Geographical Distance Calculations

For ZIPCodeWorld™ & PostalCodeWorld™ Database

In the ZIPCodeWorld™ and the PostalCodeWorld™ databases, the LATITUDE and the LONGITUDE column are not only being used to pinpoint the location of the city but also it is being used to calculate the geographical distance between two different locations. Distance calculator is used in projects such as stores/dealers locators, freight & delivery schedule estimation and proximity searches.

Introduction

The distance between two points on the earth's surface can be calculated by using its latitude and longitude coordinates. Latitude is the angle above or below the equator in degrees. Meanwhile, Longitude is the angel east or west of the Greenwich meridian. The concept used to find out a distance between two points is similar to calculate a perimeter between two points on a sphere.

Explanation of Terms

These are the standard notations used throughout this paper.

$DISTANCE$	= distance in miles between the first and the second point.
$DISTANCE_{LONG}$	= longitude distance in miles between the first & the second point.
$DISTANCE_{LAT}$	= latitude distance in miles between the first & the second points.
LAT_1	= latitude of the first point in degrees.
$LONG_1$	= longitude of the first point in degrees.
LAT_2	= latitude of the second point in degrees.
$LONG_2$	= longitude of the second point in degrees.

Approximate Formula I

This formula applies the basic of trigonometry function. It estimates 1 latitude degree is 69.1 miles. Since the distance of 1 longitude degree varies to the locations, we use the average of 53 miles per longitude degree.

$$DISTANCE_{LAT} = 69.1 \times (LAT_2 - LAT_1)$$
$$DISTANCE_{LONG} = 53 \times (LONG_2 - LONG_1)$$
$$DISTANCE = \sqrt{(DISTANCE_{LAT})^2 + (DISTANCE_{LONG})^2} \qquad \Lambda \ (1)$$

Approximate Formula II

This formula is more precise than the Approximate Formula I. The distance between longitude degrees is based on the position at the sphere.

$$DISTANCE_{LAT} = 69.1 \times (LAT_2 - LAT_1)$$
$$DISTANCE_{LONG} = 69.1 \times (LONG_2 - LONG_1) \times \cos(LAT_1 / 57.3)$$
$$DISTANCE = \sqrt{(DISTANCE_{LAT})^2 + (DISTANCE_{LONG})^2} \qquad \Lambda\ (2)$$

Great Circle Distance Formula

This formula uses spherical trigonometry formula to calculate exact distance between points precisely.

$$DISTANCE = 69.1 \times (180/\pi) \times \arccos[\sin(LAT_1) \times \sin(LAT_2) + \cos(LAT_1) \times \cos(LAT_2) \times \cos(LONG_2 - LONG_1)] \qquad \Lambda\ (3)$$

ASP (Great Circle Distance Calculation)

```
<%

'::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
':::                                                                      :::
':::  This routine calculates the distance between two points (given the     :::
':::  latitude/longitude of those points). It is being used to calculate     :::
':::  the distance between two ZIP Codes or Postal Codes using our           :::
':::  ZIPCodeWorld(TM) and PostalCodeWorld(TM) products.                     :::
':::                                                                      :::
':::  Definitions:                                                         :::
':::    South latitudes are negative, east longitudes are positive         :::
':::                                                                      :::
':::  Passed to function:                                                  :::
':::    lat1, lon1 = Latitude and Longitude of point 1 (in decimal degrees)  :::
':::    lat2, lon2 = Latitude and Longitude of point 2 (in decimal degrees)  :::
':::    unit = the unit you desire for results                            :::
':::           where: 'M' is statute miles                                :::
':::                  'K' is kilometers (default)                         :::
':::                  'N' is nautical miles                               :::
':::                                                                      :::
':::  United States ZIP Code/ Canadian Postal Code databases with latitude   :::
':::  & longitude are available at http://www.zipcodeworld.com              :::
':::                                                                      :::
':::  For enquiries, please contact sales@zipcodeworld.com                  :::
':::                                                                      :::
':::  Hexa Software Development Center ® All Rights Reserved 2003           :::
':::                                                                      :::
'::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

const pi = 3.14159265358979323846

Function distance(lat1, lon1, lat2, lon2, unit)
  Dim theta, dist
  theta = lon1 - lon2
  dist = sin(deg2rad(lat1)) * sin(deg2rad(lat2)) + cos(deg2rad(lat1)) _
         * cos(deg2rad(lat2)) * cos(deg2rad(theta))
  dist = acos(dist)
  dist = rad2deg(dist)
  distance = dist * 60 * 1.1515
  Select Case ucase(unit)
    Case "K"
      distance = distance * 1.609344
    Case "N"
      distance = distance * 0.8684
  End Select
End Function


'::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
':::  This function get the arccos function from arctan function    :::
'::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Function acos(rad)
  If Abs(rad) <> 1 Then
    acos = pi/2 - Atn(rad / Sqr(1 - rad * rad))
  ElseIf rad = -1 Then
    acos = pi
  End If
End function


'::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
':::  This function converts decimal degrees to radians             :::
```

```
':::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Function deg2rad(Deg)
  deg2rad = cdbl(Deg * pi / 180)
End Function

':::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
':::  This function converts radians to decimal degrees             :::
':::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Function rad2deg(Rad)
  rad2deg = cdbl(Rad * 180 / pi)
End Function

response.write distance(32.9697, -96.80322, 29.46786, -98.53506, "M") & "
Miles<br>"
response.write distance(32.9697, -96.80322, 29.46786, -98.53506, "K") & "
Kilometers<br>"
response.write distance(32.9697, -96.80322, 29.46786, -98.53506, "N") & "
Nautical Miles<br>"

%>
```

PHP (Great Circle Distance Calculation)

```
<?php

/*::                                                                         :*/
/*::  This routine calculates the distance between two points (given the     :*/
/*::  latitude/longitude of those points). It is being used to calculate     :*/
/*::  the distance between two ZIP Codes or Postal Codes using our           :*/
/*::  ZIPCodeWorld(TM) and PostalCodeWorld(TM) products.                     :*/
/*::                                                                         :*/
/*::  Definitions:                                                           :*/
/*::    South latitudes are negative, east longitudes are positive           :*/
/*::                                                                         :*/
/*::  Passed to function:                                                    :*/
/*::    lat1, lon1 = Latitude and Longitude of point 1 (in decimal degrees)  :*/
/*::    lat2, lon2 = Latitude and Longitude of point 2 (in decimal degrees)  :*/
/*::    unit = the unit you desire for results                               :*/
/*::           where: 'M' is statute miles                                   :*/
/*::                  'K' is kilometers (default)                            :*/
/*::                  'N' is nautical miles                                  :*/
/*::  United States ZIP Code/ Canadian Postal Code databases with latitude & :*/
/*::  longitude are available at http://www.zipcodeworld.com                 :*/
/*::                                                                         :*/
/*::  For enquiries, please contact sales@zipcodeworld.com                   :*/
/*::                                                                         :*/
/*::  Hexa Software Development Center ® All Rights Reserved 2003            :*/
/*::                                                                         :*/
/*::                                                                         :*/

function distance($lat1, $lon1, $lat2, $lon2, $unit) {

  $theta = $lon1 - $lon2;
  $dist = sin(deg2rad($lat1)) * sin(deg2rad($lat2)) +
          cos(deg2rad($lat1)) * cos(deg2rad($lat2)) *
          cos(deg2rad($theta));
  $dist = acos($dist);
  $dist = rad2deg($dist);
  $miles = $dist * 60 * 1.1515;
  $unit = strtoupper($unit);

  if ($unit == "K") {
    return ($miles * 1.609344);
  } else if ($unit == "N") {
      return ($miles * 0.8684);
    } else {
        return $miles;
      }
}

echo distance(32.9697, -96.80322, 29.46786, -98.53506, "M") . "
Miles<br>";
echo distance(32.9697, -96.80322, 29.46786, -98.53506, "K") . "
Kilometers<br>";
echo distance(32.9697, -96.80322, 29.46786, -98.53506, "N") . "
Nautical Miles<br>";

?>

deg2rad and rad2deg are built-in functions available since PHP 3.0.4.
```

C/C++ (Great Circle Distance Calculation)

```
/*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*::                                                                          :*/
/*::  This routine calculates the distance between two points (given the     :*/
/*::  latitude/longitude of those points). It is being used to calculate     :*/
/*::  the distance between two ZIP Codes or Postal Codes using our           :*/
/*::  ZIPCodeWorld(TM) and PostalCodeWorld(TM) products.                     :*/
/*::                                                                          :*/
/*::  Definitions:                                                           :*/
/*::    South latitudes are negative, east longitudes are positive           :*/
/*::                                                                          :*/
/*::  Passed to function:                                                    :*/
/*::    lat1, lon1 = Latitude and Longitude of point 1 (in decimal degrees)  :*/
/*::    lat2, lon2 = Latitude and Longitude of point 2 (in decimal degrees)  :*/
/*::    unit = the unit you desire for results                               :*/
/*::           where: 'M' is statute miles                                   :*/
/*::                  'K' is kilometers (default)                            :*/
/*::                  'N' is nautical miles                                  :*/
/*::  United States ZIP Code/ Canadian Postal Code databases with latitude & :*/
/*::  longitude are available at http://www.zipcodeworld.com                 :*/
/*::                                                                          :*/
/*::  For enquiries, please contact sales@zipcodeworld.com                   :*/
/*::                                                                          :*/
/*::  Hexa Software Development Center ® All Rights Reserved 2003            :*/
/*::                                                                          :*/
/*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/

#include <math.h>

#define pi 3.14159265358979323846

double distance(double lat1, double lon1, double lat2, double lon2,
char unit) {
  double theta, dist;
  theta = lon1 - lon2;
  dist = sin(deg2rad(lat1)) * sin(deg2rad(lat2)) + cos(deg2rad(lat1))
         * cos(deg2rad(lat2)) * cos(deg2rad(theta));
  dist = acos(dist);
  dist = rad2deg(dist);
  dist = dist * 60 * 1.1515;
  switch(unit) {
    case 'M':
      break;
    case 'K':
      dist = dist * 1.609344;
      break;
    case 'N':
      dist = dist * 0.8684;
      break;
  }
  return (dist);
}

/*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*::  This function converts decimal degrees to radians             :*/
/*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
double deg2rad(double deg) {
  return (deg * pi / 180);
}
```

```
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*::  This function converts radians to decimal degrees                      :*/
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
double rad2deg(double rad) {
  return (rad * 180 / pi);
}
```

Perl (Great Circle Distance Calculation)

```
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
#:::                                                                      :::
#:::  This routine calculates the distance between two points (given the     :::
#:::  latitude/longitude of those points). It is being used to calculate     :::
#:::  the distance between two ZIP Codes or Postal Codes using our           :::
#:::  ZIPCodeWorld(TM) and PostalCodeWorld(TM) products.                     :::
#:::                                                                      :::
#:::  Definitions:                                                         :::
#:::    South latitudes are negative, east longitudes are positive         :::
#:::                                                                      :::
#:::  Passed to function:                                                  :::
#:::    lat1, lon1 = Latitude and Longitude of point 1 (in decimal degrees)  :::
#:::    lat2, lon2 = Latitude and Longitude of point 2 (in decimal degrees)  :::
#:::    unit = the unit you desire for results                            :::
#:::           where: 'M' is statute miles                                :::
#:::                  'K' is kilometers (default)                         :::
#:::                  'N' is nautical miles                               :::
#:::                                                                      :::
#:::  United States ZIP Code/ Canadian Postal Code databases with latitude   :::
#:::  & longitude are available at http://www.zipcodeworld.com              :::
#:::                                                                      :::
#:::  For enquiries, please contact sales@zipcodeworld.com                  :::
#:::                                                                      :::
#:::  Hexa Software Development Center ® All Rights Reserved 2003           :::
#:::                                                                      :::
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

$pi = atan2(1,1) * 4;

sub distance {
  my ($lat1, $lon1, $lat2, $lon2, $unit) = @_;
  my $theta = $lon1 - $lon2;
  my $dist = sin(deg2rad($lat1)) * sin(deg2rad($lat2)) +
cos(deg2rad($lat1)) * cos(deg2rad($lat2)) * cos(deg2rad($theta));
  $dist  = acos($dist);
  $dist = rad2deg($dist);
  $dist = $dist * 60 * 1.1515;
  if ($unit eq "K") {
    $dist = $dist * 1.609344;
  } elsif ($unit eq "N") {
    $dist = $dist * 0.8684;
    }
  return ($dist);
}

#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
#:::  This function get the arccos function using arctan function   :::
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
sub acos {
      my ($rad) = @_;
      my $ret = atan2(sqrt(1 - $rad**2), $rad);
      return $ret;
}

#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
#:::  This function converts decimal degrees to radians             :::
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
sub deg2rad {
```

```
      my ($deg) = @_;
      return ($deg * $pi / 180);
}

#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
#:::  This function converts radians to decimal degrees             :::
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
sub rad2deg {
      my ($rad) = @_;
      return ($rad * 180 / $pi);
}

print distance(32.9697, -96.80322, 29.46786, -98.53506, "M") . "
Miles\n";
print distance(32.9697, -96.80322, 29.46786, -98.53506, "K") . "
Kilometers\n";
print distance(32.9697, -96.80322, 29.46786, -98.53506, "N") . "
Nautical Miles\n";
```

Visual Basic (Great Circle Distance Calculation)

```
'::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
'::: :::
'::: This routine calculates the distance between two points (given the ::: 
'::: latitude/longitude of those points). It is being used to calculate ::: 
'::: the distance between two ZIP Codes or Postal Codes using our ::: 
'::: ZIPCodeWorld(TM) and PostalCodeWorld(TM) products. ::: 
'::: :::
'::: Definitions: :::
'::: South latitudes are negative, east longitudes are positive ::: 
'::: :::
'::: Passed to function: :::
'::: lat1, lon1 = Latitude and Longitude of point 1 (in decimal degrees) ::: 
'::: lat2, lon2 = Latitude and Longitude of point 2 (in decimal degrees) ::: 
'::: unit = the unit you desire for results ::: 
'::: where: 'M' is statute miles ::: 
'::: 'K' is kilometers (default) ::: 
'::: 'N' is nautical miles ::: 
'::: :::
'::: United States ZIP Code/ Canadian Postal Code databases with latitude ::: 
'::: & longitude are available at http://www.zipcodeworld.com ::: 
'::: :::
'::: For enquiries, please contact sales@zipcodeworld.com ::: 
'::: :::
'::: Hexa Software Development Center ® All Rights Reserved 2003 ::: 
'::: :::
'::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

const pi = 3.14159265358979323846

Function distance(lat1, lon1, lat2, lon2, unit)
  Dim theta, dist
  theta = lon1 - lon2
  dist = sin(deg2rad(lat1)) * sin(deg2rad(lat2)) + cos(deg2rad(lat1)) *
cos(deg2rad(lat2)) * cos(deg2rad(theta))
  response.write "dist = " & dist & "<br>"
  dist = acos(dist)
  dist = rad2deg(dist)
  response.write "dist = " & dist & "<br>"
  distance = dist * 60 * 1.1515
  Select Case ucase(unit)
    Case "K"
      distance = distance * 1.609344
    Case "N"
      distance = distance * 0.8684
  End Select
End Function


'::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
'::: This function get the arccos function using arctan function ::: 
'::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Function acos(rad)
  If Abs(rad) <> 1 Then
    acos = pi/2 - Atn(rad / Sqr(1 - rad * rad))
  ElseIf rad = -1 Then
    acos = pi
  End If
End function
```

```
':::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
':::  This function converts decimal degrees to radians             :::
':::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Function deg2rad(Deg)
      deg2rad = cdbl(Deg * pi / 180)
End Function

':::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
':::  This function converts radians to decimal degrees             :::
':::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
Function rad2deg(Rad)
      rad2deg = cdbl(Rad * 180 / pi)
End Function

msgbox(distance(32.9697, -96.80322, 29.46786, -98.53506, "M") & "
Miles<br>")
msgbox(distance(32.9697, -96.80322, 29.46786, -98.53506, "K") & "
Kilometers<br>")
msgbox(distance(32.9697, -96.80322, 29.46786, -98.53506, "N") & "
Nautical Miles<br>")
```

Java (Great Circle Distance Calculation)

```
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*::                                                                        :*/
/*::  This routine calculates the distance between two points (given the     :*/
/*::  latitude/longitude of those points). It is being used to calculate     :*/
/*::  the distance between two ZIP Codes or Postal Codes using our           :*/
/*::  ZIPCodeWorld(TM) and PostalCodeWorld(TM) products.                     :*/
/*::                                                                        :*/
/*::  Definitions:                                                          :*/
/*::    South latitudes are negative, east longitudes are positive           :*/
/*::                                                                        :*/
/*::  Passed to function:                                                   :*/
/*::    lat1, lon1 = Latitude and Longitude of point 1 (in decimal degrees)  :*/
/*::    lat2, lon2 = Latitude and Longitude of point 2 (in decimal degrees)  :*/
/*::    unit = the unit you desire for results                               :*/
/*::           where: 'M' is statute miles                                   :*/
/*::                  'K' is kilometers (default)                            :*/
/*::                  'N' is nautical miles                                  :*/
/*::  United States ZIP Code/ Canadian Postal Code databases with latitude & :*/
/*::  longitude are available at http://www.zipcodeworld.com                 :*/
/*::                                                                        :*/
/*::  For enquiries, please contact sales@zipcodeworld.com                   :*/
/*::                                                                        :*/
/*::  Hexa Software Development Center ® All Rights Reserved 2003            :*/
/*::                                                                        :*/
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/

private double distance(double lat1, double lon1, double lat2, double
lon2, char unit) {
  double theta = lon1 - lon2;
  double dist = Math.sin(deg2rad(lat1)) * Math.sin(deg2rad(lat2)) +
                Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
                Math.cos(deg2rad(theta));
  dist = Math.acos(dist);
  dist = rad2deg(dist);
  dist = dist * 60 * 1.1515;
  if (unit == "K") {
    dist = dist * 1.609344;
  } else if (unit == "N") {
      dist = dist * 0.8684;
    }
  return (dist);
}

/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*::  This function converts decimal degrees to radians            :*/
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
private double deg2rad(double deg) {
  return (deg * Math.PI / 180.0);
}

/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
/*::  This function converts radians to decimal degrees            :*/
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
private double rad2deg(double rad) {
  return (deg / Math.PI * 180.0);
}
```

```
system.println(distance(32.9697, -96.80322, 29.46786, -98.53506, "M") +
" Miles\n");
system.println(distance(32.9697, -96.80322, 29.46786, -98.53506, "K") +
" Kilometers\n");
system.println(distance(32.9697, -96.80322, 29.46786, -98.53506, "N") +
" Nautical Miles\n");
```