**Introduction to HTAP**

**OLTP:**

* **OLTP(**online transaction processing) systems working with operational data.
* OLTP systems are optimized for dealing with discrete system or user requests immediately and responding as quickly as possible.

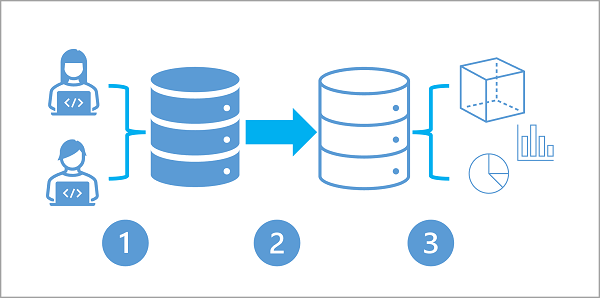
**OLAP:**

* **OLAP(**online analytical processing) systems working with historical data.
* 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.
* ETL is complex process where need is to copy large amount of data and can create delay in data being available to analyse in OLAP systems.

**What is HTAP?**

* HTAP (Hybrid transactional and analytical Processing), Is a style of data processing that combines transactional data processing with analytical processing.
* 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.
* HTAP, enables business to run advanced analytics in near-real-time on data stored and processed by OLTP systems.

**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 modelling, analytics, and reporting without impacting the transactional system.

Azure Synapse Link

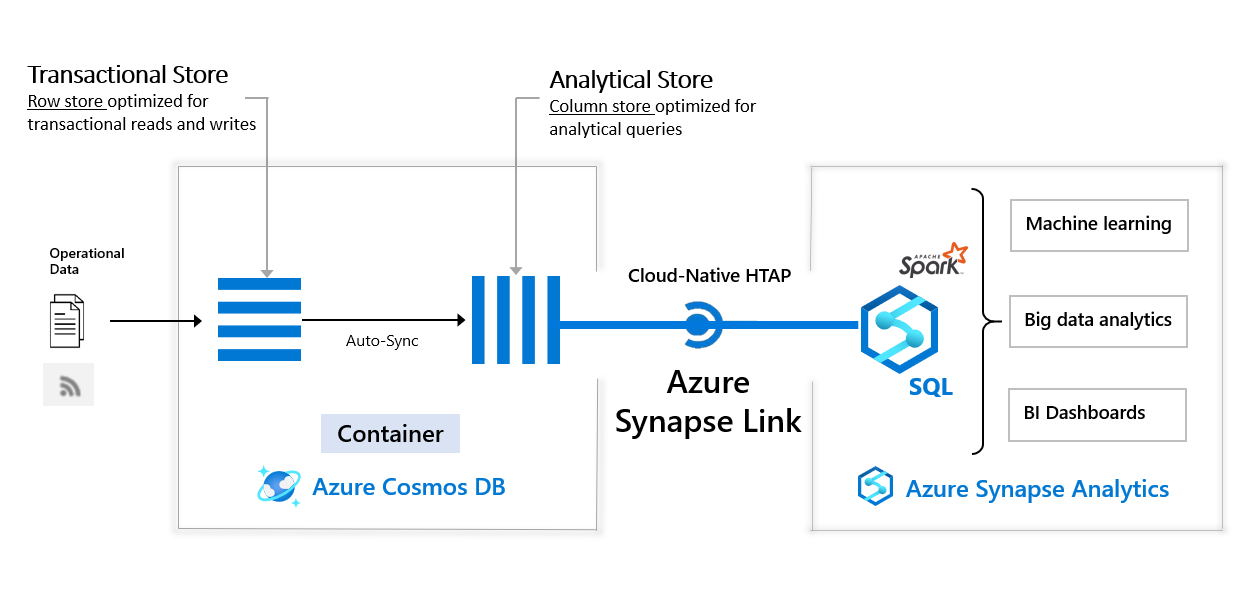
Azure Synapse Link is the general name for a set of HTAP capabilities in Azure Synapse Analytics that support commonly used operational data sources.

Azure Synapse link can be created for following data stores:

|  |  |
| --- | --- |
| A picture containing diagram, screenshot, plan, design  Description automatically generated | **Azure Cosmos DB**   * Transactional data stored in Azure Cosmos DB container * Data is synced to an analytical store in the container, and queried via linked service in Azure Synapse Analytics   **Azure SQL**   * Transactional data stored in Azure SQL Database or SQL Server * Transactions are sync’d to a dedicated SQL pool in Azure Synapse Analytics   **Microsoft Dataverse**   * Transactional data stored in Dataverse tables. * Data is sync’d to data lake and queries via linked service in Azure Synapse Analytics |

**Synapse Link for CosmosDb**

* Cloud-native implementation of HTAP (Hybrid transactional and analytical Processing)
* HTAP solutions are supported in Azure Synapse Analytics through **Azure Synapse Link**



* **Azure Synapse Link** creates a tight seamless integration between Azure Cosmos DB and Azure Synapse Analytics.
* When Azure Synapse Link is enabled on Cosmos DB Container, new read only analytical store is created.
* Data is synced in analytical store every 2 min and stored in analytical Columnar format and no throughput is consumed so no performance impact on transactional store as it is internal process.
* Azure Synapse Link provides a linked service that connects the analytical store enabled container in Cosmos DB to an Azure Synapse Analytics workspace.
* Azure Synapse Link is supported in the following types of Azure Cosmos DB account:
  + Core (SQL) API
  + MongoDB API
  + Apache Gremlin
* Synapse link can be enables using portal, Powershell, CLI

**Advantages:**

* Azure Synapse Link allows you to directly access Azure Cosmos DB analytical store **without ETL**
* You can get Near real-time insights into your operational data
* There is no impact on operational workload as you can run analytical queries against an Azure Cosmos DB analytical store (a separate column store)
* You only pay for storage of analytical store and not for throughput.

***Note: After enabling the link you can’t disable it.***

**Lab 1: Configure Azure Synapse Link in CosmosDb and Create analytical store enabled container**

1. Create CosmosDb Account with Core SQL API
2. Create a container in adworks database for storing sales information(sales.json) and enable Analytical store.

Graphical user interface, text, application, email

Description automatically generated🡪OK

1. Load items in container from **sales.json** file

Graphical user interface, text, application

Description automatically generated

1. Enable Synapse link

Integration🡪Azure Synapase Link 🡪Enable

**Lab 2: Access CosmosDb from synapse using Synapse link.**

Create Linked service in synapse.

Data Tab🡪 +(Add new source)🡪Connect to external data🡪Configure CosmosDb Core SQL API

Graphical user interface, text, application, email

Description automatically generated🡪Test connection🡪Create

Refresh the page to see that under data tab Linked section

**Lab3: Query using SQL Pool**

1. Container🡪New SQL Script🡪Select Top 100 rows
2. Copy Primary key from Cosmosdb

CosmosDb🡪Settings🡪Key🡪Copy primary key

1. Use Access key for cosmosdb to Execute command for Create Credentials in serverless pool

CREATE CREDENTIAL [mydsscosmosdb]

    WITH IDENTITY = 'SHARED ACCESS SIGNATURE',

    SECRET = 'z3YlQFLev2D7q5NmUmQrPCPLDVpBtdrMVq5FX7JZNrTBfzaaYNyg5TJSX029fcV74JUERlF7koeJLGpP7OcRrg=='

1. Execute the Query with serverless pool

SELECT TOP 100 \*

FROM OPENROWSET(​PROVIDER = 'CosmosDB',

                CONNECTION = 'Account=mydsscosmosdb;Database=Adworks',

                OBJECT = 'Sales',

                SERVER\_CREDENTIAL = 'mydsscosmosdb'

)WITH (

     OrderID VARCHAR(10) '$.id',

     OrderDate VARCHAR(10) '$.orderdate',

     CustomerID INTEGER '$.customerid',

     CustomerName VARCHAR(40) '$.customerdetails.customername',

     CustomerEmail VARCHAR(30) '$.customerdetails.customeremail',

     Product VARCHAR(30) '$.product',

     Quantity INTEGER '$.quantity',

     Price FLOAT '$.price'

 )

 AS sales

 ORDER BY OrderID;

*Note: You can also query using Spark Pool*

**Lab4:Access CosmosDb using synapase link with spark pool**

1. Use Notebook: **AzureSynapselinkDemo**
2. Create a new item in container and check the changes reflected using serverless pool and Notebook

{

"id": "SO43708",

"orderdate": "2019-07-02",

"customerid": 126,

"customerdetails": {

"customername": "Samir Nadoy",

"customeremail": "samir1@adventure-works.com"

},

"product": "Road-150 Black, 48",

"quantity": 1,

"price": 3578.27

}

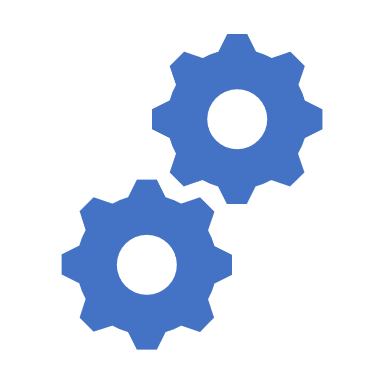
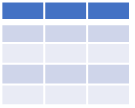
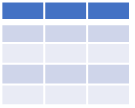
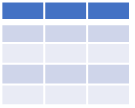
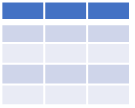
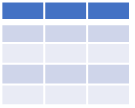
Azure Synapse link for SQL

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.

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

* Azure SQL Database
* Microsoft SQL Server 2022



1. Transactional is data stored in the *source* database (Azure SQL Database or SQL Server 2022)

Azure SQL Managed Instance is not supported by Azure Synapse Link for SQL

1. The *change feed* service captures each transaction and forwards to the *target* database

Transactions are replayed in the *target* database (dedicated SQL pool in Azure Synapse Analytics)

**Lab 5: Create Sample Azure SQL Databse**

1. Create database from Adventureworks sample .
2. Adworks database page🡪 select the **Query editor🡪Login using SQL authentication.**
3. **Verify table contains using following Query:**

Select top 100 \*

From [SalesLT].[Customer]

1. In sql server enable system assigned managed identity

Azure SQL Server 🡪 **Security** section (near the bottom)🡪 **Identity**. 🡪 **System assigned managed identity**

🡪 **Status** option to **On**. 🡪save

**Lab6: Configure Azure Synapse link in Synapse**

1. Start dedicated SQL pool
2. Create target schema

CREATE SCHEMA SalesLT;

GO

1. Create linked service for Azure SQL Database

A screenshot of a computer

Description automatically generated

1. Create link connection
2. Integrate Tab🡪 + 🡪Link Connection🡪

A screenshot of a computer

Description automatically generated with medium confidence

**Source tables**: Select the following tables:

* **SalesLT.Customer**
* **SalesLT.Product**
* **SalesLT.SalesOrderDetail**
* **SalesLT.SalesOrderHeader**

**🡪Continue**

**Target Pool:sqlpool1 🡪Continue**

A screenshot of a computer login

Description automatically generated with low confidence

Modify these structure types in the table mappings as follows:

| Source table | Target table | Distribution type | Distribution column | Structure type |
| --- | --- | --- | --- | --- |
| SalesLT.Customer **→** | [SalesLT] . [Customer] | Round robin | - | Clustered columnstore index |
| SalesLT.Product **→** | [SalesLT] . [Product] | Round robin | - | Heap |
| SalesLT.SalesOrderDetail **→** | [SalesLT] . [SalesOrderDetail] | Round robin | - | Clustered columnstore index |
| SalesLT.SalesOrderHeader **→** | [SalesLT] . [SalesOrderHeader] | Round robin | - | Heap |

🡪Click on **Start to synchronize data at the begining**

1. Verify Replicated Data:

SELECT oh.SalesOrderID, oh.OrderDate,

p.ProductNumber, p.Color, p.Size,

c.EmailAddress AS CustomerEmail,

od.OrderQty, od.UnitPrice

FROM SalesLT.SalesOrderHeader AS oh

JOIN SalesLT.SalesOrderDetail AS od

ON oh.SalesOrderID = od.SalesOrderID

JOIN SalesLT.Product AS p

ON od.ProductID = p.ProductID

JOIN SalesLT.Customer as c

ON oh.CustomerID = c.CustomerID

ORDER BY oh.SalesOrderID;