Group Assignment

ASQL

by:Adrian Krebs, Kyle Fowler, Matt Anselmo

date:10/12/14

# Transform Requirements

Our approach for all 3 of these transformations were the same. They shared the same approach as they all live in the same sphere of altering data that comes from the csv file. We decided to alter these fields before actually placing the values into the database, for ease, efficiency and consistency.

### Convert Fahrenheit

Before we place a temperature into the database we call the following function,

tmin = GetTemp(count, row[18], "temp min");

tmax = GetTemp(count, row[19], "temp max");

tavg = GetTemp(count, row[4], "temp average");

This function looks like the following

double CelciusValue = 0;

CelciusValue = 5f/9f\*(FahrenheitValue - 32);

CelciusValue = Math.Round(CelciusValue, 2);

return CelciusValue.ToString();

Pretty much in a nutshell we take the value within the csv file, pump it through a function that changes the Fahrenheit to Celsius, then place the new value in the DB.

### Convert Inches

The methodology is the same for as above. Here is the function for converting inches.

double MillimetreValue = 0;

MillimetreValue = InchesValue \* 25.4;

MillimetreValue = Math.Round(MillimetreValue, 2);

return MillimetreValue.ToString();

### Separate Year & Month

This is done by taking the YYYYMM value out of the file, and parsing it into 2 different values, again before putting those new values into the database. It doesn't have its own function as it was not seen as efficient, because it happens every time no matter what. Here are the 2 rows that do the string parse while calling the DB queries.

Weather.Rows.Add("(SELECT ID FROM YEAR WHERE yearname = " + row[2].Substring(0, 4) + ")," +

"(SELECT ID FROM MONTH WHERE monthname = " + row[2].Substring(4, 2) + ")," +

Notice the SUBSTRING() call, this is what splits the date into the 2 database tables. You'll also notice that these are Select statements, and not Inserts. That is because we have already created prior a YEAR and MONTH table, this ROWS.ADD is only for the weather table, which references the YEAR, and MONTH table.

# Live Graphing Requirements

### Series Requirements

You can see in image 3 that I record the locations of where the images are taken. Below I will list some code samples of how I did this. This method is the method called when a picture has been

### Region Requirement

You can see in image 3 that I record the locations of where the images are taken. Below I will list some code samples of how I did this. This method is the method called when a picture has been

### Summary Data Requirement

You can see in image 3 that I record the locations of where the images are taken. Below I will list some code samples of how I did this. This method is the method called when a picture has been

### X/Y Range Requirements

You can see in image 3 that I record the locations of where the images are taken. Below I will list some code samples of how I did this. This method is the method called when a picture has been

### Time Slider Requirement

You can see in image 3 that I record the locations of where the images are taken. Below I will list some code samples of how I did this. This method is the method called when a picture has been

### Printing Requirement

You can see in image 3 that I record the locations of where the images are taken. Below I will list some code samples of how I did this. This method is the method called when a picture has been

# Appendix of DB scripts

### Views

We had to include the 3 views in 3 different files, otherwise we could not get azure to accept them. (Azure requires them to be run in separate batches). Here are the 3 views.

CREATE VIEW getPrecipitationView AS

SELECT State.statecode, Month.monthname AS [month], Year.yearname AS [year], Weather.pcp

FROM Weather

INNER JOIN Year on Weather.YID = Year.id

INNER JOIN Month on Weather.MID = Month.id

INNER JOIN State on Weather.SID = State.id

CREATE VIEW getCoolAndHeatView AS

SELECT State.statecode, Month.monthname AS [month], Year.yearname AS [year], Weather.cdd, Weather.hdd

FROM Weather

INNER JOIN Year on Weather.YID = Year.id

INNER JOIN Month on Weather.MID = Month.id

INNER JOIN State on Weather.SID = State.id

CREATE VIEW getTemperatureView AS

SELECT State.statecode, Month.monthname AS [month], Year.yearname AS [year], Weather.Tmin, Weather.Tmax, Weather.Tavg

FROM Weather

INNER JOIN Year on Weather.YID = Year.id

INNER JOIN Month on Weather.MID = Month.id

INNER JOIN State on Weather.SID = State.id

These views are used to return all data pertaining to a state, for a specific graph type (temperature, cooling/heating days, precipitation)

### Stored Procedures

These stored procedures are what are called when specific UI elements are activated. Again the procedures had to be in different files as Azure could not handle multiple commands in a single batch.

CREATE PROCEDURE getPrecipitationForArea @AreaCode int

AS

SELECT [month], [year], pcp

FROM getPrecipitationView

WHERE statecode = @AreaCode

ORDER BY [year],[month]

CREATE PROCEDURE getCoolAndHeatForArea @AreaCode int

AS

SELECT [month], [year], cdd, hdd

FROM getCoolAndHeatView

WHERE statecode = @AreaCode

ORDER BY [year],[month]

CREATE PROCEDURE getTemperatureForArea @AreaCode int

AS

SELECT [month], [year], Tmin, Tmax, Tavg

FROM getTemperatureView

WHERE statecode = @AreaCode

ORDER BY [year],[month]