CLOUD APPLICATION DEVELOPMENT

PROJECT

Data Warehousing with IBM Cloud Db2 Warehouse

Phase 2:Consider incorporating advanced analytics tools or machine learning models for predictive analysis within the data warehouse

Db2 Warehouse on Cloud is a fully managed data warehouse service.
One powerful use case in Db2 Warehouse on Cloud is to analyze born-on-the cloud data sets.
Those data sets can be generated in volumes from your mobile apps, web apps, or Internet of Things (IoT) services.
An organization might choose to store born-on-the-cloud data sets in Object Storage services as a staging area before choosing which sets to load into a data warehouse for further analysis.
Because Db2 Warehouse on Cloud supports loading data from Object Storage services with compatible APIs, data sets can be generated, stored, and analyzed on the cloud.
This guided demo illustrates how to use an external table to insert data from Cloud Object Storage to Db2 Warehouse on Cloud.
Typically, tables are in a database while external table data is in a text-based, delimited file outside of the database.
You can use an external table to store data outside the database but still be able to query the data and load it into a table in the database with casts, joins, and by dropping columns to manipulate data during loading.

An advantage of using external tables for extract, transform, and load (ETL) processes is that they can be carried out by using plain SQL.
Because you can start an SQL-based ETL process from any SQL client, you don't need special ETL tools.
To load data into the database from an external table, use a FROM clause in a SELECT SQL statement as you would for any other table. To unload data from the database into an external file, specify the external table as the target table in one of these

SQL statements:

- INSERT SQL
- SELECT INTO SQL
- CREATE EXTERNAL TABLE AS SELECT SQL

you load the data from a file that is stored on Cloud Object Storage. Db2 Warehouse on Cloud supports external tables that reference data files that are stored in Cloud Object Storage with compatible Swift or S3 APIs. Cloud Object Storage is a highly scalable cloud storage service that is designed for high durability, resiliency, and security. You can store, manage, and access your data through the self-service portal and RESTful APIs, or connect applications directly to Cloud Object Storage and integrate with IBM Cloud services.

Task 1: Set up Cloud Object Storage

When the demo starts, you are already logged in to IBM Cloud. If you follow these steps outside of this demo, log in to IBM Cloud by entering your credentials.
Create an instance of Cloud Object Storage

In the upper-right corner, click **Catalog**.

In the Storage section, click **Object Storage**. The description of the service is displayed.

Review the information about the service and make sure that the Lite plan is selected. Click **Create**.

Create a bucket and upload a data file
A *bucket* is where you organize and store the objects in Cloud Object Storage. To learn more about buckets and objects, see the Cloud Object Storage documentation.

Click the **Create your first bucket** link.

In the window that opens, click the **Name** field. The name `db2woc.demo.001` is displayed.

Note: Typically, you need to type the bucket name. In this demo, the name is automatically entered when you click the field.

Use the default values for the other settings and click **Create**.

Start to add your files to the bucket by clicking **Click here to add objects**.

In the window that opens, click **Add Files**. Click **bank_customers** and click **Choose**.

Click **Upload**. When the upload is complete, the file is listed on the page.

Create a service credential

From the left menu, click **Service credentials**.

On the right, click **New Credential**.

Click the **Add Inline Configuration Parameters (Optional)** field.
The `{"HMAC":true}` value is displayed.

Note: Typically, you need to type the value in the field. In this demo, the value is automatically entered when you click the field.

By default, Cloud Object Storage provides Tivoli Information Archive Manager token-based authentication. By adding the `{HMACS:true}` parameter, you ensure that the HMAC keys, such as `access_key_id` and `secret_access_key`, are generated. Those keys are compatible with most S3-compatible services, such as Db2 Warehouse on Cloud. To create an external table, you need the `access_key_id` and `secret_access_key` values to authenticate to Cloud Object Storage.

.

.

Click **Add**. A new credential is created. To see the values of the **access_key_id** and **secret_access_key** fields, in the Actions column, click **View credentials**. You need those values later to create the external table in Db2 Warehouse on Cloud.

From the left menu, click **Endpoint**. Take note of the use-geo endpoint information in the PUBLIC column. You need this information later to create the external table in Db2 Warehouse on Cloud.

.

Start Task 2

Task 2: Create a table in Db2 Warehouse on Cloud

An instance of Db2 Warehouse on Cloud Entry plan was already created for this demo. The “Services information” page for this instance is displayed in a second browser tab.
In this task, you create a typical table in Db2 Warehouse on Cloud that you insert data into later.
Go to the Db2 Warehouse on Cloud web console

Click the second browser tab: **Service Details – IBM Cloud**. The tab shows information for the Db2WoC_demo instance.

Click **OPEN** to open the Db2 Warehouse on Cloud console in a new tab.

Create the bank_customers table

In the web console, click **Run SQL** to open the SQL editor.

Open a script file:

a. Next to **Run selected**, click the folder icon. A browser window opens.

b. Click **1_bank_customers_ddl** and click **Choose**.

In the confirmation window, click **Confirm**. The warning is shown because opening a script replaces the content in the SQL editor. Make sure that you see the Create Table statement in the SQL editor.

Click **Run all** to run the DDL statement. The table is created in the console near the end of the page.

Start Task 3

Task 3: Create an external table and insert data into a typical table

Next, you create an external table that references the `bank_customers.csv` file that is stored in Cloud Object Storage. Cloud Object Storage is compatible with the S3 API, so you can use the s3 parameter in your definition of the external table
The external table syntax for Cloud Object Storage (s3) is `S3 (endpoint, authKey1, authKey2, bucket)`.
For more information about the external table syntax, see IBM Documentation.
In this demo, follow the naming convention of adding `et_` as prefix to the external table names.

Open a script file that contains the Create External Table DDL statement:

a. On the RUN SQL page, next to **Run Selected**, click the folder icon.

b. In the window that opens, click **2_create_ext_table_script** and click **Choose**.

In the confirmation message, click **Confirm**. Make sure that you see the Create External Table statement in the SQL editor along with four other SQL statements.

Select the entire `CREATE EXTERNAL TABLE` DDL statement by clicking the statement.

Observe the parameters, including the ones that you noted in Task 1, such as the credentials and the endpoint. This part allows Db2 Warehouse to locate and authenticate with the Object Storage service.

You also included extra parameters to suit the format of the data file, including defining the delimiter and null value. You can customize the load parameters for future data loads. For more information about the supported parameters, see IBM Documentation.

Click **Run Selected** to run only the Create External table statement.

Verify that the external table was created successfully by querying the table. Select the `select count(*) from et_bank_customers;` command by clicking the statement. Then, click **Run Selected**.

The result shows that the et_bank_customers table has 3000 rows.

Before you insert data into the bank_customers table, verify that the table is empty. Select the `select count(*) from bank_customers;` command by clicking the statement. Then, click **Run Selected**.

The result shows that the bank_customers table has 0 rows. The table is currently empty.

Insert data from the et_bank_customers external table into the bank_customers table. To do so, use an `insert from select` statement. Select the `insert into bank_customers select * from et_bank_customers;` command by clicking the statement. Then, click **Run Selected**.

In this case, you are selecting every record from the external table. If you want to filter out certain records, you can provide a WHERE condition. A major benefit of using an external table is being able to use SQL to filter the data before loading.

Check the row count in your target table. Select the `select count(*) from bank_customers;` command by clicking the statement. Then, click **Run Selected**.

The result shows that 3000 rows were inserted into the table. Your target table was loaded successfully.

To further explore the data, click **Explore**. A list of schemas is shown.
Click **DASH6308**.

Two tables were created in the schema: BANK_CUSTOMERS and ET_BANK_CUSTOMERS.

Click **BANK_CUSTOMERS**. The table definition is displayed. Click **View Data** to see the records. To return to the previous page, click **Back**.

Click **ET_BANK_CUSTOMERS**. You can see the same table definition, which is expected as you mirrored the external table definition to that of bank_customers.

Even though you can query and view data, the data that is associated with this external table is in a flat file. In this case, the file is stored in Cloud Object Storage