Data Engineering in the Cloud

Ingest and Load Data

Xuemao Zhang
East Stroudsburg University

January 18, 2025

Outline

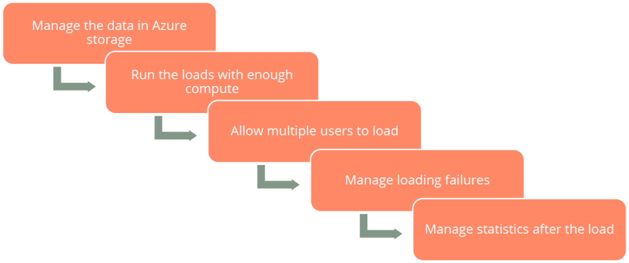
- Prerequisites
- Data import in Azure Synapse Analytics
- Loading Data with PolyBase and Copy using T-SQL

Prerequisites

- Prerequisites (5 minutes):
  - Create a datalake (storage account with hierarchical namespace enabled)
    - To lower the cost, you may choose `Redundancy` as `LRS`
  - Create a container
  - Upload the data set `new_customers.csv` to the container
  - Create a Synapse workspace and a dedicated SQL pool
  - Open the Synapse Studio

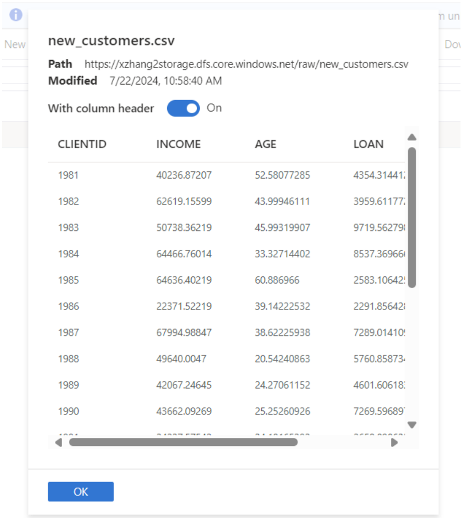
Data import in Azure Synapse Analytics

- As a Azure Data Engineer, tt is important to load the data into Azure Synapse and integrate SQL in Azure Synapse Analytics.
- Loading Data into Azure Synapse Analytics:


Manage the data in Azure storage
Run the loads with enough compute
Allow multiple users to load
Manage loading failures
Manage statistics after the load

Data import in Azure Synapse Analytics

- You can have a preview of the data



new_customers.csv

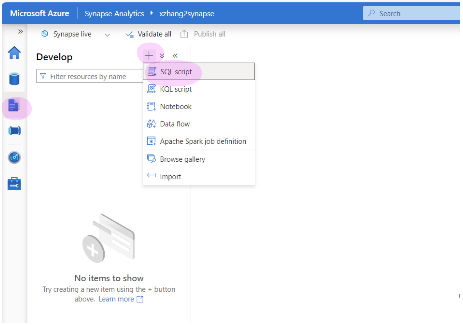
Path https://xzhang2storage.dfs.core.windows.net/raw/new_customers.csv
Modified 7/22/2024, 10:58:40 AM

With column header On

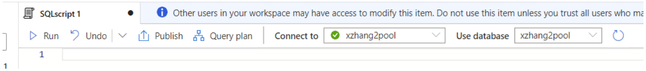
CLIENTID	INCOME	AGE	LOAN
1981	40236.87207	52.58077285	4354.31441
1982	62619.15599	43.99946111	3959.61177
1983	50738.36219	45.99319907	9719.56279
1984	64466.76014	33.32714402	8537.36966
1985	64636.40219	60.886966	2583.10642
1986	22371.52219	39.14222532	2291.85642
1987	67994.98847	38.62225938	7289.01410
1988	49640.0047	20.54240863	5760.85873
1989	42067.24645	24.27061152	4601.60618
1990	43662.09269	25.25260926	7269.59689

Data import in Azure Synapse Analytics

- Go to the `Develop` tab and click on + and `SQL script`, as shown below.



- After the above task, Connect the script to the dedicated SQL pool



Other users in your workspace may have access to modify this item. Do not use this item unless you trust all users who ma

Connect to xzhang2pool
Use database xzhang2pool

Data import in Azure Synapse Analytics

- Let us create a schema

```
CREATE SCHEMA [customers];
```

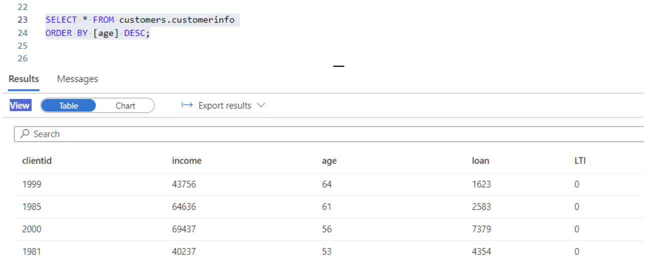
- Now, let us create a new table and copy the data into it

```
CREATE TABLE [customers].customerinfo
    (
        [clientid] [int]  NOT NULL,
        [income] [NUMERIC]  NOT NULL,
        [age] [NUMERIC]  NOT NULL,
        [loan] [NUMERIC]  NOT NULL,
        [LTI] [NUMERIC]  NOT NULL
    )
GO

COPY INTO customers.customerinfo
FROM 'https://xzhang2storage.dfs.core.windows.net/raw/new_customers.csv'
WITH (
    FILE_TYPE = 'CSV',
    FIELDTERMINATOR=',',
    ROWTERMINATOR='\n',
    FIRSTROW = 2
)
GO
```

Data import in Azure Synapse Analytics

- To view the imported data use the command below
  - Square brackets are used in SQL Server to delimit identifiers, such as schema names, table names, or column names, especially when those identifiers include spaces, special characters, or reserved words.

```
SELECT * FROM customers.customerinfo
ORDER BY [age] DESC;
```



```
SELECT * FROM customers.customerinfo
ORDER BY [age] DESC;
```

clientid	income	age	loan	LTI
1999	43756	64	1623	0
1985	64636	61	2583	0
2000	69437	56	7379	0
1981	40237	53	4354	0

Loading Data with PolyBase and Copy using T-SQL

- PolyBase is a mechanism to access external data storage in the Azure Data Lake with the help of T-SQL.
- To use PolyBase, define the external tables in your SQL pool before loading the data.
- It utilizes the external tables to access the data from the Azure storage.

Loading Data with PolyBase and Copy using T-SQL

Loading Data with PolyBase: The following options can load the data with PolyBase:

- PolyBase with T-SQL
- PolyBase with SSIS
- PolyBase with Azure Data Factory
- PolyBase with Azure Databricks

Loading Data with PolyBase and Copy using T-SQL

- ETL Process:
  - During the ETL process, data is first extracted from the source systems and loaded into staging tables.
  - Data transformations, such as cleansing, deduplication, and formatting, are applied to the data in the staging tables.
  - Once the data is transformed, it is loaded from the staging tables into the final destination tables in the data warehouse.
- A **staging table** is a temporary storage area used during the data extraction, transformation, and loading (ETL) process in a data warehouse. It acts as an intermediary space where data is initially loaded before it undergoes further processing and transformation and is subsequently moved to its final destination, such as a data warehouse
  - Staging tables provide a controlled environment to apply data quality checks and transformations before the data is loaded into the final destination.
  - This ensures that only clean and accurate data is moved to the data warehouse.

Loading Data with PolyBase and Copy using T-SQL

- Upload the parquet file `parquet.parquet` to your container
- Go to the opened synapse studio, right click the parquet file → `New SQL script` → `Create external table` and then modify the code as in the next slide.
  - `DISTRIBUTION = ROUND_ROBIN`: Distributes the rows evenly across all distributions (nodes).
  - `HEAP`: Specifies that the table is a heap, meaning it has no clustered index and data is not stored in any particular order.
- Inserting data into a **heap table** can be faster than into an indexed table because there is no need to maintain index structures. The data is simply added to the next available space.
  - To optimize read performance, consider adding non-clustered indexes.

```
CREATE NONCLUSTERED INDEX IX_CustomerID ON SalesOrders(CustomerI
```

Loading Data with PolyBase and Copy using T-SQL

```
CREATE SCHEMA [product_staging];
GO

CREATE TABLE [product_staging].[ProductHeapp]
(
    [Id] int,
    [Correlationid] nvarchar(4000),
    [Operationname] nvarchar(4000),
    [Status] nvarchar(4000),
    [Eventcategory] nvarchar(4000),
    [Level] nvarchar(4000),
    [Time] datetime2(7),
    [Subscription] nvarchar(4000),
    [Eventinitiatedby] nvarchar(4000),
    [Resourcetype] nvarchar(4000),
    [Resourcegroup] nvarchar(4000)
    )
WITH
    (
    DISTRIBUTION = ROUND_ROBIN,
    HEAP
    )
GO
```

Loading Data with PolyBase and Copy using T-SQL

- `CREATE TABLE` creates a regular table within the database. The data for this table is stored within the SQL Server or Azure SQL Data Warehouse.
  - Accessing data is managed within the SQL Server.
  - It is used for storing and managing data directly within SQL Server.
- `CREATE EXTERNAL TABLE` creates a table definition that points to data stored outside of the database, such as in an Azure Data Lake. The data itself is not stored in SQL Server but is accessible through it.
  - Accessing data involves reading from an external source.
  - It is used for querying and analyzing large datasets stored outside of SQL Server without the need to import them.

Loading Data with PolyBase and Copy using T-SQL

- Create another table called Product in the product_staging schema for comparing the load
  - `DISTRIBUTION = HASH ( [Correlationid] )`: Distributes the rows based on a hash value of the Correlationid column. This ensures that rows with the same Correlationid are stored on the same node.
  - `CLUSTERED COLUMNSTORE INDEX`: Uses a columnstore index, which is optimized for read-heavy operations on large datasets.

```
CREATE TABLE [product_staging].[Product]
(
    [Id] [int] NULL,
    [Correlationid] [varchar](200) NULL,
    [Operationname] [varchar](200) NULL,
    [Status] [varchar](100) NULL,
    [Eventcategory] [varchar](100) NULL,
    [Level] [varchar](100) NULL,
    [Time] [datetime] NULL,
    [Subscription] [varchar](200) NULL,
    [Eventinitiatedby] [varchar](1000) NULL,
    [Resourcetype] [varchar](1000) NULL,
    [Resourcegroup] [varchar](1000) NULL
)
WITH
(
    DISTRIBUTION = HASH ( [CorrelationId] ),
    CLUSTERED COLUMNSTORE INDEX
)
```

Loading Data with PolyBase and Copy using T-SQL

- Now, let us configure and run polybase operations. Create an external table using the command below
  - Follow the same naming convention for file and storage account
  - `TYPE = HADOOP`: Specifies that the data source type is HADOOP

```
CREATE EXTERNAL DATA SOURCE ABSS
WITH
( TYPE = HADOOP,
    LOCATION = 'abfss://raw@xzhang2storage.dfs.core.windows.net'
);
```

Loading Data with PolyBase and Copy using T-SQL

- Now create an external file format and an external table using the command below and run the command

```
CREATE EXTERNAL FILE FORMAT [ParquetFormat]
WITH (
    FORMAT_TYPE = PARQUET,
    DATA_COMPRESSION = 'org.apache.hadoop.io.compress.SnappyCodec'
)
GO

CREATE SCHEMA [product_external];
GO

CREATE EXTERNAL TABLE [product_external].Product
    (
        [Id] [int] NULL,
        [Correlationid] [varchar](200) NULL,
        [Operationname] [varchar](200) NULL,
        [Status] [varchar](100) NULL,
        [Eventcategory] [varchar](100) NULL,
        [Level] [varchar](100) NULL,
        [Time] [datetime] NULL,
        [Subscription] [varchar](200) NULL,
        [Eventinitiatedby] [varchar](1000) NULL,
        [Resourcetype] [varchar](1000) NULL,
        [Resourcegroup] [varchar](1000) NULL
    )
WITH
    (
        LOCATION = '/parquet.parquet',
        DATA_SOURCE = ABSS,
        FILE_FORMAT = [ParquetFormat]
    )
GO
```

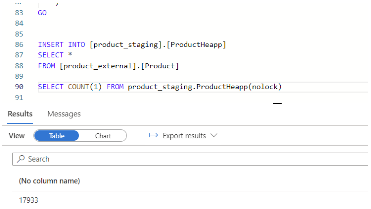
Loading Data with PolyBase and Copy using T-SQL

- Now load the data into the heap table using the command below.
  - Loading Data from External Table to Staging Table:

```
INSERT INTO [product_staging].[ProductHeapp]
SELECT *
FROM [product_external].[Product]
```

- Now select COUNT to see if the data and its count (number of rows) was loaded.
  - The NOLOCK hint allows reading rows without acquiring shared locks.

```
SELECT COUNT(1) FROM product_staging.ProductHeapp(nolock)
```



```
GO


INSERT INTO [product_staging].[ProductHeapp]
SELECT *
FROM [product_external].[Product]

SELECT COUNT(1) FROM product_staging.ProductHeapp(nolock)
```

```
(No column name)
17933
```

Loading Data with PolyBase and Copy using T-SQL

- Now, configure the copy statement and run it. Truncate the heap table and load data using the COPY statement.
  - Removes all rows from the ProductHeapp table, resetting the table to an empty state and then
  - load data directly from the specified external source into the ProductHeapp table.

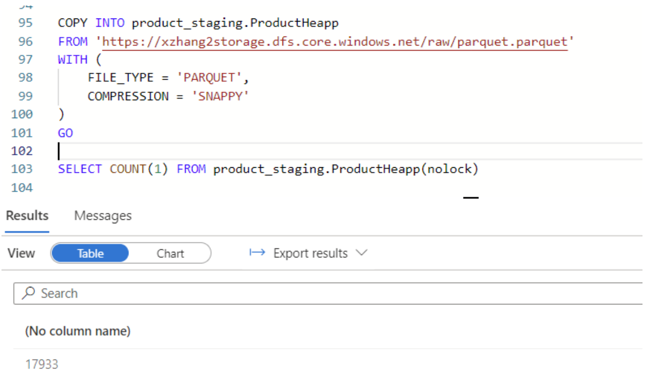
```
TRUNCATE TABLE product_staging.ProductHeapp;
GO

COPY INTO product_staging.ProductHeapp
FROM 'https://xzhang2storage.dfs.core.windows.net/raw/parquet.parquet'
WITH (
    FILE_TYPE = 'PARQUET',
    COMPRESSION = 'SNAPPY'
)
GO
```

Loading Data with PolyBase and Copy using T-SQL

- Now select the data using the command below:

```
SELECT COUNT(1) FROM product_staging.ProductHeapp(nolock)
```

- You can see that the number of rows and the time have matched.



```
COPY INTO product_staging.ProductHeapp
FROM 'https://xzhang2storage.dfs.core.windows.net/raw/parquet.parquet'
WITH (
    FILE_TYPE = 'PARQUET',
    COMPRESSION = 'SNAPPY'
)
GO

SELECT COUNT(1) FROM product_staging.ProductHeapp(nolock)
```

```
(No column name)
17933
```

Loading Data with PolyBase and Copy using T-SQL

- This sequence of commands demonstrates a typical ETL (Extract, Transform, Load) process using SQL Server and PolyBase, where data is extracted from an external source, loaded into a staging table, and then processed as needed.

License