100 XP

Introduction

1 minute

Hybrid Transactional / Analytical Processing (HTAP) is a style of data processing that combines transactional data processing, such as is typically found in a business application, with analytical processing, such as is used in a business intelligence (BI) or reporting solution. The data access patterns and storage optimizations used in these two kinds of workload are very different, so usually a complex extract, transform, and load (ETL) process is required to copy data out of transactional systems and into analytical systems; adding complexity and latency to data analysis. In an HTAP solution, the transactional data is replicated automatically, with low-latency, to an analytical store, where it can be queried without impacting the performance of the transactional system.

In Azure Synapse Analytics, HTAP capabilities are provided by multiple **Azure Synapse Link** services, each connecting a commonly used transactional data store to your Azure Synapse Analytics workspace and making the data available for processing using Spark or SQL.

After completing this module, you'll be able to:

- Describe Hybrid Transactional / Analytical Processing patterns.
- Identify Azure Synapse Link services for HTAP.

Next unit: Understand hybrid transactional and analytical processing patterns

Continue

Understand hybrid transactional and analytical processing patterns

6 minutes

Many business application architectures separate transactional and analytical processing into separate systems with data stored and processed on separate infrastructures. These infrastructures are commonly referred to as OLTP (online transaction processing) systems working with operational data, and OLAP (online analytical processing) systems working with historical data, with each system is optimized for their specific task.

OLTP systems are optimized for dealing with discrete system or user requests immediately and responding as quickly as possible.

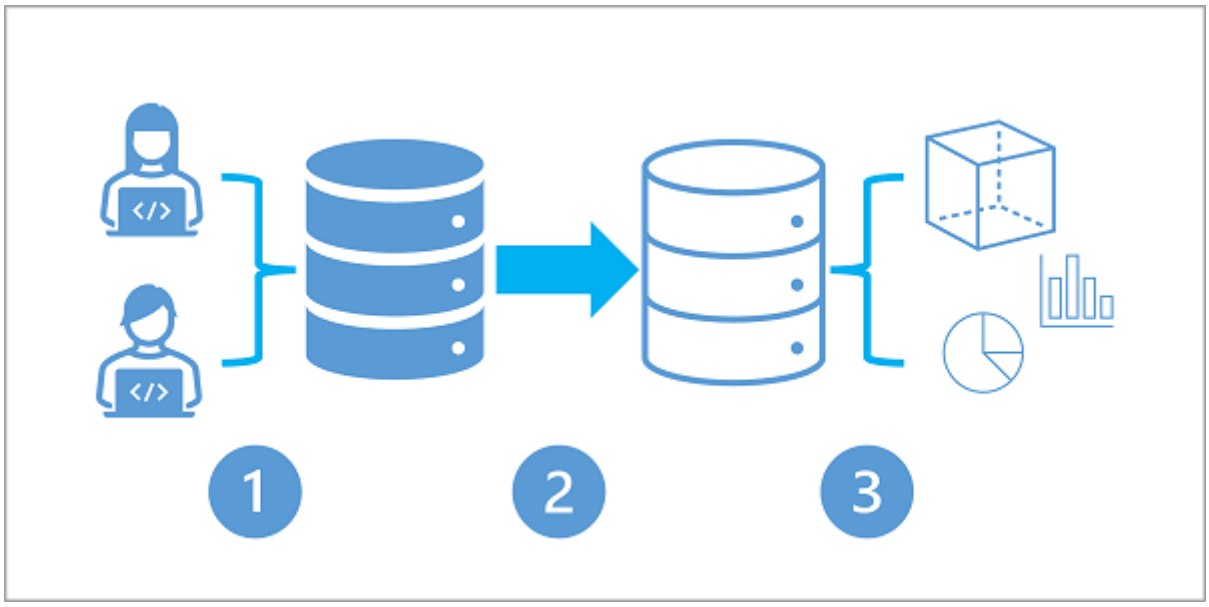
OLAP systems are optimized for the analytical processing, ingesting, synthesizing, and managing large sets of historical data. The data processed by OLAP systems largely originates from OLTP systems and needs to be loaded into the OLAP systems by ETL (Extract, Transform, and Load) batch processes.

Due to their complexity and the need to physically copy large amounts of data, this approach creates a delay in data being available to analyze in OLAP systems.

Hybrid Transactional / Analytical Processing (HTAP)

As more businesses move to digital processes, they increasingly recognize the value of being able to respond to opportunities by making faster and well-informed decisions. HTAP (Hybrid Transactional/Analytical processing) enables business to run advanced analytics in near-real-time on data stored and processed by OLTP systems.

The following diagram illustrates the generalized pattern of an HTAP architecture:



1. A business application processes user input and stores data in a transactional database that is optimized for a mix of data reads and writes based on the application's expected usage profile.
2. The application data is automatically replicated to an analytical store with low latency.
3. The analytical store supports data modeling, analytics, and reporting without impacting the transactional system.

Next unit: Describe Azure Synapse Link

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Describe Azure Synapse Link

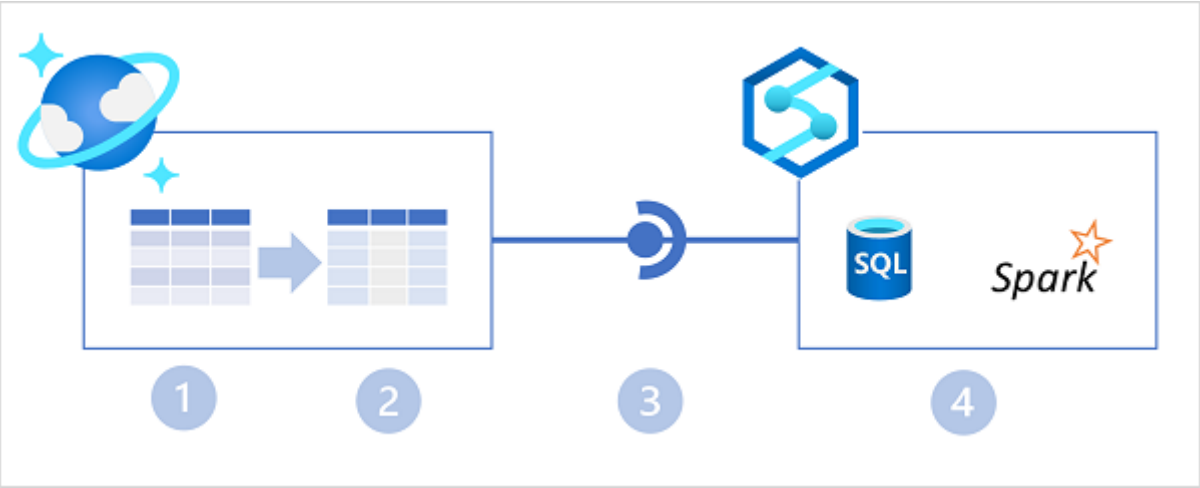
6 minutes

HTAP solutions are supported in Azure Synapse Analytics through **Azure Synapse Link**; a general term for a set of linked services that support HTAP data synchronization into your Azure Synapse Analytics workspace.

Azure Synapse Link for Cosmos DB

Azure Cosmos DB is a global-scale NoSQL data service in Microsoft Azure that enables applications to store and access operational data by using a choice of application programming interfaces (APIs).

Azure Synapse Link for Azure Cosmos DB is a cloud-native HTAP capability that enables you to run near-real-time analytics over operational data stored in a Cosmos DB container. Azure Synapse Link creates a tight seamless integration between Azure Cosmos DB and Azure Synapse Analytics.



In the diagram above, the following key features of the Azure Synapse Link for Cosmos DB architecture are illustrated:

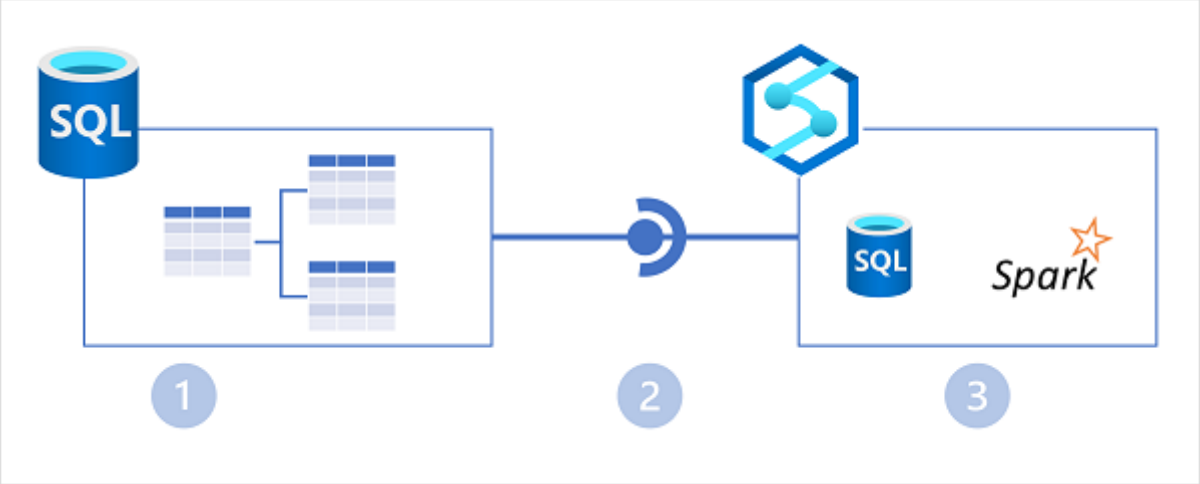
1. An Azure Cosmos DB container provides a row-based transactional store that is optimized for read/write operations.
2. The container also provides a column-based analytical store that is optimized for analytical workloads. A fully managed autosync process keeps the data stores in sync.
3. Azure Synapse Link provides a linked service that connects the analytical store enabled container in Azure Cosmos DB to an Azure Synapse Analytics workspace.

4. Azure Synapse Analytics provides Synapse SQL and Apache Spark runtimes in which you can run code to retrieve, process, and analyze data from the Azure Cosmos DB analytical store without impacting the transactional data store in Azure Cosmos DB.

Azure Synapse Link for SQL

Microsoft SQL Server is a popular relational database system that powers business applications in some of the world's largest organizations. Azure SQL Database is a cloud-based platform-as-a-service database solution based on SQL Server. Both of these relational database solutions are commonly used as operational data stores.

Azure Synapse Link for SQL enables HTAP integration between data in SQL Server or Azure SQL Database and an Azure Synapse Analytics workspace.



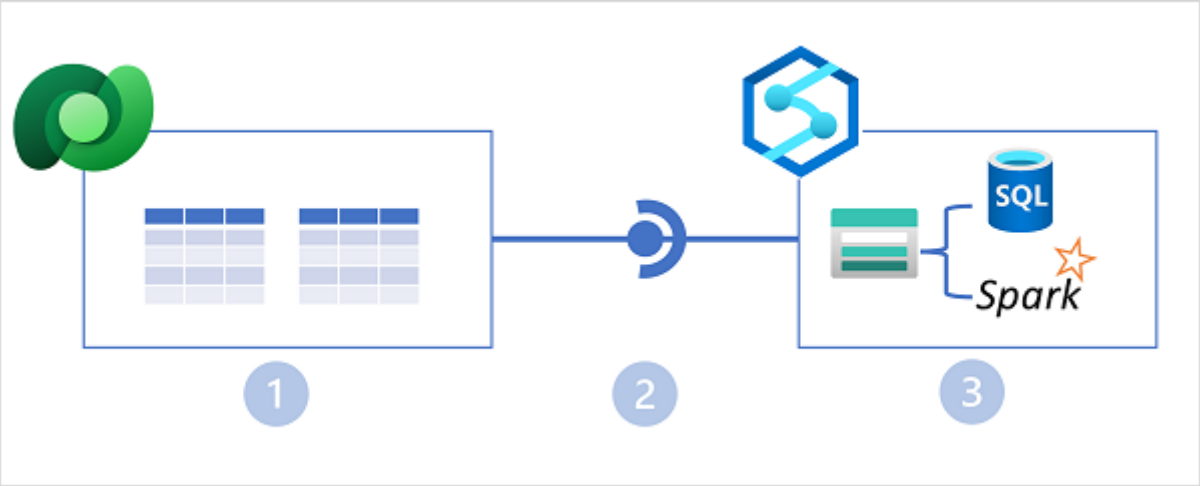
In the diagram above, the following key features of the Azure Synapse Link for SQL architecture are illustrated:

1. An Azure SQL Database or SQL Server instance contains a relational database in which transactional data is stored in tables.
2. Azure Synapse Link for SQL replicates the table data to a dedicated SQL pool in an Azure Synapse workspace.
3. The replicated data in the dedicated SQL pool can be queried in the dedicated SQL pool, or connected to as an external source from a Spark pool without impacting the source database.

Azure Synapse Link for Dataverse

Microsoft Dataverse is data storage service within the Microsoft Power Platform. You can use Dataverse to store business data in tables that are accessed by Power Apps, Power BI, Power Virtual Agents, and other applications and services across Microsoft 365, Dynamics 365, and Azure.

Azure Synapse Link for Dataverse enables HTAP integration by replicating table data to Azure Data Lake storage, where it can be accessed by runtimes in Azure Synapse Analytics - either directly from the data lake or through a Lake Database defined in a serverless SQL pool.



In the diagram above, the following key features of the Azure Synapse Link for Dataverse architecture are illustrated:

1. Business applications store data in Microsoft Dataverse tables.
2. Azure Synapse Link for Dataverse replicates the table data to an Azure Data Lake Gen2 storage account associated with an Azure Synapse workspace.
3. The data in the data lake can be used to define tables in a lake database and queried using a serverless SQL pool, or read directly from storage using SQL or Spark.

Next unit: Knowledge check

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

200 XP

Knowledge check

5 minutes

1. Which of the following descriptions matches a hybrid transactional/analytical processing (HTAP) architecture. *

- Business applications store data in an operational data store, which is also used to support analytical queries for reporting.
- Business applications store data in an operational data store, which is synchronized with low latency to a separate analytical store for reporting and analysis.

✔ **Correct. an HTAP solution replicates operational data to an analytical store, enabling you to perform analytics and reporting without impacting the performance of the operational system.**

- Business applications store operational data in an analytical data store that is optimized for queries to support reporting and analysis.

2. You want to use Azure Synapse Analytics to analyze operational data stored in a Cosmos DB for NoSQL container. Which Azure Synapse Link service should you use? *

- Azure Synapse Link for SQL
- Azure Synapse Link for Dataverse

✘ **Incorrect. Azure Synapse Link for Dataverse integrates with the Dataverse Power Platform data store.**

- Azure Synapse Link for Azure Cosmos DB

✔ **Correct. Azure Synapse Link for Azure Cosmos DB integrates with multiple Azure Cosmos DB APIs, including Azure Cosmos DB for NoSQL.**

3. You plan to use Azure Synapse Link for Dataverse to analyze business data in your Azure Synapse Analytics workspace. Where is the replicated data from Dataverse stored? *

- In an Azure Synapse dedicated SQL pool

- ◯ In an Azure Data Lake Gen2 storage container.

  ✔ **Correct. Azure Synapse Link for Dataverse replicates data to an Azure Data Lake Gen2 storage account.**

- ◯ In an Azure Cosmos DB container.

Next unit: Summary

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

Introduction

1 minute

Azure Synapse Analytics Link for Cosmos DB enables *hybrid transactional/analytical processing* (HTAP) integration between Azure Cosmos DB and Azure Synapse Analytics. By using this HTAP solution, organizations can make operational data in Azure Cosmos DB available for analysis and reporting in Azure Synapse Analytics in near-real time without the need to develop a complex ETL pipeline.

In this module, you'll learn how to:

- Configure an Azure Cosmos DB account to use Azure Synapse Link.
- Create an analytical store enabled container.
- Create a linked service for Azure Cosmos DB.
- Analyze linked data using Spark.
- Analyze linked data using Synapse SQL.

Next unit: Enable Cosmos DB account to use Azure Synapse Link

Continue >

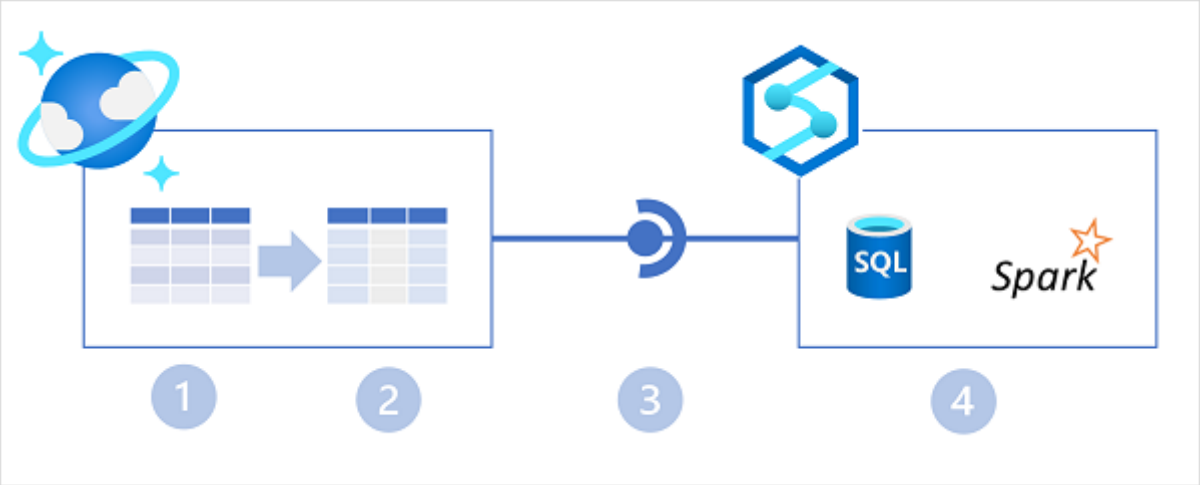
---

100 XP

Enable Cosmos DB account to use Azure Synapse Link

5 minutes

Azure Synapse Link for Azure Cosmos DB is a cloud-native HTAP capability that enables integration between Azure Cosmos DB and Azure Synapse Analytics.



In the diagram above, the following key features of the Azure Synapse Link for Cosmos DB architecture are illustrated:

1. An Azure Cosmos DB container provides a row-based transactional store that is optimized for read/write operations.
2. The container also provides a column-based analytical store that is optimized for analytical workloads. A fully managed autosync process keeps the data stores in sync.
3. Azure Synapse Link provides a linked service that connects the analytical store enabled container in Azure Cosmos DB to an Azure Synapse Analytics workspace.
4. Azure Synapse Analytics provides Synapse SQL and Apache Spark runtimes in which you can run code to retrieve, process, and analyze data from the Azure Cosmos DB analytical store without impacting the transactional data store in Azure Cosmos DB.

Enabling Azure Synapse Link in Azure Cosmos DB

The first step in using Azure Synapse Link for Cosmos DB is to enable it in an Azure Cosmos DB account. Azure Synapse Link is supported in the following types of Azure Cosmos DB account:

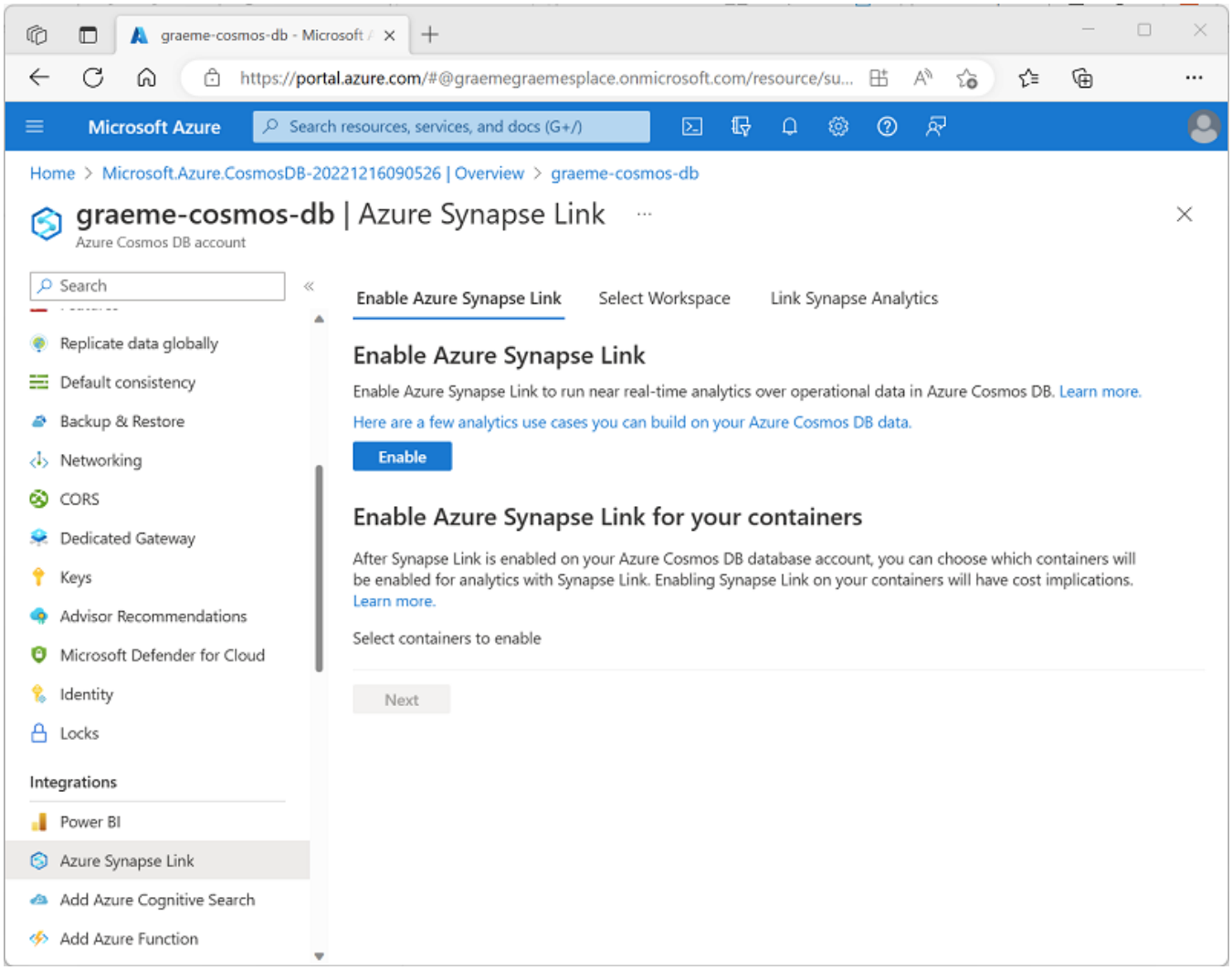
- Azure Cosmos DB for NoSQL
- Azure Cosmos DB for MongoDB

- Azure Cosmos DB for Apache Gremlin (*preview*)

You can enable Azure Synapse Link in the Azure portal page for your Cosmos DB account, or by using the Azure CLI or Azure PowerShell from a command line or in a script.

Using the Azure portal

In the Azure portal, you can enable Azure Synapse Link for a Cosmos DB account on the **Azure Synapse Link** page in the **Integrations** section, as shown below.



> **Tip**
>
> For Azure Cosmos DB for NoSQL accounts, there's also a link on the **Data Explorer** page.

Using the Azure CLI

To enable Azure Synapse Link using the Azure CLI, run the `az cosmosdb create` command (to create a new Cosmos DB account) or `az cosmosdb update` command (to configure an existing

Cosmos DB account) with the `--enable-analytical-storage true` parameter. For example, the following command updates an existing Cosmos DB account named **my-cosmos-db** to enable Azure Synapse Link.

```
az cosmosdb update --name my-cosmos-db --resource-group my-rg --enable-analytical-storage true
```

To enable Azure Synapse Link for an Azure Cosmos DB for Apache Gremlin account, include the `--capabilities EnableGremlin` parameter.

Using Azure PowerShell

To enable Azure Synapse Link using Azure PowerShell, run the `New-AzCosmosDBAccount` cmdlet (to create a new Cosmos DB account) or `Update-AzCosmosDBAccount` cmdlet (to configure an existing Cosmos DB account) with the `-EnableAnalyticalStorage 1` parameter. For example, the following command updates an existing Cosmos DB account named **my-cosmos-db** to enable Azure Synapse Link.

```
Update-AzCosmosDBAccount -Name "my-cosmos-db" -ResourceGroupName "my-rg" -EnableAnalyticalStorage 1
```

Considerations for enabling Azure Synapse Link

When planning to enable Azure Synapse Link for a Cosmos DB account, consider the following facts:

- After enabling Azure Synapse Link for an account, you can't disable it.
- Enabling Azure Synapse Link doesn't start synchronization of operational data to an analytical store - you must also create or update a container with support for an analytical store.
- When enabling Azure Synapse Link for a Cosmos DB for NoSQL account using the Azure CLI or PowerShell, you can use the `--analytical-storage-schema-type` (Azure CLI) or `-AnalyticalStorageSchemaType` (PowerShell) parameter to specify the schema type as `WellDefined` (default) or `FullFidelity`. For a Cosmos DB for MongoDB account, the default (and only supported) schema type is `FullFidelity`.

- After a schema type has been assigned, you can't change it.

> **Note**
>
> You'll learn more about the analytical store and its schema types in the next unit.

Next unit: Create an analytical store enabled container

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Create an analytical store enabled container

5 minutes

After enabling Azure Synapse Link in an Azure Cosmos DB account, you can create or update a container with support for an analytical store.

An analytical store is a column-based store within the same container as a row-based operational store. An *auto-sync* process synchronizes changes in the operational store to the analytical store; from where it can be queried without incurring processing overhead in the operational store.

Analytical store schema types

As the data from the operational store is synchronized to the analytical store, the schema is updated dynamically to reflect the structure of the documents being synchronized. The specific behavior of this dynamic schema maintenance depends on the analytical store schema type configured for the Azure Cosmos DB account. Two types of schema representation are supported:

- **Well-defined**: The default schema type for an Azure Cosmos DB for NoSQL account.
- **Full fidelity**: The default (and only supported) schema type for an Azure Cosmos DB for MongoDB account.

The analytical store receives JSON data from the operational store and organizes it into a column-based structure. In a well-defined schema, the first non-null occurrence of a JSON field determines the data type for that field. Subsequent occurrences of the field that aren't compatible with the assigned data type aren't ingested into the analytical store.

For example, consider the following two JSON documents:

JSON

```
{"productID": 123, "productName": "Widget"}
{"productID": "124", "productName": "Wotsit"}
```

The first document determines that the **productID** field is a numeric (integer) value. When the second document is encountered, its **productID** field has a string value, and so isn't imported

into the analytical store. The document and the rest of its field is imported, but the incompatible field is dropped. The following columns represent the data in the analytical store:

productID	productName
123	Widget
	Wotsit

In a full fidelity schema, the data type is appended to each instance of the field, with new columns created as necessary; enabling the analytical store to contain multiple occurrences of a field, each with a different data type, as shown in the following table:

productID.int32	productName.string	productID.string
123	Widget	
	Wotsit	124

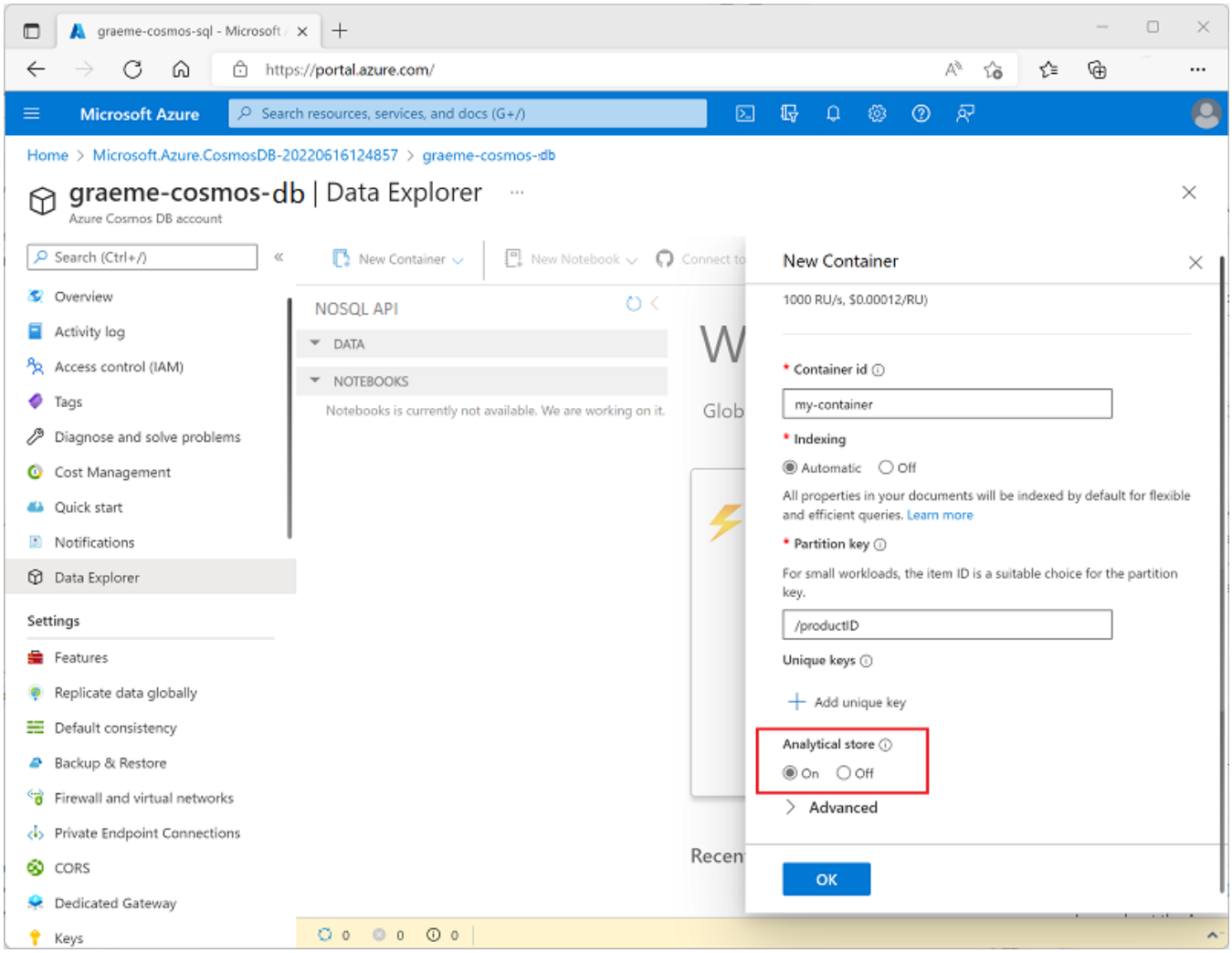
> **Note**
>
> For more information, see **What is Azure Cosmos DB analytical store?**.

Enabling analytical store support in a container

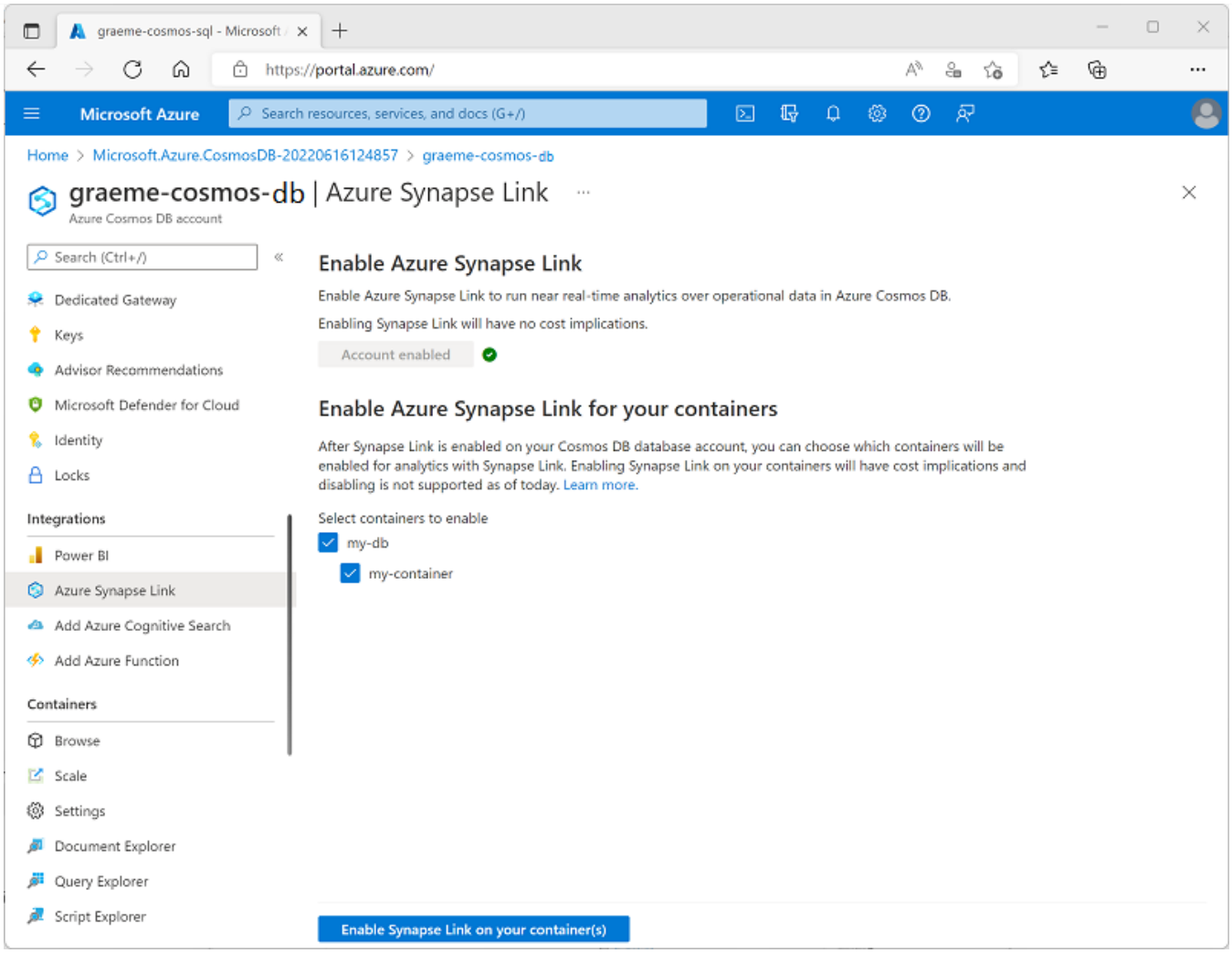
You can enable analytical store support when creating a new container or for an existing container. To enable analytical store support, you can use the Azure portal, or you can use the Azure CLI or Azure PowerShell from a command line or in a script.

Using the Azure portal

To enable analytical store support when creating a new container in the Azure portal, select the **On** option for **Analytical Store**, as shown here:



Alternatively, you can enable analytical store support for an existing container in the **Azure Synapse Link** page in the **Integrations** section of the page for your Cosmos DB account, as shown here:



Using the Azure CLI

To use the Azure CLI to enable analytical store support in an Azure Cosmos DB for NoSQL container, run the `az cosmosdb sql container create` command (to create a new container) or `az cosmosdb sql container update` command (to configure an existing container) with the `--analytical-storage-ttl` parameter, assigning a retention time for analytical data. Specifying an `-analytical-storage-ttl` parameter of **-1** enables permanent retention of analytical data. For example, the following command creates a new container named **my-container** with analytical store support.

```
az cosmosdb sql container create --resource-group my-rg --account-name my-cosmos-db --database-name my-db --name my-container --partition-key-path "/productID" --analytical-storage-ttl -1
```

For an Azure Cosmos DB for MongoDB account, you can use the `az cosmosdb mongodb collection create` or `az cosmosdb mongodb collection update` command with the `--analytical-storage-ttl` parameter. For an Azure Cosmos DB for Apache Gremlin account,

use the `az cosmosdb gremlin graph create` or `az cosmosdb gremlin graph update` command with the `--analytical-storage-ttl` parameter.

Using Azure PowerShell

To use Azure PowerShell to enable analytical store support in n Azure Cosmos DB for NoSQL container, run the `New-AzCosmosDBSqlContainer` cmdlet (to create a new container) or `Update-AzCosmosDBSqlContainer` cmdlet (to configure an existing container) with the `-AnalyticalStorageTtl` parameter, assigning a retention time for analytical data. Specifying an `-AnalyticalStorageTtl` parameter of **-1** enables permanent retention of analytical data. For example, the following command creates a new container named **my-container** with analytical store support.

```
New-AzCosmosDBSqlContainer -ResourceGroupName "my-rg" -AccountName "my-cosmos-db" -DatabaseName "my-db" -Name "my-container" -PartitionKeyKind "hash" -PartitionKeyPath "/productID" -AnalyticalStorageTtl -1
```

For an Azure Cosmos DB for MongoDB API account, use the `New-AzCosmosDBMongoDBCollection` or `Update-AzCosmosDBMongoDBCollection` cmdlet with the `-AnalyticalStorageTtl` parameter.

Considerations for enabling analytical store support

Analytical store support can't be disabled without deleting the container. Setting the analytical store TTL value to 0 or *null* effectively disables the analytical store by no longer synchronizing new items to it from the operational store and deleting items already synchronized from the analytical store. After setting this value to 0, you can't re-enable analytical store support in the container.

Next unit: Create a linked service for Cosmos DB

Continue >

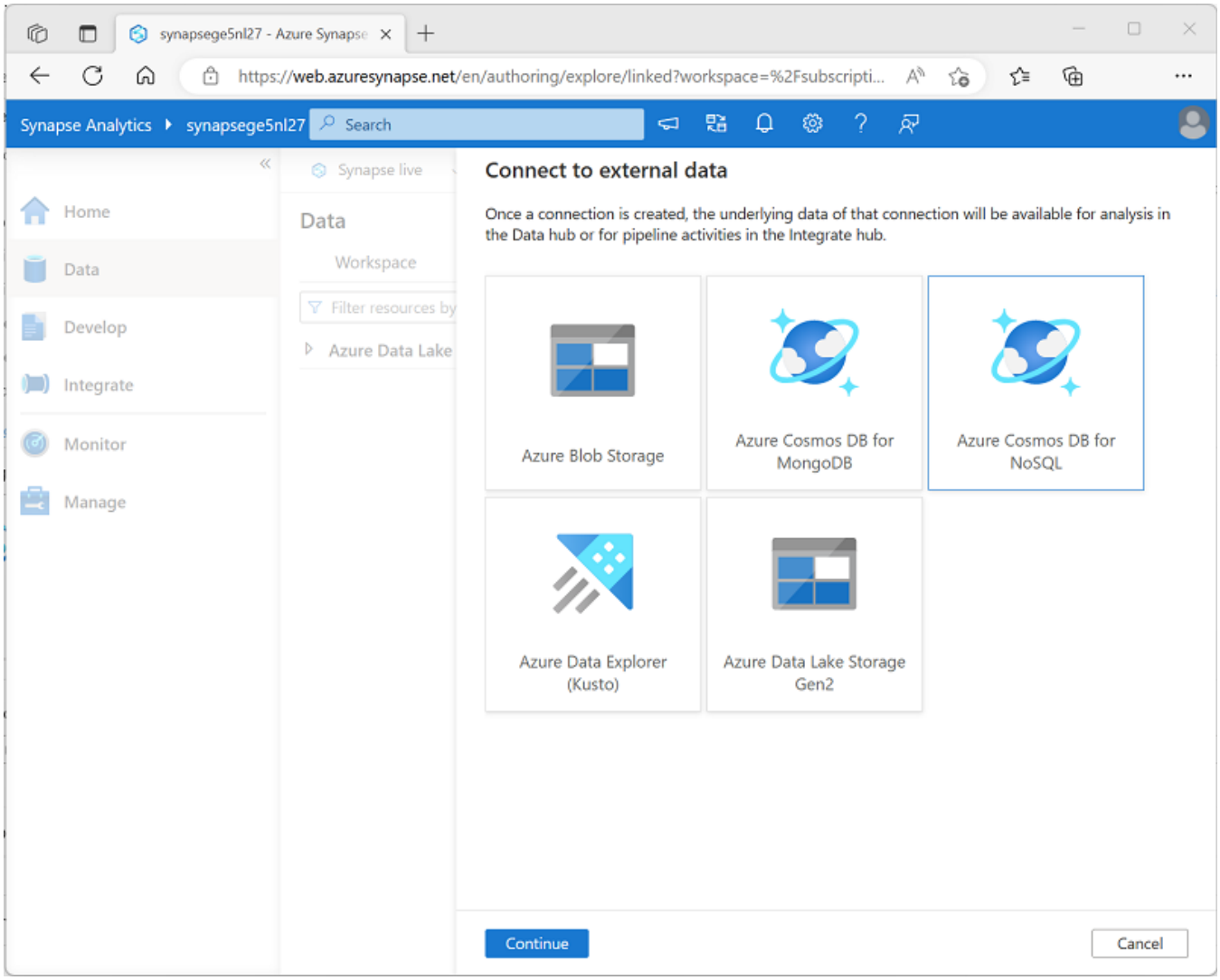
100 XP

Create a linked service for Cosmos DB

5 minutes

When you have an Azure Cosmos DB container with analytical store support, you can create a linked service in an Azure Synapse Analytics workspace to connect to it.

To create a linked service to an Azure Cosmos DB analytical data store, use Azure Synapse Studio, and add a linked service on the **Data** page by selecting the **Connect to external data** option, as shown here:



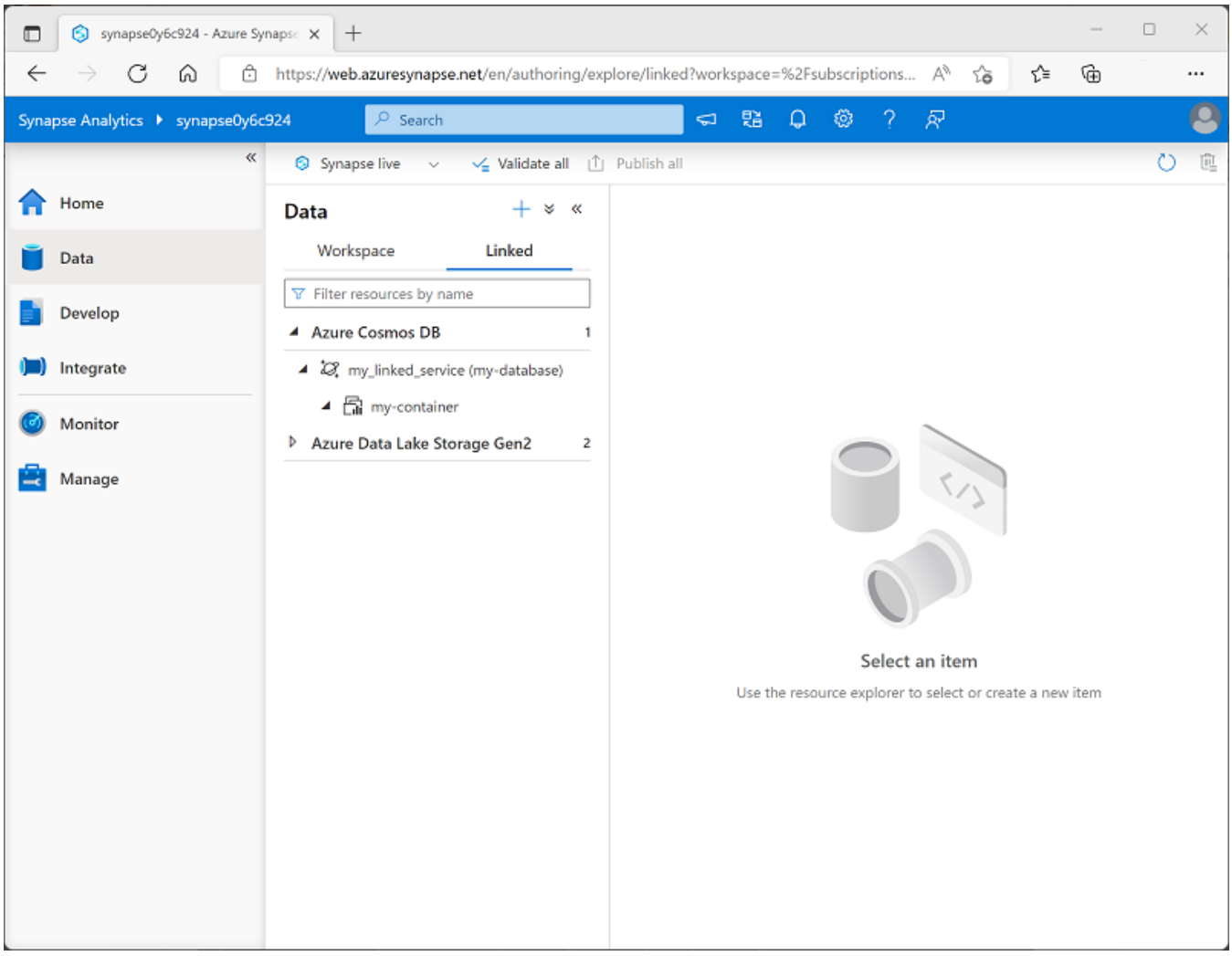
As you complete the steps to create your linked service, select the type of Azure Cosmos DB account and then assign your linked service a meaningful name and provide the necessary information to connect to your Azure Cosmos DB database.

To connect to the Azure Cosmos DB database, you can use any of the following authentication options:

- **Account key**: Specify an authentication key for your Cosmos DB account.
- **Service Principal**: Use the identity of the Azure Synapse Analytics service.
- **System Assigned Managed Identity**: Use system-assigned managed identity.
- **User Managed Identity**: Use a user-defined managed identity.

> **Tip**
>
> For more information about using managed identities in Azure Active Directory, see **What are managed identities for Azure resources?**

After creating a linked service, the Azure Cosmos DB database and its containers will be shown in the **Data** page of Azure Synapse Studio, as shown here:



> **Note**
>
> The user interface differentiates between containers with analytical store support and those without by using the following icons:

Analytical store enabled	Analytical store not enabled
	

You can query a container without an analytical store, but you won't benefit from the advantages of an HTAP solution that offloads analytical query overhead from the operational data store.

Next unit: Query Cosmos DB data with Spark

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Query Cosmos DB data with Spark

5 minutes

After you've added a linked service for your analytical store enabled Azure Cosmos DB database, you can use it to query the data using a Spark pool in your Azure Synapse Analytics workspace.

Loading Azure Cosmos DB analytical data into a dataframe

For initial exploration or quick analysis of data from an Azure Cosmos DB linked service, it's often easiest to load data from a container into a dataframe using a Spark-supported language like PySpark (A Spark-specific implementation of Python) or Scala (a Java-based language often used on Spark).

For example, the following PySpark code could be used to load a dataframe named **df** from the data in the **my-container** container connected to using the **my_linked_service** linked service, and display the first 10 rows of data:

Python

```
df = spark.read
    .format("cosmos.olap")\
    .option("spark.synapse.linkedService", "my_linked_service")\
    .option("spark.cosmos.container", "my-container")\
    .load()

display(df.limit(10))
```

Let's suppose the **my-container** container is used to store items similar to the following example:

JSON

```
{
    "productID": 123,
    "productName": "Widget",
    "id": "7248f072-11c3-42b1-a368-...",
    "_rid": "mjMaAL...==",
    "_self": "dbs/mjM...==/colls/mjMaAL...=/docs/mjMaAL...==/",
    "_etag": "\"54004b09-0000-2300-...\"",
```

```
    "_attachments": "attachments/",
    "_ts": 1655414791
}
```

The output from the PySpark code would be similar to the following table:

_rid	_ts	productID	productName	id	_etag
mjMaAL...==	1655414791	123	Widget	7248f072-11c3-42b1-a368-...	54004b09-0000-2300-...
mjMaAL...==	1655414829	124	Wotsit	dc33131c-65c7-421a-a0f7-...	5400ca09-0000-2300-...
mjMaAL...==	1655414835	125	Thingumy	ce22351d-78c7-428a-a1h5-...	5400ca09-0000-2300-...
...	...	...	...	...	...

The data is loaded from the analytical store in the container, not from the operational store; ensuring that there's no querying overhead on the operational store. The fields in the analytical data store include the application-defined fields (in this case **productID** and **productName**) and automatically created metadata fields.

After loading the dataframe, you can use its native methods to explore the data. For example, the following code creates a new dataframe containing only the **productID** and **productName** columns, ordered by the **productName**:

Python

```
products_df = df.select("productID", "productName").orderBy("productName")

display(products_df.limit(10))
```

The output of this code would look similar this table:

productID	productName
125	Thingumy
123	Widget
124	Wotsit
...	...

Writing a dataframe to a Cosmos DB container

In most HTAP scenarios, you should use the linked service to read data into Spark from the analytical store. However you *can* write the contents of a dataframe to the container as shown in the following example:

Python

```
mydf.write.format("cosmos.oltp")\
    .option("spark.synapse.linkedService", "my_linked_service")\
    .option("spark.cosmos.container", "my-container")\
    .mode('append')\
    .save()
```

> **Note**
>
> Writing a dataframe to a container updates the *operational* store and can have an impact on its performance. The changes are then synchronized to the analytical store.

Using Spark SQL to query Azure Cosmos DB analytical data

Spark SQL is a Spark API that provides SQL language syntax and relational database semantics in a Spark pool. You can use Spark SQL to define metadata for tables that can be queried using SQL.

For example, the following code creates a table named **Products** based on the hypothetical container used in the previous examples:

SQL

```
%%sql

-- Create a logical database in the Spark metastore
CREATE DATABASE mydb;

USE mydb;

-- Create a table from the Cosmos DB container
CREATE TABLE products using cosmos.olap options (
    spark.synapse.linkedService 'my_linked_service',
    spark.cosmos.container 'my-container'
);

-- Query the table
SELECT productID, productName
FROM products;
```

Tip

The `%%sql` keyword at the beginning of the code is a *magic* that instructs the Spark pool to run the code as SQL rather than the default language (which is usually set to PySpark).

By using this approach, you can create a logical database in your Spark pool that you can then use to query the analytical data in Azure Cosmos DB to support data analysis and reporting workloads without impacting the operational store in your Azure Cosmos DB account.

Next unit: Query Cosmos DB with Synapse SQL

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

100 XP

Query Cosmos DB with Synapse SQL

5 minutes

In addition to using a Spark pool, you can also query an Azure Cosmos DB analytical container by using a built-in *serverless* SQL pool in Azure Synapse Analytics. To do this, you can use the `OPENROWSET` SQL function to connect to the linked service for your Azure Cosmos DB database.

Using OPENROWSET with an authentication key

By default, access to an Azure Cosmos DB account is authenticated by an authentication key. You can use this key as part of a connection string in an `OPENROWSET` statement to connect through a linked service from a SQL pool, as shown in the following example:

SQL

```
SELECT *
FROM OPENROWSET(
    'CosmosDB',
    'Account=my-cosmos-db;Database=my-db;Key=abcd1234....==',
    [my-container]) AS products_data
```

Tip

You can find a primary and secondary key for your Cosmos DB account on its **Keys** page in the Azure portal.

The results of this query might look something like the following, including metadata and application-defined fields from the items in the Azure Cosmos DB container:

_rid	_ts	productID	productName	id	_etag
mjMaAL...==	1655414791	123	Widget	7248f072-11c3-42b1-a368-...	54004b09-0000-2300-...
mjMaAL...==	1655414829	124	Wotsit	dc33131c-65c7-	5400ca09-0000-

_rid	_ts	productID	productName	id	_etag
				421a-a0f7-...	2300-...
mjMaAL...==	1655414835	125	Thingumy	ce22351d-78c7-428a-a1h5-...	5400ca09-0000-2300-...
...	...	...	...	...	...

The data is retrieved from the analytical store, and the query doesn't impact the operational store.

Using OPENROWSET with a credential

Instead of including the authentication key in each call to OPENROWSET, you can define a *credential* that encapsulates the authentication information for your Cosmos DB account, and use the credential in subsequent queries. To create a credential, use the `CREATE CREDENTIAL` statement as shown in this example:

SQL

```
CREATE CREDENTIAL my_credential
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
SECRET = 'abcd1234....==';
```

With the credential in place, you can use it in an `OPENROWSET` function like this:

SQL

```
SELECT *
FROM OPENROWSET(PROVIDER = 'CosmosDB',
                CONNECTION = 'Account=my-cosmos-db;Database=my-db',
                OBJECT = 'my-container',
                SERVER_CREDENTIAL = 'my_credential'
) AS products_data
```

Once again, the results include metadata and application-defined fields from the analytical store:

_rid	_ts	productID	productName	id	_etag
mjMaAL...==	1655414791	123	Widget	7248f072-11c3-42b1-a368-...	54004b09-0000-2300-...
mjMaAL...==	1655414829	124	Wotsit	dc33131c-65c7-421a-a0f7-...	5400ca09-0000-2300-...
mjMaAL...==	1655414835	125	Thingumy	ce22351d-78c7-428a-a1h5-...	5400ca09-0000-2300-...
...	...	...	...	...	...

Specifying a schema

The `OPENROWSET` syntax includes a `WITH` clause that you can use to define a schema for the resulting rowset. You can use this to specify individual fields and assign data types as shown in the following example:

SQL

```
SELECT *
FROM OPENROWSET(PROVIDER = 'CosmosDB',
                CONNECTION = 'Account=my-cosmos-db;Database=my-db',
                OBJECT = 'my-container',
                SERVER_CREDENTIAL = 'my_credential'
)
WITH (
   productID INT,
   productName VARCHAR(20)
) AS products_data
```

In this case, assuming the fields in the analytical store include **productID** and **productName**, the resulting rowset will resemble the following table:

productID	productName
123	Widget
124	Wotsit
125	Thingumy
...	...

You can of course specify individual column names in the `SELECT` clause (for example, `SELECT productID, productName ...`), so this ability to specify individual columns may seem of limited use. However, consider cases where the source JSON documents stored in the operational store include multiple levels of fields, as show in the following example:

JSON

```
{
    "productID": 126,
    "productName": "Sprocket",
    "supplier": {
        "supplierName": "Contoso",
        "supplierPhone": "555-123-4567"
    }
    "id": "62588f072-11c3-42b1-a738-...",
    "_rid": "mjMaAL...==",
    ...
}
```

The `WITH` clause supports the inclusion of explicit JSON paths, enabling you to handle nested fields and to assign aliases to field names; as shown in this example:

SQL

```
SELECT *
FROM OPENROWSET(PROVIDER = 'CosmosDB',
                CONNECTION = 'Account=my-cosmos-db;Database=my-db',
                OBJECT = 'my-container',
                SERVER_CREDENTIAL = 'my_credential'
)
WITH (
   ProductNo INT '$.productID',
   ProductName VARCHAR(20) '$.productName',
   Supplier VARCHAR(20) '$.supplier.supplierName',
```

```
    SupplierPhoneNo VARCHAR(15) '$.supplier.supplierPhone'
) AS products_data
```

The results of this query would include the following row for product 126:

ProductNo	ProductName	Supplier	SupplierPhoneNo
126	Sprocket	Contoso	555-123-4567

Creating a view in a database

If you need to query the same data frequently, or you need to use reporting and visualization tools that rely on `SELECT` statements that don't include the `OPENROWSET` function, you can use a *view* to abstract the data. To create a view, you should create a new database in which to define it (user-defined views in the **master** database aren't supported), as shown in the following example:

SQL

```
CREATE DATABASE sales_db
   COLLATE Latin1_General_100_BIN2_UTF8;
GO;

USE sales_db;
GO;

CREATE VIEW products
AS
SELECT *
FROM OPENROWSET(PROVIDER = 'CosmosDB',
                CONNECTION = 'Account=my-cosmos-db;Database=my-db',
                OBJECT = 'my-container',
                SERVER_CREDENTIAL = 'my_credential'
)
WITH (
    ProductNo INT '$.productID',
    ProductName VARCHAR(20) '$.productName',
    Supplier VARCHAR(20) '$.supplier.supplierName',
    SupplierPhoneNo VARCHAR(15) '$.supplier.supplierPhone'
) AS products_data
GO
```

Tip

When creating a database that will access data in Cosmos DB, it's best to use a UTF-8 based collation to ensure compatibility with strings in Cosmos DB.

After the view has been created, users and client applications can query it like any other SQL view or table:

SQL

```
SELECT * FROM products;
```

Considerations for Serverless SQL pools and Azure Cosmos DB

When planning to use a serverless SQL pool to query data in an Azure Cosmos DB analytical store, consider the following best practices:

- Provision your Azure Cosmos DB analytical storage and any client applications (for example Microsoft Power BI) in the same region as serverless SQL pool.

  Azure Cosmos DB containers can be replicated to multiple regions. If you have a multi-region container, you can specify a `region` parameter in the OPENROWSET connection string to ensure queries are sent to a specific regional replica of the container.

- When working with string columns, use the OPENROWSET function with the explicit WITH clause and specify an appropriate data length for the string data.

Next unit: Exercise - Implement Azure Synapse Link for Cosmos DB

Continue >

---

200 XP

Knowledge check

5 minutes

1. You have an Azure Cosmos DB for NoSQL account and an Azure Synapse Analytics workspace. What must you do first to enable HTAP integration with Azure Synapse Analytics? *

- ○ Configure global replication in Azure Cosmos DB.
  - ✗ Incorrect. Global replication helps scale Cosmos DB, but it's not required for HTAP integration with Azure Synapse Analytics.
- ○ Create a dedicated SQL pool in Azure Synapse Analytics.
- ○ Enable Azure Synapse Link in Azure Cosmos DB.
  - ✓ Correct. The first step in setting up HTAP integration is to enable Azure Synapse Link in Azure Cosmos DB.

2. You have an existing container in a Cosmos DB core (SQL) database. What must you do to enable analytical queries over Azure Synapse Link from Azure Synapse Analytics? *

- ○ Delete and recreate the container.
- ○ Enable Azure Synapse Link in the container to create an analytical store.
  - ✓ Correct. Before a container can be used for analytical queries, you need to enable Synapse link for the container; which creates an analytical store.
- ○ Add an item to the container.
  - ✗ Incorrect. Adding an item to a container does not enable Synapse Link integration.

3. You plan to use a Spark pool in Azure Synapse Analytics to query an existing analytical store in Azure Cosmos DB. What must you do? *

- ○ Create a linked service for the Azure Cosmos DB database where the analytical store enabled container is defined.

✓ Correct. A linked service that connects to the Azure Cosmos DB account containing the analytical store enabled container is required.

Disable automatic pausing for the Spark pool in Azure Synapse Analytics.

Install the Azure Cosmos DB SDK for Python package in the Spark pool.

✗ Incorrect. The Azure Cosmos DB SDK for Python is not required for Synapse Link integration.

4. You're writing PySpark code to load data from an Azure Cosmos DB analytical store into a dataframe. What `format` should you specify? *

`cosmos.json`

`cosmos.olap`

✓ Correct. `cosmos.olap` is the appropriate format to read data from a Cosmos DB analytical store.

`cosmos.sql`

✗ Incorrect. `cosmos.sql` isn't a valid format.

5. You're writing a SQL code in a serverless SQL pool to query an analytical store in Azure Cosmos DB. What function should you use? *

`OPENDATASET`

`ROW`

✗ Incorrect. While `ROW` is a valid SQL function, it isn't used to query external data.

`OPENROWSET`

✓ Correct. `OPENROWSET` is used to query external data, including analytical stores in Cosmos DB.

Next unit: Summary

Continue >

100 XP

Introduction

1 minute

Azure Synapse Link for SQL is a *hybrid transactional / analytical processing* (HTAP) capability in Azure Synapse Analytics that you can use to synchronize transactional data in Azure SQL Database or Microsoft SQL Server with a dedicated SQL pool in Azure Synapse Analytics. This synchronization enables you to perform near real-time analytical workloads on operational data with minimal impact on the transactional store used by business applications.

In this module, you'll learn how to:

- Understand key concepts and capabilities of Azure Synapse Link for SQL.
- Configure Azure Synapse Link for Azure SQL Database.
- Configure Azure Synapse Link for Microsoft SQL Server.

Next unit: What is Azure Synapse Link for SQL?

Continue

100 XP

What is Azure Synapse Link for SQL?

5 minutes

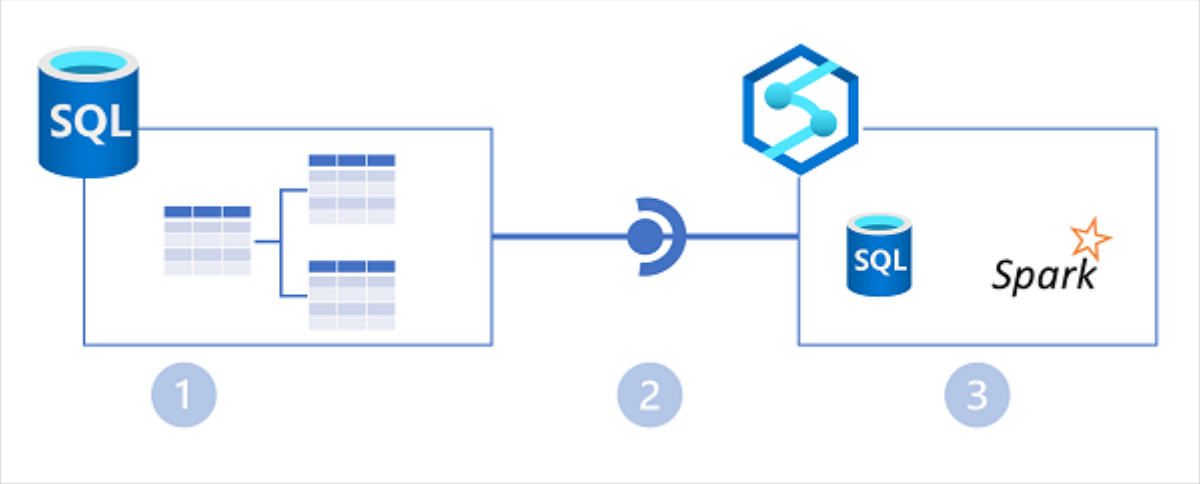
Many organizations use a relational database in Azure SQL Database or Microsoft SQL Server to support business applications. These databases are optimized for transactional workloads that store and manipulate operational data. Performing analytical queries on the data in these databases to support reporting and data analysis incurs resource contention that can be detrimental to application performance.

A traditional approach to resolving this problem is to implement an *extract, transform, and load* (ETL) solution that loads data from the operational data store into an analytical store as a batch operation at regular intervals. While this solution supports the analytical workloads required for reporting and data analysis, it suffers from the following limitations:

- The ETL process can be complex to implement and operate.
- The analytical store is only updated at periodic intervals, so reporting doesn't reflect the most up-to-date operational data.

Azure Synapse Link for SQL

Azure Synapse Link for SQL addresses the limitations of a traditional ETL process by automatically replicating changes made to tables in the operational database to corresponding tables in an analytical database. After the initial synchronization process, the changes are replicated in near real-time without the need for a complex ETL batch process.



In the diagram above, the following key features of the Azure Synapse Link for SQL architecture are illustrated:

1. An Azure SQL Database or SQL Server 2022 instance contains a relational database in which transactional data is stored in tables.
2. Azure Synapse Link for SQL replicates the table data to a dedicated SQL pool in an Azure Synapse workspace.
3. The replicated data in the dedicated SQL pool can be queried in the dedicated SQL pool, or connected to as an external source from a Spark pool without impacting the source database.

Source and target databases

Azure Synapse Link for SQL supports the following source databases (used as operational data stores):

- Azure SQL Database
- Microsoft SQL Server 2022

> **Note**
>
> Azure Synapse link for SQL is not supported for Azure SQL Managed Instance.

The target database (used as an analytical data store) must be a dedicated SQL pool in an Azure Synapse Analytics workspace.

The implementation details for Azure Synapse Link vary between the two types of data source, but the high-level principle is the same - changes made to tables in the source database are synchronized to the target database.

Change feed

Azure Synapse Link for SQL uses the *change feed* feature in Azure SQL Database and Microsoft SQL Server 2022 to capture changes to the source tables. All data modifications are recorded in the transaction log for the source database. The change feed feature monitors the log and applies the same data modifications in the target database. In the case of Azure SQL Database, the modifications are made directly to the target database. When using Azure Synapse Link for SQL Server, the changes are recorded in files and saved to a *landing zone* in Azure Data Lake Gen2 storage before being applied to the target database.

> **Note**
>
> Change feed is similar to the *change data capture* (CDC) feature in SQL Server. The key difference is that CDC is used to reproduce data modifications in a table in the same

database as the modified table. Change feed caches the data modification in memory and forwards it to Azure Synapse Analytics.

After implementing Azure Synapse Link for SQL, you can use system views and stored procedures in your Azure SQL Database or SQL Server database to monitor and manage change feed activity.

Tip

Learn more:

- For more information about change feed, see **Azure Synapse Link for SQL change feed**.
- To learn more about monitoring and managing change feed, see **Manage Azure Synapse Link for SQL Server and Azure SQL Database**.

Next unit: Configure Azure Synapse Link for Azure SQL Database

Continue

How are we doing? ☆ ☆ ☆ ☆ ☆

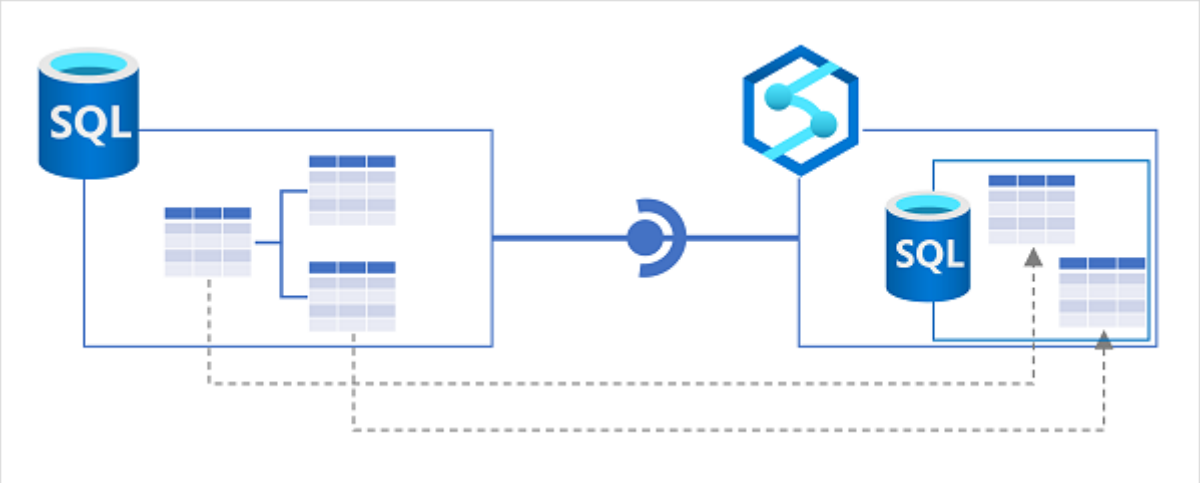
100 XP

Configure Azure Synapse Link for Azure SQL Database

5 minutes

Azure SQL Database is a platform-as-a-service (PaaS) relational database service based on the SQL Server database engine. It's commonly used in cloud-native applications as a scalable, secure, and easy to manage relational database store for operational data.

Azure Synapse Link for Azure SQL Database uses a *link connection* to map one or more tables in an Azure SQL Database instance to tables in a dedicated SQL pool in Azure Synapse Analytics. When the link connection is started, the tables are initialized by copying a snapshot of the source tables to the target tables. Subsequently, the change feed process applies all modifications made in the source tables to the target tables.



Implementing Azure Synapse Link for Azure SQL Database

To use Azure Synapse Link for Azure SQL Database, you need to configure some settings in your Azure SQL Database server, before creating a link connection in Azure Synapse Analytics.

Configure Azure SQL Database

Before you can use Azure SQL Database as a source for a linked connection in Azure Synapse Analytics, you must ensure the following settings are configured in the Azure SQL Database server that hosts the database you want to synchronize:

- **System assigned managed identity** - enable this option so that your Azure SQL Database server uses a system assigned managed identity.
- **Firewall rules** - ensure that Azure services can access your Azure SQL Database server.

In addition to these server-level settings, if you plan to configure the link connection from Azure Synapse Analytics to use a managed identity when connecting to Azure SQL Database, you must create a user for the workspace identity in the database and add it to the **db_owner** role, as shown in the following code example:

SQL

```
CREATE USER my_synapse_workspace FROM EXTERNAL PROVIDER;
ALTER ROLE [db_owner] ADD MEMBER my_synapse_workspace;
```

> **Tip**
>
> If you intend to use SQL authentication, you can omit this step.

Prepare the target SQL pool

Azure Synapse Link for Azure SQL Database synchronizes the source data to tables in a dedicated SQL pool in Azure Synapse Analytics. You therefore need to create and start a dedicated SQL pool in your Azure Synapse Analytics workspace before you can create the link connection.

The database associated with the dedicated SQL pool must include the appropriate schema for the target table. If source tables are defined in a schema other than the default **dbo** schema, you must create a schema of the same name in the dedicated SQL pool database:

SQL

```
CREATE SCHEMA myschema;
```

Create a link connection

To create a linked connection, add a **linked connection** on the **Integrate** page in Azure Synapse Studio. You'll need to:

1. Select or create a *linked service* for your Azure SQL Database. You can create this separately ahead of time, or as part of the process of creating a linked connection for

Azure Synapse Link. You can use a managed identity or SQL authentication to connect the linked service to Azure SQL Database.
2. Select the tables in the source database that you want to include in the linked connection.
3. Select the target dedicated SQL pool in which the target tables should be created.
4. Specify the number of CPU cores you want to use to process synchronization. Four driver cores will be used in addition to the number of cores you specify.

After creating the linked connection, you can configure the mappings between the source and target tables. In particular, you can specify the table structure (index) type and distribution configuration for the target tables.

> **Note**
>
> Some data types in your source tables may not be supported by specific dedicated SQL pool index types. For example, you cannot use a clustered columnstore index for tables that include VARBINARY(MAX) columns. You can map such tables to a *heap* (an unindexed table) in the dedicated SQL pool.

When the linked connection is configured appropriately, you can start it to initialize synchronization. The source tables are initially copied to the target database as snapshots, and then subsequent data modifications are replicated.

> **Tip**
>
> Learn more:
>
> - For more information about Synapse Link for Azure SQL Database, see **Azure Synapse Link for Azure SQL Database**.
> - To learn about limitations and restrictions that apply to Synapse Link for Azure SQL Database, see **Known limitations and issues with Azure Synapse Link for SQL**.
> - For a step-by-step guide to setting up Synapse Link for Azure SQL Database, see **Get started with Azure Synapse Link for Azure SQL Database**. You'll also get a chance to try configuring Synapse Link for Azure SQL Database in the exercise, later in this module.

Next unit: Configure Azure Synapse Link for SQL Server 2022

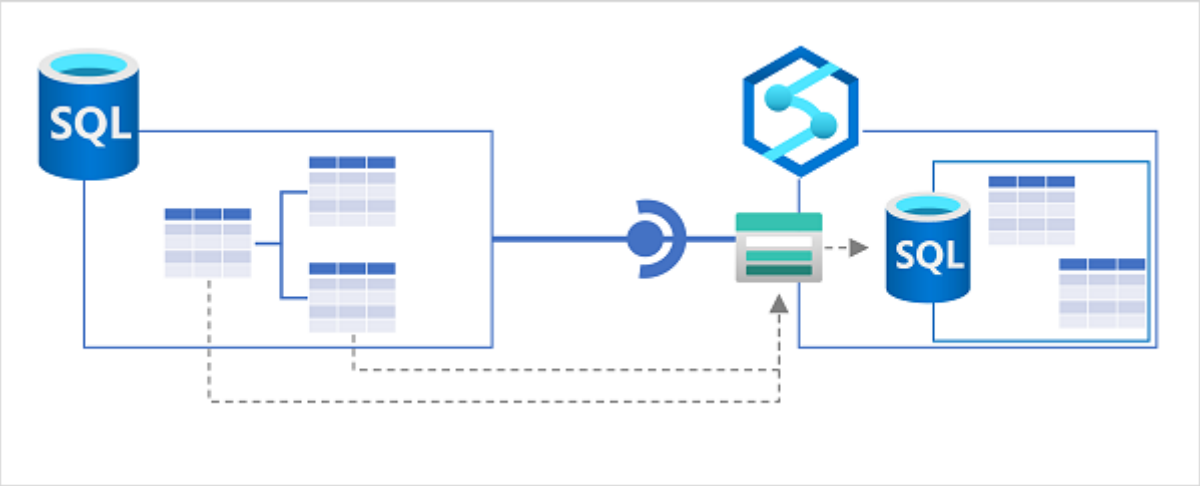
100 XP

Configure Azure Synapse Link for SQL Server 2022

5 minutes

Microsoft SQL Server is one of the world's most commonly used relational database systems. SQL Server 2022 is the latest release, and includes many enhancements and new features; including the ability to be used as a source for Azure Synapse Link.

Azure Synapse Link for SQL Server uses a *link connection* to map one or more tables in an Azure SQL Database instance to tables in a dedicated SQL pool in Azure Synapse Analytics. When the link connection is started, the tables are initialized by copying a .parquet file for each source table to a *landing zone* in Azure Data Lake Storage Gen2; from where the data is imported into tables in the dedicated SQL pool. Subsequently, the change feed process copies all changes as .csv files to the landing zone where they're applied to the target tables.



Synchronization between SQL Server (which can be on-premises or in a private network) and Azure Synapse Analytics is achieved through a self-hosted integration runtime. An integration runtime is a software agent that handles secure connectivity when using Azure Data Factory or Azure Synapse Analytics to transfer data across networks. It must be installed on a Microsoft Windows computer with direct access to your SQL Server instance.

Tip

For more information about using a self-hosted integration runtime to work with Azure Synapse Analytics, see **Create and configure a self-hosted integration runtime**.

Implementing Azure Synapse Link for SQL Server 2022

To use Azure Synapse Link for SQL Server 2022, you need to create storage for the landing zone in Azure and configure your SQL Server instance before creating a link connection in Azure Synapse Analytics.

Create landing zone storage

You need to create an Azure Data Lake Storage Gen2 account in your Azure subscription to use as a landing zone. You can't use the default storage for your Azure Synapse Analytics workspace.

> **Tip**
>
> For more information about provisioning an Azure Data Lake Storage Gen2 account, see **Create a storage account to use with Azure Data Lake Storage Gen2**.

Create a master key in the SQL Server database

To support Azure Synapse Link, your SQL Server database must contain a master key. You can use a `CREATE MASTER KEY` SQL statement like the following example to create one:

SQL

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'my$ecretPa$$w0rd';
```

Create a dedicated SQL pool in Azure Synapse Analytics

In your Azure Synapse Analytics workspace, you need to create a dedicated SQL pool where the target tables will be created. You also need to create master key in this database by using the following SQL statement:

SQL

```
CREATE MASTER KEY
```

Create a linked service for the SQL Server source database

Next, in Azure Synapse Analytics, create a linked service for your SQL Server database. When you do this, you need to specify the self-hosted integration runtime to be used for connectivity between SQL Server and Azure Synapse Analytics. If you haven't already configured a self-hosted integration runtime, you can create one now, and then download and install the agent onto a Windows machine in the network where your SQL Server instance is located.

Create a linked service for your Data Lake Storage Gen2 account

In addition to the linked service for SQL Server, you need a linked service for the Data Lake Storage Gen2 account that will be used as a landing zone. To support this, you need to add the managed identity of your Azure Synapse Analytics Workspace to the **Storage Blob Data Contributor** role for your storage account and configure the linked service to use the managed identity for authentication.

Create a link connection for Azure Synapse Link

Finally, you're ready to create a link connection for Azure Synapse Link data synchronization. As you do so, you'll specify the service link for the SQL Server source database, the individual tables to be replicated, the number of CPU cores to be used for the synchronization process, and the Azure Data Lake Storage Gen2 linked service and folder location for the landing zone.

After the link connection is created, you can start it to initialize synchronization. After a short time, the tables will be available to query in the dedicated SQL pool, and will be kept in sync with modifications in the source database by the change feed process.

> **Tip**
>
> Learn more:
>
> - For more information about Synapse Link for SQL Server 2022, see **Azure Synapse Link for SQL Server 2022**.
> - To learn about limitations and restrictions that apply to Synapse Link for Azure SQL Database, see **Known limitations and issues with Azure Synapse Link for SQL**.
> - For a step-by-step guide to setting up Synapse Link for SQL Server 2022, see **Get started with Azure Synapse Link for SQL Server 2022**.

Next unit: Exercise - Implement Azure Synapse Link for SQL

200 XP

Knowledge check

3 minutes

1. From which of the following data sources can you use Azure Synapse Link for SQL to replicate data to Azure Synapse Analytics? *

- ○ Azure Cosmos DB

  ✗ Incorrect. Azure Synapse Link does not support Cosmos DB. You can use Azure Synapse Link for Cosmos DB instead.

- ○ SQL Server 2022

  ✓ Correct. You can use Azure Synapse Link for SQL to replicate data from SQL Server 2022.

- ○ Azure SQL Managed Instance

2. What must you create in your Azure Synapse Analytics workspace as a target database for Azure Synapse Link for Azure SQL Database? *

- ○ A serverless SQL pool
- ○ An Apache Spark pool

  ✗ Incorrect. You cannot use a Spark pool for Azure Synapse Link.

- ○ A dedicated SQL pool

  ✓ Correct. Use a dedicated SQL pool as a target database.

3. You plan to use Azure Synapse Link for SQL to replicate tables from SQL Server 2022 to Azure Synapse Analytics. What additional Azure resource must you create? *

- ○ An Azure Storage account with an Azure Data Lake Storage Gen2 container

  ✓ Correct. An Azure Data Lake Storage Gen2 account is required to be used as a landing zone when using Azure Synapse Link for SQL Server 2022.

- ○ An Azure Key Vault containing the SQL Server admin password

✕ Incorrect. You do not need Azure key Vault to use Azure Synapse Link for SQL.

An Azure Application Insights resource

Next unit: Summary

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆