European Study Group with Industry'95 (ESGI'95)
September 23 – 27, 2013, Sofia, Bulgaria

Compressed representation of a partially defined integer function over multiple arguments

Adiss Lab Lts.

Zhivko Angelov

Problem description

In OLAP (OnLine Analitical Processing) data are analysed in an n-dimensional cube. The cube may be represented as a partially defined function over n arguments

$$\boldsymbol{f}(x_1, x_2, \ldots, x_n), \text{ where } x_i \in [1 \div N_i] \text{ for } i \in [1 \div n].$$

In the general case the values of ***f*** are real numbers but quite often they are whole numbers (integers). Most commercial applications store values as fixed precision numbers, so if we assume that ***f*** is an integer function this will encompass a significant part of the applications.

Let $(*, *, \ldots, \boldsymbol{a_i}, *, \ldots, *)$ denote an $(n - 1)$ – dimensional cube. The most frequently solved problem in OLAP is the computation of the sum of values ***F*** of a given sub-cube of the n-dimensional cube. In the case of the $(n - 1)$ – dimensional cube the values ***F*** are:

$$\boldsymbol{F}(*, *, \ldots, \boldsymbol{a_i}, *, \ldots, *) = \\ = \Sigma_{j1}^{N1} \Sigma_{j2}^{N2} \ldots \Sigma_{ji-1}^{Ni-1} \Sigma_{ji+2}^{Ni+1} \ldots \Sigma_{jn}^{Nn} \boldsymbol{f}(j_1, j_2, \ldots, j_{i-1}, \boldsymbol{a_i}, j_{i+1}, \ldots, j_n).$$

Other frequently computed functions beside summation are arithmetic average, minimum and maximum.

Question 1: Considering that often the function ***f*** is not defined everywhere is there a known way of representing ***f*** or the points in which it is defined in a more compact manner than the trivial $a_1, a_2, \ldots, a_n, \boldsymbol{f}(a_1, a_2, \ldots, a_n)$? The goal is to reduce the time necessary for moving or storing ***f*** and using part of the time gained for computations for restoring the original ***f*** **.**

Question 2: Due to natural limitations the visualization of a *n*-dimensional cube is reduced to the representation of a rectangle in a particular plain.

One possible approach is to compute the values of all sub-cubes

$\boldsymbol{F}(*, x_2, x_3, \ldots, x_n)$,

$\boldsymbol{F}(x_1, *, x_3, \ldots, x_n)$,

...

$\boldsymbol{F}(*, *, x_3, \ldots, x_n)$,

...

$\boldsymbol{F}(*, *, \ldots, *)$

and to show the values of cubes located in the defined rectangle.

In this case the transition from one rectangle to another in the the considered plain and the transition to a plain another dimension will be accomplished in approximately one and the same short time interval (includes only the reading of several hundred values). This is in contrast to the time needed for the computation of the values and the memory necessary for their storage which may be significant.

In the visualization process only a subset of set of all sub-cubes is used, only the values of theses cubes are necessary. How can an algorithm be constructed that makes a balance between preliminarily computed values and the computation of values that are not stored but fall into the defined visualization area? If there exists a compressed representation of data (Question 1) is it possible to base the algorithms on this representation? Considering the fact that in many applications the function ***f*** is of integer type do more efficient algorithms exist under the restriction of ***f*** being integer?

An important fact that has to be taken into account is that almost always the values of the whole cube take part in the visualization process as well as the values of several sub-cubes (as illustrated below). This means that these values have to be computed before the visualization start.

Arrivals

Facts

Arr Date	Country	Plovdiv	Sofia	Total
		Airport		
05.09.2013			1	1
06.09.2013		1	17	18
07.09.2013		1	10	11
08.09.2013		2	46	48
09.09.2013	Belgium		1	1
	England		1	1
	France		1	1
	Germany		1	1
	Luxembourg		1	1
	Sfax- Tunisia		1	1
	South Africa		1	1
	Spain		1	1
	Total		8	8
10.09.2013			1	1
11.09.2013			4	4
12.09.2013			2	2
Total		4	89	93