http://www.sqlservercentral.com/articles/SQL+Server/67215/
Printed 2009/07/09 01:58PM

Map IP Address to a Geographical Location

By Oscar D Garcia, 2009/06/10

Introduction

On the Internet there is the concept of web site analytics that helps track all the visitors' activities and usage patterns. One of the dimensions to track is the geographical information of the visitors, which can be obtained by using the IP address information that is collected when a user lands on a Website. In this article, I will describe a simple process that enables your reporting system to display the geographical information of the visitors.

Scope

This article does not describe the process needed to capture the IP information from the user. This process is an application level solution that can be implemented on the front end with software like ASP.Net, PHP etc. The scope of this article is limited to the usage of the IP address to discover the geographical data. This data is composed of the Country, Region, City, Postal code and Area code where applicable. Some countries may not have the concept of postal code.

What is an IP Address?

When a user lands on a web page, the web application has the ability to collect information from this user. One of these data elements is the IP address. The IP address has a format of 192.15.10.125, and it is a logical address assigned to a device. This is what identifies your Internet address, and it is composed of segments that identify your geographical location.

What Do I need to Map an IP Address to a Geo Location?

In order to map the IP address to a geographical representation, the system needs the geographical data mapping. This data is provided by several companies. In this case, we are using the GEOLiteCity data, which is free. To obtain this data, visit http://www.maxmind.com/app/geolitecity and download the ZIP file which contains two files, Blocks and Location CSV. The block file maps an IP number range to a location. The location file has the geographical information. Please note that

there are frequent updates to these files, so make sure you read the description of their services.

In order to import this data to your database, one first needs to create the table definitions. We need to create the Geo Location and Block tables. This are the table definitions: (you can also download the scripts)

Import the data using your preferred method. The first line on the CSV file is a copyright statement. The second line has the column headers, so make sure to skip the first line during the import process. I have also included a visitor log table which can be used to track the user's data. This table is very simple, and it does not include all the possible data elements that can be collected.

Solution

Once the data has been imported, you will notice that the IP address in the data does not look anything like 192.15.10.125. The information is actually stored as an IP number. This numeric value is what allows us to do a range comparison. An IP number range is mapped to a particular location. This is what allows us to do the association, but the first step is to figure out how to convert an IP address to an IP number. This is where a user defined function can help us. We first need to convert the IP address to an IP number by using the ConvertIP2Num function below:

This function first splits the IP address into its four segments (delimited by a dot). This is where XML becomes really handy. We just create an XML string and use the parser to do the split for us by selecting the XML nodes. We use the Row_Number() function to create a factor which will help us arrive to the segment weight (i.e. segment: 192 has rownumber:1 and weigh: 4-1 = 3) . We now apply the conversion formula which consists of assigning a zero based weight to each segment (zero starting from the right segment) and multiplying this segment by (256 ^ n) OR POWER (256,n) where n = weight. The final step is to add all the segment results. For example, IP 192.15.10.125 is converted as follows:

The result is the number that can be used to query the geo tables. To do this, you can create a query similar to the one below which returns the geographical information.

One item to notice here is that the conversion from IP address to number would not perform as well when you are trying to query thousand of rows and using the function to do the conversion at run time. A better approach would be to have an additional column on the table the holds the IP address information. This column can store the IP number as well. (See IPNum column on the Visitor_log Table) This way the import job can populate this field, so there is no performance hit

during the query or creation of the report. In addition, by storing both fields (IP Address and Num), you will not have a need to reverse the IP Num to IP Address.

If you do need to reverse an IP Num to IP Address (to validate the process, etc), you can use the ConvertNum2IP function below:

This function just reverses the process by dividing the IP Num by Power(256,n) and subtracting the result from the IP Num. Each segment result is then appended into a string with the corresponding IP Address format.

Conclusion:

With this article, I was able to show a simple process to create your own GEO lookup database and provide a solution to map the IP address to its location information. There are still other items to consider like automating the import process to download the new files, convert the IP address to the IP Num value, integrate this information in a data warehouse, and create reports that will show your marketing team in what regions the customers are located.