

# DAMG 7275

## Advanced Database Management Systems

### P3 submission

## Project - Farm Environmental Monitoring System






Project Group : 3

---

### \* Introduction :-

For this submission, we have implemented our architecture diagram and pipeline on Azure cloud platform. Some of the services that we have used are :

- Azure Blob Storage
- Azure SQL Server
- Azure SQL database
- Azure Comsos DB
- Azure Data Factory

<input type="checkbox"/> Name ↑↓	Type ↑↓	Resource group ↑↓	Location ↑↓	Subscription ↑↓
<input type="checkbox"/>  adbms-finalproject	Azure Cosmos DB account	<a href="#">shalom_adbms_RG</a>	East US 2	<a href="#">shalomd_adbms_finalproject</a>
<input type="checkbox"/>  adbms-sql-server	SQL server	<a href="#">shalom_adbms_RG</a>	East US 2	<a href="#">shalomd_adbms_finalproject</a>
<input type="checkbox"/>  adbms_sql_db (adbms-sql-server/adbms_sql_db)	SQL database	<a href="#">shalom_adbms_RG</a>	East US 2	<a href="#">shalomd_adbms_finalproject</a>
<input type="checkbox"/>  adbmsstorage	Storage account	<a href="#">shalom_adbms_RG</a>	East US 2	<a href="#">shalomd_adbms_finalproject</a>
<input type="checkbox"/>  shalom-adbms-data-factory	Data factory (V2)	<a href="#">shalom_adbms_RG</a>	East US 2	<a href="#">shalomd_adbms_finalproject</a>

## \* Dataset :-

We have 3 primary sources of data for this project implementation :

- Weather data (JSON)
- Soil moisture data (CSV)
- Camera images (JPG)

soil\_moisture\_data

soil_reading_id	reading_date	reading_time	sensor_id	moisture_reading
1	2023-11-13	08:00:00	101	25.5
2	2023-11-14	12:30:00	102	30.2
3	2023-11-15	15:45:00	103	22.8
4	2023-11-16	10:10:00	104	18.6
5	2023-11-17	14:20:00	105	28.1
6	2023-11-18	09:55:00	106	33.7
7	2023-11-19	11:40:00	107	19.3

```
{
  "weather_reading_id": 1,
  "data": "2023-01-01",
  "time": "12:00:00",
  "temp": 25.5,
  "humidity": 70.2,
  "rainfall": 0.0,
  "longitude": -73.975,
  "latitude": 40.783
},
{
  "weather_reading_id": 2,
  "data": "2023-01-02",
  "time": "14:30:00",
  "temp": 22.3,
  "humidity": 68.8,
  "rainfall": 0.2,
  "longitude": -74.006,
  "latitude": 40.712
},
{
  "weather_reading_id": 3,
  "data": "2023-01-03",
  "time": "10:45:00",
  "temp": 28.1,
  "humidity": 75.5,
  "rainfall": 0.0,
  "longitude": -73.986,
  "latitude": 40.748
},
{
  "weather_reading_id": 4,
  "data": "2023-01-04",
  "time": "08:15:00",
  "temp": 19.8,
  "humidity": 62.4,
  "rainfall": 0.5,
  "longitude": -73.943,
  "latitude": 40.669
},
}
```

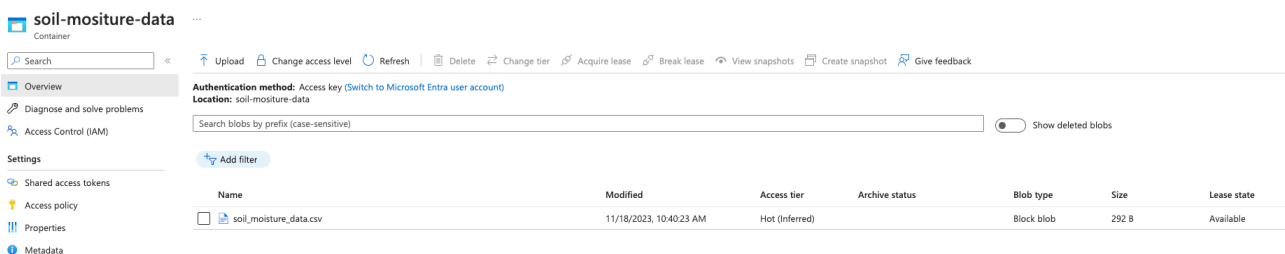
## \* Implementation :-

We are using Azure Blob Storage to manage our data. All data from each source is stored in respective containers on Azure like below.

Name	Last modified
<input type="checkbox"/> \$logs	11/18/2023, 10:10:45 AM
<input type="checkbox"/> camera-images	11/18/2023, 12:02:07 PM
<input type="checkbox"/> soil-mositure-data	11/18/2023, 10:11:23 AM
<input type="checkbox"/> weather-api-data	11/18/2023, 10:57:27 AM

Each container will have its own data file :

### *Soil moisture data in csv format -*



soil-mositure-data

Container

Search

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Shared access tokens

Access policy

Properties

Metadata

Authentication method: Access key (Switch to Microsoft Entra user account)

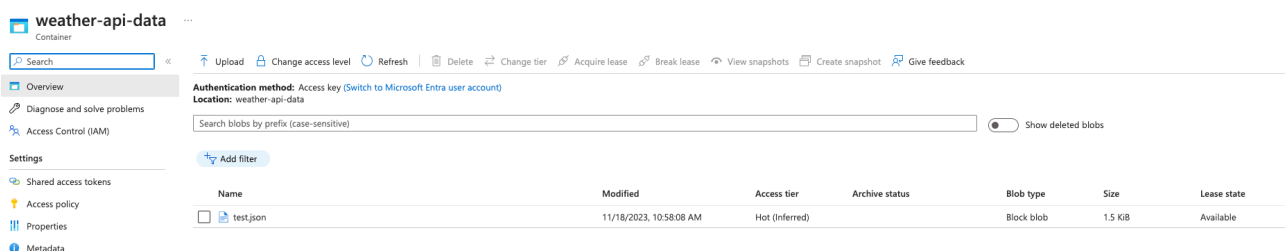
Location: soil-mositure-data

Search blobs by prefix (case-sensitive)

Show deleted blobs

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
<input type="checkbox"/> soil_moisture_data.csv	11/18/2023, 10:40:23 AM	Hot (Inferred)		Block blob	292 B	Available

### *Weather data in json format -*



weather-api-data

Container

Search

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Shared access tokens

Access policy

Properties

Metadata

Authentication method: Access key (Switch to Microsoft Entra user account)

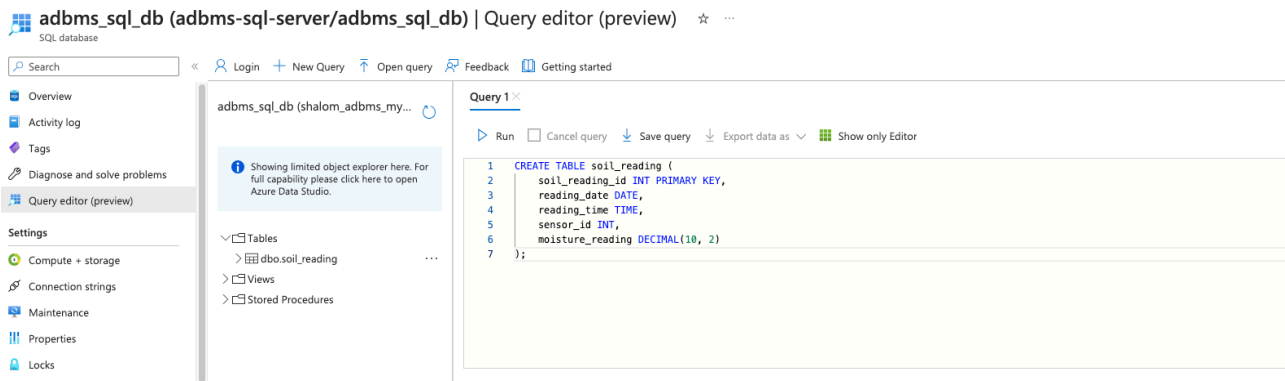
Location: weather-api-data

Search blobs by prefix (case-sensitive)

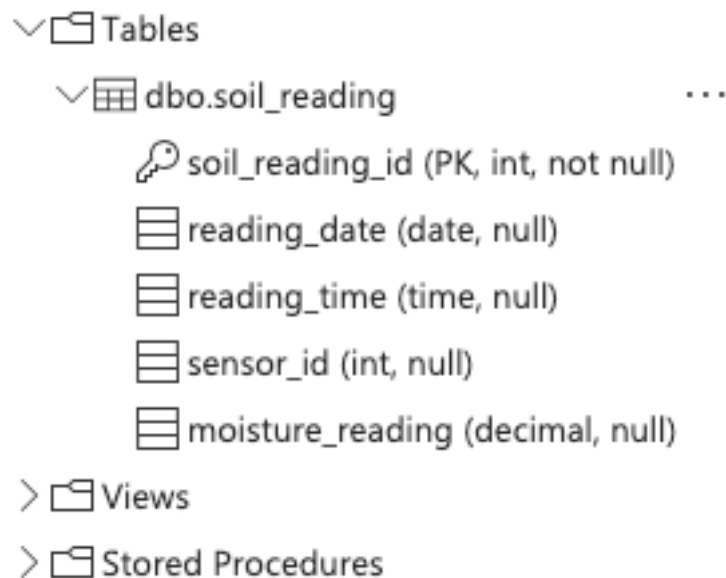
Show deleted blobs

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
<input type="checkbox"/> test.json	11/18/2023, 10:58:08 AM	Hot (Inferred)		Block blob	1.5 KiB	Available

We are using Azure SQL DB to handle our relational data model which in this case is Soil Moisture Data.



We have created a table on Azure SQL db replicating the schema of our soil moisture data. Once sql command query is successful, we see the table is created as below.



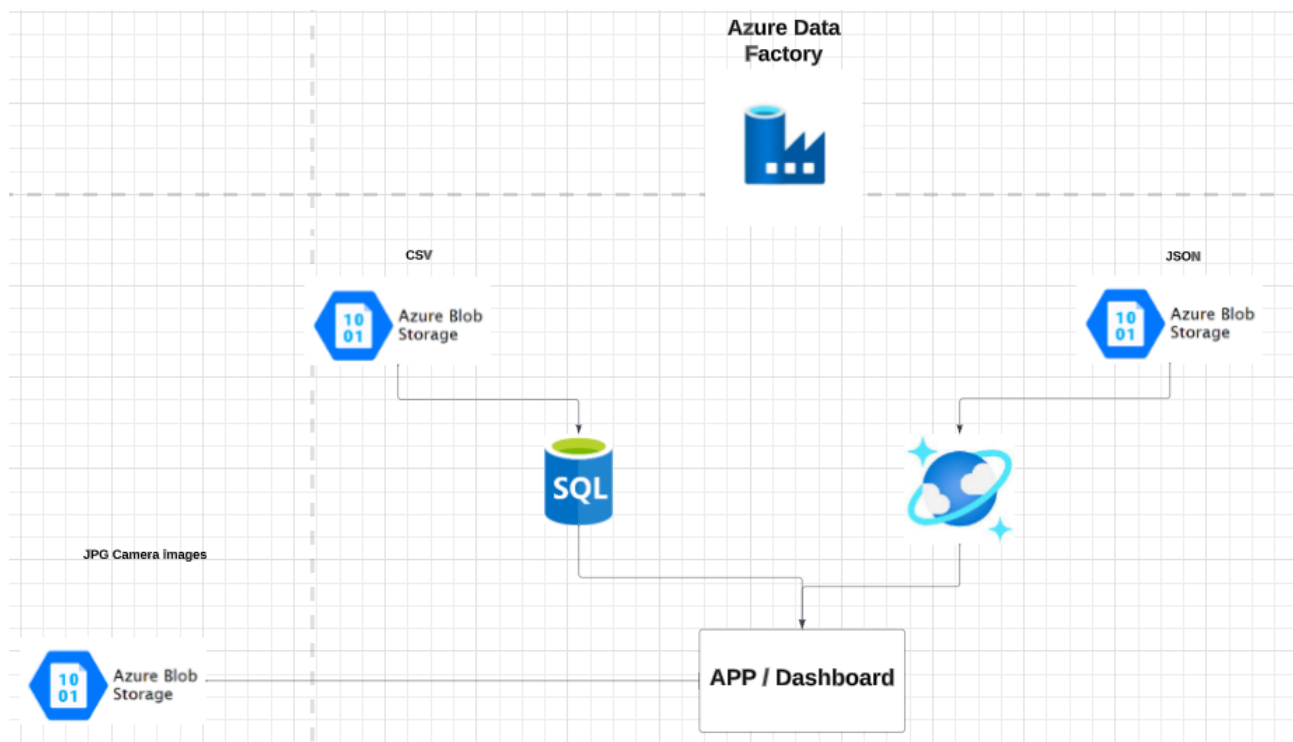
We are using Azure Cosmos DB to handle our document data model which in this case is Weather Data.

On Cosmos DB, we have created a db named *weather\_api* & a container within it named *weather\_data* to handle incoming weather data in JSON format.

The screenshot shows the Azure Cosmos DB Data Explorer interface. At the top, the account name 'adbms-finalproject | Data Explorer' is displayed with a star icon and a menu icon. Below this, the text 'Azure Cosmos DB account' is visible. A search bar is located on the left. The main navigation pane on the left lists various options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Cost Management, Quick start, Notifications, Data Explorer (which is highlighted), and Settings. Under Settings, there are links for Features and Replicate data globally. The main content area on the right is titled 'NOSQL API' and shows a tree view of the database structure. The 'DATA' section is expanded, showing the 'weather\_api' database. Under 'weather\_api', there is a 'Scale' section and a 'weather\_data' container, which is highlighted. Below the container, there are links for 'Items', 'Settings', 'Stored Procedures', 'User Defined Functions', and 'Triggers'. The 'NOTEBOOKS' section is also visible, with a message stating 'Notebooks is currently not available. We are working on it.'

We are using Azure Data Factory for our data pipeline which will be used to orchestrate each of the services and ingest data into their respective databases.

Below is the pipeline and data flow.



**Note:** JPG camera images are not part of the pipeline but they are stored in Azure blob storage directly.  
The app or dashboard will directly pull images from Azure storage container.

We have created 3 Linked Services on Data Factory to handle the data models.





### Linked services

Linked service defines the connection information to a data store or compute. [Learn more](#)

+ New

Filter by name Annotations : Any

Showing 1 - 4 of 4 items

Name ↑↓	Type ↑↓	Related ↑↓
 CosmosDbNoSql1	Azure Cosmos DB for NoSQL	1
 soil_moisture_blob	Azure Blob Storage	1
 soil_moisture_sql	Azure SQL Database	1
 weather_api_data	Azure Blob Storage	1

We have created 3 Datasets on Data Factory to replicate the schema of the data models.

Microsoft Azure | Data Factory ▶ shalom-adbms-data-factory

»

Data Factory ▼

Validate all

Publish all

Home

5

Factory Resources

Filter resources by name

Pipelines 1

adbms\_pipeline

Change Data Capture (preview) 0

Datasets 4

soilmoisturedata

soilmoisturesqldataset

weathercosmosdataset

weatherdataset

adbms\_

Activitie

Search

Move a

Synapsi

Azure C

Azure F

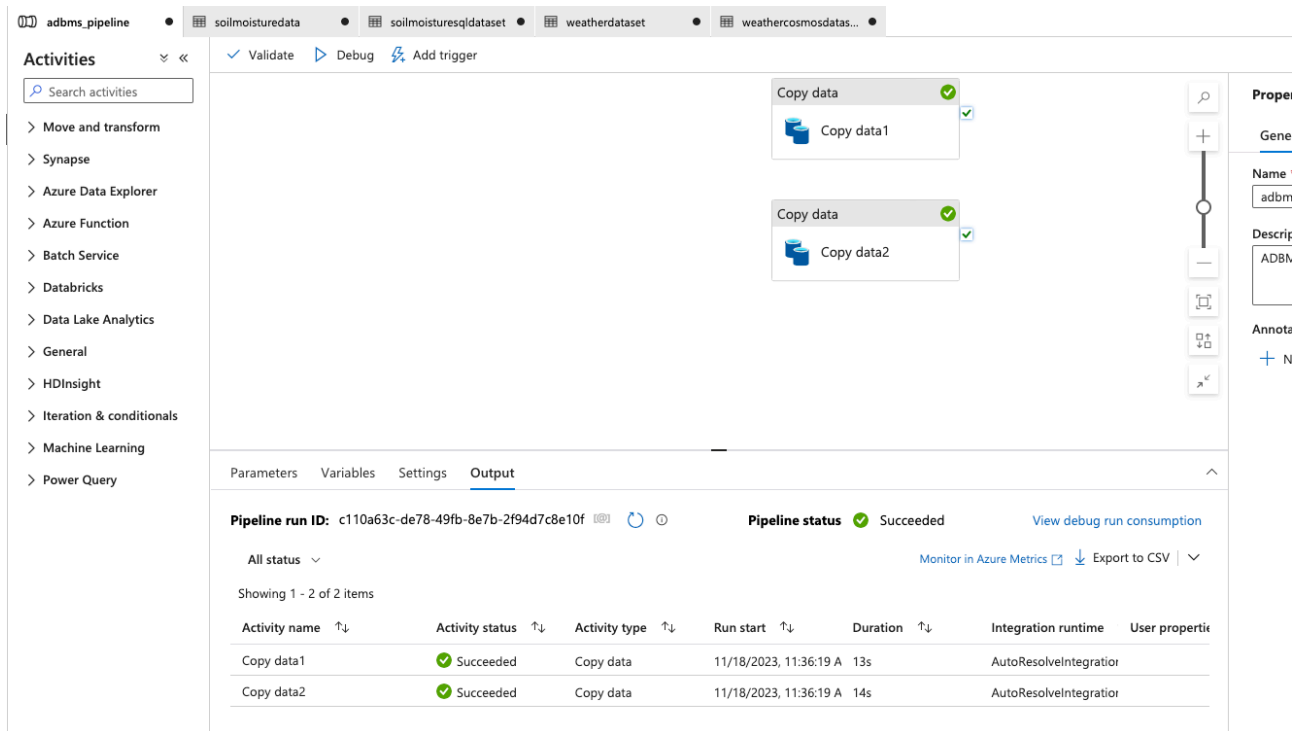
Batch S

Databri

Data La



In our pipeline, we have added 2 Copy steps to ingest csv data from blob storage to Sql db and json data from blob storage to Cosmos db.

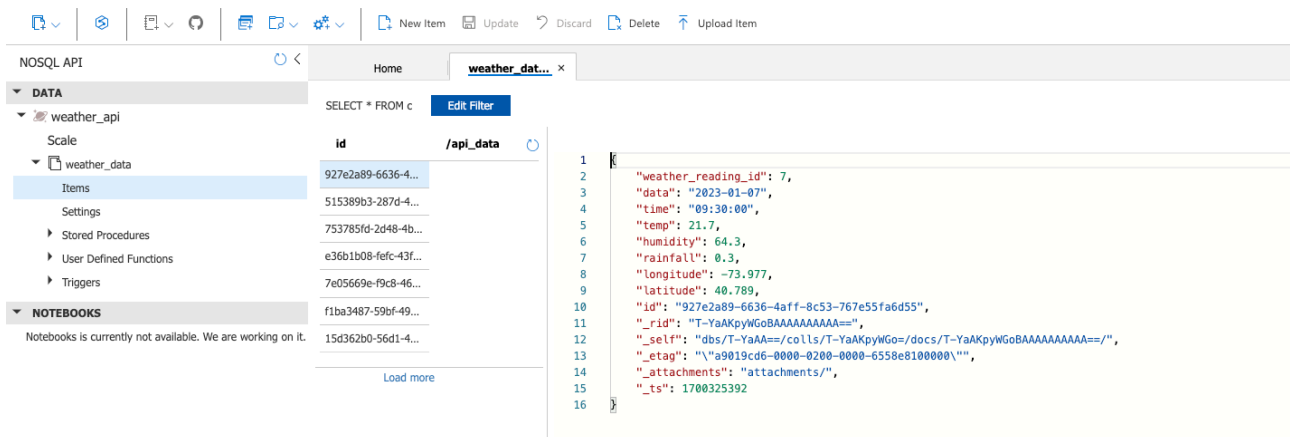


**Pipeline run ID:** c110a63c-de78-49fb-8e7b-2f94d7c8e10f **Pipeline status:** Succeeded

**Activity details table:**

Activity name	Activity status	Activity type	Run start	Duration	Integration runtime	User properties
Copy data1	Succeeded	Copy data	11/18/2023, 11:36:19 A	13s	AutoResolveIntegration	
Copy data2	Succeeded	Copy data	11/18/2023, 11:36:19 A	14s	AutoResolveIntegration	

We ran the Debug option to test our pipeline and it succeeded. Below is the result of the pipeline. The Sql DB table and Cosmos DB container is correctly populated with expected data models.



**Table: weather\_data**


id	/api_data
927e2a89-6636-4...	[{"weather_reading_id": 7, "data": "2023-01-07", "time": "09:30:00", "temp": 21.7, "humidity": 64.3, "rainfall": 0.3, "longitude": -73.977, "latitude": 40.789, "id": "927e2a89-6636-4aff-8c53-767e55fa6d55", "rid": "T-YaAKpyWGoBAAAAAAAAAA==", "self": "dbs/T-YaAA=/colls/T-YaAKpyWGo=/docs/T-YaAKpyWGoBAAAAAAAAAA=/", "etag": "\"a9019cd6-0000-0200-0000-6558e8100000\"", "attachments": "attachments/", "ts": 1700325392}

Query 1


 Run ☐ Cancel query  Save query  Export data as  Show only Editor

```
1 Select * from [dbo].[soil_reading]
```

Results Messages

 Search to filter items...

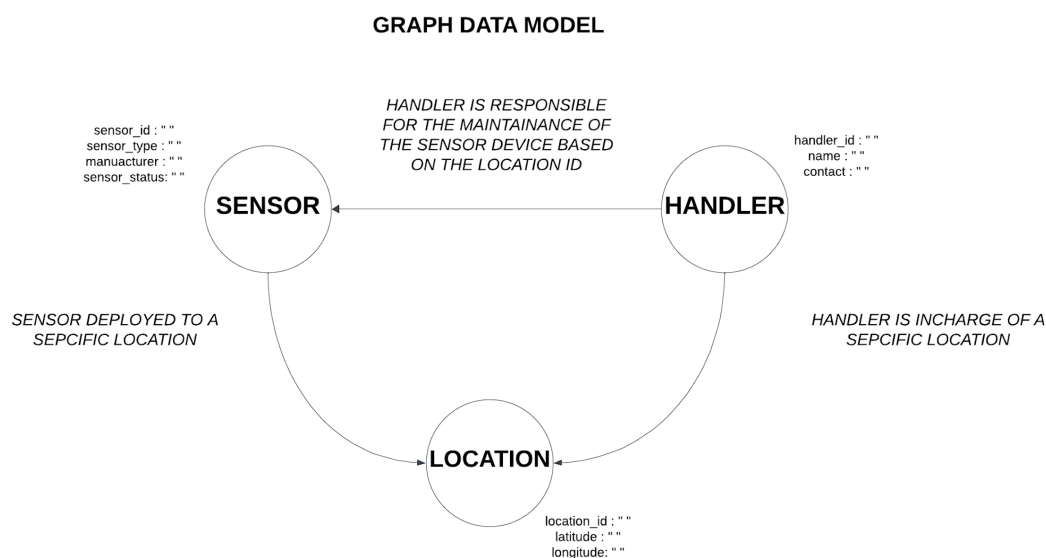
soil_reading_id	reading_date	reading_time	sensor_id	moisture_reading
1	2023-11-13T00:00:00.0000000	08:00:00	101	25.50
2	2023-11-14T00:00:00.0000000	12:30:00	102	30.20
3	2023-11-15T00:00:00.0000000	15:45:00	103	22.80
4	2023-11-16T00:00:00.0000000	10:10:00	104	18.60
5	2023-11-17T00:00:00.0000000	14:20:00	105	28.10
6	2023-11-18T00:00:00.0000000	09:55:00	106	33.70
7	2023-11-19T00:00:00.0000000	11:40:00	107	19.30

 Query succeeded | 0s

For the graph data model implementation, we are using a python script to extract vertices and edges from our relational data models : Location, Handler & Sensor.

The Python script also gives us the Gremlin API queries which we will use to create the vertices and edges on Azure CosmosDB with Gremlin API account.

Below is our Graph data model -



Below are the Gremlin queries we ran on the Gremlin API console -

a) Creating Location vertex:

```
g.addV('location').property('locationID', 1).property('lat', 123.45).property('long', 67.89).property('city', 'City1').property('country', 'Country1')
```

b) Creating Handler vertex and edge:

```
g.addV('handler').property('handlerID', 101).property('name',  
'Handler1').property('contact', 'Contact1')
```

```
g.V().has('locationID',  
1).addE('hasHandler').to(g.V().has('handlerID', 101))
```

c) Creating Sensor node and edge:

```
g.addV('sensor').property('sensorID', 1001).property('type',  
'Type1').property('manufacturer',  
'Manufacturer1').property('sensor_status', 'Active')
```

```
g.V().has('locationID',  
1).addE('hasSensor').to(g.V().has('sensorID', 1001))
```

d) Creating edge between Handler and Sensor:

```
g.V().has('handlerID',  
101).addE('handles').to(g.V().has('sensorID', 1001))
```

We have create Azure cosmos's with gremlin api account and configured a new database with a graph as shown below:

Home > graphdatamodel-acc-adbms

## graphdatamodel-acc-adbms | Data Explorer

Azure Cosmos DB account

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Cost Management
- Quick start
- Notifications
- Data Explorer**

Settings

- Features
- Replicate data globally

APACHE GREMLIN API

DATA

- adbms-graph2
  - Scale
  - graph1
    - Graph**
    - Settings
    - Stored Procedures
    - User Defined Functions
    - Triggers

NOTEBOOKS

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

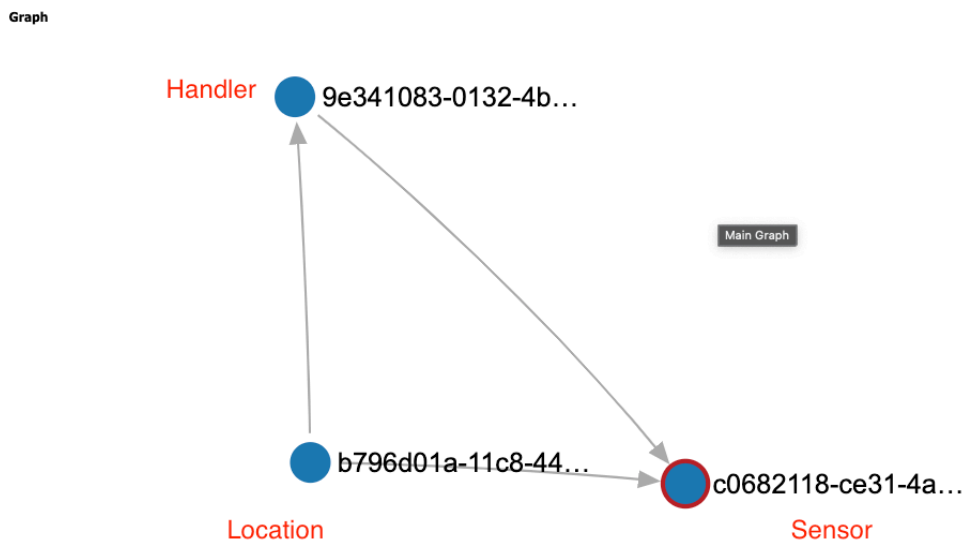
g.V()

JSON Graph Query Stats

**Results**

- b796d01a-11c8-44ac-9108...
- 9e341083-0132-4b55-8a9...
- c0682118-ce31-4a9e-87b1...

Below is the screenshot of the graph created with the vertices and edges specified through the gremlin api:



**Note:** Azure Data Factory does not have a connector for Azure CosmosDB with Gremlin API yet, so we could not integrate our graph data model with our existing pipeline.

We will be running our python script mentioned above whenever there is a change in our data sources to refresh the graph data model.

## \* Next steps & Future Scopes :-

- 1) Implement an application and dashboard to make sense of the data that we have used and for visualisations. (P5)
- 2) Auto data fresh. (P4)
- 3) Try and increase size of datasets for better visibility & analysis.

## \* Group members :-

- 1) Ankita Patil
- 2) Aditya Pande
- 3) Keshni Mulrajani
- 4) Sachit Wagle
- 5) Shalom Daniel

\*\*\*\*\*