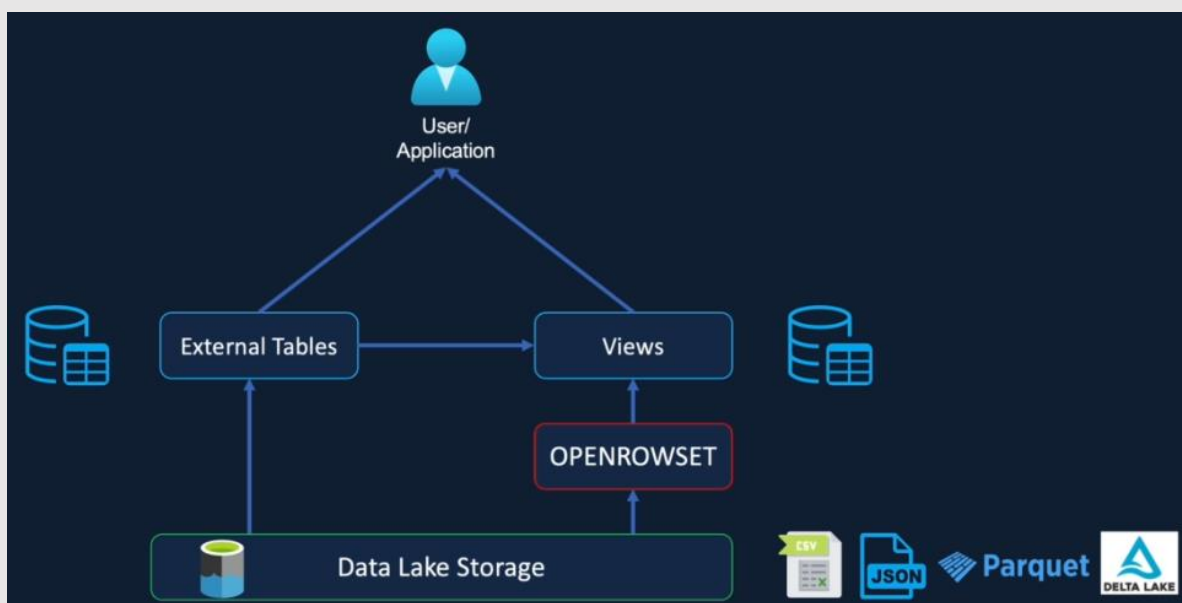


# Data Virtualization

En esta sección, vamos a ver algunas de las capacidades proporcionadas por Synapse Serverless SQL Pool para lograr la virtualización de datos. Entonces, **¿qué es Data Virtualization?** La virtualización de datos es una capa de datos lógica que nos permite combinar datos de múltiples fuentes en tiempo de consulta sin tener que escribir complejos pipelines ETL para cargar los datos en una base de datos.

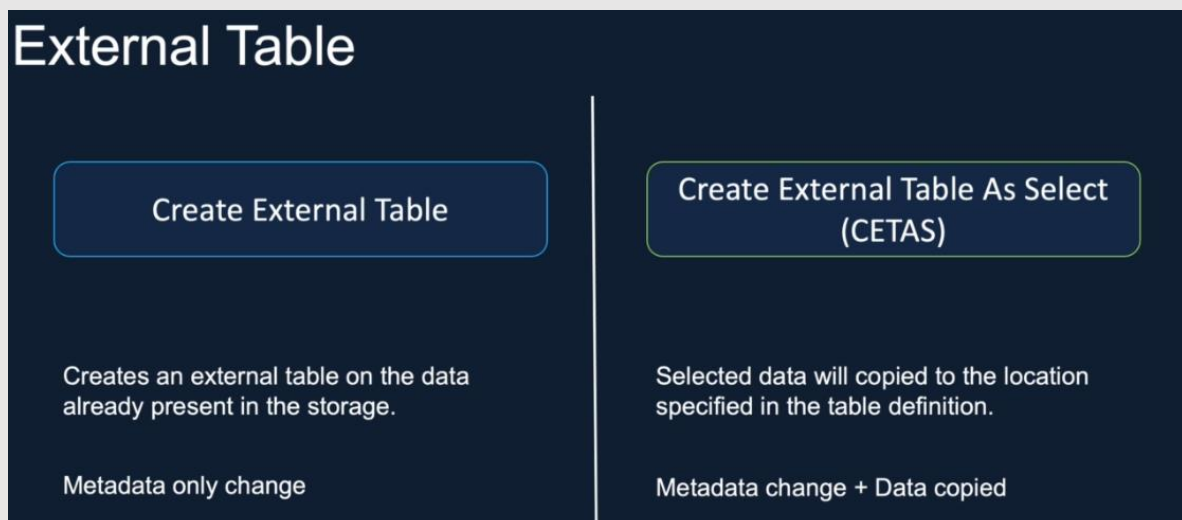
**¿No sería bueno que los consumidores de los datos pudiesen acceder a los archivos como si fuesen tablas o vistas SQL y no se preocuparan por el tipo de los archivos y el formato de los mismos, etc.?** Sí, es posible, con Serverless SQL Pool.

Serverless SQL Pool proporciona dos tipos de objetos de base de datos para hacer realidad esta experiencia. **El primero es la EXTERNAL TABLE**, que se puede crear sobre los datos del Data Lake. Como parte de la creación de la External table puede definir cosas como su array de la ubicación de almacenamiento, formatos de archivo, nombres de columnas y tipos de datos. Una vez creada la External table, se puede acceder a ella como a cualquier otra tabla de una base de datos relacional donde los usuarios no necesitan conocer ninguna de esas características que han definido la tabla. **La otra opción es utilizar VISTAS**. Las vistas en Serverless SQL es similar a las vistas en cualquier otro sistema de base de datos relacional. Se puede crear una vista sobre la selección de los datos. Como hemos visto anteriormente, podemos seleccionar los datos utilizando la función OPENROWSET y podemos crear una vista sobre esa selección de los datos. También se pueden crear vistas sobre los datos seleccionados de las External tables o una combinación de External tables y funciones OPENROWSET.



## CREATE EXTERNAL TABLE

Empecemos por entender lo que son las tablas externas. Hay dos formas de crear tablas externas en Serverless SQL. La primera es usando el comando **CREATE EXTERNAL TABLE**. Este comando crea una tabla externa sobre los datos que ya están presentes en el storage, por ejemplo, como tu **Data Lake Storage Gen2**. Esto simplemente va a poner una estructura a los datos en carpetas y archivos y no va a mover esos datos a ninguna parte en absoluto. En términos simples, el efecto que la creación de la tabla externa es sólo cambios de metadatos. Ningún dato en los archivos o carpetas se verá afectado por este proceso. La segunda forma de crear una tabla externa es utilizando la sentencia **CREATE EXTERNAL TABLE AS SELECT (CETAS)**. Este comando nos permite seleccionar datos del storage usando una sentencia SELECT y copiar esos datos a otra ubicación en el storage y también hace una tabla disponible sobre esos datos para que pueda ser consultada de forma similar al comando CREATE EXTERNAL TABLE. Esto crea los metadatos para definir la fuente de datos de la tabla y el formato del archivo, y también copia los datos que están siendo seleccionados. Una nota importante que quiero reiterar aquí es que **Serverless SQL no tiene ninguna opción de almacenamiento disponible dentro de él**, y los datos siempre residen externamente por ejemplo en un Azure Data Lake Storage Gen2.

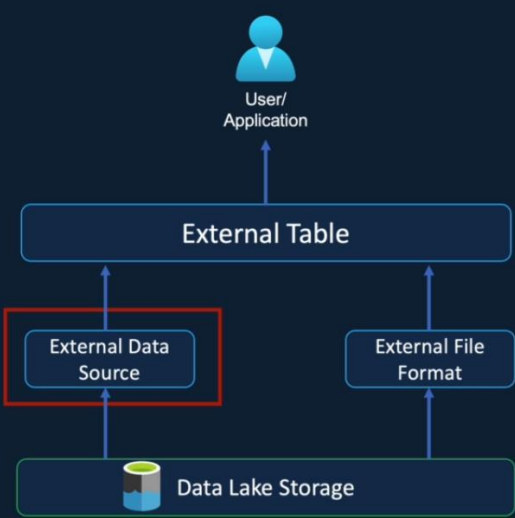


Para crear una tabla externa, necesitamos tener un par de otros objetos de la base de datos en su lugar. Como sabemos, creamos tablas externas en el Data Lake Storage o en el Blob storage. **El primer objeto que necesitamos es la fuente de datos externa (External Data Source)**. Esto define que es derecho de la cuenta de almacenamiento y el pliegue aparte. También puede tener las credenciales necesarias para acceder a esos datos. **Lo siguiente que necesitamos tener es el formato de archivo externo (External File Format)**. Este objeto tiene la información sobre el formato de los archivos, como el tipo de archivo, por ejemplo, si es un archivo delimitado o un archivo parquet, etc. y también tiene la información sobre el delimitador y la codificación. Utilizando estos dos objetos de la base de datos, podemos crear la tabla externa que contiene la referencia a la fuente de datos, así como el formato del archivo. También contiene los nombres de las columnas, los tipos de datos y la información sobre qué hacer con los rechazos cuando hay fallos. Los usuarios y las aplicaciones pueden ahora acceder a la tabla externa sin preocuparse por ninguno

de los detalles subyacentes. La principal conclusión es que la creación de la tabla externa también implica la creación de la fuente de datos externa y el formato de archivo externo.

Toma tres argumentos: **LOCATION**, **CREDENTIAL** y **TYPE**.

## External Data Source



```
graph BT; User[User/ Application] --> Table[External Table]; Table --> Source[External Data Source]; Table --> Format[External File Format]; Source --> Storage[(Data Lake Storage)]; Format --> Storage
```

The diagram illustrates the architecture of an external data source. At the top, a 'User/ Application' icon points to an 'External Table' box. Below the 'External Table', there are two boxes: 'External Data Source' (highlighted with a red border) and 'External File Format'. Both of these boxes point to the 'External Table'. At the bottom, a 'Data Lake Storage' box (represented by a cylinder icon) has arrows pointing up to both the 'External Data Source' and 'External File Format' boxes.

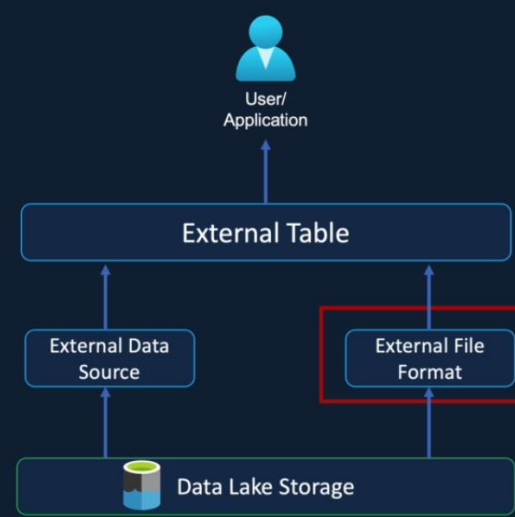
```
CREATE EXTERNAL DATA SOURCE <Data Source Name>
WITH
(
    LOCATION = <folder path URI> ,
    CREDENTIAL = <Credential Name> ,
    TYPE = {HADOOP}
);
```

```
CREATE EXTERNAL DATA SOURCE nyc_taxi_data
WITH
(
    LOCATION = 'abfss://nyc-taxi-
data@synapsecoursedl.dfs.core.windows.net',
    CREDENTIAL = nyc_taxi_data_cred
);
```

Udemy

Aquí hay una plantilla de comandos SQL para crear la fuente de datos externa. Es necesario proporcionar un <file\_format\_name> único dentro de la base de datos. Como dijimos, toma los tres argumentos: tipo de formato, compresión de datos y las opciones de formato.

## External File Format



```
graph BT; User[User/ Application] --> Table[External Table]; Table --> Source[External Data Source]; Table --> Format[External File Format]; Source --> Storage[(Data Lake Storage)]; Format --> Storage
```

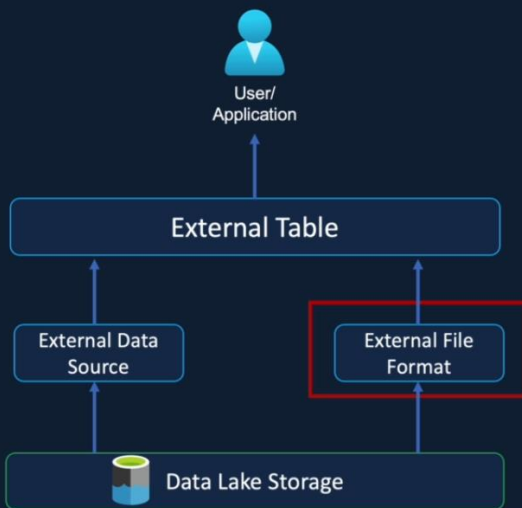
The diagram illustrates the architecture of an external file format. It is identical to the one in the previous slide, showing the flow from 'User/ Application' to 'External Table', which then branches to 'External Data Source' and 'External File Format' (highlighted with a red border), both of which connect to 'Data Lake Storage'.

```
CREATE EXTERNAL FILE FORMAT <file format name>
WITH
(
    FORMAT_TYPE = {DELIMITEDTEXT | PARQUET| DELTA}
    [, DATA_COMPRESSION =
        'org.apache.hadoop.io.compress.GzipCodec'
        | 'org.apache.hadoop.io.compress.SnappyCodec']
    [, FORMAT_OPTIONS ( <format_options> [ ,...n ] ) ]
);
```

```
<format_options> ::=
{
    FIELD_TERMINATOR = field_terminator
    | STRING_DELIMITER = string_delimiter
    | First_Row = integer
    | USE_TYPE_DEFAULT = { TRUE | FALSE }
    | Encoding = {'UTF8' | 'UTF16'}
    | PARSER_VERSION = {'parser_version'}
}
```

Udemy

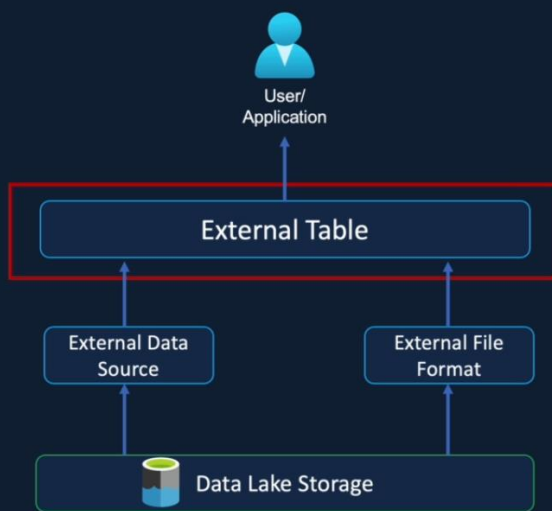
# External File Format (Example)



```
CREATE EXTERNAL FILE FORMAT csv_file_format
WITH
(
    FORMAT_TYPE = DELIMITEDTEXT,
    FORMAT_OPTIONS
    (
        FIELD_TERMINATOR = ',',
        STRING_DELIMITER = '"',
        FIRST_ROW = 2,
        USE_TYPE_DEFAULT = False,
        ENCODING = 'UTF8',
        PARSER_VERSION = '2.0'
    )
);
```

Odemy

# External Table

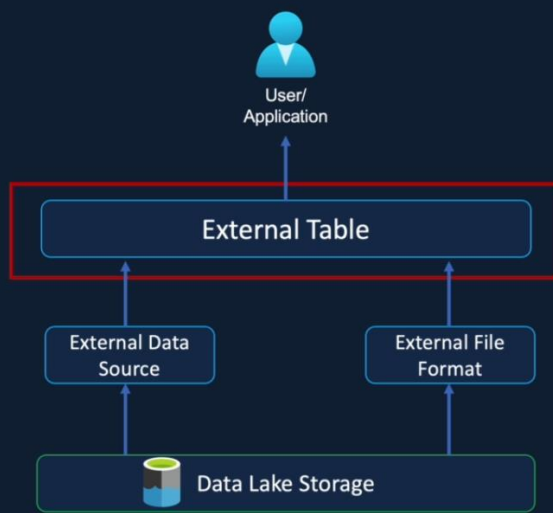


```
CREATE EXTERNAL TABLE
<[database_name].[schema_name].table_name>
( <column name> <data type> )
WITH (
    LOCATION = '<folder_or_filepath>',
    DATA_SOURCE = <external_data_source_name>,
    FILE_FORMAT = <external_file_format_name>
    [, TABLE_OPTIONS =
    N'{"READ_OPTIONS":{"ALLOW_INCONSISTENT_READS"}}'
    ]
    [, <reject_options>]
)
```

```
<reject_options> ::=
{
    | REJECT_TYPE = value,
    | REJECT_VALUE = reject_value,
    | REJECT_SAMPLE_VALUE = reject_sample_value,
    | REJECTED_ROW_LOCATION = '/REJECT_Directory'
}
```

Odemy

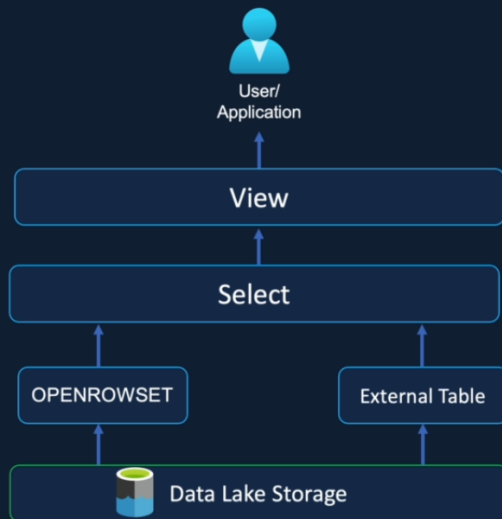
# External Table



```
CREATE EXTERNAL TABLE demo.ldw.taxi_zone
(
    location_id SMALLINT,
    borough VARCHAR(1),
    zone VARCHAR(50) ,
    service_zone VARCHAR(15)
)
WITH (
    LOCATION = 'raw/taxi_zone.csv',
    DATA_SOURCE = nyc_taxi_data,
    FILE_FORMAT = csv_file_format,
    REJECT_VALUE = 10,
    REJECTED_ROW_LOCATION = 'rejections/ldw/taxi_zone'
);
```

## VIEWS

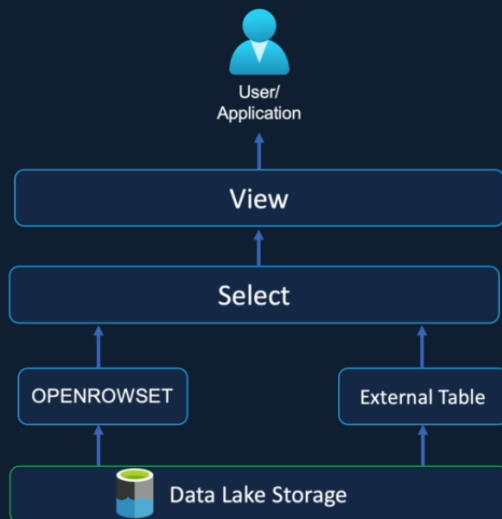
### Create View



```
CREATE VIEW [ schema_name . ] view_name  
    [ ( column_name [ ,...n ] ) ]  
AS  
<select_statement> [;]
```

Udemy

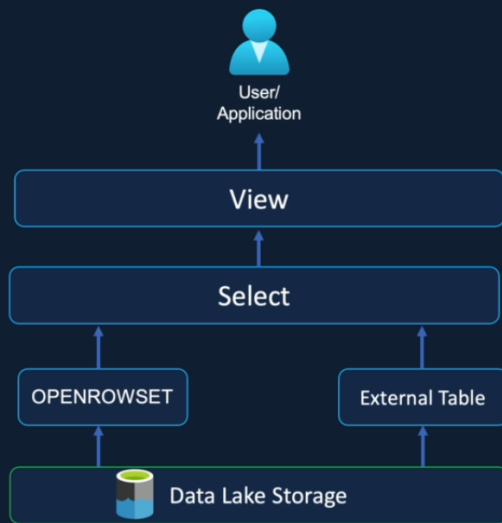
### Create View - OPENROWSET



```
CREATE VIEW bronze.vw_vendor  
AS  
SELECT *  
FROM OPENROWSET(  
    BULK 'vendor.csv',  
    DATA_SOURCE = 'nyc_taxi_data_raw',  
    FORMAT = 'CSV',  
    PARSER_VERSION = '2.0',  
    HEADER_ROW = TRUE  
) AS vendor;
```

Udemy

# Create View – External Table



```
CREATE VIEW bronze.vw_taxi_zone_brooklyn  
AS  
SELECT location_id, zone, service_zone  
FROM bronze.taxi_zone  
WHERE borough = 'Brooklyn';
```