Ordered Set Data Manipulation Language (Osdm): An Overview

Yong Feng (yf21@columbia.edu)

June 1, 2006

Introduction

Osdm language is a simple interpret language designed to process and manipulate sets of data associated with time or string as index, such as stock quote or volumes at each point of time during a time period. Osdm can be used to operate on two values with timeline data, or do analysis or aggregate operations on a timeline data, such as calculating the sum of the data values within a time period. Current programming languages do not support timeline data as a first-class object. You have to write complicate function or class in order to represent the data and to do those operations.

Osdm

Osdm is a simple, high-level, interpreted, architecture neutral, and portable language specializing on manipulating ordered set or timeline data in particular.

Simple

Osdm is very simple and easy to use. Because ordered set data becomes the first-class object of the language, Osdm can easily represent timeline data or ordered set data in general and make it very simple to operate on those kinds of data. Some analysis and aggregate functions are included as part of the language to provide common analysis functionality, such as aggregating data from much longer time period into one-week or one-month data. Therefore, users just need to use the functions provided by Osdm to specify what they want instead of using user-defined data structure to hold the data and writing complicate algorithm of implementing it.

High Level

The OSet (Ordered Set) data type of Osdm can directly represent a set of data indexed and ordered by time or string. Basic operations are provided to directly operate on values with OSet data type. Osdm also provides some very high-level functions to analyze and aggregate data values with OSet data type.

Interpreted

Osdm has only a few arithmetic operations and very high-level functions available. The programs written in Osdm should be relative small, and those programs most probably will run only once a while. So it makes sense to implement it as an interpreted language. Therefore, user can input the data, execute some operation, and examine the immediate result step by step.

Architecture Neutral

Osdm is an interpreted language implemented using java. So the code written in Osdm will work on any platforms where java is supported.

Portable

Osdm is portable because it is architecture neutral, and there is no implementation dependent aspect in the language specification.

Example

Here is a program written in Osdm to show some basic operations available in the language:

```
Set DATE_FORMAT “YYYY/MM/DD”

OSet x = ((“2001/03/05”, 200), (“2001/03/09”, 200));
OSet y = ((“2001/03/09”, 200), (“2001/04/5”, 200));

OSet z = x + y;
double totalValue = sum(z);
double maxValue = max(z);

OSet u = aggregate (z, “????/??/DD”, “SUM”);
OSet v = aggregate(z, “YYYY/MM/??”, “SUM”);
```

In this example, date format is set at the beginning so that the date strings for the index that matches the format specified can be parsed as dates. If date format is not presented, the index appears in the Oset data would be treated as strings. Next two lines define two OSet variables as timeline values. One of the basic arithmetic operations is used in the next line to calculate the addition of those two values. The result of z is ((“2001/03/05”, 200), (“2001/03/09”, 400), (“2001/04/05”, 200)). The next two lines calculate the sum and the maximum value of the values across all the data points. The next line merges the data points that have the same day within a month into one data point using the operation specified by the third parameter. In this case, u = ((“5”, 400), (“9”, 400)). The last line aggregates the data points in a same month of a same year into one data point using the operation specified by the third parameter. So the value of v will be ((“2001/03”, 600), (“2001/04”, 200)). The symbol “?” is used as a mask so that the characters of the index at the same positions as symbol “?” are removed from the index.

Note, values stored in the data of OSet data type will be automatically sorted using its index.

Possible extensions

Since this language is designed to deal with lots of data points, it is very desirable to be able to accept input from a file and save the result into an output file. It would be nice to display the result using a chart, or save the result as a XML file so that it can be further processed or to be displayed by a third party tool that accepts XML file. Also it would be nice if the code could accept XML file as input. Currently the languages contains only some basically operations and some basic statements for flow control. More operations and control flow statements might be

needed to handle complicated applications. It might be interesting to extend this language to handle multi-dimension data such as data that contains time, country, and populations.

Summary

The Osdm language provides a native data type and functions to deal with two dimension data such as timeline data. Its simplicity, powerful functions, and interpreted way of executing commands will hopefully make it easy to manipulate two dimension data.