Global Journal of researches in engineering
Numerical Methods
Volume 11 Issue 7 Version 1.0 December 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 2249-4596 & Print ISSN: 0975-5861

# A New Approach for Calculating Average Value (Including Null) Without Aggregate Function

By Mridul Kanti Das, Goutam Biswas, Md. Masudur Rahman, Dr. Md. Nasim Akhtar

*Dhaka University of Engineering & Technology (DUET), Gazipur, Bangladesh*

*Abstract -* An evaluation of aggregate functions in relational database system considerable impact on performance in many application areas like geographic information systems and statistical and scientific databases. The problem with existing systems is inefficient execution of aggregate functions with large data volumes and lack of flexibility. It is not possible to extend the systems with new aggregate (average) functions. We show how this could be implemented into a database. We also describe how support for special kinds of aggregate queries and data structures can help in designing future high performance systems.

*Keywords :* *Aggregate Function, NULL, SQL, Boolean Algebra, Sub Queries.*

*GJRE Classification :* *FOR Code: 050299*


*Strictly as per the compliance and regulations of :*

# A New Approach for Calculating Average Value (Including Null) Without Aggregate Function

Mridul Kanti Das[α], Goutam Biswas[Ω], Md. Masudur Rahman[β], Dr. Md. Nasim Akhtar[ψ]

*Abstract -* An evaluation of aggregate functions in relational database system considerable impact on performance in many application areas like geographic information systems and statistical and scientific databases. The problem with existing systems is inefficient execution of aggregate functions with large data volumes and lack of flexibility. It is not possible to extend the systems with new aggregate (average) functions. We show how this could be implemented into a database. We also describe how support for special kinds of aggregate queries and data structures can help in designing future high performance systems.

*Keywords : Aggregate Function, NULL, SQL, Boolean Algebra, Sub Queries.*

## I. Introduction

Aggregate functions perform a calculation on a set of values and return a single value. Aggregate function avg( ) only calculate average without null values. It provides average result, eliminating null values. Null does not have a value (and is not a member of any data domain) but it is a placeholder or "mark" for missing information. Comparisons with Null can never result in either True or False but always in the third logical result is Unknown. So comparing two null is difficult. We discus about

(1) review of the research for handling null values in database system using aggregate function (2) problem structure with null value with respect to database (3) describes existing solution and proposed solution and its algorithm as well as how it works (4) details the experimental work that has been carried out. The experimental evaluation has been performed using a large amount of datasets.

**Application:**

Database System & its related Application software.

*Author α: Bachelor of Science in Engineering degree from Department of Computer Science and Engineering (CSE), Dhaka University of Engineering & Technology (DUET), Gazipur, Bangladesh. PH-+8801715543754. E-mail : mridul.duet@yahoo.com*

*Author Ω : Bachelor of Science in Engineering degree from Department of Computer Science and Engineering (CSE), Dhaka University of Engineering & Technology (DUET), Gazipur, Bangladesh. PH:+8801710741874. E-mail : welovegoutam@yahoo.com*

*Author β : Bachelor of Science in Engineering degree from Department of Computer Science and Engineering (CSE), Dhaka University of Engineering & Technology (DUET), Gazipur, Bangladesh. PH-+8801710370054. Email : masudcse21@gmail.com*

*Author ψ : Asst. Prof. of Department of CSE, DUET and he achieved his B.Sc & M.Sc from NTUU, Ukraine and Ph. D. from MITXT, Russia . PH-+8801711395537. Email : nasim_duet@yahoo.com*

**Related woks:**

Baekgraard and Mark [1] on this work, developing a nested algebra for describing sub-queries, but without considering aggregates, duplicates, or null values,

*Jeff Smith [2] is software developer, he compare all columns in database, and also handles comparing NULL values to other NULLs, Using UNION operator.*

- Aggregate function avg (X) returns the average value of all non-NULL X within a group ignoring null values .

So, we do not get appropriate result.

*Aggregate avg ( ) function with Null value problem solution:*

- *SQL>select avg (Amount) from bank;*

*Table bank*

| *name* | *Address* | *account_no* | *amount* |
|---|---|---|---|
| *Mridul* | *Sylhet* | *49501* | *500* |
| *outam* | *Gopalganj* | *49502* | *NULL* |
| *Masud* | *Narial* | *49503* | *600* |
| *Ashim* | *Khulna* | *49504* | *600* |
| *Swapan* | *Dinajpur* | *49505* | *800* |
| *Kishor* | *Kushtia* | *49506* | *600* |
| *Shahid* | *Khulna* | *49507* | *1100* |

**Normal Avg Query Result is**

**Avg=(4200)/6**

**=700**

## II. Existing Solution

- Aggregate avg ( ) function with Null value problem solution:
- SQL>select sum (amount)/count(*) from bank;

| Table bank | | | |
|---|---|---|---|
| *name* | *Address* | *account_no* | *amount* |
| *Mridul* | *Sylhet* | *49501* | *500* |
| *outam* | *Gopalganj* | *49502* | *NULL* |
| *Masud* | *Narial* | *49503* | *600* |
| *Ashim* | *Khulna* | *49504* | *600* |
| *Swapan* | *Dinajpur* | *49505* | *800* |
| *Kishor* | *Kushtia* | *49506* | *600* |
| *Shahid* | *Khulna* | *49507* | *1100* |

**Normal Avg Query Result is**

**Avg=(4200)/7**

**=600**

## III. Performance Measure Table

| Table bank | | |
|---|---|---|
| *Amount of Data* | *Amount of NULL* | *Existing solution Execution Time (sec)* |
| 40000 | 1000 | 0.2840 |
| 80000 | 4000 | 0.5720 |
| 120000 | 6000 | 0.8440 |
| 160000 | 7000 | 1.1841 |
| 180000 | 7000 | 1.3301 |
| 200000 | 7000 | 1.5511 |
| 220000 | 7000 | 1.7511 |
| 240000 | 7000 | 1.9371 |
| 260000 | 8000 | 2.0601 |
| 280000 | 8000 | 2.1851 |

## IV. Proposed Algorithm

1. **Algorithm** Average_WN (d, Avg, size )
2. // d is a data table which like two dimensional //array, size is maximum data row. r and c //mean row and column of data table //respectively.
3. // By this algorithm find the average value //considering NULL
4. {
5. sum:=0.0, Avg:=0.0;
6. size:=row size of data table;
7. For r:=0 to size-2 step 2
8. {
9. If( d[r][c]=NULL and d[r+1][c] = NULL ) then ;
10. else if( d[r][c]=NULL ) then
11. sum:=sum+ d[r+1][c] ;
12. else if( d[r+1][c]=NULL ) then
13. sum:=sum+ d[r][c] ;
14. else
15. sum:=sum+ d[r][c] + d[r+1][c] ;
16. }
17. If( size mod 2=1)
18. {
19. If(size=1) then
20. sum := sum + d[r][c];
21. else
22. sum := sum + d[r-1][c];
23. }
24. Avg:=sum/size ;
25. Print AVG ;
26. }

## V. Performance Measure Table of Proposed Solution

| Table bank | | |
|---|---|---|
| *Amount of Data* | *Amount of NULL* | *Proposed Solution Execution Time (sec)* |
| 40000 | 1000 | 0.2460 |
| 80000 | 4000 | 0.5040 |
| 120000 | 6000 | 0.8250 |
| 160000 | 7000 | 1.1591 |
| 180000 | 7000 | 1.3011 |
| 200000 | 7000 | 1.5011 |
| 220000 | 7000 | 1.5781 |
| 240000 | 7000 | 1.7641 |
| 260000 | 8000 | 1.9671 |
| 280000 | 8000 | 2.0762 |

In this solution we see that if the number of data in database gradually increased then the execution time is increased. Built-in function sum() and count() scan record individually from top to bottom in a table. Therefore Existing query are not efficient to calculating average value in large amount of data with Null values.

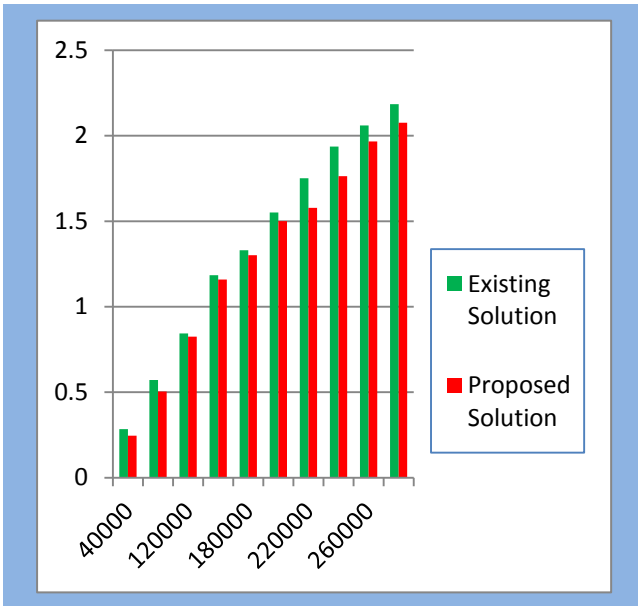
## VI. Comparison Table Between Existing and Proposed Solution

| *Table bank* | | |
|---|---|---|
| *Amount of Data* | *Existing solution Execution Time (sec)* | *Proposed Solution Execution Time (sec)* |
| 40000 | 0.2840 | 0.2460 |
| 80000 | 0.5720 | 0.5040 |
| 120000 | 0.8440 | 0.8250 |
| 160000 | 1.1841 | 1.1591 |
| 180000 | 1.3301 | 1.3011 |
| 200000 | 1.5511 | 1.5011 |
| 220000 | 1.7511 | 1.5781 |
| 240000 | 1.9371 | 1.7641 |
| 260000 | 2.0601 | 1.9671 |
| 280000 | 2.1851 | 2.0762 |

From comparison table we see that our propose system takes less times than existing system. By proposed system can reduce time and reduce the problem of existing system. To understand easily a graph chart is given below.

## VII. Graph of Existing Solution VS. Proposed Solution

Our propose solution is efficient to calculate average value with Null values from large amount of data.




From the above graph Green bar indicates Existing solution time and Red Bar indicates proposed solution time. We see that in proposed system needs execution time less than existing system.

## VIII. Conclusion

At the age of globalization most of all bank already has been computerized. They store their customer information, balance, transaction etc. in database. And they need to calculate average number of transaction after a certain period of time. Even stock exchange Ltd. Hospital, Airlines etc. need to calculate average number of transaction frequently. So our proposed system will be best for them which can save their times.

## References References Referencias

1. "A Relational Model of Data for Large Shared Data Banks" E. F. Codd, Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377–387.
2. Domain (Data Type) and null. Posted June 17,2009 http://dbtips.wordpress.com/2009/05/31/some-notes-about-null/
3. Jeff Smith is software developer, he using UNION operator comparing NULL values to other NULLs. www.weblogs.sqlteam.com.
4. Analytic Functions from Oracle® Database SQL Reference 10g Release 1 (10.1) Part Number B10759-01
5. L. Baekgraard and L. Mark. Incremental computation of nested relational query expression. ACM TODS, 20(2):111-148, June 1995.
6. Comparing Tables By Bill Graziano on 07 January 2002.
7. SQL Functions from Oracle Database Globalization Support Guide
8. "Structured Query Language (SQL)". International Business Machines. October 27, 2006. Retrieved 2007-06-10.

This page is intentionally left blank