Speeding Data Access using Arrays

Pradeep Chand
Shroff S.R. Rotary Institute of Chemical Technology (SRICT)

Saumitra S.Wagh
Shroff S.R. Rotary Institute of Chemical Technology (SRICT)

Kishan Gupta
Shroff S.R. Rotary Institute of Chemical Technology (SRICT)

Shrikant J.Wagh
Shroff S.R. Rotary Institute of Chemical Technology (SRICT)

Abstract - Online attendance system is a very useful tool mainly for academic institutions and organizations maintaining huge databases on regular or periodic time intervals. This essentially needs optimal utilization of memory (and therefore time with enhanced speed) and smart management of storage. In this article, we present the array as an alternative way of storage instead of the SQL database. While our approach aids the client programmer in reasoning about the performance of his program in terms of the source code, it also makes the intermediate code easier to transform at compile-time, resulting in faster compilation and more reliable runtimes. We demonstrate how our new approach improves both, the clarity and performance of several end users.

Keywords : SQL

I. INTRODUCTION

Academic institutions, industries, and Government/Non-Government offices need to maintain huge attendance records of their students and/or employees for various obvious reasons. One of the major requirements of the software programmers used for such application is SQL Database[1] However, using SQL Database has difficulties [2] such as, interfacing an SQL database is more complex than adding a few lines of code. It makes system slow and we have to search data again and again from the SQL database.

In the systems using SQL databases, a stepwise method is followed - the attendance of student is marked after the LECTURE/TUTORIAL/LABORATORY (L/T/P) hour has been engaged by the faculty member, the mobile number of all parents are fetched from SQL database from the application, and a pre-designed stored SMS about the absente student is sent to the mobile number of the concerned parent. In such applications, when a faculty member marks a student absent for L/T/P, an appropriate mobile number of the parent of the student is fetched from the SQL database. Since absent students might be more than 1, every time the mobile number is searched from the SQL database and that mobile number from the SQL database is required to be linked with the SMS API [3]. This process requires a two time usage of SQL database, which increases the time of application.
All or some of these difficulties can be obviated by using arrays as storage [4]. We have built an online attendance application based on array storage for our institute to be used daily by faculty members as a routine practice.
We have reduced the above multi-step process to a one step process using an array for connecting with/to SMS API. Also, our application can handle more than one absentee in only one step, so that the message of absenteeism is sent "in one go" to all concerned parents.

1. Traditional Storage Method

Traditionally, the data is stored in SQL database or alternative databases and during programming the data is accessed. For printing or website display of data, the data must be accessed from the database; this requires connectivity with the SQL database. Every time the data is accessed, it is searched in the database which consumes time and memory proportional to size of the data to be accessed.

METHOD OF ARRAY, used in the present work.

- **Array** data structure, is an arrangement of items at equally spaced addresses in computer memory.
- **Array** data type, is used in a programming language to specify a variable that can be indexed.

Array is a collection mainly using similar data types that are stored into a common variable, forming (at least conceptually that may even be replicated into the memory hardware) a linear data structure of an array.
The alternate method used in the present work is the array storage which is the temporary storage that stores the data till the scope of an array variable exists. But in our case the scope of variable will remain until our requirement is met. We have used the ASP.net as front end in our programming using c# for programming and SQL as the back end. We will access the SQL database once and store in the array for the rest of programming. Now all the application gets the data from the array. Since array is a variable and if we create different array variables for all the events, all the variables will consume memory space and will slow down the program execution. To do away with this, we have created a single variable for array for different events, so that the declaration of variable does not take much space in the memory. All other events share that variable. But the multiple access of data simultaneously from the array may create a Race Condition [5]. The present work has dealt with this problem and successfully sorted it out. Our program can effectively solve this problem during the online process of execution.

II. REMEDIATION OF PROGRAM EXECUTION PROBLEMS

We encountered mainly two problems in the execution of our online program.

THE PROBLEM OF Shared Data [6]

Shared Data: Problem of sharing data by Multiple Tasks and Routines:
The shared data problem can be explained as follows:
When several functions (ISR or Task) share a variable and at that instant the value of that variable operates and during its operation, only a part of the operation is completed and a part remains incomplete at that moment, that there is an another task request the variable the value of the variable may differ from the one expected if the earlier operation had been completed.

THE PROBLEM OF Race Condition

A race condition is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly. In computer memory or storage, a race condition may occur if commands to read and write a large amount of data are received at almost the same instant, and the machine attempts to overwrite some or all of the old data while that old data is still being read. The result may be one or more of the following: a computer crash, an "illegal operation," notification and shutdown of the program, errors reading the old data or errors writing the new data. A race condition can also occur if instructions are processed in the incorrect order.

III. SOLUTION OF RACE CONDITION AND SHARED DATA PROBLEM

We have created the session for all the applications running under our website and all the application has particular time out and so that no such application has same timings to access the array variable at that particular time. The solution for shared data problem is that creating atomic instructions [7] that will complete its execution before it can be requested by other event. This part is called the critical section [8].
In our program, we have set the connection with the SQL database and items which are checked are searched from the database and, are then stored in Arrays from where we can perform different operations. Here, we have made different Data Set for different operations and the array variable remains active for the particular Data Set.

Therefore, the Race Condition practically does not arise. The program# has been successfully implemented and executed using different test conditions

Data Set

As previously described, this project assumes that data stays resident for a particular period of time and thus neglects the up-front cost of moving data to the SQL. Based on the read-only nature of the SQL queries in this project and the characteristics of the Array programming model, data is stored on the array in row-column form. SQL stores its data in different data base .When A faculty has to mark the attendance (ABSENT) then he/she will mark the enrollment no. (ID NUMBER) of the students, USE THE SELECT command to select the mobile number of the parents of students from the currently open database and will be stored in the array, and use API link of SMS pack to send the message on selected mobile number which is stored in the array .

Data Size Results (Observations)

As data size increases, fetching data every time from SQL database reduces the speed of operation. But storing data in an array obviates the problem of the increase in data.

Our application does not slow down or crash, if the number of users increase. We have checked the program with 80 faculty members simultaneously accessing attendance data of 1200 students. It is expected to take even more than these numbers at a time.

Future Scope and Applications

This program can be used for different applications, such as:

1. On-line Examination.
2. Railway reservation system
3. Sending Bulk SMS / Queries
4. To improve online transaction speed from online shopping sites.
5. To analyze huge scientific experimental and/or model-generated data and solve equations using such data for comparison, evaluation, and validation purposes

Thus future scope of this application, when we have to deal with bulk data using SQL database, is large enough to accommodate the desired output.

IV. CONCLUSION

The present software serves the end user, who is not a computer professional, to work more effectively with respect to the speed, efficacy, and ease of operation. The software further in principle satisfies the programmer providing a great intellectual challenge to his creativity and command on the language.

V. ACKNOWLEDGEMENTS

We thank SRICT Management for providing us the necessary infrastructure for working and development of the software and Mr. Kiran Patel for the support and help.

REFERENCES

[1] Bakkum, P. and K. Skadron. *Accelerating SQL database operations on a GPU with CUDA.* in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*. 2010. ACM.

[2] Greene, S.L., L.M. Gomez, and S.J. Devlin. *A cognitive analysis of database query production*. in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. 1986. SAGE Publications.

[3] Hassinen, M. and S. Markovski, *Secure SMS messaging using Quasigroup encryption and Java SMS API.* SPLST, 2003. **3**: p. 187.

[4] Busser, R.W., *Recovering data from arrays of storage devices after certain failures*. 2006, Google Patents.

[5] Bishop, M. and M. Dilger, *Checking for race conditions in file accesses.* Computing systems, 1996. **2**(2): p. 131-152.

[6] Bal, H.E. and A.S. Tanenbaum, *Distributed programming with shared data.* Computer Languages, 1991. **16**(2): p. 129-146.

[7] Hovemeyer, D., W. Pugh, and J. Spacco, *Atomic instructions in java*, in *ECOOP 2002—Object-Oriented Programming*. 2002, Springer. p. 133-154.

[8] Dijkstra, E.W., *Solution of a problem in concurrent programming control*, in *Pioneers and Their Contributions to Software Engineering*. 2001, Springer. p. 289-294.