IJCSN International Journal of Computer Science and Network, Volume 3, Issue 5, October 2014
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 0.274

Comparative Approaches of Query Optimization for Partitioned Tables

[1] **Dipmala Salunke** , [2] **Girish Potdar**

[1, 2] Computer Engineering Department, University of Pune,
PICT Pune, Maharashtra, India

Abstract - Due to vast use of Internet data grows explosively. A query that is fired on table may require a complete table scan which can take a long time as it has to inspect every row in table. Since, there is no way to identify this problem, becomes more sever for historical tables for which many queries concentrate, access on rows that were generated recently. Partition helps to solve this problem. Partition divide table into partitions. A query that only requires rows that correspond to a single partition or range of partitions can be executed using a partition scan rather than a table scan. Here, proposed Partition Algorithm, based on Rank Partition Tree structure, (PARPT) will map rows to partitions for optimizing the query performance of Max/Min type with mass data. We will compare amount of time required to execute queries on existing Range partition method and proposed partitioning method. The experimental result shows that the implemented method to solve the specific type of queries is much more effective than Range partitioning method.

Keywords - *Rank Decision Tree (RDT), Join, Range Partition Method, Query processing, Relational databases.*

1. Introduction

Data warehouse contains big tables and need to manage those to provide good query performance and timely response. In order to achieve more useful and timely response from big database, database partitioning divides the table into smaller parts that can be accessed, stored and maintained independent of one another. Partitioning is a powerful mechanism to improve the overall manageability of database. Advantages of database partitioning are Easy roll-in and roll-out of data, Easier administration of large tables, Flexible index placement and faster query processing. In the process of analyzing the enterprise data, to make business decisions, we often encounter the following database query: query the biggest commodity sales in certain areas, or query the most commodities consumption by customer in some age group. Such queries are often multi-dimensional, and they are changed when the demand is varied. A common solution is like this: Divide the table into smaller parts, and build a rank index for every partition, then search the index sequentially. This method is simple and easy to achieve relatively. But there are two problems:

i. Query range may not exist in every partition, that is, there is no satisfied record in some partitions.
ii. If the result of query is in the end range of the index, we may have to search the entire index.

It is obvious that this method is extremely low efficiency in the mass data environment, and in case of large number of partitions the wasting of resources is enormous. So here, we focus on partitioning algorithm on data. Through this we can aggregate the records satisfying the query conditions into a certain partition. Our partition search algorithm can quickly navigate to the partition, greatly reducing the join operations of the partition tables, and improving the query efficiency.

Road map of paper is as follows:

Section II: Related work gives the efforts made in this area by other people.
Section III: Programmer's Design defines the proposed system in terms of mathematical and algorithmic views. Section IV: Results and Discussion talks about the result and efficiency issues of the proposed system.
Section V: Conclusion concludes the paper with the future scope of the proposed system.
Section VI: References gives the references in IEEE format.

2. Literature Survey

Various table partitioning schemes as well as techniques to find a good partitioning scheme automatically have been proposed.[1,2,3] Selectivity-based partitioning [4], content-based routing [5],and conditional plans [6] are techniques that enable different plans to be used for different subsets of the input data .Unlike our work, these techniques focus on dynamic partitioning of (unpartitioned) tables and data streams rather than exploiting the properties of existing partitions. Previously, query optimizers had to consider only the restricted partitioning schemes specified by the DBA on base tables. Today, the optimizer faces a diverse mix of partitioning schemes that expand on traditional schemes like hash and equi-range partitioning. Hierarchical (or multidimensional) partitioning is one such scheme to deal with multiple granularities or hierarchies in mass data[7]. So here, we present a partition algorithm based on Rank Decision Tree[19], and the corresponding partition localization algorithm which are related closely to data localization in distributed DBMS [20], to optimize the query performance of max/min type with mass data.

3. Programmers Design

Figure 1 shows the System Design. Database partition in proposed system is based on Rank Decision Tree Structure [19].In this system the database used is Employee database.

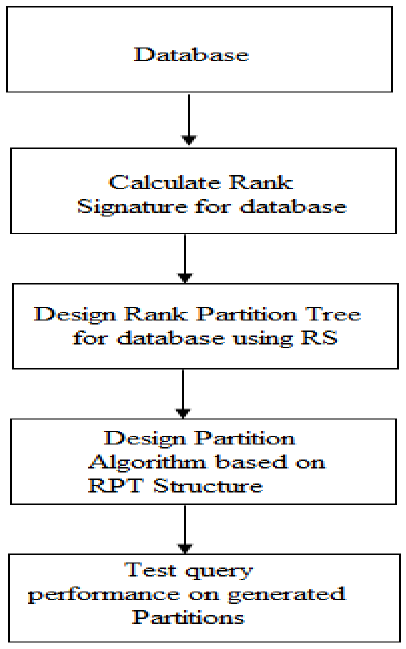
Database
Calculate Rank Signature for database
Design Rank Partition Tree for database using RS
Design Partition Algorithm based on RPT Structure
Test query performance on generated Partitions

Figure 1: System Design

To partition database first we need to calculate rank signature. Rank Signature (RS) is an m-bit vector, where each bit indicates whether value of cell is ranked above the half or not. RS calculation is useful if database is large. Based on m-bit vector of RS we design Rank Partition Tree (RPT). RPT is a binary tree through which we can partition a table quickly. Partition Algorithm is developed for a table to aggregate records into certain partitions satisfying the query condition. Result for query is available in certain partition instead of taking joins of partition tables, thus improving query efficiency of Max/min type. Query performance checked on generated partitions. Here we exploit properties of existing partition methods instead of dynamic partitioning of database to claim the query optimization for partitioned table. Here, we proposed an effective way of using the techniques of partition optimization to solve the specific type of queries with mass data.

3.1 Mathematical Model

Relevant mathematics associated with the dissertation is as given below. Let S represents Query Optimization for partitioned table system. This system provides partition by two methods, one is Range Partitioning Method (RPM) and another is Partition Algorithm based on Rank Partition Tree structure (PARPT). It can be shown in following form,

$\mathbf{S= \{T, D, V_s, D_{l:u}, R_I, R_{s(l:u)}, R_{PT}, P_n\}}$ Where,

T: Table having minimum 50,000 records.
Having any of column C1or C2 or C3.......or Cn ∈ numeric values.
D: Data array of size n. Data array contains values of column of Table T.

R: is the rank index of D, with the size of n, value of R[i] is an index for the data array D which satisfied the following inequality:

$$D[R[i]] \geq D[R[j]], \text{ if } i<j \qquad (1)$$

$\mathbf{V_s}$: The Rank Signature of D is a bit vector of size m ,
$\mathbf{V_s}$ =(b_0, b_1, ... b_{m-1}), satisfied:

$$b_{R[i]} = \begin{cases} 1, & \text{if } 0 \leq i < [m/2] \\ 0, & otherwise \end{cases} \qquad (2)$$

$\mathbf{D_{l:u:}}$ is a data array(child table) of size u-l+1, and
{ $D_{l:u}[i]$ | i=0,...,u-l } = { D[R[k]] | k=l,...,u } is satisfied.

$\mathbf{R_{s(l:u)}}$= is defined as a rank signature of $D_{l:u}$.
$D_{l;u}$ consists of u-l+1 elements of D whose ranks in D are from (l+1)th to (u+1)th.

Here, the 'l:u' is called a rank interval.

$\mathbf{R_{PT}}$= Partition tree structure based on RD-Tree.

$\mathbf{P_n}$= Number of partitions generated i.e. $P_1, P_2, P_3 \ldots\ldots P_n$

Functions

i. Calculate Rank Index

$R_I = (D) \rightarrow R[i]$

Function R_I takes the D as input and generates rank index for data array D.

Where,
R[i] should satisfy following inequality:

$D[R[i]] \geq D[R[j]]$ if $i<j$

ii. Calculate Rank Signature for D

$R_{s1} = (D, R[i]) \rightarrow V_s$

Function R_{S1} takes input data array D and Rank index R[i] to calculate rank signature for D and gives output as bit vector.

$V_s = (b_0, b_1, \ldots b_{m-1})$, should satisfy :

$$b_{R[i]} = \begin{cases} 1, & \text{if } 0 \leq i < [m/2] \\ 0, & otherwise \end{cases}$$

iii. Calculate Rank Signature for $D_{l:u}$

$R_{s2} = (D_{l:u}) \rightarrow R_{s(l:u)}$

Function R_{S2} takes input data array D and Rank index R[i] to calculate rank signature for D and gives output as bit vector.

$V_s = (b_0, b_1, \ldots b_{m-1})$, should satisfy :

$$b_{R[i]} = \begin{cases} 1, & \text{if } 0 \leq i < [m/2] \\ 0, & otherwise \end{cases}$$

$D_{l:u}$ is a data array of size u-l+1, and

$\{D_{l:u}[i] | i=0,\ldots,u-l\} = \{D[R[k]] | k=l,\ldots,u\}$ is satisfied.

The relative orders among the elements in $D_{l:u}$ are equal to those in D.

$R_{s(l:u)}$ is defined as a rank signature of $D_{l:u}$. $D_{l:u}$ consists of u-l+1 elements of D whose ranks in D are from $(l+1)^{th}$ to $(u+1)^{th}$. Here, the 'l: u' is called a rank interval.

iv. Form Rank Partition Tree (RPT)

$F = (R_{s(l:u)}) \rightarrow R_{PT}$

Let D be a data array of size m (m>1), n is the number of distributed nodes. Then the R_{PT} for D is a binary tree such that:

i. The root node of the tree is $R_{s(0:m-1)}$

ii. A leaf node of the tree is $R_{s(l:u)}$, where $u-l=m/n$.

iii. A left child of $R_{s(l:u)}$ is $R_{sl:l+w-1}$ where w is the weight of $R_{s(l:u)}$

- iv. A right child of $R_{s(l:u)}$ is $R_{s(1+w):u}$ if the weight of $R_{s(l:u)}$ is greater than 2.

Otherwise, the right child does not exist.

Figure 2 shows an example of RPT construction.

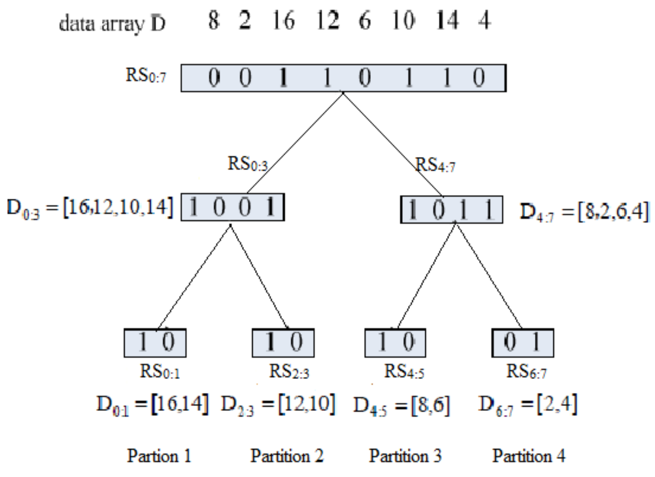

Figure 2: An Example of RPT construction

3.2 Serialization

i. First complete the calculation of rank signature on database which yields m-bit vector to decide whether value of cell is ranked above the half or not.

ii. Then construct Rank Partition Tree based on RD tree structure using calculated RS value in last step.

iii. Design partition algorithm based on constructed RPT so that result of max/min type of queries will be available in certain partition instead of taking joins of partitioned tables.

iv. Finally, we have to check performance of queries on generated partitions.

3.3 Turing Machine

Figure 3 shows the different states of the system according to the events occurred in the system. State diagram consist of events such as Login to the system, database connection, selection of proper field/attribute, then selection of partition method to partition database, testing of queries against generated partitions and finally result collection.

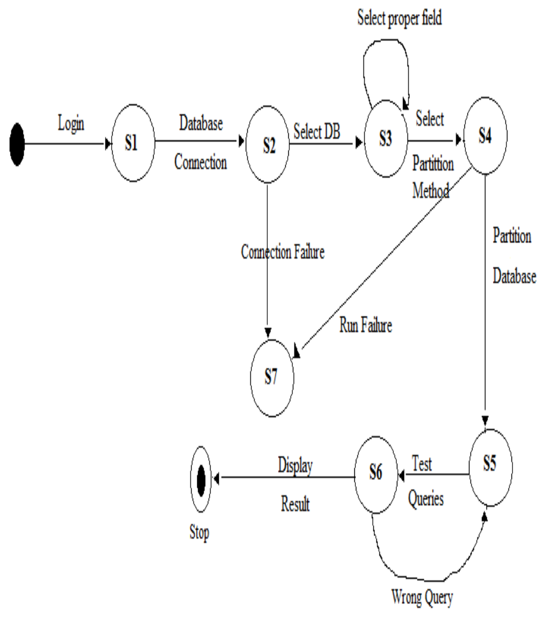

Figure 3: State Diagram

Table 1: State Table

State	Description	State	Description
S1	LogIn	S5	Partition the Database
S2	Database Connection	S6	Display Result
S3	Select Database	S7	Logout
S4	Select Partition method		

Table 1 Shows description of states in the system.

3.4 Partition Algorithm based on RPT Structure

Define a few constants,
m: the record of table;
n: the number of partitions;
logn: the extended layer of R_{PT};
$D_{l:u}$: the child table with u-l+1 record
Input: table $D_{0:m-1}$, $R_{S\ (0:m-1)}$
Output: [$D_{(0:m/n-1)}$, $Rs_{(0:m/n-1)}$] , [$D_{(m/n:2m/n-1)}$,$R_{S\ (m/n:2m/n-1)}$]……[$D_{(m(n-1)/n:m-1)}$, $R_{S\ m(n-1)/n:m-1}$]

```
BEGIN
l=0;
u=m-1;
for i=0 to logn-1 do
{
for j=1 to $2^i$ do
{
        if (j=1 && i!=1)
        {
            l=0;
            u=$m/2^i$ -1;
        }

getLeftChildPatition($D_{l:u}$ , $R_{s(l:u)}$ );
getRightChildPatition(($D_{l:u}$ , $R_{s(l:u)}$);

        l=$j*m/2^i$ ;
        u=$l+m/2^i$ -1 ;
}}
getLeftChildPatition($D_{l:u}$ , $R_{s(l:u)}$)
    {
            $D_{l:(u-1)/2}$ =getOnebit($R_{s(l:u)}$);
            $R_{S\ l:(u-1)/2}$ =ConvertToRS($D_{l:(u-1)/2}$ );
    }
getRightChildPatition($D_{l:u}$ , $R_{s(l:u)}$)
    {
            $D_{(u+1)/2:u}$ = getZerobit($R_{s(l:u)}$);
            $R\ S_{(u+1)/2:u}$ =ConvertToRS($D_{(u+1)/2:u}$ );
}
END
```

Features of PARPT Algorithm:

i. i. Records in first partition are greater than the all records value behind.
ii. ii. Number of partitions are decided based on number of nodes.
iii. For max/min type queries result will finally fall on a specific partition.
iv. We can run this system in distributed system to meet the requirements of data load balance.

4. Results

In this paper, two methods of partitioning are provided Range Partition method (RPM) and Partition algorithm based on RPT structure (PARPT). Performance of both Partitioning techniques is compared in terms of time required for partition and accuracy of the system. Different types of queries are executed on both method and result of query execution is shown by performance graph shown in figure 4. From graph, we can say that query efficiency is enhanced obviously in the experimental with query Q3 and query Q6. Q3 and Q6 are two queries are related to the max/min, and take full advantage of the characteristics of the RPT partition.

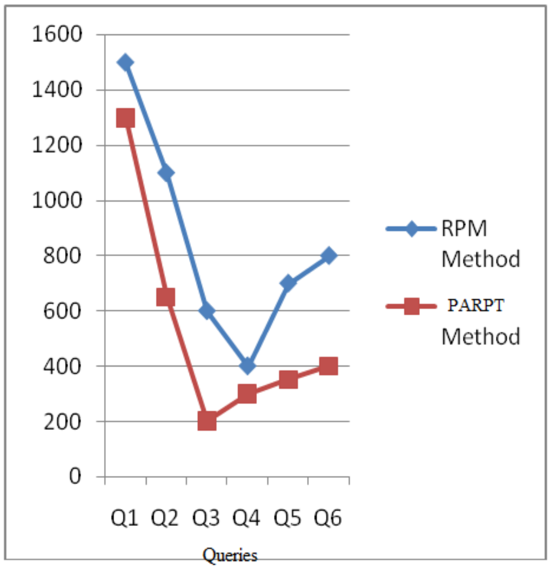

Figure 4: Comparative Graph

Thus, PARPT based on RD-Tree is more efficient for max/min type of queries in mass data than range partitioning method.

5. Conclusion

Now days, data grows explosively due to electronic commerce and use of internet. Business decisions have relied more and more on big data. To face mass data we need to divide data into partitions to meet the request of timely response. Thus, in this paper we have proposed an effective database partition algorithm to solve max/min type of queries. Here, we proposed RPT structure based on calculated RS for an attribute. Then proposed new partition algorithm based on constructed RPT structure. Finally, we compared partition algorithm with existing Range partition method. Experimental results show that proposed method greatly improved the query efficiency of max/min type.

In future, if we run proposed algorithm in distributed environment we may achieve data load balance and also come up with minimum consumption of resource as result of max/min type query is available in one partition. RPT construction can be unstable in some high dynamic environment. So, we can improve algorithm to adapt reconstruction of RPT structure.

References

[1] D. G. Shin, and K. B. Irani, “Fragmenting relations horizontally using a knowledge based approach,” IEEE Transactions on Software Engineering (TSE), Vol. 17, No. 9, pp. 872–883, 1991.

[2] S. Agrawal, V. Narasayya, and B. Yang, “Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design” ACM SIGMOD, 2004.

[3] Rao, C. Zhang, N. Megiddo, and G. M. Lohman, Automating Physical Database Design in a Parallel Database”, SIGMOD, 2002.

[4] N. Polyzotis, “Selectivity-based Partitioning: A Divide-and-union Paradigm for Effective Query Optimization”, CIKM, 2005.

[5] P. Bizarro, S. Babu, D. J. DeWitt, and J. Widom, “Content-based Routing: Different Plans for Different Data”, VLDB, 2005.

[6] A. Deshpande, C. Guestrin, W. Hong, and S. Madden,“Exploiting Correlated Attributes in Acquisitional Query Processing”, ICDE, 2005.

[7] C. Baldwin, T. Eliassi-Rad, G. Abdulla, and T. Critchlow, “The Evolution of a Hierarchical Partitioning Algorithm for Large- Scale Scientific Data:Three Steps of Increasing Complexity”,SSDB 2003.

[8] Herodotou, N. Borisov, S. Bbu, “Query Optimization Techniques for Partitioned Tables”, SIGMOD’11, Athens, Greece, *June 2011*

[9] F. Afrati and J. D. Ullman. Optimizing Joins in a MapReduce Environment. In EDBT, 2010.

[10] Sharma, Sakshi, Narendra Kumar Kumawat, and Mukesh Maheshwari. "Automatic Traffic Congestion Detection and Alert System." ijcat.org, pg 296 – 300.

[11] T. . Morales.Oracle(R) Database VLDB and Partitioning Guide 11g Release 1(11.1). Oracle Corporation,2007 http://downloaduk.oracle.com/docs/cd/B28359_01/server111/b32024/toc.html

[12] ‘TPC Benchmark H Standard Specification’, 2009 http://www.tpc.org/tpch/spec/tpch2.9.0.pdf

[13] R. Talmage. Partitioned Table and Index Strategies Using SQL , Server 2008. Microsoft, 2009. http://msdn.microsoft.com/en-us/library/dd578580.aspx

[14] Chun-Hung Cheng, Wing-Kin Lee, and Kam-Fai Wong, A Genetic Algorithm-Based Clustering Approach for Database Partitioning, IEEE Transactions On Systems, Man, And Cybernetics Part C: Applications And Reviews,Vol. 32, No. 3, Pp 215- 230,August 2002.

[15] Hadj Mahboubi and J´ er ˆome Darmont Enhancing XML Data Warehouse Query Performance by Fragmentation, SAC’09.ACM, March 2009, Honolulu, Hawaii, U.S.A.

[16] Lisbeth Rodríguez and Xiaoou Li, A Dynamic Vertical Partitioning Approach for Distributed Database System, man and Cybernetics (SMC), 2011 International Conference on 9-12 Oct. 2011, pp-1853-1858, Anchorage, AK.

[17] Ali A. Amer and Hassan I. Abdalla, An Integrated Design Schema for Performance Optimization in Distributed Environments,2012.International

Conference on Education and e-Learning Innovations IEEE, 978-1-4673-2225-6/12.

[18] Valentina Ciriani, Sabrina De Capitani di Vimercati, Sara Foresti Fragmentation Design for Efficient Query Execution over Sensitive Distributed Databases**,** 2009 29th IEEE, International Conference on Distributed Computing Systems,DOI10.1109/ICDCS 2009.

[19] Ajit M. Tamhankar and Sudha Ram, Database Fragmentation and Allocation: An Integrated Methodology and Case Study, IEEE Transactions On Systems, Man, And Cybernetics—Part A:Systems And Humans, Vol. 28, No.3, pp. 288-305 May 1998.

[20] Yon Dohn Chung , Woo Suk Yang , Myoung Ho Kim ,An efficient, robust method for processing of partial top- k/bottom-kqueries using the RD-Tree in OLAP, ACM Decision Support Systems 43 (2007) pp. 313–321.

[21] T. M. Ozsu and P. Valduriez, "Principles of Distributed Database Systems", Prentice Hall, 1999.