U-SQL Catalog and Database Creation

Overview

In this demo, you will create an Azure Data Lake database that contains some tables and views for ongoing big data processing and reporting.

Creating an Azure Data Lake Analytics Account

In this exercise, you will create an Azure Data Lake Analytics Account and associated Azure Data Lake store.

Create an Azure Data Lake Analytics Account

Before you can use Azure Data Lake Analytics to process data, you must create an Azure Data Lake Analytics account, and associate it with at least one Azure Data Lake store.

1. In a web browser, navigate to http://portal.azure.com, and if prompted, sign in using the Microsoft account that is associated with your Azure subscription.
2. In the Microsoft Azure portal, in the Hub Menu, click **New**. Then in the **Intelligence and analytics** menu, click **Data Lake Analytics**.
3. In the **New Data Lake Analytics Account** blade, enter the following settings, and then click **Create**:
   - **Name**: *Enter a unique name (and make a note of it!)*
   - **Subscription**: *Select your Azure subscription*
   - **Resource Group**: *Create a new resource group with a unique name*
   - **Location:** *Select any available region*
   - **Data Lake Store:** *Create a new Data Lake Store with a unique name (and make a note of it!)*
   - **Pin to dashboard**: *Not selected*
4. In the Azure portal, view **Notifications** to verify that deployment has started. Then wait for the resources to be deployed (this can take a few minutes.)

Upload Source Data Files

In this lab, you will use Azure Data Lake Analytics to process web server log data.

1. In the folder where you extracted the lab files, open the **iislogs** folder and then use a text editor to view the **2008-01.txt** file.
2. Review the contents of the file, noting that it contains some header rows (prefixed with a # character) and some space-delimited web server request records for the month of January in

2008. Then close the file without saving any changes. The other files in this folder contain similar data for February to June 2008.

3. In the Azure portal, view the **Data Explorer** page for your Azure Data Lake Analytics account, and create a new folder named **iislogs** in the root of your Azure Data Lake store.
4. Open the newly created **iislogs** folder. Then click **Upload**, select all of the files in the local **iislogs** folder (you can hold the CTRL key to select multiple files) and upload them.

Creating an Azure Data Lake Database

Although Azure Data Lake Analytics enables you to extract and process data directly from files in a data lake store, you can achieve better performance and reusability by using the Azure Data Lake catalog to create a database for your data.

Create a Database

Creating a database enables you to store data in a structured format, ready to be queried by jobs.

1. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click **New Job**.
2. In the **New U-SQL Job** blade, in the **Job Name** box, type **Create DB**.
3. In the code editor, enter the following code:

```
CREATE DATABASE IF NOT EXISTS webdata;
```

4. Click **Submit Job**, and observe the job status as it runs.
5. When the job has finished running, return to the blade for your Azure Data Lake Analytics account and click **Data Explorer**.
6. In the **Data Explorer** blade, under **Catalog**, verify that the **webdata** database is now listed in your Azure Data Lake analytics account (alongside the default **master** database).

Create a Schema and a Table

Now that you have a database, you can create tables in it.

1. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click **New Job**.
2. In the **New U-SQL Job** blade, in the **Job Name** box, type **Create Table**.
3. In the code editor, enter the following code:

```
USE DATABASE webdata;

CREATE SCHEMA IF NOT EXISTS iis;

CREATE TABLE iis.log
(date string,  time
string,  client_ip
string,  username
string,  server_ip
string,  port int,
method string,  stem
string,  query
string,  status
string,
server_bytes int,
client_bytes int,
time_taken int?,
user_agent string,
referrer string,
 INDEX idx_logdate CLUSTERED (date))
DISTRIBUTED BY HASH(client_ip);
```

This code creates a schema named **iis** and a table named **log** in the **webdata** database. The log table has a clustered index on the date column, and the data will be distributed in the data lake store based on a hash of the **client_ip** column. Note that the **time_taken** column is declared with the data type **int?**. The **?** character indicates that this numeric column allows NULL values (the string data type allows NULLs by default).

4. Click **Submit Job**, and observe the job status as it runs.
5. When the job has finished running, return to the blade for your Azure Data Lake Analytics account and click **Data Explorer**.
6. In the **Data Explorer** blade, under **Catalog**, expand your account, expand the **webdata** database, expand **Tables**, and select **iis.log**. Note that the details of this table, including its columns and indexes are shown here.

Insert Data into the Table

The table you have created is initially empty. You can use a U-SQL job to insert data into it.

1. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click **New Job**.
2. In the **New U-SQL Job** blade, in the **Job Name** box, type **Load Table**.
3. In the code editor, enter the following code:

```
USE DATABASE webdata;

@log =
EXTRACT date string,
time string,
client_ip string,
username string,
server_ip string,
port int,         method
string,         stem
string,         query
string,         status
string,
server_bytes int,
client_bytes int,
time_taken int?,
user_agent string,
referrer string FROM
"/iislogs/{*}.txt"
USING Extractors.Text(' ', silent:true);

INSERT INTO iis.log
SELECT * FROM @log;
```

This code reads all files with a .txt extension from the **iislogs** folder into a schema that matches the table, and then inserts the extracted data into the table.

4. Click **Submit Job** and observe the job details as it is run.
5. When the job has finished running, return to the blade for your Azure Data Lake Analytics account and click **Data Explorer**.
6. In the **Data Explorer** blade, under **Catalog**, expand your account, expand the **webdata** database, expand **Tables**, and select **iis.log**. Then click **Query Table**.
7. Modify the default query as follows:

```
@table = SELECT * FROM [webdata].[iis].[log]
         ORDER BY date, time          FETCH FIRST 100;

OUTPUT @table
    TO "/Outputs/webdata.iis.log.tsv"
    USING Outputters.Tsv();
```

This code queries the table and returns the first 100 rows ordered by date and time.

8. Click **Submit Job** and observe the job details as it is run.
9. When the job has finished, click the **Output** tab and select **webdata.iis.log.tsv** to see a preview of the results, and verify that the table now contains data.

Creating and Querying a View

Views encapsulate complex queries, and provide a layer of abstraction over tables. They are commonly used in relational databases, and are supported in U-SQL.

Create a View

To create a view, you can use the CREATE VIEW statement.

1. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click **New Job**.
2. In the **New U-SQL Job** blade, in the **Job Name** box, type **Create View**.
3. In the code editor, enter the following code:

```
USE DATABASE webdata;

CREATE VIEW iis.summary
AS
SELECT date,
       COUNT(*) AS hits,
       SUM(server_bytes) AS bytes_sent,
       SUM(client_bytes) AS bytes_received
FROM iis.log
GROUP BY date;
```

This code creates a view named **summary** that retrieves daily aggregated values from the **log** table.

4. Click **Submit Job** and observe the job details as it is run.

5. When the job has finished running, return to the blade for your Azure Data Lake Analytics account and click **Data Explorer**.

Querying a View

1. In the **Data Explorer** blade, under **Catalog**, expand your account, expand the **webdata** database, expand **Views**, and select **iis.summary**. Then click **Query View**.
2. Modify the default query as follows:

   ```
   @view = SELECT * FROM [webdata].[iis].[summary];

   OUTPUT @view
       TO "/Outputs/webdata.iis.summary.tsv"
       ORDER BY date
       USING Outputters.Tsv();
   ```

   This code queries the view and returns the output ordered by date.

3. Click **Submit Job** and observe the job details as it is run.
4. When the job has finished, click the **Output** tab and select **webdata.iis.summary.tsv** to see a preview of the results.
5. Download the output file and open it in a text editor or spreadsheet application, and note that each row in the data contains a daily summary of hits, bytes sent, and bytes received for January to June 2008.

Using Table-Valued Functions

Create a Table-Valued Function

Table-values functions provide another way to encapsulate a query that returns a rowset. Additionally, table-values functions can include parameters; making them more flexible than views in some scenarios.

1. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click **New Job**.
2. In the **New U-SQL Job** blade, in the **Job Name** box, type **Create TVF**.
3. In the code editor, enter the following code:

   ```
   USE DATABASE webdata;

   CREATE FUNCTION iis.summarizelog(@Year int, @Month int)
   RETURNS @summarizedlog TABLE
   (   date string,
   hits long?,
   bytes_sent long?,
   bytes_received long?
   )
   AS
   BEGIN
     @summarizedlog =
   SELECT date,
   hits,
   bytes_sent,
   bytes_received
       FROM iis.summary
   ```

```
    WHERE DateTime.Parse(date).Year == @Year
AND DateTime.Parse(date).Month == @Month;
END;
```

This code creates a function named **summarizelog** that retrieves data from the **summary** view for a specified year and month.

4. Click **Submit Job** and observe the job details as it is run.
5. When the job has finished running, return to the blade for your Azure Data Lake Analytics account and click **Data Explorer**.

Querying a Table-Valued Function

1. In the **Data Explorer** blade, under **Catalog**, expand your account, expand the **webdata** database, expand **Table values Functions**, and verify that **iis.summarizelog** is listed.
2. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click **New Job**.
3. In the **New U-SQL Job** blade, in the **Job Name** box, type **Query TVF**.
4. In the code editor, enter the following code:

```
USE DATABASE webdata;

@june = iis.summarizelog(2008, 6);

OUTPUT @june
    TO "/Outputs/june.csv"
    ORDER BY date
    USING Outputters.Csv();
```

This code calls the **summarizelog** function, specifying the parameters for June 2008, returns the output ordered by date.

5. Click **Submit Job** and observe the job details as it is run.
6. When the job has finished, click the **Output** tab and select **june.csv** to see a preview of the results, and note that each row in the data contains a daily summary of hits, bytes sent, and bytes received for June 2008

Using Procedures

Create a Procedure

Procedures provide a way to encapsulate tasks, such as extracting data from files and inserting it into tables.

1. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click **New Job**.
2. In the **New U-SQL Job** blade, in the **Job Name** box, type **Create Procedure**.
3. In the code editor, enter the following code:

```
USE DATABASE webdata;

CREATE PROCEDURE iis.LoadLog (@File string)
AS
BEGIN
  @log =
    EXTRACT date string,
time string,
client_ip string,
username string,
server_ip string,
port int,              method
string,              stem
string,              query
string,              status
string,
server_bytes int,
client_bytes int,
time_taken int?,
user_agent string,
referrer string
    FROM @File
    USING Extractors.Text(' ', silent:true);

  INSERT INTO iis.log
  SELECT * FROM @log;
END;
```

This code creates a procedure named **LoadLog** that loads data from the specified file path into the **log** table.

4. Click **Submit Job** and observe the job details as it is run.
5. When the job has finished running, return to the blade for your Azure Data Lake Analytics account and click **Data Explorer**.

Running a Procedure

1. In **Data Explorer**, under **Storage Accounts**, browse to the **iislogs** folder, which currently contains log files for January to June.
2. Click **Upload**, browse to the **July** folder in the local folder where you extracted the lab files, and upload **2008-07.txt**.
3. In the **Data Explorer** blade, under **Catalog**, expand your account, expand the **webdata** database, expand **Procedures**, and select **iis.LoadLog**. Then click **Run Procedure**.
4. In the code editor, modify the code as follows:
```
[webdata].[iis].[LoadLog]("/iislogs/2008-07.txt");
```

   This code uses the **LoadLog** procedure to load the data from the **2008-07.txt** file in the **iislogs** folder into the **log** table.

5. Click **Submit Job** and observe the job details as it is run.
6. When the job has finished, on the blade for your Azure Data Lake Analytics account, click **New Job**.
7. In the **New U-SQL Job** blade, in the **Job Name** box, type **Get July**.
8. In the code editor, enter the following code:
```
USE DATABASE webdata;
```

```
@july = iis.summarizelog(2008, 7);

OUTPUT @july
    TO "/Outputs/july.csv"
    ORDER BY date
    USING Outputters.Csv();
```

This code calls the function, specifying the parameters for July 2008, returns the output ordered by date.

9. Click **Submit Job** and observe the job details as it is run.
10. When the job has finished, click the **Output** tab and select **july.csv** to see a preview of the results, and note that each row in the data contains a daily summary of hits, bytes sent, and bytes received for July 2008. This data was inserted into the **log** table by the **LoadLog** procedure, and then queried through the **summary** view by the **summarizelog** table-valued function.