

# Data Engineering in the Cloud

## Hybrid Transactional Analytical Processing

Xuemao Zhang  
East Stroudsburg University

January 18, 2025

# Outline

- In the labs, we Query Azure Cosmos DB with Apache Spark Pool and Serverless SQL Pools
- Lab 1: Querying Azure Cosmos DB with Apache Spark Pool
- Lab 2: Querying Azure Cosmos DB with Serverless SQL Pools

# Prerequisites

- Prerequisites (10-15 minutes):
  - ▶ Create a datalake (storage account with hierarchical namespace enabled)
    - ★ To lower the cost, you may choose Redundancy as LRS
    - ★ Create a container in the storage account
  - ▶ Create an Azure Synapse workspace and
    - ★ create a Apache spark pool in the Manage tab
  - ▶ Create an Azure Cosmos DB database with a container
    - ★ with Analytical store enabled and then create two key-value items
  - ▶ In the Synapse Studio, go to the Manage tab to add a new linked service for Azure Cosmos DB for NoSQL
  - ▶ Go to the Data tab of the Synapse Studio, click on + and choose “Integration dataset” to create a dataset.
    - ★ We do not need the Copy pipeline

# Lab 1: Querying Azure Cosmos DB with Apache Spark Pool

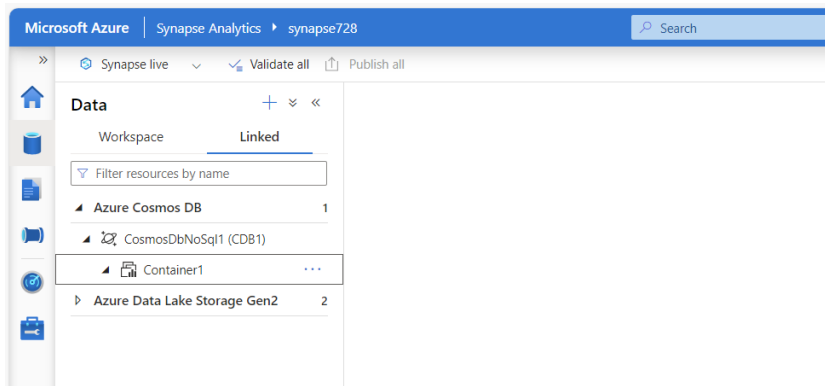
- 1 Open “synapse studio”, go to “Manage” tab, and add a new Cosmos DB linked service.

The screenshot displays the Azure Synapse Analytics interface for a workspace named 'synapse728'. The left-hand navigation pane is expanded to show the 'Linked services' option under the 'External connections' category. The main content area, titled 'Linked services', provides a description of linked services and includes a '+ New' button to add a new one. Below this, there is a search bar and a filter for 'Annotations'. A table lists the existing linked services:

Name	Type	Related
CosmosDbNoSql1	Azure Cosmos DB for NoSQL	0
synapse728-WorkspaceDefaultSqlServer	Azure Synapse Analytics	0
synapse728-WorkspaceDefaultStorage	Azure Data Lake Storage Gen2	0

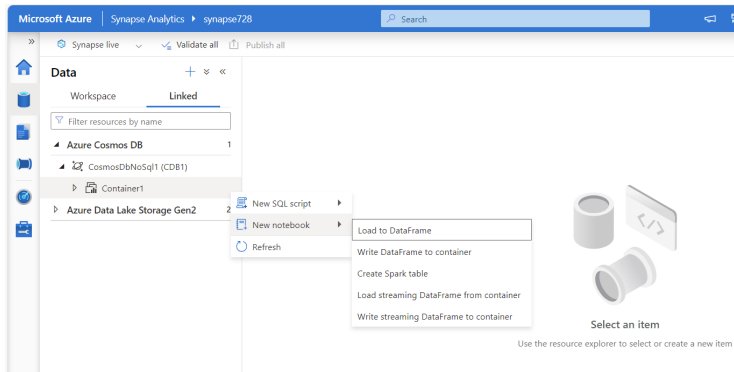
# Lab 1: Querying Azure Cosmos DB with Apache Spark Pool

- 2 Go to “Data” tab, and choose “Linked”.



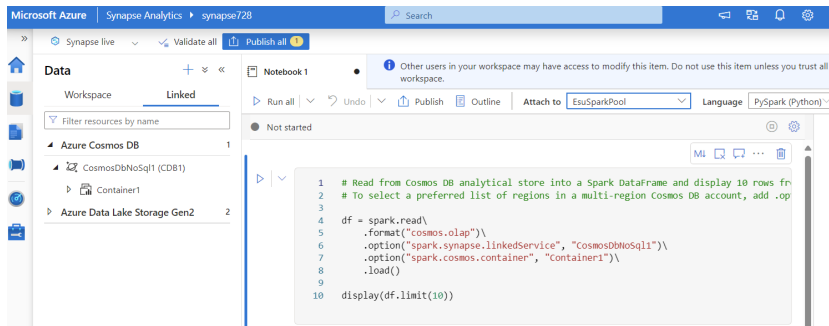
# Lab 1: Querying Azure Cosmos DB with Apache Spark Pool

- 3 Expand the Azure Cosmos DB, choose the container “Container1” created in the Cosmos DB, click on three dots, choose “New notebook”, and choose “Load to Dataframe”.



# Lab 1: Querying Azure Cosmos DB with Apache Spark Pool

- 4 Attach the spark pool to the notebook.

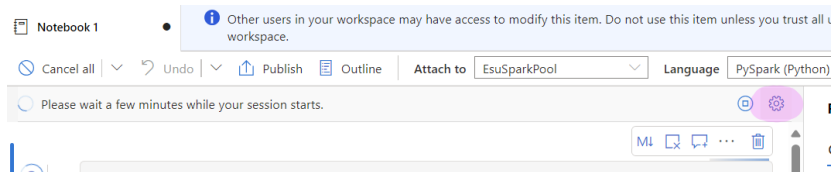


The screenshot displays the Microsoft Azure Synapse Analytics workspace 'synapse728'. On the left, the 'Data' pane shows a 'Linked' workspace with a filter 'Filter resources by name'. Under 'Azure Cosmos DB', there is a resource 'CosmosDbNoSql1 (CDB1)' with a 'Container1' sub-resource. Below that is 'Azure Data Lake Storage Gen2' with 2 resources. The main area shows 'Notebook 1' with a status of 'Not started'. A dropdown menu for 'Attach to' is set to 'EsuSparkPool', and the 'Language' is 'PySpark (Python)'. A code editor contains the following PySpark script:

```
1 # Read from Cosmos DB analytical store into a Spark DataFrame and display 10 rows fr
2 # To select a preferred list of regions in a multi-region Cosmos DB account, add .op
3
4 df = spark.read\
5     .format("cosmos.olap")\
6     .option("spark.synapse.linkedService", "CosmosDbNoSql1")\
7     .option("spark.cosmos.container", "Container1")\
8     .load()
9
10 display(df.limit(10))
```

# Lab 1: Querying Azure Cosmos DB with Apache Spark Pool

- 5 Click on “Run All”. It takes some time to run.
  - ▶ If you see similar message, then update the configuration of the spark session

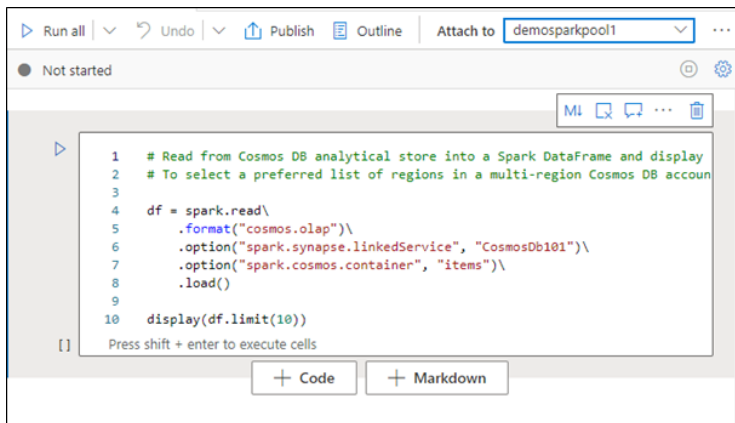


InvalidHttpRequestToLivy: Your Spark job requested 48 vcores. However, the workspace has a 12 core limit. Try reducing the numbers of vcores requested or increasing your vcore quota.

Quota can be increased using Azure Support request <https://learn.microsoft.com/en-us>



# Lab 1: Querying Azure Cosmos DB with Apache Spark Pool



The screenshot shows a Jupyter Notebook interface. At the top, there is a toolbar with buttons for 'Run all', 'Undo', 'Publish', 'Outline', and 'Attach to'. The 'Attach to' dropdown menu is set to 'demosparkpool1'. Below the toolbar, a status bar indicates 'Not started'. The main area contains a code cell with the following Python code:

```
1 # Read from Cosmos DB analytical store into a Spark DataFrame and display
2 # To select a preferred list of regions in a multi-region Cosmos DB account
3
4 df = spark.read\
5     .format("cosmos.olap")\
6     .option("spark.synapse.linkedService", "CosmosDb101")\
7     .option("spark.cosmos.container", "items")\
8     .load()
9
10 display(df.limit(10))
```

Below the code cell, there is a prompt: '[ ] Press shift + enter to execute cells'. At the bottom of the interface, there are two buttons: '+ Code' and '+ Markdown'.

## Lab 1: Querying Azure Cosmos DB with Apache Spark Pool

- 6 You must be able to see the output as given below:

```
1 # Read from Cosmos DB analytical store into a Spark DataFrame and display 10 rows from the DataFrame
2 # To select a preferred list of regions in a multi-region Cosmos DB account, add .option("spark.cosmos.preferredRegions", "<Region1>,<Region2>")
3
4 df = spark.read\
5     .format("cosmos.olap")\
6     .option("spark.synapse.linkedService", "CosmosDbNoSql1")\
7     .option("spark.cosmos.container", "Container1")\
8     .load()
9
10 display(df.limit(10))
```

[1] ✓ 3 min 28 sec - Apache Spark session started in 3 min 2 sec 344 ms. Command executed in 25 sec 972 ms by niuyuxia728 on 2:59:48 PM, 7/28/24

> Job execution Succeeded Spark 1 executors 4 cores [View in monitoring](#) [Open Spark UI](#)

View Table Chart [Export results](#)

_rid	_ts	id	category	description	_etag
2J4oAMexroYBAAAAAAAAA==	1722190945	1	fruits	vrious fruits	"06007145-0"
2J4oAMexroYCAAAAAAAAAA==	1722190981	2	vegetables	various vegetables	"06005846-0"

# Lab 1: Querying Azure Cosmos DB with Apache Spark Pool

- ⑦ You can remove the unwanted column using the below command:

```
unwanted_cols = {'_rid', '_ts', '_etag'}  
cols = list(set(df.columns) - unwanted_cols)  
df2 = df.select(cols)
```

```
display(df2.limit(10))
```

The screenshot shows a Databricks notebook interface. The code cell contains the following Spark code:

```
7 .option("spark.cosmos.container", "Container1")\n8 .load()\n9 \n10 display(df.limit(10))\n11 \n12 unwanted_cols = {'_rid', '_ts', '_etag'}\n13 cols = list(set(df.columns) - unwanted_cols)\n14 df2 = df.select(cols)\n15 \n16 display(df2.limit(10))\n17
```

Below the code cell, a status bar indicates: "2 sec - Command executed in 2 sec 807 ms by niyuxia728 on 3:03:55 PM, 7/28/24".

The "Job execution Succeeded" message is followed by a "Spark 1 executors 4 cores" status.

Below the job execution, there are two table views. The first table view shows the results of the first query (df.limit(10)) with columns: \_rid, \_ts, id, category, and description. The second table view shows the results of the second query (df2.limit(10)) with columns: category, id, and description.

_rid	_ts	id	category	description
2J4oAMexroYBAAAAAAAAAA==	1722190945	1	fruits	vnious fruits
2J4oAMexroYCAAAAAAAAAAA==	1722190981	2	vegetables	various vegetables

category	id	description
fruits	1	vnious fruits
vegetables	2	various vegetables

# Lab 2: Querying Azure Cosmos DB with Serverless SQL Pools

- 1 Go to “Data” in the Synapse Studio and right click Container1 to add an SQL script.

The screenshot shows the Microsoft Azure Synapse Studio interface. The top bar indicates the environment is 'synapse728'. The left sidebar shows the 'Data' pane with a 'Linked' view. Under 'Workspace', there is a search bar and a list of resources: 'Azure Cosmos DB' (1), 'CosmosDbNoSql1 (CDB1)', 'Container1' (highlighted with a pink circle), and 'Azure Data Lake Storage Gen2'. A context menu is open over 'Container1', with options: 'New SQL script' (highlighted with a pink circle), 'New notebook', and 'Refresh'. The 'New SQL script' option has a sub-menu open showing 'Select TOP 100 rows' (highlighted with a pink circle). The main pane shows 'SQL script 1' with the following code:

```
1 IF (NOT EXISTS(SELECT * FROM sys.credentials WHERE name = 'esucosmos'))
2     THROW 50000, 'As a prerequisite, create a credential with Azure Cosmos DB key in
3     CREATE CREDENTIAL [esucosmos]
4     WITH IDENTITY = 'SHARED ACCESS SIGNATURE', SECRET = '<Enter your Azure Cosmos
5 GO
6
7 SELECT TOP 100 *
8 FROM OPENROWSET( PROVIDER = 'CosmosDB',
9                  USER_ID = 'esucosmos',
10                 PASSWORD = 'esucosmos',
11                 SERVER_CREDENTIAL = 'esucosmos',
12                 SOURCE = 'Container1')
```

## Lab 2: Querying Azure Cosmos DB with Serverless SQL Pools

- 2 Run the first block query see if a credential is needed.

```
IF (NOT EXISTS(SELECT * FROM sys.credentials WHERE name = 'esucosmos'))  
    THROW 50000, 'As a prerequisite, create a credential with Azure Cosmos DB key in SECRET option:  
    CREATE CREDENTIAL [esucosmos]  
    WITH IDENTITY = 'SHARED ACCESS SIGNATURE', SECRET = 'a4WXnMntqa0yJfMR4D84rcfNgDCDR33dfo70PMeum8n0vNeshAEqkD0TdyycfyHUFyrDrQUAbWUNACDb8oZYVg==' ;  
GO
```

- 3 Create a credential first

```
CREATE CREDENTIAL [esucosmos]  
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',  
SECRET = 'a4WXnMntqa0yJfMR4D84rcfNgDCDR33dfo70PMeum8n0vNeshAEqkD0TdyycfyHUFyrDrQUAbWUNACDb8oZYVg==' ;
```

## Lab 2: Querying Azure Cosmos DB with Serverless SQL Pools

- 4 Run the query:

```
SELECT TOP 100 *  
FROM OPENROWSET( PROVIDER = 'CosmosDB',  
                  CONNECTION = 'Account=esucosmos;Database=CDB1',  
                  OBJECT = 'Container1',  
                  SERVER_CREDENTIAL = 'esucosmos'  
) AS [Container1]
```

# Lab 2: Querying Azure Cosmos DB with Serverless SQL Pools

```
10
11 SELECT TOP 100 *
12 FROM OPENROWSET([PROVIDER = 'CosmosDB',
13                 CONNECTION = 'Account=esucosmos;Database=CDB1',
14                 OBJECT = 'Container1',
15                 SERVER_CREDENTIAL = 'esucosmos'
16 ]) AS [Container1]
17
```

Results Messages

View **Table** Chart [Export results](#)

Search

description	category	_rid	_etag	_ts	id
various fruits	fruits	2J4oAMexroYBAAA...	"06007145-000...	1722190945	1
various vegetables	vegetables	2J4oAMexroYCAAA...	"06005846-000...	1722190981	2

## Lab 2: Querying Azure Cosmos DB with Serverless SQL Pools

- 5 Now let's create a database in Azure Synapse

```
-- Create the Profiles database if it does not exist
USE master;
GO
IF DB_ID(N'Profiles') IS NULL
BEGIN
    CREATE DATABASE Profiles;
END
GO

-- Switch to the Profiles database
USE Profiles;
GO

-- Drop the items view if it exists
DROP VIEW IF EXISTS items;
GO
```



## Lab 2: Querying Azure Cosmos DB with Serverless SQL Pools

```
-- Create the items view to access Cosmos DB
CREATE VIEW items
AS
SELECT
    *
FROM OPENROWSET(
    PROVIDER = 'CosmosDB',
    CONNECTION = 'Account=esucosmos;Database=CDB1',
    OBJECT = 'Container1',
    SERVER_CREDENTIAL = 'esucosmos'
)
WITH (
    id varchar(1000),
    category varchar(50),
    description varchar(max)
) AS profiles;
GO
```

# Lab 2: Querying Azure Cosmos DB with Serverless SQL Pools

- 6 Check the new database

```
USE Profiles;  
GO
```

```
SELECT TOP 10 *  
FROM items;
```

The screenshot shows the Azure Data Studio interface with a notebook titled 'Notebook 2' and a script named 'SQL script 1'. The script contains the following SQL code:

```
47 WITH (  
48     id varchar(1000),  
49     category varchar(50),  
50     description varchar(max)  
51 ) AS profiles;  
52 GO  
53  
54 USE Profiles;  
55 GO  
56  
57 SELECT TOP 10 *  
58 FROM items;  
59
```

The interface includes a toolbar with 'Run', 'Undo', 'Publish', and 'Query plan' buttons. The 'Connect to' dropdown is set to 'Built-in' and the 'Use database' dropdown is set to 'Profiles'. Below the script editor, the 'Results' tab is active, showing a table with the following data:

id	category	description
1	fruits	vrrious fruits
2	vegetables	various vegetables

# Lab 2: Querying Azure Cosmos DB with Serverless SQL Pools

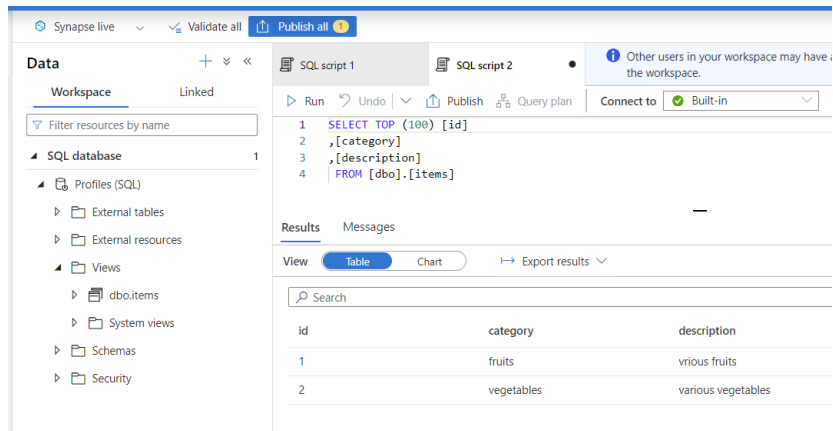
- Check the new database: Refresh the Synapse Studio, then go to “Data” tab, expand the “Profiles (SQL)” and under “Views” there must be a view created.

The screenshot shows the Microsoft Azure Synapse Studio interface. The top bar indicates the environment is 'Synapse Analytics' for workspace 'synapse728'. The left sidebar shows the 'Data' tab selected, with 'Workspace' and 'Views' highlighted. The 'Views' folder is expanded, showing a list of views including 'dbo.items', 'System views', 'Schemas', and 'Security'. The main pane displays a T-SQL script in the 'SQL script 1' editor. The script includes a conditional statement to create a credential if it does not exist, followed by a query using the OPENROWSET function to access data from an external source.

```
1 IF (NOT EXISTS(SELECT * FROM sys.c
2     THROW 50000, 'As a prerequisite
3     CREATE CREDENTIAL [esucosmos]
4     WITH IDENTITY = ''SHARED ACCESS
5 GO
6
7 CREATE CREDENTIAL [esucosmos]
8 WITH IDENTITY = 'SHARED ACCESS SIGI
9 SECRET = 'a4wXnMntqa0yJfMR4D84rcfN,
10
11 SELECT TOP 100 *
12 FROM OPENROWSET( PROVIDER = 'Cosmos
13                  CONNECTION = 'Acco
14                  OBJECT = 'Containe
15                  SERVER_CREDENTIAL :
16 ) AS [Container1]
17
18
19
20 -- Create the Profiles database if
```

# Lab 2: Querying Azure Cosmos DB with Serverless SQL Pools

- 8 Click on “dbo.items”, choose “New SQL script”, click “Select Top 100 rows” and you will be able to see the auto generated query. Run the query.



The screenshot shows the Synapse Studio interface. On the left, the 'Data' pane shows the 'Workspace' with a 'Filter resources by name' search bar. Under 'SQL database', there is a 'Profiles (SQL)' section with a tree view showing 'External tables', 'External resources', 'Views', 'dbo.items', 'System views', 'Schemas', and 'Security'. The 'dbo.items' view is selected. The main pane shows 'SQL script 1' with the following query:

```
1 SELECT TOP (100) [id]
2 , [category]
3 , [description]
4 FROM [dbo].[items]
```

Below the query, the 'Results' tab is active, showing a table view of the results. The table has three columns: 'id', 'category', and 'description'. The results are as follows:

id	category	description
1	fruits	vrrious fruits
2	vegetables	various vegetables

# License



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#).