ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

A Survey Paper on Database Partitioning

[I]Dipmala T. Salunke, [II]Girish P. Potdar
[I]Post Graduate Student [II]Associate Prof.HOD
[I,II]Dept. of Computer Engineering, Pune Institute of Computer Technology, University of Pune.

Abstract
Now a day's most of the enterprises are running mission critical databases having several gigabytes of data. These enterprises need support system to manage big data. Database partitioning works as a support system for such enterprises with extremely high availability requirements. Also query processing on partitioned database should work fine.
The goal of this paper is to understand different partitioning techniques proposed recently to manage big data. The next step in the path would be to come up with a better partitioning techniques to handle mission critical databases. Undoubtedly the following discussed techniques are extremely good. A collaborative approach using few of them can be a better solution.

Keywords
Mass Data, Database Partitioning, Join, Range Partitioning, Query Processing

I. Introduction

Due to use of internet data grows explosively. 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, data warehouse divide the database into partitions. 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.[1]
Two types of database partitioning:

1. Vertical Partitioning:- Vertical partitioning consists of subdividing a relation into sub relations that are projections of the original relation according to a subset of attributes.
2. Horizontal Partitioning:- the horizontal partitioning divides a relation into subsets of tuples based on selection operations.

Fig.1: shows partition of table horizontally and vertically.

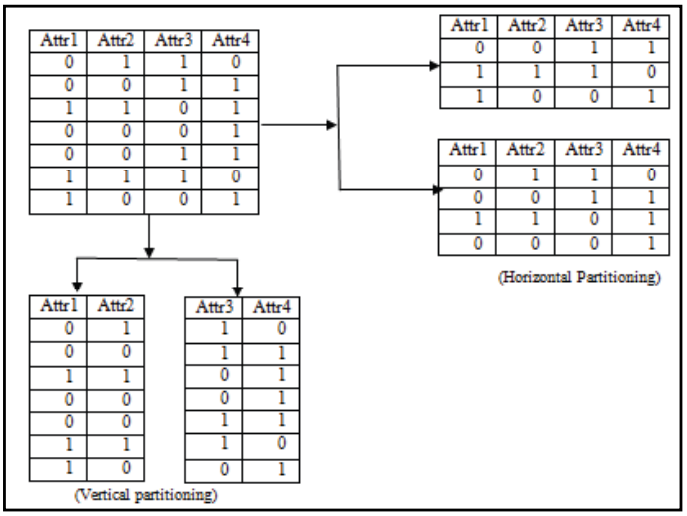

Fig.1: Partitioning Types

Partitioning Criterion

Current high end relational database management systems provide for different criteria to split the database. They take a partitioning key and assign a partition based on certain criteria. Common criteria are:

1. Range partitioning

Selects a partition by determining if the partitioning key is inside a certain range. An example could be a partition for all rows where the column zip code has a value between 80000 and 99999.

2. List partitioning:

A partition is assigned a list of values. If the partitioning key has one of these values, the partition is chosen. For example all rows where the column Country is either eland, Norway, Sweden, Finland or Denmark could build a partition for the Nordic countries.

3. Hash partitioning:

The value of a hash function determines membership in a partition. Assuming there are four partitions, the hash function could return a value from 0 to 3.

4. Composite partitioning:

Allows for certain combinations of the above partitioning schemes, by for example first applying a range partitioning and then a hash partitioning. Consistent hashing could be considered a composite of hash and list partitioning where the hash reduces the key space to a size that can be listed.[3]

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.

II. Few Recently Implemented Techniques Of Database Partitioning

1. Selectivity Based Partitioning: A Divide and Union Paradigm for Effective Query Optimization [4]

In this paper, Authors have initiated the study of selectivity-based partitioning, a novel approach to query optimization that adopts a divide and-union approach to query evaluation. The basic idea is to carefully partition a relation according to the selectivity's of the join operations, and subsequently rewrites the query as a union of constituent queries over the computed partitions. The optimization problem is been decided and presented an analysis on the characteristics of an optimal solution. Based on this analysis, developed an efficient algorithm for computing an effective partitioning of the input query while considering a limited fraction of the total search space. Experimental results verify the effectiveness of selectivity-based partitioning and demonstrate its potential as a paradigm for query optimization.

2. What can Partitioning do for your Data Warehouses and Marts? [5]

Efficient query processing is a critical requirement for data warehousing systems as decision support applications often require minimum response times to answer complex, ad-hoc queries having aggregations, multi-ways joins over vast repositories of data. This can be achieved by fragmenting warehouse data. The data fragmentation concept in the context of distributed databases aims to reduce query execution time and facilitates the parallel execution of queries. In this paper, authors have implemented an algorithm for fragmenting the tables of a star schema. During the fragmentation process, it is observed that the choice of the dimension tables used in fragmenting the fact table plays an important role on overall performance. Therefore, greedy algorithm is developed to select "best" dimension tables. An analytical cost model for executing a set of OLAP queries on a fragmented star schema is built. Some experiments are conducted to evaluate the utility of the fragmentation for efficiently executing OLAP queries.

3. Database Partitioning Techniques to Support Reload in a Main Memory Database System: Mars [6]

In main memory database system the primary copy of database is placed in volatile memory. When a crash occurs, a partial or complete reload of the database from archive memory (AM) into main memory (MM) is required. Authors have done complete analysis of possible partitioning techniques on database to solve reload problem and number of MM references incurred during transaction processing. The best technique is the one that yields the minimum overall cost consisting of both properties. Their analysis shows that horizontal and single vertical are actually the only possible candidates. Physical vertical never yields the best result. In some very rare cases, group vertical outperforms the other techniques. If the database system encountered performs more selections than projections and joins, and performs more tuple modifications or tuple deletions than tuple insertions then horizontal is the best technique. Otherwise, single vertical is the chosen technique. If reload is the only concern, that is if we do not take into account the transaction performance, then single vertical is always the best choice.

2.4 A Genetic Algorithm-Based Clustering Approach for Database Partitioning [7]

In a typical distributed/parallel database system, a request mostly accesses a subset of the entire database. It is, therefore, natural to organize commonly accessed data together and to place them on nearby, preferably the same, machine(s)/site(s). For this reason, data partitioning and data allocation are performance critical issues in distributed database application design. Data partitioning requires the use of clustering. In this paper, Authors have explored the use of a genetic search-based clustering algorithm for data partitioning to achieve high database retrieval performance. The Vertical Partitioning (VP) model is represented by a transaction-attribute matrix[14]. VP requires that the matrix rows and columns be rearranged to produce a special structural. The rearrangement problem has been shown to be equivalent to a Travelling Salesperson Problem [13]. Two new genetic operators, SE and SP, as well as a modified version of the existing enhanced Edge Recombination crossover operators, are proposed for solving the TSP. The performance for several operators has been evaluated. Results indicate that proposed crossovers have different contributions in solving the TSP and that they outperform other operators in solving the problem.

2.5 Physical and Virtual Partitioning in OLAP Database Clusters [8]

On-line analytical processing (OLAP) applications require high performance database support to achieve good response time. Database clusters provide a cost-effective alternative to parallel database systems.

For OLAP applications, that typically use heavy weight queries, intra-query parallelism yields better performance as it reduces the execution time of individual queries. Intra-query parallelism is based on processing the same query on different subsets of the query table. Adaptive Virtual Partitioning is essentially a sequential algorithm that by running on different subsets of the database can obtain intra-query parallelism in a database cluster [12]. The speedup obtained reduces significantly the time of typical OLAP heavy weight queries and confirms this can be a good basis for future works in high-availability for database clusters and dynamic load balancing. Combining physical and virtual partitioning to define table subsets provides flexibility in intra-query parallelism while optimizing disk space usage and data availability. Experiments with our partitioning technique using TPC-H benchmark queries on a 32-dual node cluster gave linear and super-linear speedup, thereby reducing significantly the time of typical OLAP heavy weight queries.

2.6 Near-uniform Range Partition Approach for Increased Partitioning in Large Database [9]

Database partitioning technique which adopts "divide and conquer" method can efficiently simplify the complexity of managing massive data and improve the performance of the system, especially the range partitioning. The traditional range partitioning approach brings heavy burden to the system without an increased partitioning algorithm, so it does not adapt to the partitioning in the real time data environment. To speed up the partitioning algorithm, the current partitioning technology is well studied and three effective range partitioning algorithms for the massive data are proposed, which are based on allowing the fluctuation of data amount in each range of partitions. Some experiments and applications show that the proposed algorithms are more effective and efficient to partitioning and re-partitioning tables in the large database or real-time environment. It is easy to maintain the partition automatically, rapidly and effectively.

2.7 A Dynamic Vertical Partitioning Approach for Distributed Database System [10]

Vertical and horizontal partitioning are physical database design techniques that can considerably improve query response time in distributed database system. Although most current database management systems support horizontal partitioning, they do not implement vertical partitioning because it is based on user queries and it is necessary to monitor queries in order to generate a good vertical partitioning solution. In this paper, authors use active rules to develop an active system for dynamic vertical partitioning of distributed database. An active-rule based statistic collector accumulates information about attributes, queries and fragments without the explicit intervention of the DBA[16]. It means the system vertically fragment and re-fragment a database without intervention of a database administrator. The fragment configuration will change dynamically according to the changes

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

in the information of the queries in order to find the best solution and not affect the performance of the database. All the vertical partitioning process is implemented inside the database using rules, the attribute usage matrix used by most of the vertical partitioning algorithms is implemented as a database table order to use rules to change the fragment configuration automatically. An active rule-based partitioning reorganizer that automatically creates the fragments on disk when is triggered by the partitioning analyzer. Experiments on a Benchmark database TPC-H demonstrate acceptable query response time.

2.8 Partitioning Techniques for Fine-grained Indexing [11]

Many data-intensive websites use databases that grow much faster than the rate that users access the data. Such growing datasets lead to ever-increasing space and performance overheads for maintaining and accessing indexes. Furthermore, there is often considerable skew with popular users and recent data accessed much more frequently. These observations led authors to design Shinobi, a system which uses horizontal partitioning as a mechanism for improving query performance to cluster the physical data, and increasing insert performance by only indexing data that is frequently accessed present database design algorithms that optimally partition tables, drop indexes from partitions that are infrequently queried, and maintain these partitions as workloads change [15]. Authors have shown a 60× performance improvement over traditionally indexed tables using a real-world query workload derived from a traffic monitoring application. Experiments show partitioning significantly reduces query costs when the dataset is not clustered on the partition keys, whereas selective indexing can dramatically reduce the index size, and correspondingly the index costs, even for clustered datasets. We show dramatic performance improvements on a real-world two-dimensional query workload from a traffic analysis website, with average performance that is 60× better than an un partitioned, fully indexed database.

III. Conclusion

Due to vast use of internet corporations, government agencies and other organizations need to manage mass data. To improve query response time and to manage very large databases, partitioning is the solution. In this paper, we have studied some static and dynamic database partitioning techniques to support mission critical databases. The techniques discussed above are extremely useful but next step in the path would be to come up with modifications in existing partitioning methods or to come up with collaborative approach to find more efficient partitioning algorithm.

References

[1]. *Hong Yin, Shuqiang Yang, Hui Zhao, Zhikun Chen, "Partial query optimization Techniques for partitioned table," IEEE 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012) ,pp. 792–797.*

[2]. *http://ace.ucv.ro/sintes13/SINTES13_2007/ComputerEngineering/C16_ID14.pdf.*

[3]. *http://en.wikipedia.org/wiki/Partition_(database).*

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

[5]. *Ladjel Bellatreche,Kamalakar," What can Partitioning Do For your Data Warehouses and Data Ma-" rts?",IDEAS'00,Proceedings of the 2000 International Symposium On Database Engineering and A-lications,pp-437-443.*

[6]. *LeGruenwaldMargaretH.Eich, "Database Partitioning TechniquesToSupport Reload In A Main Memory Database System: Mars",IEEE 1990,Published in:Databases,Parallelarchitecturesandtheir Applications,PARBASE-90,InternationalConfe rence, pp:107-109.*

[7]. *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, August 2002", pp-215-230.*

[8]. *Camille Furtado1, Alexandre A. B. Lima1, Esther Pacitti2, Patrick Valduriez2 and Marta Mattoso1, Physical and Virtual Partitioning in OLAP Database Clusters, Proceedings of the 17th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'05).*

[9]. *Wen Qi, Jie Song, Yu-bin Bao ,Near-uniform Range Partition Approach for Increased Partitioningin Large Database, IEEE,2010.*

[10]. *Lisbeth Rodríguez and Xiaoou Li,"A Dynamic Vertical Partitioning Approach for Distributed Database System",IEEE 2011,pp- 1853-1858.*

[11]. *Eugene Wu, Samuel Madden, "Partitioning Techniques for Fine-grained Indexing",IEEE 2011, pp1127-1138.*

[12]. *C.Furtado, A.Lima, E.Pacitti, et al. Physical and Virtual Partitioning in OLAP Database Clusters Proc. of Conf. on Computer Architecture and High Performance Computing, 2005, pp.143-150.*

[13]. *J.K.Lenstra and A.H.G.Kan Rinnooy,"Some simple applications of the traveling salesman problem," Oper. Res. Q.,vol. 26, pp.717–733,1975.*

[14]. *S.Navathe, S.Ceri,G.Wiederhold,and J.Dou, "Vert partitioning algorithm for database design," ACM Trans. Database Syst., vol. 9, pp. 680–710, 1984.*

[15]. *E. Wu, "Shinobi: Insert-aware partitioning and indexing techniques for skewed database workloads" Master's thesis, MIT, 2010.*

[16]. *N. W. Paton and O. Diaz, "Active database systems," ACM Computing Surveys, 31(1), 1999, pp.64-103.*