Adaptation of the Algorithms for Execution of the Aggregate Operations for the Purposes of the Virtual Laboratory on “Databases”

Elitsa Arsova, Silyan Arsov, Angel Smrikarov

Abstract: *Studying a course “Databases” involves not only understanding of complex notions, methods and operations, that are specific for this field, but also common notions and operations from the field of mathematics and informatics like aggregate operations. In this paper, we define a set of aggregate operations, namely summing, counting, averaging, finding minimal and maximal element, and sorting, and adapt algorithms for their execution to work with databases' records for the purposes of Virtual Laboratory on “Databases” course. In parallel, we report and explain the evaluation of the algorithms’ complexity.*

Key words: *Virtual Laboratory, Relational Operations; Algorithms; Algorithms’ Complexity; Databases.*

INTRODUCTION

The database queries could consist of three types of operations: operations on relations, aggregate and actualizing operations. To find a record that fulfilled a certain set of criteria, operations on relations are usually combining with aggregate functions. An aggregate query is a query which requests information concerning a group of records in the aggregate. Students are familiar with the aggregate operation from the field of mathematics and informatics, but it is necessary the functions to be adapted for working with database and also to visualize their execution as algorithms. Our idea is to design and implement these algorithms in an educational computer-tool that will be a part of Virtual Laboratory on course “Database”.

During the last years the Virtual Learning Environments (VLE) rapidly influence the educational process as they are considered to substitute the traditional methods of learning. At present, there is a trend to interchange and combine full-time with distance form of study [2]. This tendency seems to be a substantial component of the educational policy of many leading universities. Some of the Bulgarian universities that *implemented* e-learning systems for *enhanced learning*, are the University of Ruse – the eLSe system [1], the University of Sofia – the ARCADE system [7], the University of Plovdiv – PeU [8], Defense and Staff College “G.S. Racovski”, Sofia – the FLAME system [3] and the Technical University-Sofia[6]. Two Virtual Laboratories (VL) are implemented in the department of “Computer systems and technologies” in the University of Ruse. Interactive models of a different CPU blocks are created within the framework of the VL on “Computer Organization” [9]. Tools for logical schemes design and test system are developed in “Analysis and synthesis of logical schemes”, named Digital Logic Design Virtual Laboratory (DLDVL) [4].

ALGORITHMS FOR EXECUTION OF THE AGGREGATE OPERATIONS

Summing ($SUM(A_i)$). The algorithm is realized by adding an attribute A_i from the loaded in the operating memory tuple to the sum $SUM(A_i)$ of the values of the attribute A_i (fig. 1).

Let have a sequence of attribute values $A: a_1, a_2, ..., a_n$. The *summing* can be executed by consecutively accumulation, e.g. with the execution of the following operations:

$S=0,\ S:=S+A_i,\ при\ \ i=1,2,....,N.$

The second operator should be executed N times due to obtain the sum of all N values and the summing algorithm complexity is expressed by the expression (1):

$$T_{executions} = O(N), \tag{1}$$

where $T_{executions}$ is the number of the execution of "*read* " and *"calculate"* operations.

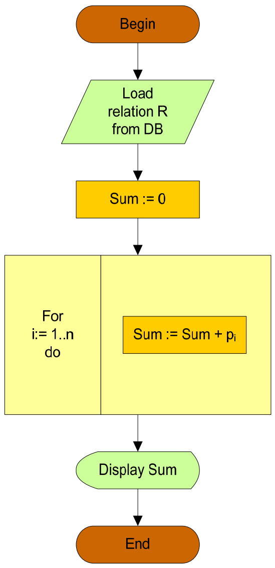

Figure 1. An algorithm for execution of the aggregate operation summing

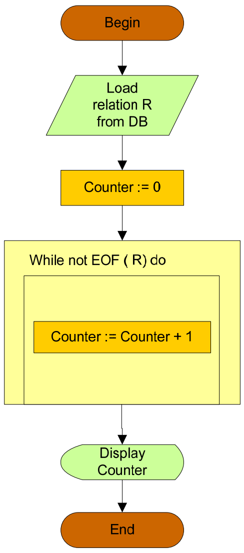

Figure 2. An algorithm for execution of the aggregate operation counting

Counting $COUNT\ (A_i)$. The algorithm is realized by increasing the value of a counter with one for every attribute that belongs to the reviewing tuple (fig. 2).

To obtain the number of the values of a given attribute $A: a_1, a_2, ..., a_n$, it is necessary to save the result from operation $k = k+1$ in a variable k, which at the beginning is zero. The operation summation is executed N times. The expression (2) presents the *counting* algorithm complexity:

$$T_{executions} = O(N), \tag{2}$$

where $T_{executions}$ is the number of the execution of "*read*" operation.

Averaging $Avg(A)$**.** The realization of this algorithm requires presence of two accumulating variables – the first one is used for a counter of a number of tuples, and the second one is used for saving the sum of the values of the attribute A_i from these tuples. All of the values are calculated by using the described above algorithms for counting $COUNT\ (A_i)$ and sum $SUM(A_i)$ (fig. 5). An average value of the attribute $A: a_1, a_2, ..., a_n$ is equal to the quotient of sum $SUM(A_i)$. and count $COUNT(A)$.

The sum of the attribute values and their count could be find by N times consecutive execution of the following operations:

$$S := S + A_i,\ \ npu\ \ i = 1, 2, ..., N;\ k := k+1.$$

After these operations, the operation division of the sum and count is executed.

The estimation of the algorithm complexity for execution of the operation *average* is the following:

$$T_{executions} = O(N) + 1, \tag{3}$$

where $T_{executions}$ is the number of the execution of "*read* " and *"calculate"* operations.

Finding minimal and maximal element of an attribute - $Min(A_i)$ **and** $Max(A_i)$. The algorithm of finding minimal element of a given attribute: the values of an attribute from the loaded tuple are compared with last saved minimal value. If the current value of A_i is smaller than the minimal, then the first one is saved as a minimal value. The algorithm for finding maximal element is similar.

The expression (4) presents the algorithm complexity for finding minimal and maximal element of an attribute:

$$T_{executions} = O(N), \tag{4}$$

where $T_{executions}$ is the number of the execution of "*read*" operation.

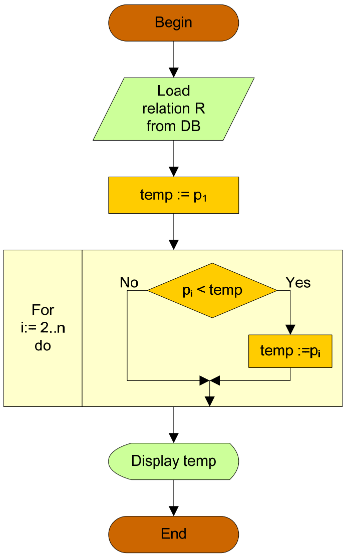

Figure 3. An algorithm for execution of the aggregate operation minimum

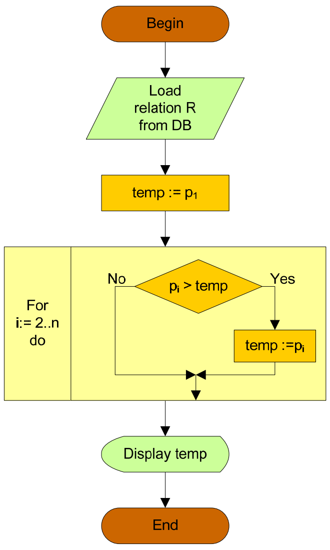

Figure 4. An algorithm for execution of the aggregate operation maximum

Sorting $SORT(A)$**.** The realization of the algorithm is based on the traditional method for sorting by swapping (bubble sort).

The values (N) of a given attribute A could be ordered by comparing consecutively two neighbour values and swapping them if the left value is bigger then the right value (in increasing order). The values of the attributes from the databases are loaded in an array and the result from the sorting is in the same array. During the first step the biggest value (in position N) emerges. During the second step the second biggest element emerges in position $N-1$ and so on (fig. 6).

The estimations of the algorithm complexity for execution of the operation *average* is the following:

In the best case, when the input data is ordered in advance, the number of comparisons is $N-1$ and there is only one step. The next expression (5) presents the estimation of algorithm complexity:

$$T_{\min}(N) = N-1; \tag{5}$$

In the worst case, when the input data is in reverse order, the number of comparisons is $N-1$ in the first step, $N-2$ in the second step and so on till the last comparison which is 1, e.g:

$$T_{\max}(N) = (N-1)+(N-2)+\ldots+1 = N(N-1)/2; \tag{6}$$

The average estimation of the algorithm complexity is:

$$T \le 1/2(N^2 - N\ln N) + O(N), \tag{7}$$

where T is the number of the comparison operations.

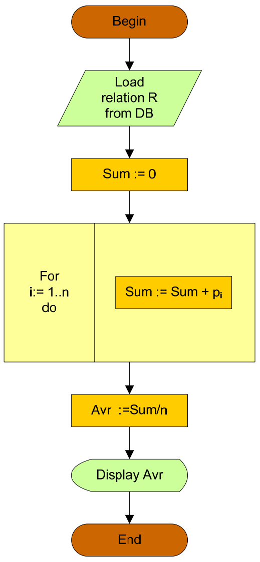

Figure 5. An algorithm for execution of the aggregate operation average

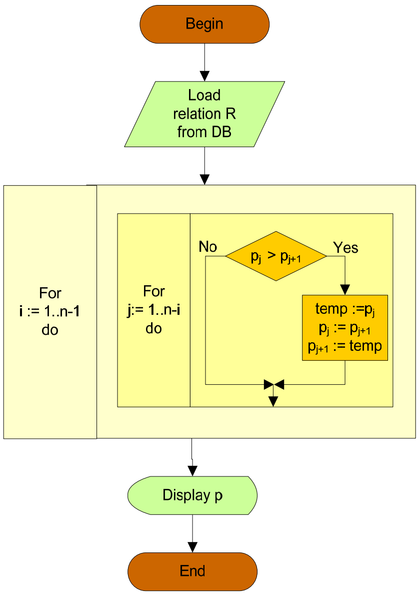

Figure 6. Algorithm for execution of the aggregate operation sorting

IMPLEMENTATION OF THE ALGORITMS

The adapted and developed algorithms are used for the implementation of a web-based programming system. The idea is to demonstrate the execution of the operations on relations and aggregate operations. The demonstration presents the result from the execution of the chosen operations with the indicated operands. The queries, which are built in a graphical environment, perform only one operation. Thereby every operation is defined and implement for itself and the students easily can understand theoretical notions and definitions. Furthermore, the system has the possibility to display learning materials, as definitions and algorithms, in additional windows. The implementation will be proposed in the future works.

CONCLUSIONS AND FUTURE WORK

The adaptation of the algorithms for executing the aggregate operations summing, counting, averaging, finding minimal and maximal element, and sorting is used for implementation of the functions into the VL on "Databases" course. The algorithms present the aggregate operations as independent functions which work with records from databases, not only combined with operations on relations or operations for actualization.

REFERENCES

[1] Hristov, T., S. Smrikarova, A. Vasileva, A. Smrikarov. An Approach to Development of an e-Learning Software Platform. Proceedings of CompSysTech'02, Sofia, 2002.

[2] Ivanov, S., J. Peneva. Distance Learning Courses in Computer Science – Initiation and Design. Proceedings of CompSysTech'2007, Ruse, 14-15 June, 2007.

[3] FLAME. Web-Based System for Distance Learning. TU Sofia, 2003.

[4] Mateev, V., S. Todorova, A. Smrikarov. Test Construction and Distribution in Digital Logic Design Virtual Laboratory. Istanbul, Turkey, 27-28 August, 2007.

[5] Molina, H., J. Ullman, and J. Widom. Database Systems: The Complete Book. USA, Prentice Hall, New Jersey, 2002.

[6] Shoikova, E., V. Denishev, I. Pandiev. Development of an eLearning Architecture Based on Microsoft Class Server. International Conference "New Technologies in Higher Education and Training", Sofia, 6-17 May, 2003.

[7] Stefanov, K., S. Stoyanov, R. Nikolov. Design Issues of a Distance Learning Course on Business on the Internet. JCAL (Journal of Computer Assisted Learning), Volume 14, No 2, 1998.

[8] Totkov, G., R. Doneva. Computerized environment for integrated maintenance of distance education course modules. Proceedings of the 1998 EDEN Conference, vol. 2, Italy, 1998.

[9] Vasileva, A., A. Smrikarov, T.Hristov. A Conceptual Model of a Virtual Laboratory on "Computer Organization". Proceedings of CompSysTech'2002, Sofia, 20-21 June, 2002.

ABOUT THE AUTHORS

Elitsa Arsova, PhD Student, Dept. of Computer Systems and Technologies, University of Ruse, Phone: +359 (0)82 888 827, E-mail: EArsova@ecs.ru.acad.bg.

Assist. Prof. Silyan Arsov, PhD, Dept. of Computer Systems and Technologies, University of Ruse, Phone: +359 (0)82 888 276, E-mail: SArsov@ecs.ru.acad.bg.

Assoc. Prof. Angel Smrikarov, PhD, Dept. of Computer Systems and Technologies, University of Ruse, Phone: +359 (0)82 888 249, E-mail: ASmrikarov@ecs.ru.acad.bg.

Докладът е рецензиран.