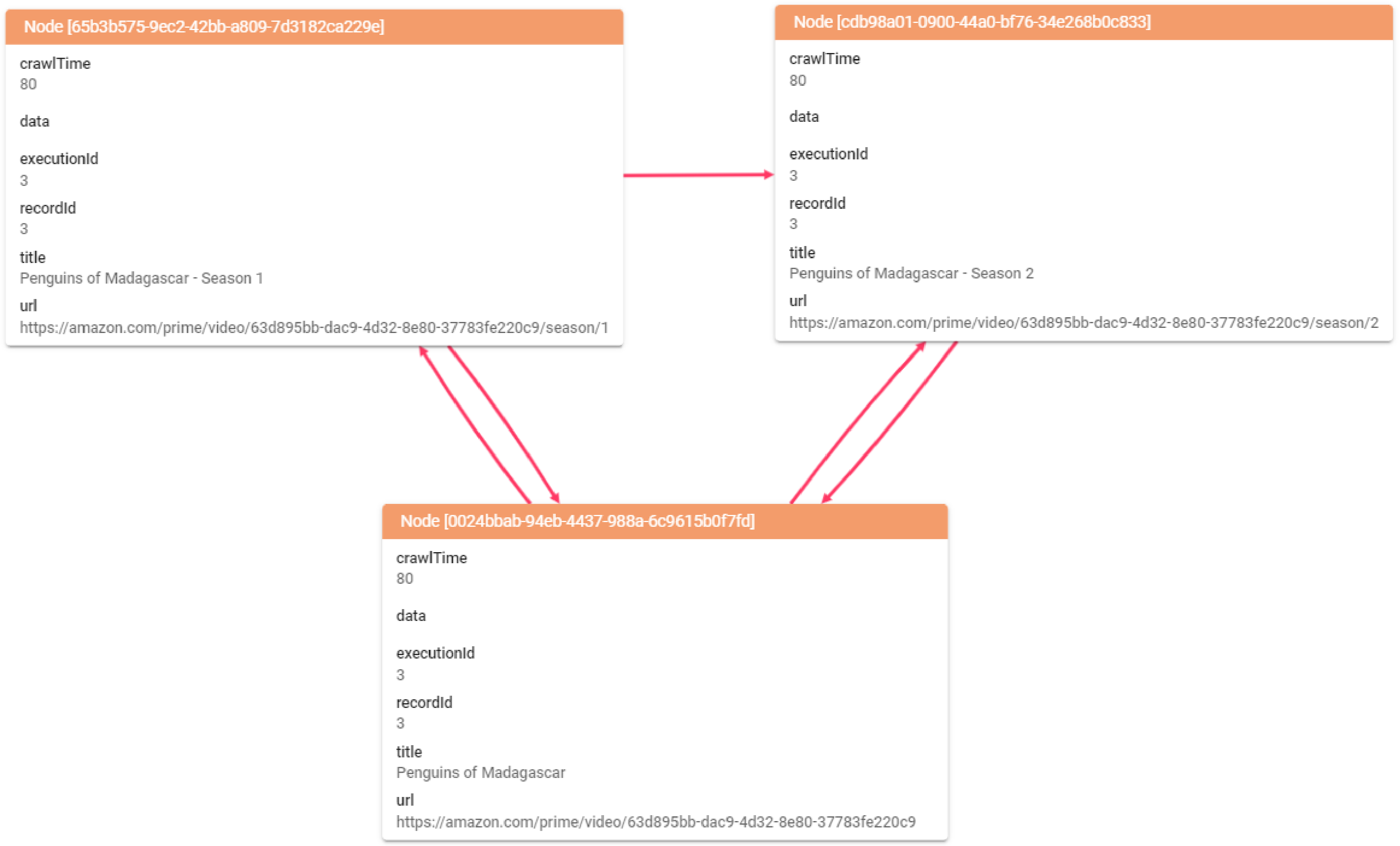
```
[
    {
        "title": "Amazon Prime",
        "url": "https://amazon.com/prime",
        "degree": 18
    },
    {
        "title": "Penguins of Madagascar - Season 1",
        "url": "https://amazon.com/prime/video/63d895bb-dac9-4d32-8e80-
37783fe220c9/season/1",
        "degree": 11
    },
    ...,
    {
        "title": "Penguins of Madagascar - Season 2 - Episode 3",
        "url": "https://amazon.com/prime/video/63d895bb-dac9-4d32-8e80-
37783fe220c9/season/2/episode/3",
        "degree": 1
    }
]
```

3. Query: Cycle Detection

- Detect the presence of a cycle in the graph (i.e. a loop)

```
g.V().as("cycle").
  repeat(out().simplePath()).times(2).
  where(out().as("cycle")).
  path().
  dedup().by(unfold().order().by(id).dedup().fold())
```

3. Query Result (G.V() visualisation)



Node [65b3b575-9ec2-42bb-a809-7d3182ca229e]

crawlTime
80

data

executionId
3

recordId
3

title
Penguins of Madagascar - Season 1

url
https://amazon.com/prime/video/63d895bb-dac9-4d32-8e80-37783fe220c9/season/1

Node [cdb98a01-0900-44a0-bf76-34e268b0c833]

crawlTime
80

data

executionId
3

recordId
3

title
Penguins of Madagascar - Season 2

url
https://amazon.com/prime/video/63d895bb-dac9-4d32-8e80-37783fe220c9/season/2

Node [0024bbab-94eb-4437-988a-6c9615b0f7fd]

crawlTime
80

data

executionId
3

recordId
3

title
Penguins of Madagascar

url
https://amazon.com/prime/video/63d895bb-dac9-4d32-8e80-37783fe220c9

Query Execution Profile

- Simple measure of Execution Time and other information by appending **.executionPlan()** at the end of a Gremlin Query
- https://learn.microsoft.com/en-us/azure/cosmos-db/gremlin/execution-profile

```
[
  {
    // The Gremlin statement that was executed.
    "gremlin": "g.V('mary').out().executionProfile()",
    // Amount of time in milliseconds that the entire operation took.
    "totalTime": 28,
    // An array containing metrics for each of the steps that were executed.
    // Each Gremlin step will translate to one or more of these steps.
    // This list is sorted in order of execution.
    "metrics": [
      {
        ...
       }
     ]
  }
[
```

Query Results – Execution Times

Query/time	Execution #1	Execution #2	Execution #3	Execution #4
Query 1	169ms	71ms	34ms	35ms
Query 2	150ms	145ms	148ms	142ms
Query 3	64ms	20ms	20ms	20ms

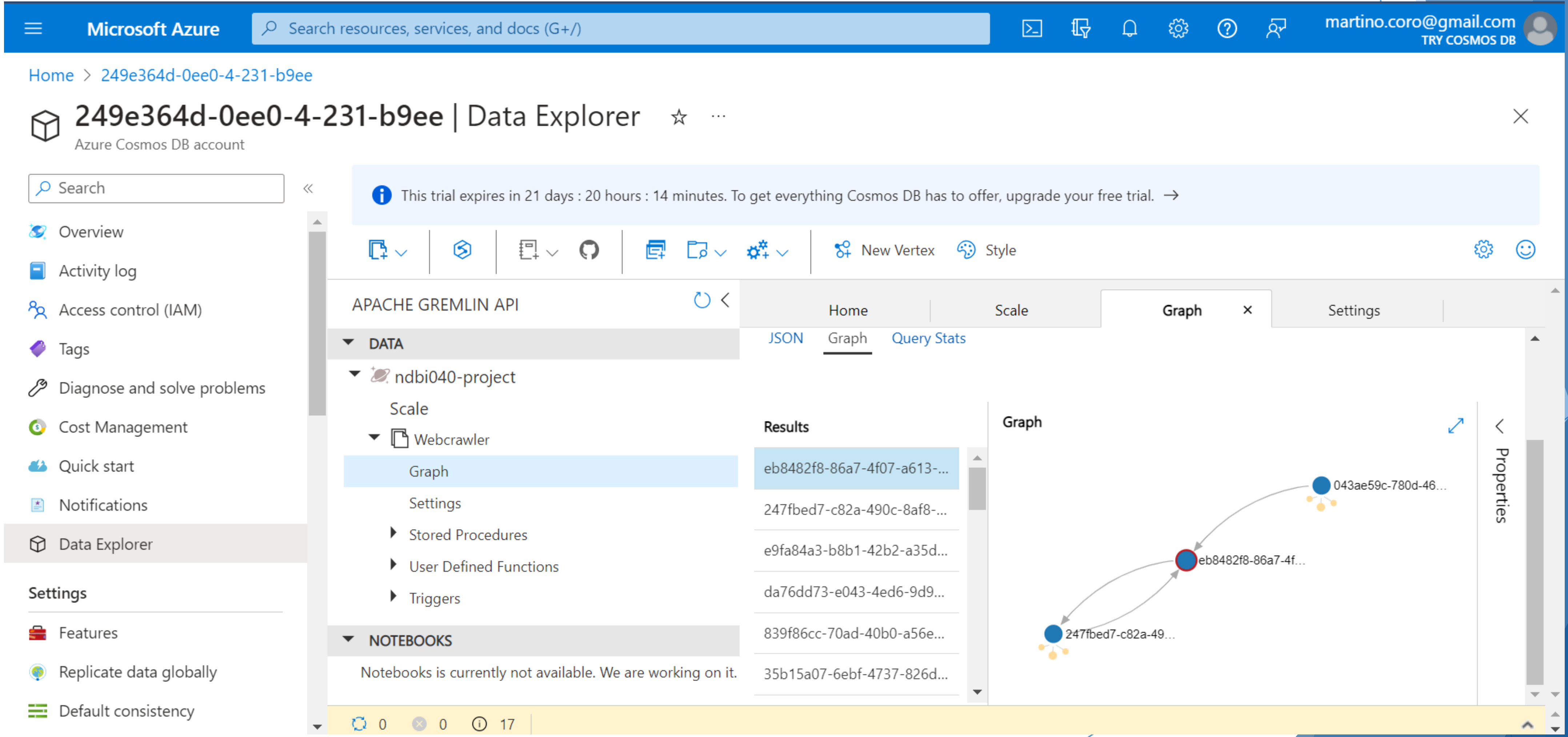
- The results show an interesting correlation between 1st and subsequent executions
- Based on Cosmos DB documentation, the 1st execution is expensive but is cached afterwards to produce faster retrievals on next attempts
- Based on results it takes at least 3 executions to stabilize
- 2nd execution (edge counts) is always Full Scan, so it cannot go any faster

- Full Execution Plans: https://github.com/corovcam/NDBI040-CosmosDB-Presentation

Useful Tools & Platforms

- Azure Cosmos DB Web Interface
  - Data Explorer to execute Gremlin queries
  - Weak visualising tool to show only subset of queried data
- Gremlin Console
  - For automated script execution
  - Playground & Debug tool alongside other Programming SDKs (.NET, Java, Python,...)
- G.V()
  - Feature-rich and robust ecosystem for debugging Gremlin queries, visualising Cosmos DB Gremlin API, Amazon Neptune, JanusGraph,...
  - 30-day Free Trial for new installations only

Azure Web Interface



Microsoft Azure

Search resources, services, and docs (G+/)

martino.coro@gmail.com
TRY COSMOS DB

Home > 249e364d-0ee0-4-231-b9ee

249e364d-0ee0-4-231-b9ee | Data Explorer

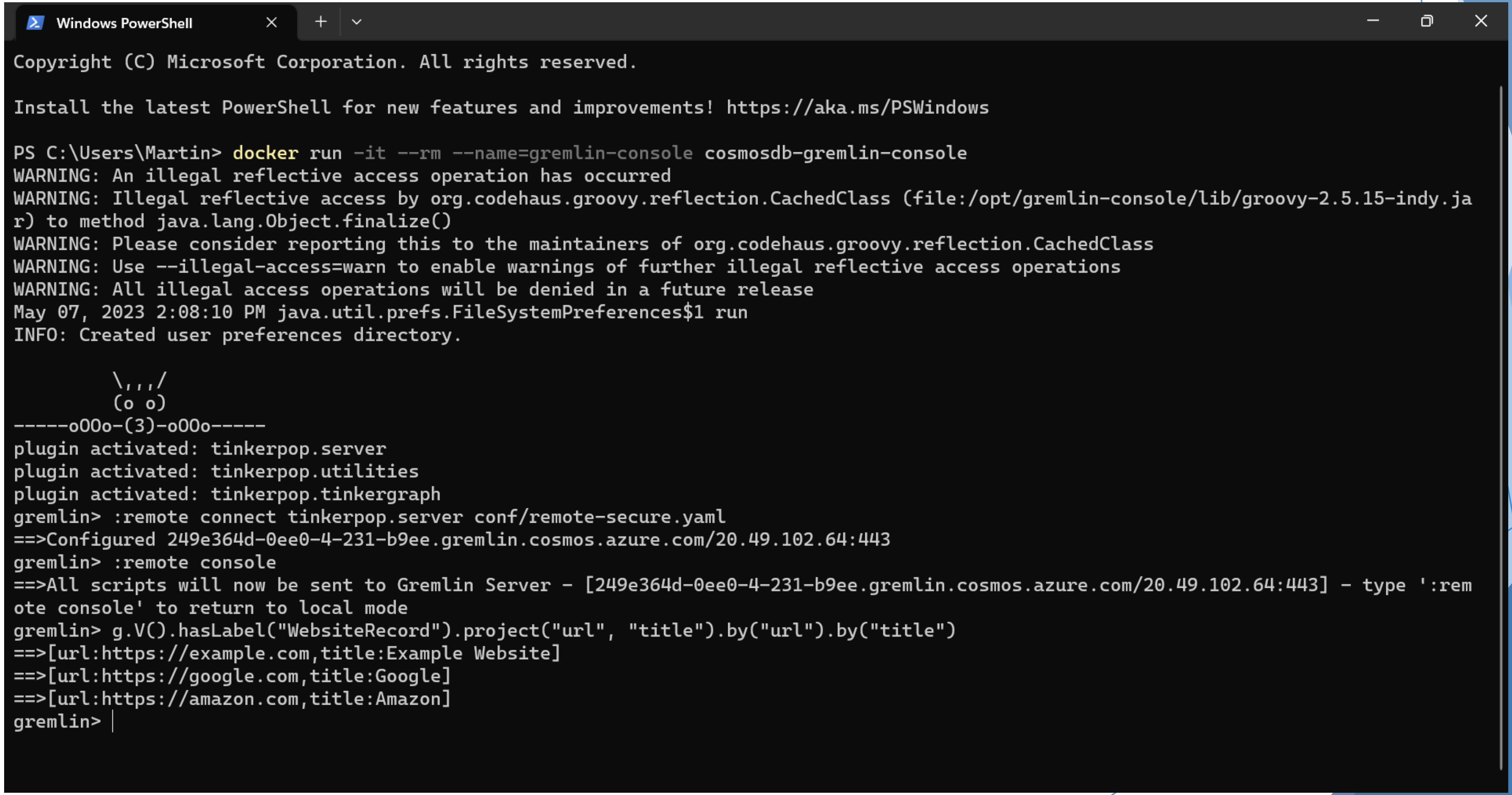
Azure Cosmos DB account

This trial expires in 21 days : 20 hours : 14 minutes. To get everything Cosmos DB has to offer, upgrade your free trial. →

Notebooks is currently not available. We are working on it.

Results
eb8482f8-86a7-4f07-a613-...
247fbed7-c82a-490c-8af8-...
e9fa84a3-b8b1-42b2-a35d...
da76dd73-e043-4ed6-9d9...
839f86cc-70ad-40b0-a56e...
35b15a07-6ebf-4737-826d...

Gremlin Console



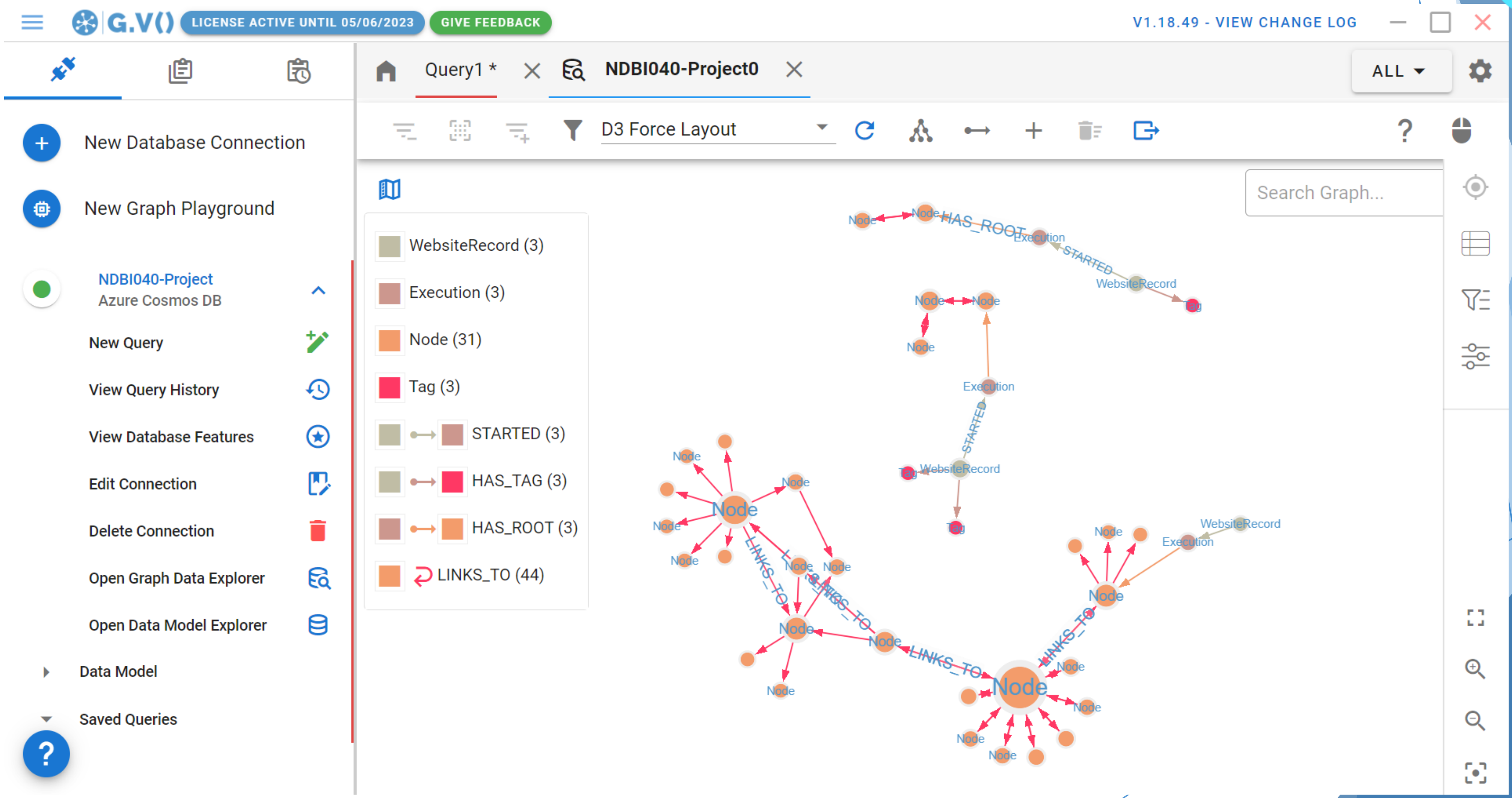
```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Martin> docker run -it --rm --name=gremlin-console cosmosdb-gremlin-console
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.reflection.CachedClass (file:/opt/gremlin-console/lib/groovy-2.5.15-indy.ja
r) to method java.lang.Object.finalize()
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.reflection.CachedClass
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
May 07, 2023 2:08:10 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> :remote connect tinkerpop.server conf/remote-secure.yaml
==>Configured 249e364d-0ee0-4-231-b9ee.gremlin.cosmos.azure.com/20.49.102.64:443
gremlin> :remote console
==>All scripts will now be sent to Gremlin Server - [249e364d-0ee0-4-231-b9ee.gremlin.cosmos.azure.com/20.49.102.64:443] - type ':rem
ote console' to return to local mode
gremlin> g.V().hasLabel("WebsiteRecord").project("url", "title").by("url").by("title")
==>[url:https://example.com,title:Example Website]
==>[url:https://google.com,title:Google]
==>[url:https://amazon.com,title:Amazon]
gremlin>
```

G.V() – Gremlin IDE & Visualisation Tool



Re/sources

- https://learn.microsoft.com/en-us/azure/cosmos-db/
- https://cosmos.azure.com/try/
- https://gdotv.com/
- https://azurecosmosdb.github.io/labs/
- https://en.wikipedia.org/wiki/Gremlin_%28query_language%29
- https://kepty.cz/wp-content/uploads/2021/11/AzureCosmosDB.png

List of Additional Files

- „cosmos-db-recording.mp4“ – video recording of the presentation
- Repository:
  - „/scripts/data.groovy“ – data generation script
  - „/scripts/queries.groovy“ – Gremlin queries script
  - „/conf/remote-secure.yaml“ – Gremlin Console config
  - „/conf/indexing-policy.json“ – Azure Web Interface -> Webcrawler -> Settings -> Indexing Policy configuration

The End of Cosmos