Microsoft SQL Server 2022 with Hitachi Content Platform

Backup to Object and PolyBase Validation

March 2023

Table of Contents

Introduction

The purpose of this report is to document the deployment, configuration and lab validation results achieved during the integration testing between Microsoft SQL Server 2022 with Hitachi Content Platform v 9.4.0.267. In the testing HCP was backed by Hitachi Content Platform S Series S10.

SQL Server 2022 (16.x) Preview introduces object storage integration to the MS SQL data platform, enabling you to integrate SQL Server with S3 compatible object storage. Hitachi Content Platform (HCP) provides fully compatible S3 object storage and is well suited to integration with MS SQL Server to provide the ideal combination of price and performance. The MS SQL release extends the existing BACKUP/RESTORE syntax to support S3 storage, allowing users to use economical Hitachi object storage for their backup targets. This release also introduces Data Virtualization with a new feature called PolyBase. With PolyBase, table data stored on Hitachi S3 storage can be directly queried from the object store. This allows customers to share data across applications in an open format and eliminates the need to import redundant copies of data into proprietary table and storage formats.

This guide is written by Hitachi Vantara after lab validation for IT professionals charged with object storage management who are Content Platform administrators, storage administrators and those who have good knowledge on implementing Microsoft SQL Server and other cloud services. It is expected that you have a basic knowledge of SQL, SAN, compute, and networking, and have working experience in managing and administering a Content Platform system and SQL Server Management Studio (SSMS).

Tested Solution Components

This section describes the configuration that was tested, Microsoft SQL Server 2022 with Hitachi Content Platform v 9.4.0.267. These components are part of the environment used to test the implementation procedures in this document. Always follow best practices from Microsoft and Hitachi. Hardware and software requirements for MS SQL Server 2022 release are the same as SQL Server 2019 except for .NET

Software Components

Hardware Components	

Hitachi Content Platform	• Single-instance HCP running on 1 VM
	• VM Details:
	  ◦ CentOS7 minimal install
	  ◦ 8 CPUs
	  ◦ 32 GB RAM
	  ◦ 935.43 GB disk

Hitachi Content Platform S Series	• S10 Newisys NDS-SB1EA • Using Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz
Virtual Machine	• 12 vCPUs, 32 GB Memory, 2 HDDs 100 GB each

Framework 4.7.2

Hitachi Content Platform	● 9.4.0.267
Hitachi Content Platform S Series	● 2.2.3.3
Microsoft SQL Server 2022 (CTP 2.1)	● 16.0.700.4 Microsoft Corporation Enterprise Evaluation Edition (64-bit) on Windows Server 2019 Standard 10.0 <X64>
SQL Server Management Studio	● v19.0 Preview 3

Hitachi Content Platform (HCP) is a distributed object store that provides advanced storage and data management capabilities. This helps you address challenges posed by ever-growing volumes of data. Divide a single HCP into multiple tenants, secure access to each store, and uniquely configure each store for your workloads.

Eliminate storage silos using Content Platform with a single object storage infrastructure. This supports a wide range of data types, applications, and users with different service level needs in enterprise and cloud environments. Consolidate application storage using Content Platform a highly scalable, economical, and cloud-ready object storage infrastructure.

Hitachi Content Platform is designed to enable tiering of application data in a manner that does the following:

- Ensures content integrity, authenticity, security, completeness, and accessibility over the long term, in accordance with relevant laws and regulations.
- Allows integrated searching and indexing of the object store, including search of file contents.
- Supports business continuity, data recovery, compliance search and retention need.
- Scales horizontally to support multiple applications and content types; and vertically to support continued data growth.

Hitachi Content Platform (HCP) S Series Nodes are highly efficient, highly available, cost-effective storage devices that support very large amounts of data. Each S Series Node consists of two cooperating server modules and multiple high-density drives in some number of enclosures.

During normal operation, the two server modules in an S Series Node actively share responsibility for all S Series Node functions. Because the server modules are peers, if one module becomes unavailable, the other can still provide full, uninterrupted S Series Node functionality, although performance may be degraded.

The S Series Node data storage implementation ensures that data is well-protected using erasure coding.

Additionally, S Series Nodes use several internal processes to continuously check the integrity of both the stored data and the storage media.

Microsoft: SQL Server 2022 (16.x) Preview introduces S3-compatible object storage integration to the data platform, enabling you to integrate SQL Server with HCP. SQL Server 2022 (16.x) Preview extends the existing BACKUP/RESTORE TO/FROM URL to allow users to store their backups on economical S3-compatible object storage. The release also extends PolyBase Data Virtualization to directly create and query open format table data on the S3-compatible object store.

PolyBase now enables creating and querying CSV, Parquet, and Delta table data directly on S3-compatible object storage. Use T-SQL queries to Create External Tables (CET & CETAS), and to query and join data from external sources to relational tables in SQL Server while your data remains in its original location and format. This enables sharing of data across applications in open table formats, reduces ETL processes, and eliminates redundant copies of data in proprietary storage formats. The following T-SQL Operations are among the many that now support querying data on S3-compatible storage:

OPENROWSET: Lightweight command that allows SQL engine to access data outside SQL Server, either a file or another database. Recommended for loading data or data exploration.

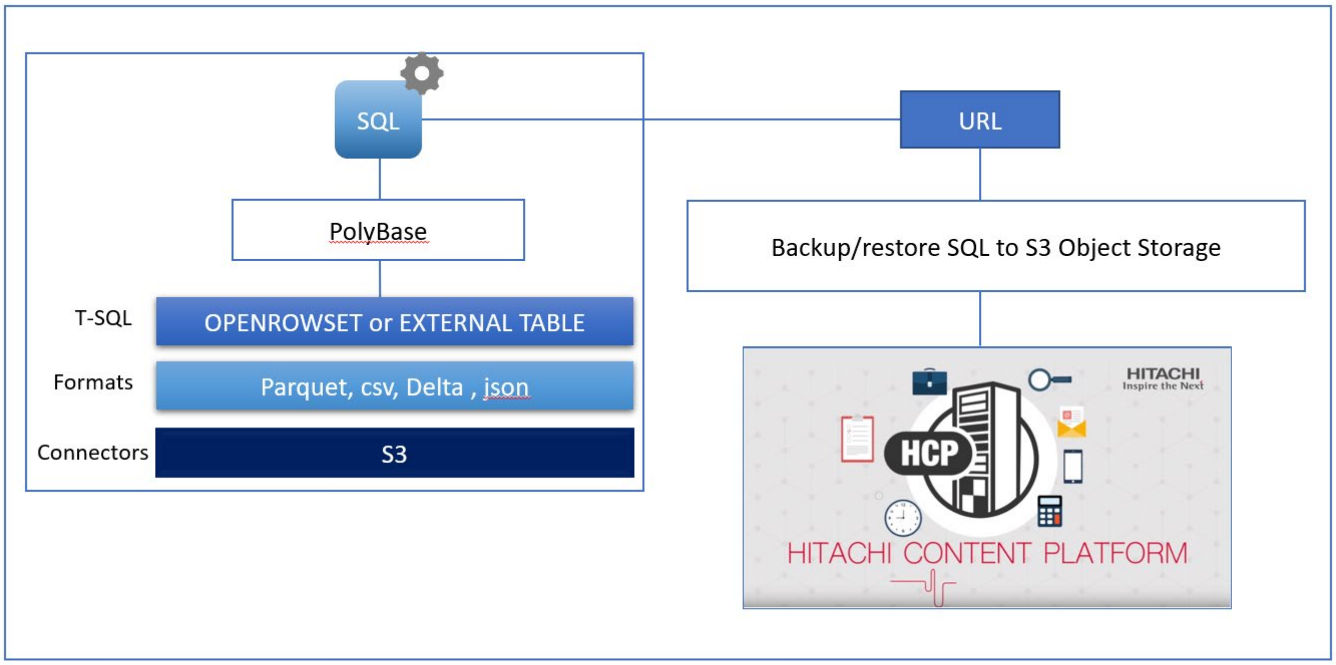
CREATE EXTERNAL TABLE (CET): Creates a table where the data stays in its original location outside of SQL Server, and when selected, the SQL engine will provide the requested data to the user. External table benefits from reusability and can leverage the use of statistics for better performance.

CREATE EXTERNAL TABLE as SELECT (CETAS): It performs a combination of operations in a single command. First, it allows SQL Server to transform and convert a given data stored inside or outside the database. Second, it then exports the data to a different location, either a network location or Azure. Finally, it creates an external table targeting the newly exported data.

These operations are secured by a combination of database master key and external credentials for simplified management.

Click on the arrow to navigate to **“Configure PolyBase to access external data in S3-compatible object storage”**

Solution Implementation


SQL
URL
PolyBase
Backup/restore SQL to S3 Object Storage
T-SQL
OPENROWSET or EXTERNAL TABLE
Formats
Parquet, csv, Delta , json
Connectors
S3
HCP
HITACHI
Inspire the Next
HITACHI CONTENT PLATFORM

Prerequisites:

For the HCP S3 endpoint:

TLS must be configured. It is assumed that all connections will be securely transmitted over HTTPS not HTTP. The HCP SSL certificate must contain subject alternate name (SAN) entries that include both the tenant and bucket domain names. The certificate must be signed by a trusted certificate authority, or if self-signed the MS SQL Server instances must be configured to trust the certificate or the CA with which the certificate was signed. This guide does not cover that procedure.

HCP Tenant Configuration

1. The HCP tenant must have versioning enabled.
2. The HCP tenant must support local authentication.

HCP Bucket Configuration

Note: In the HCP Tenant Management Console (TMC) the **S3 Bucket** is referred to as **Namespace**

1. In the TMC create a local data access user to be used exclusively by SQL Server Management Studio
   a. Provide a username and password.
      i. Name the user something like to indicate the user is a non-person user.
   b. Do not select any administrative roles.
   c. Do not allow namespace management.
   d. Optionally provide a description
   e. Click “Create User Account” and **be sure to copy the “Authorization token” string** in the confirmation message.
      i. The text before the colon is the S3 Access Key, the text following the colon is the S3 Secret Key (i.e.: access**:**secret).
      ii. If you forget you can recover keys by clicking the TMC => Security => Users => Generate Authorization Token button.

2. In the TMC create a dedicated namespace to be used by the Microsoft SQL Server Management Studio
    a. Provide a name, this can be anything for easy reference.
    b. In the namespace owner field begin typing the name of the user created above and then select the name from the list. You must select the name from the list for the owner to be properly applied.
    c. Set your namespace quotas.
    d. Under Protocol Optimization choose Cloud Protocols
    e. Under Directory Usage choose Unbalanced
    f. Enable Versioning
        a. Set versioning options to prune versions older than 0 days old.
        b. Do not enable delete markers.
        c. Do not enable overwrite.
    g. Click the Create Namespace button.
3. In the TMC => Namespaces, expand the setting for your new namespace and adjust as follows
    a. Policies => Versioning: Uncheck keep deletion records
    b. Settings => ACLs: Enable ACLs; Enforce ACLs
    c. Protocols => HTTP(S)
        i. Uncheck Enable REST API
        ii. Check Enable Hitachi API for Amazon S3
            1. Authenticated Only
            2. If checked, uncheck enable active directory.

MS SQL 2022 Installed on Windows Server

- Setup Microsoft SQL Server 2022 by deploying a Windows Server 2019 VM in VMware vSphere with 12 vCPUs, 32 GB Memory, 2 HDDs 100 GB each.
- Setup the Microsoft SQL Server Management Studio (SSMS) inside the Windows OS.
- Enable PolyBase

Lab Validation Functional Test

This test evaluates the basic functionality of copying data to object storage from the SQL Server Studio and make use of the data virtualization feature of PolyBase.

No	Test Cases	Result
1	Backup a test Database to HCP	Pass
2	Restore a test Database to HCP	Pass
3	Backup/Restore with some configurable parameters like MAXTRANSFERSIZE, split URL etc.	Pass
4	Use **OPENROWSET** and **Create External Table** to query on .csv , .json , .parquet ,.delta files stored on S3	Pass
5	**Create External Table As Select** (CETAS) for Delta file format	Pass
6	Demo.sql (Microsoft defined) use cases	Pass

Backup a test Database to HCP

```
create database tryhcp;

use tryhcp;

CREATE CREDENTIAL   [s3://partner.hcpdemo.hitachivantara.com/sqlserver22]
WITH
      IDENTITY    = 'S3 Access Key',
      SECRET      = 'amFzcw==:a3b9c163f6c520407ff34cfdb83ca5c6';

BACKUP DATABASE Testing
TO URL = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22/testing.bak';


BACKUP DATABASE tryhcp
TO URL = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22/tryhcp.bak';

```

Restore a test Database from HCP

```
RESTORE DATABASE Testing
FROM URL = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22/testing.bak';

```

Backup/Restore with configurable parameters.

Backup/Restore with some configurable parameters like MAXTRANSFERSIZE, split URL etc. (used for large databases):

Parts and file size limitations:

Unlike traditional file systems, object storage splits and stores its data in blocks called parts, similar to the Azure block blobs when using Azure Blob Storage. Each part can vary its size from 5 MB to 20 MB, where the default value is 10 MB. This behaviour is controlled by SQL Server through the parameter MAXTRANSFERSIZE in combination with COMPRESSION.

- Each backup file can be split into 10,000 parts.
- Each part size can be between 5 to 20 MB. You can control this range using the MAXTRANSFERSIZE in the T-SQL backup statement.
- SQL Server 2022 uses by default 10 MB value for the parameter MAXTRANSFERSIZE.
- It supports the maximum file size of a single file as 10,000 parts * MAXTRANSFERSIZE.
- You can split a more extensive backup into up to 64 URLs. Therefore, the maximum supported file size is 10,000 parts * URL * MAXTRANSFERSIZE.
- A single SQL database backup file can be up to 200,000 MB per URL (20*10,000)

Note: You must specify COMPRESSION in the backup statement to change MAXTRANSFERSIZE values.

```
BACKUP DATABASE Testing
TO URL = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22/testingc.bak'
WITH   FORMAT
       ,MAXTRANSFERSIZE = 20971520
       ,STATS  = 10
       ,COMPRESSION;

BACKUP DATABASE Testing
TO
 URL = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22/Testing2.bak'
,URL = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22/Testing3.bak'
,URL = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22/Testing4.bak'
WITH   FORMAT
       ,MAXTRANSFERSIZE = 20971520
       ,STATS  = 10
       ,COMPRESSION;

```

After deleting the Testing database, restored the database using the below queries:

```
 RESTORE DATABASE Testing
FROM
 URL = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22/Testing2.bak'
,URL = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22/Testing3.bak'
,URL = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22/Testing4.bak'

     WITH   REPLACE -- overwrite
, STATS             = 10;

use Testing;
SELECT * FROM sys.symmetric_keys;
SELECT * FROM sys.external_tables;

```

Use OPENROWSET and Create External Table

Use OPENROWSET and Create External Table to query on .csv , .json , .parquet ,.delta files stored on HCP S3. Both OPENROWSET and EXTERNAL TABLES are used to query data files stored in S3-compatible object storage. However, OPENROWSET is typically used for ad-hoc queries and one-off data imports, while EXTERNAL TABLES is used for creating a permanent schema on top of S3 data for repeated querying. EXTERNAL TABLES supports data manipulation language (DML) operations such as INSERT, UPDATE, and DELETE, while OPENROWSET only supports read operations.

Set of queries that need to be run before using OPENROWSET or Create External Table to query data files in S3-compatible object storage:

```
use tryhcp;          --to use the tryhcp database

--enable polybase:
exec sp_configure @configname = 'polybase enabled', @configvalue = 1
;
RECONFIGURE
;
exec sp_configure @configname = 'polybase enabled'
;

--Before you create a database scoped credential, the user database must have a master key to protect the credential
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Start123!';

--Verify the presence of the new key
SELECT * FROM sys.symmetric_keys;

--The following creates a database scoped credential s3_dchcp that needs to be further used in creating a datasource
OPEN MASTER KEY DECRYPTION BY PASSWORD = 'Start123!';
GO
IF NOT EXISTS(SELECT * FROM sys.credentials WHERE name = 's3_dchcp')
BEGIN
 CREATE DATABASE SCOPED CREDENTIAL s3_dchcp
 WITH IDENTITY = 'S3 Access Key',
 SECRET = 'amFzcw==:a3b9c163f6c520407ff34cfdb83ca5c6' ;
END
GO

-- Verify the new database-scoped credential
SELECT * FROM sys.database_scoped_credentials;

–The following creates an external data source which can then be referenced to for the location of files
CREATE EXTERNAL DATA SOURCE s3_dshcp
WITH
(   LOCATION = 's3://partner.hcpdemo.hitachivantara.com/sqlserver22'
,   CREDENTIAL = s3_dchcp
);
GO

-- Verify the new external data source
SELECT * FROM sys.external_data_sources;

```

For Parquet File format

```
SELECT  *
FROM    OPENROWSET
        (   BULK 'sampleparquetfile.parquet'
        ,   FORMAT       = 'PARQUET'
        ,   DATA_SOURCE  = 's3_dshcp'
        ) AS [cc];

CREATE EXTERNAL FILE FORMAT ParquetFileFormat WITH(FORMAT_TYPE = PARQUET);
GO

SELECT * FROM sys.external_file_formats;

CREATE EXTERNAL TABLE parq2
(
GageHeight2 VARCHAR (152),
Flow VARCHAR(152),
site_no VARCHAR(152),
datetime VARCHAR(152),
Precipitation VARCHAR(152),
GageHeight VARCHAR(152)
)
WITH
(LOCATION = 'sampleparquetfile.parquet', DATA_SOURCE = s3_dshcp, FILE_FORMAT = ParquetFileFormat);

GO

SELECT * FROM sys.external_tables;

select * from parq2;
select * from parq2 WHERE GageHeight2 = '4.830000' AND GageHeight = '4.750000';
select GageHeight2 from parq2 where datetime='2018-06-27 00:15';
select GageHeight2 from parq2 where datetime='2018-06-27 00:15';

SELECT  *
FROM    OPENROWSET
        (   BULK 'sampleparquetfile.parquet'
        ,   FORMAT       = 'PARQUET'
        ,   DATA_SOURCE  = 's3_dshcp'
        ) AS [cc]

                    WHERE GageHeight2 = '4.830000' AND GageHeight = '4.750000'
                    ;

SELECT  GageHeight2
FROM    OPENROWSET
        (   BULK 'sampleparquetfile.parquet'
        ,   FORMAT       = 'PARQUET'
        ,   DATA_SOURCE  = 's3_dshcp'
        ) AS [cc]

                    where datetime='2018-06-27 00:15'
                    ;

```

For CSV file format

```
SELECT * FROM sys.database_scoped_credentials;

 SELECT  *
FROM    OPENROWSET
        (   BULK 'emp_data.csv'
        ,   FORMAT      = 'CSV'
        ,   DATA_SOURCE  = 's3_dshcp'
                        ,   FIRSTROW     = 2
                        )
WITH    (
                        empno varchar(150),
                        ename varchar(150),
                        designation varchar(150),
                        manager varchar(150),
                        hire_date varchar(150),
                        sal varchar(150),
                        deptno varchar(150)
        )
    AS [cc]
    ;

CREATE EXTERNAL FILE FORMAT csv_ff
WITH
(   FORMAT_TYPE = DELIMITEDTEXT
,   FORMAT_OPTIONS  (    FIELD_TERMINATOR = ','
                    ,    STRING_DELIMITER = '"'
                    ,    FIRST_ROW = 2 )
);

CREATE EXTERNAL TABLE csvtable2
(
                        empno varchar(150),
                        ename varchar(150),
                        designation varchar(150),
                        manager varchar(150),
                        hire_date varchar(150),
                        sal varchar(150),
                        deptno varchar(150)
)
WITH
(
LOCATION = 'emp_data.csv', DATA_SOURCE = s3_dshcp, FILE_FORMAT = csv_ff
);

select * from csvtable2;

select * from csvtable2 WHERE empno= 7566;

select empno from csvtable2 WHERE deptno=20;

```

For Delta File Format

Uploaded the “people-10m” Dataset inside the HCP bucket:

Note: The people-10m delta table is a sample delta table from a sample dataset from Databricks as found at https://docs.microsoft.com/en-us/azure/databricks/data/databricks-datasets#sql. This dataset contains names, birthdates, and SSN which are all fictional and don't represent actual people.

The below query uses the OPENROWSET query on top of the people-10m dataset present in S3 and fetches 10 million rows of data taking around 2 minutes:

```
CREATE EXTERNAL FILE FORMAT DELTA
WITH (
      FORMAT_TYPE = DELTA
    );

SELECT *
FROM   OPENROWSET
     (  BULK 'people-10m'
     ,  FORMAT      = 'DELTA'
     ,  DATA_SOURCE  = 's3_dshcp'
     ) AS [r];
```

The following query creates External Table to query the DELTA dataset:

```
CREATE EXTERNAL TABLE EXT_PEOPLE10M2
(
id INT,
firstName VARCHAR(8000),
middleName VARCHAR(8000),
lastName VARCHAR(8000),
gender VARCHAR(8000),
birthDate datetime2,
ssn VARCHAR(8000),
salary int,
)
WITH
(
      LOCATION = 'people-10m'
     ,DATA_SOURCE = s3_dshcp
     ,FILE_FORMAT = DELTA
)

SELECT Top 1000 * FROM EXT_PEOPLE10M2

```

JSON Dataset

- External tables do not support JSON format - https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/develop-tables-external-tables
- https://learn.microsoft.com/en-us/sql/t-sql/statements/create-external-file-format-transact-sql?view=sql-server-ver16&tabs=json
- Creating JSON External File Format (for creating external tables) applies to Azure SQL Edge only. For information on using OPENROWSET to import JSON data in other platforms, see Import JSON documents into SQL Server or Query JSON files using serverless SQL pool in Azure Synapse Analytics.

The following sample query reads JSON and line-delimited JSON files and returns every document as a separate row.

```
SELECT  *
FROM    OPENROWSET
        (   BULK 'large.json'
                        ,  FORMAT ='csv'
                        ,fieldterminator ='0x0b'
        ,fieldquote = '0x0b'
        ,   DATA_SOURCE  = 's3_dshcp'
        )
with (jsonfile nvarchar(max)) as rows;

```

The queries in the previous examples return every JSON document as a single string in a separate row of the result set. You can use functions JSON_VALUE and OPENJSON to parse the values in JSON documents and return them as relational values:

JSON_VALUE Function:

```
select
   JSON_VALUE(j, '$.id') AS id,
   JSON_VALUE(j, '$.first_name') AS first_name,
   JSON_VALUE(j, '$.last_name') as last_name,
   JSON_VALUE(j, '$.email') as email,
   JSON_VALUE(j, '$.gender') AS gender,
   JSON_VALUE(j, '$.designation') AS designation,
   JSON_VALUE(j, '$.phone') as phone,
   JSON_VALUE(j, '$.country') as country

from openrowset(
      bulk 'large.json',
      data_source = 's3_dshcp',
      format = 'csv',
      fieldterminator ='0x0b',
      fieldquote = '0x0b'
   ) with (j nvarchar(max)) as rows

```

openjson Function:

```
select
   *
from openrowset(
       bulk 'large.json',
       data_source = 's3_dshcp',
       format = 'csv',
       fieldterminator ='0x0b',
       fieldquote = '0x0b'
    ) with (j nvarchar(max)) as rows
    cross apply openjson (j)
       with (  id int,
               first_name varchar(100),
               last_name varchar(100) ,
               country varchar(100) )

```

Create External Table As Select (CETAS) with Delta Format

Creating an external table from HCP data source and at the same time export the data to another place. In the below example, we will extract only a certain set of people from a HCP source to create/export/write a new folder on HCP S3 which has birthdate of people ranging between 1959 and 1970 :

```
---------------------------------------------------------------------------------
/*
Use CETAS to query the delta file and only extract a certain set of people to create/export/write a new folder on S3
which has
birthdate of people ranging
*/
----------------------------------------------------------------------------------

CREATE EXTERNAL FILE FORMAT ParquetFileFormat WITH(FORMAT_TYPE = PARQUET);

DROP EXTERNAL TABLE PEOPLE10M_60s;

CREATE EXTERNAL TABLE PEOPLE10M_60s
WITH
(   LOCATION = '1960s',
    DATA_SOURCE = s3_dshcp,
    FILE_FORMAT = ParquetFileFormat)
AS
SELECT * FROM OPENROWSET
(BULK 'people-10m', FORMAT = 'DELTA', DATA_SOURCE = 's3_dshcp') as [people]
WHERE YEAR(people.birthDate) > 1959 AND YEAR(people.birthDate) < 1970;

SELECT * FROM PEOPLE10M_60s ORDER BY birthDate;

```

After running the above queries , a New Folder (1960s) gets created with exported parquet files from the query above.

Demo.sql (Microsoft defined) Test Cases:

We have been provided a demo.sql file by Microsoft along with a people-10m Dataset.

1. Since we have a dataset with 10 million rows of people of different birthdates. We can initiate different queries on top of it:

Select Query from OPENROWSET to group users per decade

```sql
SELECT left(cast(r.birthDate as varchar(4)),3) + '0s' as decade, count(r.id) as users
FROM OPENROWSET
(BULK '/delta/people-10m',
FORMAT = 'DELTA', DATA_SOURCE = 's3_delta_eds') as [r]
GROUP by left(cast(r.birthDate as varchar(4)),3) + '0s'

```

```sql
SELECT TOP 1000 * FROM OPENROWSET
(BULK '/delta/people-10m/', FORMAT = 'DELTA', DATA_SOURCE = 's3_delta_eds') as [r]

```

```sql
IF OBJECT_ID('EXT_PEOPLE10M', 'ET') IS NOT NULL
      DROP EXTERNAL TABLE EXT_PEOPLE10M
GO

CREATE EXTERNAL TABLE EXT_PEOPLE10M
(
id INT,
firstName VARCHAR(8000),
middleName VARCHAR(8000),
lastName VARCHAR(8000),
gender VARCHAR(8000),
birthDate datetime2,
ssn VARCHAR(8000),
salary int,
)
WITH
(
       LOCATION = '/delta/people-10m'
      ,DATA_SOURCE = s3_delta_eds
      ,FILE_FORMAT = DeltaFileFormat
)

SELECT Top 1000 * FROM EXT_PEOPLE10M

```

2. In the below example, we are taking data for all the people born in their respective decades 50s, 60s ,70s , 80s, 90s, 2000s using the Select OPENROWSET query and then load that into the External table. Create External Tables from the OPENROWSET Data.

Note: the data is being taken from the delta files stored in **people-10m** and export those parquet files into the new location defined in “decades/19**“and then load that data into the External tables.

```
-- CETAS DECADES DEMO 50s -- ~5s (1.662.873)
CREATE EXTERNAL TABLE EXT_PEOPLE10M_50s
WITH
(   LOCATION = '/decades/1950s',
    DATA_SOURCE = s3_delta_eds,
    FILE_FORMAT = ParquetFileFormat)
AS
SELECT * FROM OPENROWSET
(BULK '/delta/people-10m', FORMAT = 'DELTA', DATA_SOURCE = 's3_delta_eds') as [r]
WHERE year(r.birthDate) < 1960

-- CETAS DECADES DEMO 60s -- ~7s (3.743.270)
CREATE EXTERNAL TABLE EXT_PEOPLE10M_60s
WITH
(   LOCATION = '/decades/1960s',
    DATA_SOURCE = s3_delta_eds,
    FILE_FORMAT = ParquetFileFormat)
AS
SELECT * FROM OPENROWSET
(BULK '/delta/people-10m', FORMAT = 'DELTA', DATA_SOURCE = 's3_delta_eds') as [r]
WHERE year(r.birthDate) > 1959 AND year(r.birthDate) < 1970

-- CETAS DECADES DEMO 70s -- ~8s (2.079.460)
CREATE EXTERNAL TABLE EXT_PEOPLE10M_70s
WITH
(   LOCATION = '/decades/1970s',
    DATA_SOURCE = s3_delta_eds,
    FILE_FORMAT = ParquetFileFormat)
AS
SELECT * FROM OPENROWSET
(BULK '/delta/people-10m', FORMAT = 'DELTA', DATA_SOURCE = 's3_delta_eds') as [r]
WHERE year(r.birthDate) > 1969 AND year(r.birthDate) < 1980

-- CETAS DECADES DEMO 80s -- ~6s (2.080.733)
CREATE EXTERNAL TABLE EXT_PEOPLE10M_80s
WITH
(   LOCATION = '/decades/1980s',
    DATA_SOURCE = s3_delta_eds,
    FILE_FORMAT = ParquetFileFormat)
AS
SELECT * FROM OPENROWSET
(BULK '/delta/people-10m', FORMAT = 'DELTA', DATA_SOURCE = 's3_delta_eds') as [r]
WHERE year(r.birthDate) > 1979 AND year(r.birthDate) < 1990

-- CETAS DECADES DEMO 90s -- ~5s (2.079.779)
CREATE EXTERNAL TABLE EXT_PEOPLE10M_90s
WITH
(    LOCATION = '/decades/1990s',
    DATA_SOURCE = s3_delta_eds,
    FILE_FORMAT = ParquetFileFormat)
AS
SELECT * FROM OPENROWSET
(BULK '/delta/people-10m', FORMAT = 'DELTA', DATA_SOURCE = 's3_delta_eds') as [r]
WHERE year(r.birthDate) > 1989 AND year(r.birthDate) < 2000

-- CETAS DECADES DEMO 2000s -- ~3s (16.758)
CREATE EXTERNAL TABLE EXT_PEOPLE10M_00s
WITH
(   LOCATION = '/decades/2000s',
    DATA_SOURCE = s3_delta_eds,
    FILE_FORMAT = ParquetFileFormat)
AS
SELECT * FROM OPENROWSET
(BULK '/delta/people-10m', FORMAT = 'DELTA', DATA_SOURCE = 's3_delta_eds') as [r]
WHERE year(r.birthDate) > 1999

```

The above query exported folders to new location (bucket) that was created manually:

3. The following will count the rows from the tables in the previous step.

```
SELECT COUNT (*) FROM EXT_PEOPLE10M_50s
SELECT COUNT (*) FROM EXT_PEOPLE10M_60s
SELECT COUNT (*) FROM EXT_PEOPLE10M_70s
SELECT COUNT (*) FROM EXT_PEOPLE10M_80s
SELECT COUNT (*) FROM EXT_PEOPLE10M_90s
SELECT COUNT (*) FROM EXT_PEOPLE10M_00s
```

4. The following will take the bucket as a whole and checks all the parquet files and search specifically with a WHERE clause

```
SELECT TOP 1000 * FROM OPENROWSET
(BULK '/decades/**', FORMAT = 'parquet', DATA_SOURCE = 's3_delta_eds')
WITH (
id INT,
firstName VARCHAR(8000),
middleName VARCHAR(8000),
lastName VARCHAR(8000),
gender VARCHAR(8000),
birthDate datetime2,
ssn VARCHAR(8000),
salary int
) as [r]
WHERE
year(r.birthDate) BETWEEN YEAR('1955-01-01') AND YEAR('1972-01-01')

```

Conclusion

Integrating Microsoft SQL Server PolyBase with the Hitachi Content Platform (HCP) provides great advantage.

Improved data access: PolyBase enables seamless access to data stored in HCP, allowing businesses to easily integrate and analyze their data.

Scalability: HCP provides scalable storage capabilities, enabling businesses to store and manage growing amounts of data without affecting performance.

Enhanced data analysis: PolyBase enables businesses to perform complex data analysis on their data stored in HCP, providing valuable insights into their data.

Cost-effectiveness: By utilizing PolyBase to access data stored in HCP, businesses can reduce the cost of data storage and improve efficiency.

Improved data security: HCP provides robust data protection features, including data replication, snapshots, and disaster recovery, ensuring that data is WORM safe and secure.

We ran extensive testing under guidance from Microsoft, and all tests passed without issue.

Further reading

Visit the following links to learn more about Hitachi Content Platform and Microsoft SQL Server:

- HCP product documentation (login required): https://knowledge.hitachivantara.com/Documents/Storage/HCP/
- Hitachi support documentation (login required):
- Configure PolyBase to access external data in S3-compatible object storage
- External tables do not support JSON format - https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/develop-tables-external-tables

  https://learn.microsoft.com/en-us/sql/t-sql/statements/create-external-file-format-transact-sql?view=sql-server-ver16&tabs=json

- Creating JSON External File Format (for creating external tables) applies to Azure SQL Edge only. For information on using OPENROWSET to import JSON data in other platforms, see Import JSON documents into SQL Server or Query JSON files using serverless SQL pool in Azure Synapse Analytics.

Hitachi Vantara

Corporate Headquarters 2535 Augustine Drive

Santa Clara, CA 95054 USA www.HitachiVantara.com community.HitachiVantara.com

Regional Contact Information

Americas: +1 866 374 5822 or info@hitachivantara.com

Europe, Middle East and Africa: +44 (0) 1753 618000 or info.emea@hitachivantara.com

Asia Pacific: +852 3189 7900 or info.marketing.apac@hitachivantara.com