Microsoft Azure

azure data lake

Hyun Suk Shin (AZURE CSA)

2017

Azure Data Lake를 통한 대용량 데이터 분석하기

본 실습을 통해 데이터 분석 플랫폼인 Azure Data Lake를 통해 비정형 데이터를 분석 환경 구성과 U-SQL로 데이터 분석을 진행합니다.

1. 실습을 위해서 사전에 계정과 구독 그리고 실습을 위한 관련 도구(tool)을 다운로드 받습니다.
   1. Azure 계정과 구독
      1. Azure에 접속하기 위해서는 Microsoft 계정이 필요합니다. Microsoft 계정이 없는 경우 <http://msn.com>에서 새로 생성할 수 있습니다.
         1. <http://msn.com> 을 방문하여 오른쪽 상단에 ‘로그인’을 클릭하고 ‘계정을 만드세요’를 클릭합니다.

|  |  |
| --- | --- |
|  |  |

* + - 1. 계정을 생성 할 때 아래와 이미 존재하는 회사 메일 주소를 사용할 수 있습니다.

|  |
| --- |
|  |

* + 1. 제공된 Azure Free Pass를 사용하여 무료 구독을 생성합니다. Free Pass는 30일 동안 제공된 금액 미만으로 사용이 가능하며 제공된 비용을 초과하게되는 경우 또는 30일이 경과하는 경우는 자동으로 중지 됩니다.
  1. 실습 비용 한도에 따른 유의 사항
     1. 실습을 종료 하고나서 가급적 사용하지 않는 서비스들은 삭제합니다. 불필요한 비용 발생으로 무료 구독이 중지되면 실습 진행이 어렵습니다.
     2. 고사양의 가상 컴퓨터나 서비스를 생성하지 않습니다. 고사양 서비스들은 고비용이 청구되어 단시간에 제공된 무료 구독의 비용을 초과할 수 있습니다.

1. 본 실습의 시나리오는 ADL (Azure Data Lake)를 사용하여 비정형 데이터를 분석 하는 것 입니다.
   1. 첫 번째 실습은 Azure Portal을 사용하여 Azure Data Lake Analytics 계정을 만들고, U-SQL로 Data Lake Analytics 작업을 정의하고, 작업을 Data Lake Analytics 계정에 제출하는 방법에 대해 알아봅니다.
   2. 두 번째 실습은 간단한 웹 서버의 로그를 전처리 합니다.
   3. 세 번째 실습은 차량의 운전자와 수집된 차량의 센서 데이터를 분석 합니다.
2. 데이터 레이크 저장소 계정을 생성합니다.
   1. Azure Portal에 로그인합니다. [+ NEW]를 클릭합니다.
   2. “Intelligence + Analytics” > “Data Lake Store”를 클릭합니다.

|  |
| --- |
|  |

* 1. 새로 생성하기 위한 정보를 입력합니다.
     1. Name: Data Lake Store 계정의 이름을 지정합니다.
     2. 구독은 Azure Free를 선택 합니다.
     3. 리소스 그룹은 새로 생성합니다. 아래 예제와 동일하게 “adl-weblogs” 명명하도록 합니다.
     4. Location은 현재 두 곳만 제공됩니다. Central US, East US 2 중 한 곳을 선택합니다.
     5. Pricing 모델 경우, 본 실습에 최소한 비용이 지출 되도록 “Pay-as-you-go”를 선택합니다.
     6. [Create]을 클릭하여 생성합니다. 생성에는 몇 초 정도 소요 됩니다.

|  |
| --- |
|  |

1. Azure Data Lake Analytics 계정을 생성합니다.
   1. “Intelligence + Analytics” > “Data Lake Store”를 클릭합니다.

|  |
| --- |
|  |

* 1. 새로 생성하기 위한 정보를 입력합니다.
     1. 이름은 인터넷 상에서 유일한 이름을 명명해야 합니다.
     2. 구독은 Azure Free를 선택합니다.
     3. 리소스 그룹은
     4. 지역 Data Late Store

|  |
| --- |
|  |

* + 1. Data Lake Store 경우는 위 단계에서 생성한 저장소 계정을 선택합니다.

|  |
| --- |
|  |

* + 1. Price Tier는 “Pay-as-you-go”를 선택합니다.
    2. [Create] 를 클릭하여 생성합니다. 서비스 생성에는 1분 정도 소요될 수 있습니다.

1. 실습을 위한 샘플 데이터를 확보 합니다.
   1. Azure Data Lake 서비스 소개 부분에 포함되어있는 “학습 (Learn)”을 클릭합니다.

|  |
| --- |
|  |

* 1. “Sample Data Missing”을 클릭합니다. Sample 데이터가 자동으로 populate 됩니다.

|  |
| --- |
|  |

* + 1. 학습을 위해 자동으로 생성된 데이터를 확인 해봅니다.
       1. “데이터 탐색기 (Data Explorer)“를 클릭합니다.

|  |
| --- |
|  |

* + - 1. 아래와같이 정상적으로 Sample이라는 경로가 생겼는지 확이납니다.

|  |
| --- |
|  |

* + - 1. “Sample” > “Data” 디렉토리를 클릭하여 아래와 같이 데이터가 저장되어있는지 확인 합니다.

|  |
| --- |
|  |

1. 실습을 위한 데이터가 확보 되었으므로 분석을 해보도록 하겠습니다.
   1. Azure Data Lake Analytics에서 [+ New Job]을 클릭합니다.

|  |
| --- |
|  |

* + 1. 아래와 같이 작업 이름을 “Query1” 이라고 입력합니다.

|  |
| --- |
|  |

* + 1. 아래 쿼리를 복사하여 쿼리 창에 붙여 넣습니다. 이 U-SQL 스크립트는 Extractors.Tsv()를 사용하여 원본 데이터 파일을 읽은 다음 Outputters.Csv()를 사용하여 csv 파일을 만듭니다. 원본 파일을 다른 위치에 복사하지 않는 한 두 경로를 수정하지 마세요. 출력 폴더가 없는 경우 Data Lake Analytics에서 해당 폴더를 만듭니다. 이런 경우에는 간단한 상대 경로가 사용됩니다.

|  |
| --- |
| @searchlog =  EXTRACT UserId int,  Start DateTime,  Region string,  Query string,  Duration int?,  Urls string,  ClickedUrls string  FROM "/Samples/Data/SearchLog.tsv"  USING Extractors.Tsv();  OUTPUT @searchlog  TO "/Output/SearchLog-from-Data-Lake.csv"  USING Outputters.Csv(); |
|  |

* + 1. [Submit Job]을 클릭하여 실행합니다. 작업 상태가 Succeeded로 바뀔 때까지 기다립니다. 약 1분 예상됩니다.

|  |
| --- |
|  |

* + 1. 작업 진척도를 보여주는 화면으로 전환됩니다. 작업 진행 과정과 진행률이 가시적으로 표시 됩니다.

|  |
| --- |
|  |

* + 1. 작업이 완료되면 “Show More…”을 클릭하여 작업에 따른 예상 비용이 얼마인지를 확인 합니다.

|  |
| --- |
|  |

* + 1. “Output”을 클릭하고 “SearchLog-from-Data-Lake.csv를 클릭하여 출럭 데이터를 확인 합니다.

|  |
| --- |
|  |

1. 비정형 파일들에 대한 U-SQL을 실습합니다.
   1. 비정형 데이터를 읽어 들여서 원하는 형탤 변환 후 파일로 저장합니다.
      1. 아래 쿼리를 실행합니다.

|  |
| --- |
| //0. Initialize some parameters/constants  DECLARE @INPUTDIR string = "/Samples/Data/AmbulanceData/";  DECLARE @INPUT\_DRIVERS string = @INPUTDIR + "Drivers.txt";  DECLARE @OUTPUTDIR string = "/output/";  DECLARE @OUTPUT string = @OUTPUTDIR + "drivers.out";  //1. Extract  @Drivers =  EXTRACT driver\_id int  , name string  , street string  , city string  , region string  , zipcode string  , country string  , phone\_numbers string  FROM @INPUT\_DRIVERS  USING Extractors.Tsv(encoding: Encoding.Unicode);  //2. SELECT Transformation  @Result =  SELECT country, city, COUNT(\*) AS NumberOfDrivers  FROM @Drivers  GROUP BY country, city  ORDER BY NumberOfDrivers DESC, country, city  FETCH FIRST 10 ROWS;  //OFFSET 0 ROWS FETCH FIRST 10 ROWS ONLY;  //3. Output  OUTPUT @Result  TO @OUTPUT  USING Outputters.Csv(); |

* 1. Azure Data Lake Store에서 두 개의 파일을 JOIN하여 결과 파일을 생성합니다.
     1. DB와 스키마가 있는 Table을 생성합니다.

|  |
| --- |
| // Create tables and index in one statement  // and in the default database (master) and default schema (dbo)  // Note that every table needs to have a clustered index before data can be inserted.  // Index names show use of quoted identifiers  // Sice tables are small, use a small number of hash buckets  DROP TABLE IF EXISTS Drivers;  CREATE TABLE IF NOT EXISTS Drivers(  driver\_id int  , name string  , street string  , city string  , region string  , zipcode string  , country string  , phone\_numbers SQL.MAP<string,string>  , INDEX [idx Drivers] CLUSTERED (driver\_id ASC)  DISTRIBUTED BY HASH (driver\_id) INTO 2 // only 2 buckets  );  DROP TABLE IF EXISTS Trips;  CREATE TABLE Trips(  date DateTime  , driver\_id int  , vehicle\_id int  , trips SQL.ARRAY<int>  , INDEX [idx\_Trips] CLUSTERED (driver\_id ASC, date, vehicle\_id)  DISTRIBUTED BY HASH (driver\_id) INTO 5 // 5 hash buckets  );  // Create a new database and new schema  // Using quoted identifiers  DROP DATABASE IF EXISTS DriverDB;  CREATE DATABASE DriverDB;  USE DATABASE DriverDB;  CREATE SCHEMA [Driver Schema];  USE SCHEMA [Driver Schema];  // Create table and create clustered index separately.  CREATE TABLE Drivers(  driver\_id int  , name string  , street string  , city string  , region string  , zipcode string  , country string  , phone\_numbers SQL.MAP<string,string>  );  CREATE CLUSTERED INDEX idx\_Drivers  ON DriverDB.[Driver Schema].Drivers(driver\_id ASC) DISTRIBUTED BY HASH (driver\_id) INTO 2;  CREATE TABLE Trips(  date DateTime  , driver\_id int  , vehicle\_id int  , trips SQL.ARRAY<int>  );  CREATE CLUSTERED INDEX idx\_Trips  ON Trips(driver\_id ASC, date, vehicle\_id) DISTRIBUTED BY HASH (driver\_id) INTO 5;  // Create a second version of the table back in the default database  USE DATABASE master;  DROP TABLE IF EXISTS USDrivers;  CREATE TABLE USDrivers(  driver\_id int  , name string  , street string  , city string  , region string  , zipcode string  , country string  , phone\_numbers SQL.MAP<string,string>  , INDEX idx1 CLUSTERED (driver\_id ASC)  DISTRIBUTED BY HASH (driver\_id) INTO 2  );  DROP TABLE IF EXISTS USTrips;  CREATE TABLE USTrips(  date DateTime  , driver\_id int  , vehicle\_id int  , trips SQL.ARRAY<int>  );  CREATE CLUSTERED INDEX idx\_Trips  ON USTrips(driver\_id ASC, date, vehicle\_id) DISTRIBUTED BY HASH (driver\_id) INTO 5; |

* + 1. Table에 데이터를 Load하기 위해 아래 Query를 실행합니다. 아울러 두 테이블 데이터를 Join하여 새로운 Output을 생성합니다. Query 실행이 완료되면 데이터 탐색기를 통해 ” Output/trips.out” 파일을 클릭하여 결과를 확인합니다.

|  |
| --- |
| //0. Initialize some parameters/constants  DECLARE @DRIVERFILE string = "Drivers.txt"; // TAB Separated Unicode file  DECLARE @TRIPFILE string = "DriverShiftTrips.csv"; // Excel CSV (ANSI)  DECLARE @ADL\_INPUTDIR string = "/Samples/Data/AmbulanceData/";  DECLARE @ADL\_TRIPS string = @ADL\_INPUTDIR + @TRIPFILE;  DECLARE @ADL\_DRIVERS string = @ADL\_INPUTDIR + @DRIVERFILE;  DECLARE @OUTPUT string = "Output/trips.out";  //1. Extract Drives from ADL  @DriversRAW =  EXTRACT driver\_id int  , name string  , street string  , city string  , region string  , zipcode string  , country string  , phone\_numbers string // Map  FROM @ADL\_DRIVERS  USING Extractors.Text(delimiter: '\t', quoting: true, encoding:Encoding.Unicode );  //alternative version with Tsv() and using defaults: USING Extractors.Tsv(encoding: Encoding.Unicode );  //2. Extract from ADL Store  @TripsRAW =  EXTRACT date DateTime  , driver\_id int  , vehicle\_id int  , trips string // Array  FROM @ADL\_TRIPS  USING Extractors.Text(quoting: true );  @Drivers =  SELECT driver\_id, name, street, city, region, zipcode, country  , phone\_numbers == "" ? null :  new SQL.MAP<string,string> (from p in phone\_numbers.Split(',') select new KeyValuePair<string, string>(p.Split(':')[0],p.Split(':')[1])) AS phone\_numbers  // Alternative:  // , new SQL.MAP<string,string> (phone\_numbers.Split(',').Select(p => new KeyValuePair<string, string>(p.Split(':')[0],p.Split(':')[1])) AS phone\_numbers  FROM @DriversRAW;  @Trips =  SELECT date  , driver\_id  , vehicle\_id  , trips == null ? null : new SQL.ARRAY<int>(trips.Split(',').Select(x => Convert.ToInt32(x))) AS trip\_ids  FROM @TripsRAW;  @Exploded\_Drivers =  SELECT driver\_id, name, street, city, region, zipcode, country  , P.phone\_type, P.phone\_number  FROM @Drivers AS d  CROSS APPLY EXPLODE (d.phone\_numbers) AS P(phone\_type, phone\_number);  @Exploded\_Trips =  SELECT date  , driver\_id  , vehicle\_id  , trip\_id  FROM @Trips CROSS APPLY EXPLODE (trip\_ids) AS T(trip\_id);  //3. Join  // Note the quoted identifier  // U-SQL JOINS:  // - Supports all kinds of joins: [INNER] JOIN, (LEFT | RIGHT | FULL) OUTER JOIN, CROSS JOIN, SEMI JOIN  // - Join conditions have to be constant comparisons (use additional rowset variables if you need to perform calculations  // - only comparison supported are equality  // - old-style joins of the form FROM A, B WHERE condition is disallowed  @DriverTrips =  SELECT d.name  , d.street  , d.city  , t.vehicle\_id  , t.date  FROM @Exploded\_Drivers AS d  LEFT OUTER JOIN @Exploded\_Trips AS t  ON d.driver\_id == t.driver\_id;  //3. Output  OUTPUT @DriverTrips  TO @OUTPUT  ORDER BY name,  date  USING Outputters.Csv(); |

1. 특화된 기능(Function)을 실습합니다.
   1. 여러개 파일을 하나에 View로 생성하고 이를 테이블로 만듭니다. View가 만들어지는지 확인 합니다.

|  |
| --- |
| // Creating a View and a table from the view with CTAS  // Assumes that vehicle CSV data has been already loaded to specified ADL store location  // View definitions cannot see variables, so we have to inline the paths explicitly  DROP VIEW IF EXISTS VehicleData\_View;  CREATE VIEW VehicleData\_View  AS  EXTRACT vehicle\_id int  , entry\_id long  , event\_date DateTime  , latitude float  , longitude float  , speed int  , direction string  , trip\_id int?  FROM  "/Samples/Data/AmbulanceDemo/vehicle1\_09172014.csv", "/Samples/Data/AmbulanceDemo/vehicle1\_09162014.csv", "/Samples/Data/AmbulanceDemo/vehicle1\_09152014.csv", "/Samples/Data/AmbulanceDemo/vehicle1\_09142014.csv",  "/Samples/Data/AmbulanceDemo/vehicle2\_09172014.csv", "/Samples/Data/AmbulanceDemo/vehicle2\_09162014.csv", "/Samples/Data/AmbulanceDemo/vehicle2\_09152014.csv", "/Samples/Data/AmbulanceDemo/vehicle2\_09142014.csv",  "/Samples/Data/AmbulanceDemo/vehicle3\_09172014.csv", "/Samples/Data/AmbulanceDemo/vehicle3\_09162014.csv", "/Samples/Data/AmbulanceDemo/vehicle3\_09152014.csv", "/Samples/Data/AmbulanceDemo/vehicle3\_09142014.csv",  "/Samples/Data/AmbulanceDemo/vehicle4\_09172014.csv", "/Samples/Data/AmbulanceDemo/vehicle4\_09162014.csv", "/Samples/Data/AmbulanceDemo/vehicle4\_09152014.csv", "/Samples/Data/AmbulanceDemo/vehicle4\_09142014.csv"  USING Extractors.Csv();  // Create and load table from CSV files  DROP TABLE IF EXISTS vehicle\_GPS;  CREATE TABLE vehicle\_GPS(INDEX vehicle\_GPS\_idx CLUSTERED (vehicle\_id ASC) PARTITIONED BY HASH (entry\_id, vehicle\_id)  ) AS SELECT \* FROM VehicleData\_View; |

* 1. Table을 Value로하는 Function을 생성합니다. 실행 후 "/output/speed0.csv” 파일과 "/output/speed35.csv” 결과 파일을 확인해 봅니다.

|  |
| --- |
| // Table-Valued Function  // Assumes that vehicle CSV data has been already loaded to specified ADL store location and that view has been created  DROP FUNCTION IF EXISTS VehicleData\_for\_Speed;  CREATE FUNCTION VehicleData\_for\_Speed  (  @minspeed int = 0  )  RETURNS @result TABLE  (  vehicle\_id int,  entry\_id long,  event\_date DateTime,  latitude float,  longitude float,  speed int,  direction string,  trip\_id int?  )  AS BEGIN  @result = SELECT \* FROM VehicleData\_View;  @result = SELECT \* FROM @result WHERE speed >= @minspeed;  END;  // Output default  OUTPUT VehicleData\_for\_Speed  (  DEFAULT  )  TO "/output/speed0.csv"  USING Outputters.Csv();  // Output above inner city speedlimit  OUTPUT VehicleData\_for\_Speed  (  35  )  TO "/output/speed35.csv"  USING Outputters.Csv(); |

* 1. 저장 프로시저를 생성합니다.

|  |
| --- |
| // Show definition and usage of a Procedure.  // Note that procedures are not allowed to contain DDL statements.  DROP PROCEDURE IF EXISTS VehicleData\_Proc;  CREATE PROCEDURE VehicleData\_Proc  (  @speed int = 0  )  AS BEGIN  @southbound =  SELECT \* FROM VehicleData\_for\_Speed(DEFAULT) AS S WHERE direction == "S";    OUTPUT @southbound  TO "/output/southbound.csv"  USING Outputters.Csv();  OUTPUT VehicleData\_for\_Speed(@speed)  TO "/output/speed.csv"  USING Outputters.Csv();  END;  // Now call it  VehicleData\_Proc(25); |

1. 데이터 분석 쿼리를 작성하여 실습해 봅니다. Query 실행 완료 후 결과 파일 3개를 각각 확인합니다.

|  |
| --- |
| // Showing Analytics capabilities over Ambulance Data using Windowing functions  //  // Requires Ambulance data to be loaded into vehicle\_GPS table.  // First define some paths  DECLARE @ADL\_DIR string = "/output/windowing/";  DECLARE @WINFCT\_OUTPUT string = @ADL\_DIR + "analytics\_window\_functions.txt";  DECLARE @RUNAGG\_OUTPUT string = @ADL\_DIR + "analytics\_running\_aggregates.txt";  DECLARE @LONGESTTRIP\_OUTPUT string = @ADL\_DIR + "ambulance\_longest\_trips.txt";  @vehicles =  SELECT  a.vehicle\_id AS vehicle\_id,  (a.event\_date).ToString("yyyy-MM-dd") AS event\_date,  a.trip\_id AS trip\_id,  (DateTime) MIN (a.event\_date) AS start\_time,  (DateTime) MAX (a.event\_date) AS finish\_time,  AVG(speed) AS avg\_speed,  MAX(speed) AS max\_speed  FROM vehicle\_GPS AS a  GROUP BY vehicle\_id, (a.event\_date).ToString("yyyy-MM-dd"), trip\_id;  @add\_duration = SELECT vehicle\_id  , event\_date  , trip\_id  , start\_time  , finish\_time  , (finish\_time).Subtract(start\_time).TotalMinutes AS duration  , (start\_time).ToString("yyyy-MM-dd") AS dispatch\_date  , avg\_speed  , max\_speed  FROM @vehicles;  @duration\_analytics = SELECT  DISTINCT dispatch\_date AS dispatch\_date  , vehicle\_id AS vehicle\_id  , AVG (duration) OVER (PARTITION BY dispatch\_date, vehicle\_id) AS avg\_trip\_length\_per\_vehicle\_eachday  , SUM (duration) OVER (PARTITION BY dispatch\_date, vehicle\_id) AS total\_trip\_length\_per\_vehicle\_eachday  , SUM (duration) OVER () AS total\_trip\_length  , PERCENTILE\_DISC(0.5) WITHIN GROUP (ORDER BY duration ASC) OVER (PARTITION BY dispatch\_date, vehicle\_id) AS median\_trip\_length\_per\_vehicle\_eachday  , PERCENTILE\_DISC(0.9) WITHIN GROUP (ORDER BY duration ASC) OVER (PARTITION BY dispatch\_date, vehicle\_id) AS top\_percentile\_trip\_length\_per\_vehicle\_eachday  , MAX (duration) OVER (PARTITION BY dispatch\_date, vehicle\_id) AS the\_longest\_trip\_per\_vehicle\_eachday  , MAX (duration) OVER (PARTITION BY dispatch\_date) AS the\_longest\_trip\_eachday  FROM @add\_duration;  OUTPUT @duration\_analytics  TO @WINFCT\_OUTPUT  ORDER BY dispatch\_date, vehicle\_id  USING Outputters.Csv();  @duration\_analytics1 = SELECT  dispatch\_date AS dispatch\_date  , vehicle\_id AS vehicle\_id  , AVG (duration) OVER (PARTITION BY dispatch\_date ORDER BY dispatch\_date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS avg\_trip\_length\_eachday  , SUM (duration) OVER (PARTITION BY dispatch\_date ORDER BY dispatch\_date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS total\_trip\_length\_eachday  FROM @add\_duration;  OUTPUT @duration\_analytics1  TO @RUNAGG\_OUTPUT  ORDER BY dispatch\_date, vehicle\_id  USING Outputters.Csv();  @ordered\_trips\_by\_duration = SELECT \*,  ROW\_NUMBER() OVER (PARTITION BY dispatch\_date ORDER BY duration DESC) AS row\_number  FROM @add\_duration;  @three\_longest\_trips = SELECT dispatch\_date  , row\_number  , duration  FROM @ordered\_trips\_by\_duration  WHERE row\_number <=3;  OUTPUT @three\_longest\_trips  TO @LONGESTTRIP\_OUTPUT  ORDER BY dispatch\_date, row\_number, duration  USING Outputters.Csv(); |

실습이 종료되었습니다. 불필요한 서비스를 삭제하여 비용 지출을 최소화 하도록 합니다.