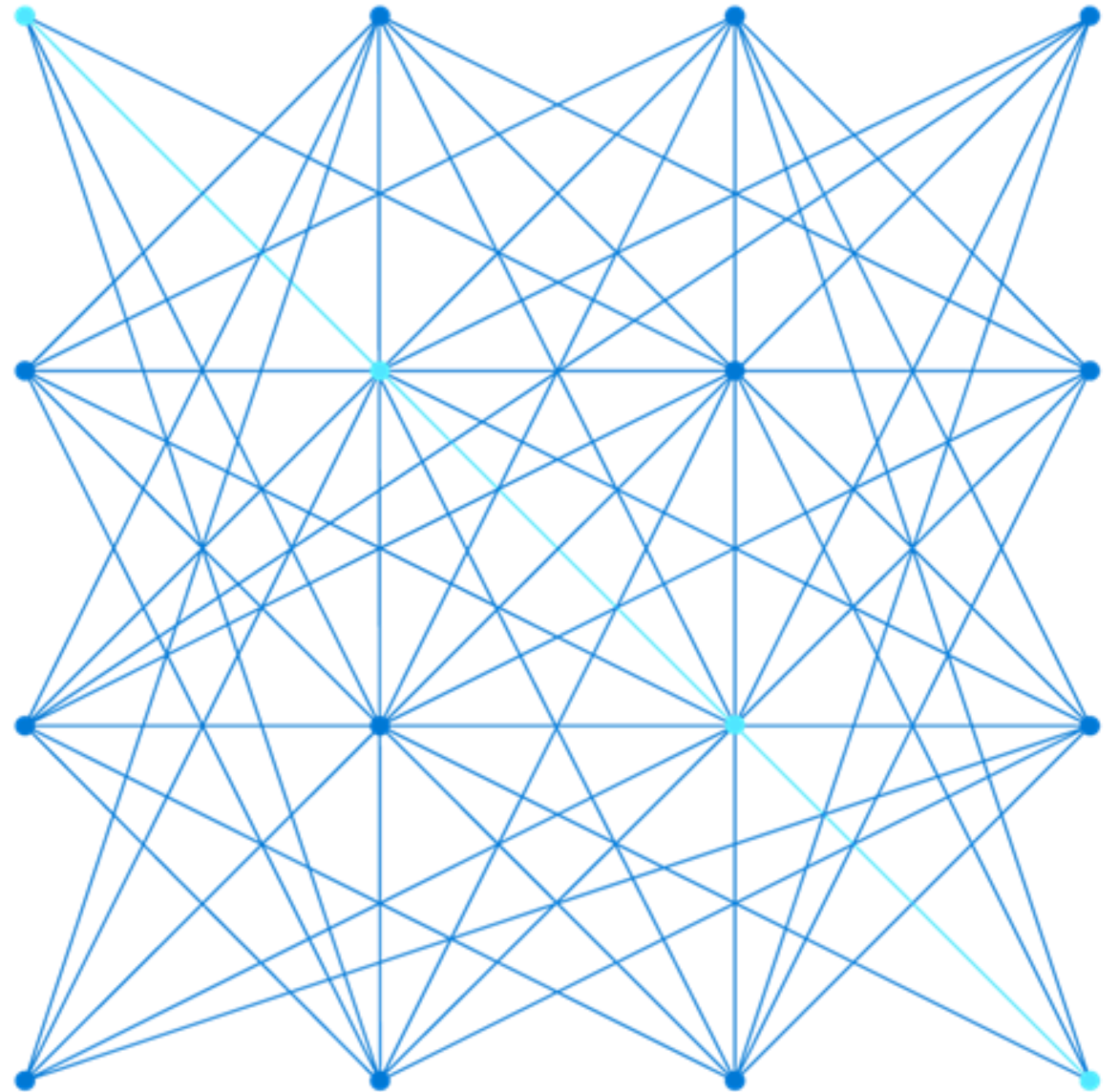


# DP-203T00: Support Hybrid Transactional Analytical Processing (HTAP) with Azure Synapse Link



# Lesson 01: Design hybrid transactional and analytical processing using Azure Synapse Analytics



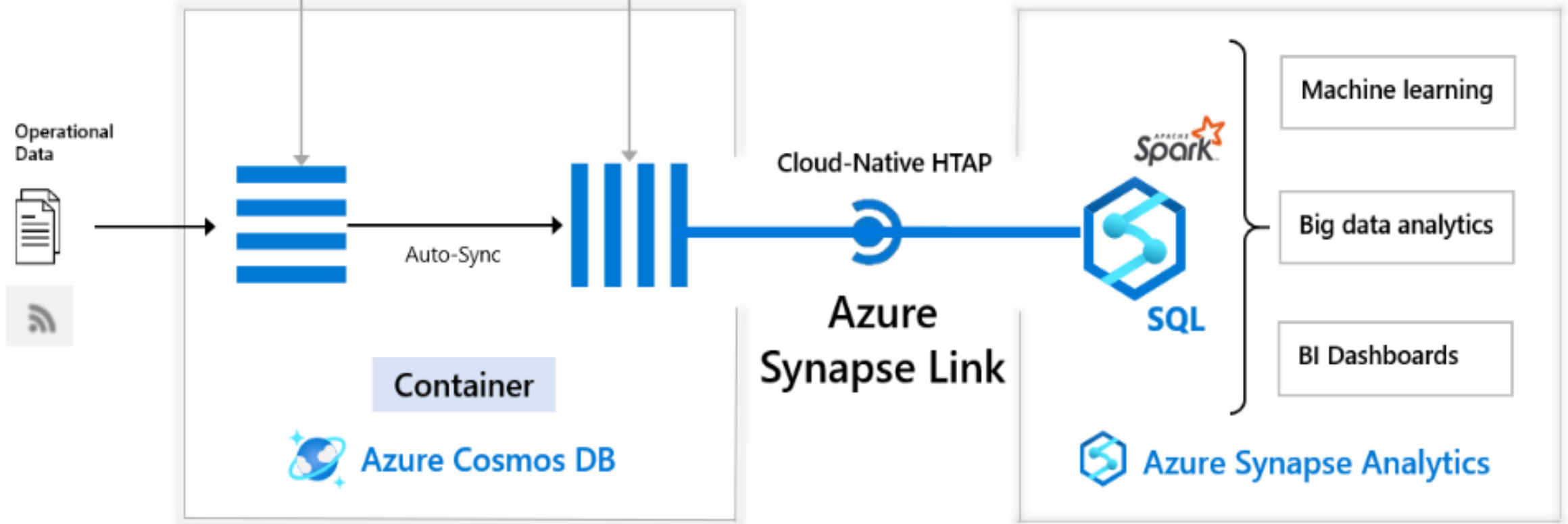
# Design hybrid transactional and analytical processing using Azure Synapse Analytics

## Transactional Store

Row store optimized for transactional reads and writes

## Analytical Store

Column store optimized for analytical queries



## Lesson 02: Configure Azure Synapse Link with Azure Cosmos DB



Home > [datasharerg](#) > [cosmosdbtestsynapse](#)

# cosmosdbtestsynapse | Features

Azure Cosmos DB account

[Refresh](#)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Quick start

Notifications

Data Explorer

Settings

Features

Feature

Azure Synapse Link

## Azure Synapse Link

Azure Synapse Link for Cosmos DB creates a tight integration between Azure Cosmos DB and Azure Synapse Analytics enabling customers to run near real-time analytics over their operational data with no-ETL and full performance isolation from their transactional workloads.

By combining the distributed scale of Cosmos DB's transactional processing with build-in analytical store and the computing power of Azure Synapse Analytics, Azure Synapse Link enables Hybrid Transactional/Analytical Processing (HTAP) architectures for optimizing your business processes. This integration eliminates ETL processes, enabling business analysts, data engineers & data scientists to self-serve and run near real-time BI, analytics and ML pipelines over operational data.

[Learn More](#)

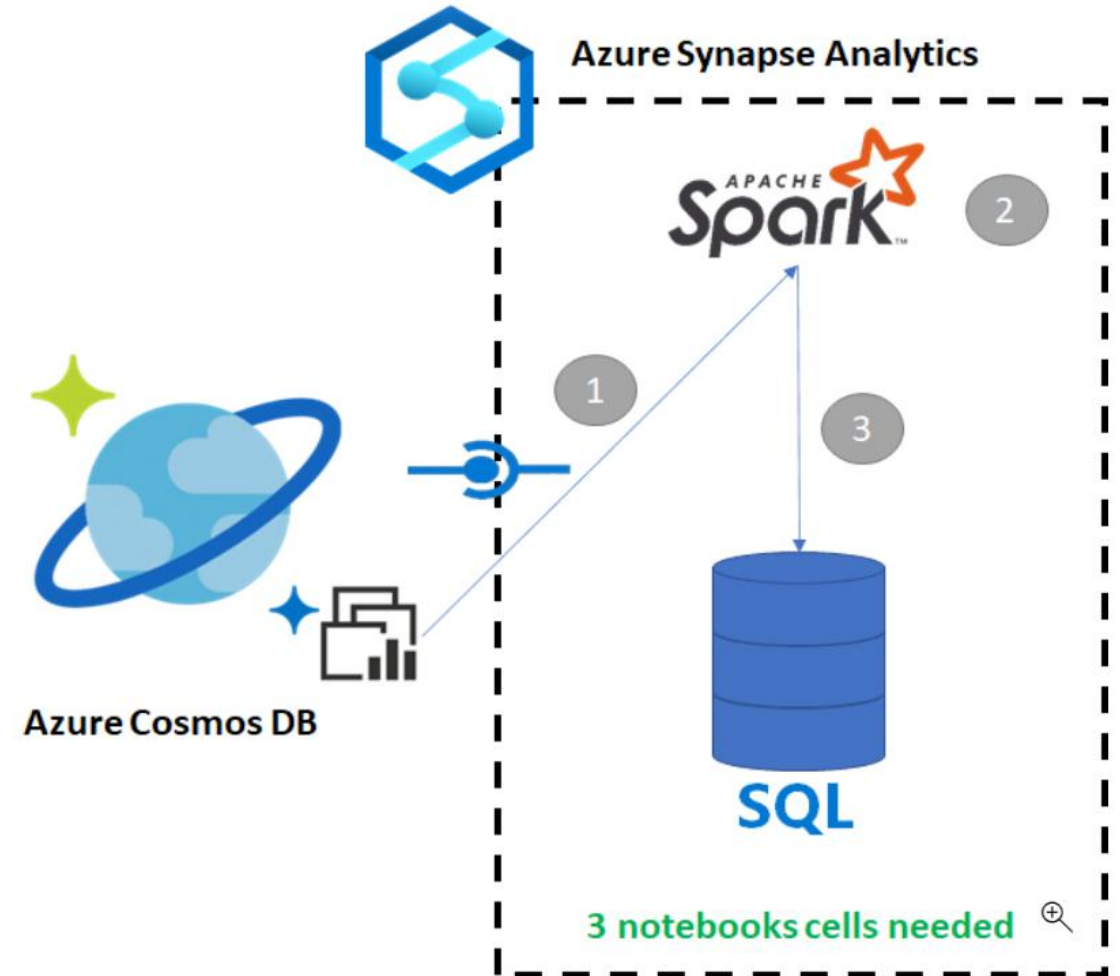
EnableClose

## Lesson 03: Query Azure Cosmos DB with Apache Spark for Azure Synapse Analytics



# Query Azure Cosmos DB with Apache Spark for Azure Synapse Analytics

- Step 1: Load the data in Spark
- Step 2: Create a base DataFrame
- Step 3: Flatten JSON data
- Step 4: Create the final DataFrame



# Query Azure Cosmos DB with Apache Spark for Azure Synapse Analytics (continued #1)

Step 1:  
Load data to  
DataFrame



Notebook 2

+ Cell Run all Undo Publish Attach to SparkPool01 Language PySpark (Python)

Cell 1

```
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.preferredRegion
3
4 df = spark.read\
5     .format("cosmos.olap")\
6     .option("spark.synapse.linkedService", "asacosmosdb01")\
7     .option("spark.cosmos.container", "UserProfileHTAP")\
8     .load()
9
10 display(df.limit(10))
```

Command executed in 3mins 33s 728ms by joel on 09-16-2020 00:22:16.665 -04:00

> Job execution Succeeded Spark 2 executors 8 cores View in monitoring Open Spark UI

View Table Chart

_rid	_ts	userId	cartId	preferredProducts
GpNQALvEY79PNQAAAAA==	1600214048	33304	af8aa699-c371-460f-ba4e-be8e4...	"[2549,1841,2089,2874



# Query Azure Cosmos DB with Apache Spark for Azure Synapse Analytics (continued #2)

Step 2:  
Create a base  
DataFrame



Cell 2

```
1 unwanted_cols = {'_attachments', '_etag', '_rid', '_self', '_ts', 'collectionType', 'id'}
2
3 # Remove unwanted columns from the columns collection
4 cols = list(set(df.columns) - unwanted_cols)
5
6 profiles = df.select(cols)
7
8 display(profiles.limit(10))
```

Command executed in 4s 513ms by joel on 09-16-2020 00:36:56.940 -04:00

> Job execution Succeeded Spark 2 executors 8 cores [View in monitoring](#)

View Table Chart

preferredProducts	userId	productReviews	cartId
▼ "[2549,1841,2089,2874,185,2842,4!] 0: "2549" 1: "1841" 2: "2089" 3: "2874" 4: "185" 5: "2842" 6: "4577" 7: "1448" 8: "1294"	33304	▼ "[{"productId":2901,"reviewText":"c ▶ 0: "{"productId":2901,"reviewTexi ▶ 1: "{"productId":738,"reviewText" ▶ 2: "{"productId":1986,"reviewTex ▶ 3: "{"productId":1163,"reviewTex	af8aa699-c371-460f-ba4e-be8e4...
▶ "[1935,1738,957,1992,2290,1495,4:	18933	▶ "[{"productId":34,"reviewText":"My	1c438ce1-d26c-4283-a8d7-7fe8...
▶ "[3704,1323,1614,438,3526,3789]"	35087	"[]"	6a25d03b-f6d1-41a3-8e0d-2e3a...
▶ "[1999,4564,2629,4986,3959,4592,i	22258	▶ "[{"productId":4999,"reviewText":"T	2c430f0f-769a-4d61-b67b-f5852...

# Query Azure Cosmos DB with Apache Spark for Azure Synapse Analytics (continued #3)

## Step 3: Flattening JSON data



Cell 4

```
1 from pyspark.sql.functions import udf, explode
2
3 preferredProductsFlat=profiles.select('userId',explode('preferredProducts').alias('productId'))
4 productReviewsFlat=profiles.select('userId',explode('productReviews').alias('productReviews'))
5 display(productReviewsFlat.limit(10))
```

Command executed in 2s 677ms by joel on 09-16-2020 01:13:23.535 -04:00

> **Job execution** Succeeded **Spark 2** executors 8 cores [View i](#)

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

userId	productReviews
52563	<div>▼ [{"productId":2793,"reviewText":"he productId: "2793" reviewText: ""heard about this on reviewDate: ""2019-05-19T00:30::</div>
52563	<div>▶ [{"productId":1486,"reviewText":"M</div>
52563	<div>▶ [{"productId":4959,"reviewText":"he</div>

# Query Azure Cosmos DB with Apache Spark for Azure Synapse Analytics (continued #4)

## Step 4: Creating the final DataFrame



Cell 7

```
1 preferredProductReviews = (preferredProductsFlat.join(productReviews,  
2 (preferredProductsFlat.userId == productReviews.userId) &  
3 (preferredProductsFlat.productId == productReviews.productId))  
4 )  
5  
6 display(preferredProductReviews.limit(100))
```

Command executed in 2s 707ms by joel on 09-16-2020 01:26:23.275 -04:00

> **Job execution** Succeeded **Spark** 2 executors 8 cores

View **Table** Chart

userId ↑	productId	reviewText
2382	1954	My raven loves to play with it.
2662	174	talk about contentment!!!
2863	4003	talk about contempt!!!
3496	2044	The box this comes in is 3 centim...
3616	3035	The box this comes in is 3 kilome...
3892	2297	My Shih-Tzu loves to play with it.
3971	2624	My goldfinch loves to play with it.
4684	2739	This deliverables works certainly ...
5014	319	this Graphic Interface is vertical

## Lesson 04: Query Azure Cosmos DB with SQL Serverless for Azure Synapse Analytics



# Query Azure Cosmos DB with SQL Serverless for Azure Synapse Analytics

## Step 1: Create a View



```
15
16 CREATE VIEW UserProfileHTAP 1
17 AS
18 SELECT
19     *
20 FROM OPENROWSET( 2
21     'CosmosDB',
22     N'account=asacosmosdbinaday84;database=CustomerProfile;key=1kxCTXbqWQ8wf0ojuzVQjCbFuAsoV6rLMR7KyD
23     UserProfileHTAP
24 )
25 WITH (
26     userId bigint,
27     cartId varchar(50),
28     preferredProducts varchar(max), 3
29     productReviews varchar(max)
30 ) AS profiles
31 CROSS APPLY OPENJSON (productReviews)
32 WITH (
33     productId bigint,
34     reviewText varchar(1000) 4
35 ) AS reviews
36 GO
```

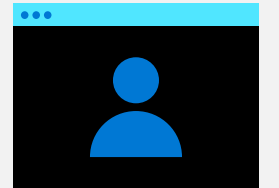
Results Messages

11:46:22 AM Started executing query at Line 1 5

(Changed database context to 'Profiles'. Changed database context to 'master'.)  
Total execution time: 00:00:04.304

✓ 00:00:04 Query executed successfully.

# Lab: Support Hybrid Transactional Analytical Processing (HTAP) with Azure Synapse Link



# Lab overview

This lab teaches you how Azure Synapse Link enables seamless connectivity of an Azure Cosmos DB account to an Azure Synapse workspace. You will understand how to enable and configure Synapse link, then how to query the Azure Cosmos DB analytical store using Apache Spark and SQL Serverless.

## Lab objectives

After completing this lab, you will be able to:

Configure Azure Synapse Link with Azure Cosmos DB

Query Azure Cosmos DB with Apache Spark for Synapse Analytics

Query Azure Cosmos DB with serverless SQL pool for Azure Synapse Analytics