EFFICIENT LIKE COMMANDS FOR DYNAMIC DATA RETRIEVAL AND REPORT GENERATION FROM PARALLEL DATABASES.

J.Vijayashree*, C.Ranichandra

Department of Information technology, VIT University, Vellore, India.

INTRODUCTION

Database contains tables and the tables contain thousands of records. Databases used in organizations are usually too large. Database tables contain huge amount of records. The most common operation performed in database is database search or data retrieval. Data search and data retrieval are based on the size of the database tables. If the database table contains huge volume of records then the search or retrieval of data will consume more time. In order to reduce the time consumption for data retrieval the like commands are used. The like commands partitions the tables horizontally and produce the result. Data partitioning splits a table into smaller parts which can be accessed, stored and maintained independent of one another. In order to improve the query performance and on the whole manageability of the database system partitioning techniques are used. Data partitioning simplifies administrative tasks like data loading, removal, backup, statistical maintenance and storage provisioning. The most common task performed in all the databases is search or retrieval of data. Retrieval of data consumes time based on the database size. If the database is of very large size then data retrieval consumes more time and if the database is of small size then data retrieval takes less time. The project deals with efficient data retrieval from normal database and parallel databases dynamically. In enhanced search, search can be done based on any attribute dynamically. The search result is produced in a table format. From the resulting table any of the records details can be printed in a report format. For the retrieval of data three different like queries are used. "$.key_word." Like query is used for numerical attributes and the "$.key_word." and "%.key_word." are used for non numeric attributes. The precision of the above like commands are analyzed. The proposed techniques improve search performance and reduce the data retrieval time.

Data partitioning is a typical feature in parallel database systems. Data partitioning splits a table into smaller parts which can be accessed, stored and maintained independent of one another. In order to improve the query performance and on the whole manageability of the database system partitioning techniques are used. Data partitioning simplifies administrative tasks like data loading, removal, backup, statistical maintenance and storage provisioning. The most common task performed in all the databases is search or retrieval of data. Retrieval of data consumes time based on the database size. If the database is of very large size then data retrieval consumes more time and if the database is of small size then data retrieval takes less time. The project deals with efficient data retrieval from normal database and parallel databases dynamically. In enhanced search, search can be done based on any attribute dynamically. The search result is produced in a table format. From the resulting table any of the records details can be printed in a report format. For the retrieval of data three different like queries are used. "$.key_word." Like query is used for numerical attributes and the "$.key_word." and "%.key_word." are used for non numeric attributes. The precision of the above like commands are analyzed. The proposed techniques improve search performance and reduce the data retrieval time.

Table partitioning is a data organization scheme in which table data is divided across multiple storage objects called data partitions and the value which determines how the partition is performed is the partition key. The partition determines to which partition the records has to be allocated. Each partition can be named independently and can be stored either in the same system or some other system. Each partition can be stored and retrieved independently. Greater query throughput is achieved through partitioning. In future some other partitions can be done or the partitioned tables can be deleted if not necessary.

Partitioning can be done in two ways namely horizontal partition and vertical data partition as in [7]. Horizontal partitioning partitions the table horizontally and each partition has some number of entire records of a particular table. Vertical data partitioning partitions the table vertically and each partitions contains all the records of the table with fewer attributes. Vertical partitioning is done based on the attributes. In most of the data partitioning techniques horizontal technique is used. Some of the well known horizontal data partitioning techniques are Round robin partitioning, Hash partitioning, Range partitioning and Random unequal partitioning.

In hash partitioning the partition is done based on single attribute. If the partition is performed based on say
designations then particular designation employees will be in one partitioned table and other employees of particular designation will be stored in other partition. Range partitioning partitions the table based on some range values that is employees of certain age say 31 to 40 will be in one partition and employees of age between 41 to 50 will be in one partition. In random unequal partitioning technique each partition contains randomly unequal number of records. In Round-robin data partitioning the records of the main table will be allocated to all the tables in clockwise manner. That is if the database table is partitioned into three tables then the first record of a table will be partitioned will be allocated to the first table and the second record will be allocated to the second table and the third record will be allocated to the third table and the fourth record will be allocated to the first table in clockwise manner.

In hash partition and range partition techniques the partitioned tables will contain unequal number of records where as in round robin data partition technique each partitioned table will contain almost same number of records. Parallel query processing can be performed on the partitioned tables. If the query is processed parallelly in tables’ partitioned using hash then the processing will be done in fewer tables and not in all tables. In range partition also the processing will be done only in fewer tables and not in all partitioned tables, whereas in case of round robin technique processing is done in all the partitioned tables.

RELATED WORK

Author [1] made a comparison between different partitioning techniques. The partition techniques compared in this paper are document-based partitioning, term based partitioning and hybrid partitioning. Comparison is based on average query response time. The partitioning is made by dividing the collection in to sub collections. When a query is given it is processed by joining all the collections or some of the collections. In document based partition one or more partitions takes part in answering a query. In term based partition each partition takes part in answering a query. High load balance is obtained in document based partition but the through is very low. In term based partition, the load balancing is poor but the query throughput is high. Hybrid partition combines both document and term based where the load balancing and high throughput is achieved. Based on the comparison the author concluded that term based partition is out performed by document based and hybrid partition.

Author [2] proposed a new partitioning algorithm called security aware partition algorithm. Large file systems usually have many users and the access rights should be specified for every user, so that the confidential files can be maintained safely. The security aware partition algorithm does partitioning as well as provides security features for the file system. Thus searching the files more efficiently and securely is made possible in large file systems. Partitioning algorithms are mainly used for improving the search speeds in file system. Security aware partition is more useful in file systems which contain multiple users with different security permissions. In this new partition technique at the time of creating partition itself the security permissions are used. The user can query for those files only for which he has accessed.

Author [3] proposed the partitioning in terms of physical and virtual partitioning in OLAP database clusters. For physical and virtual partitioning intra query parallelism mechanism is used. Parallel database systems are used for achieving higher performance. Partitioning and replication are the two methods used for achieving parallel database system. Execution time of individual query is reduced through intra query parallelism. This technique by reducing the query execution time improves the performance of the system. Tables in database can be partitioned in two ways namely physical and virtual. In physical method the tables are partitioned physically. In virtual database partition the database is replicated and the query can be processed by any of the partition.

Author [4] proposed an enhanced hybrid range partition for parallel database systems. This algorithm is an extension of traditional range partitioning. The most common issues in parallel systems is load balancing, the author combined the three conventional partitions such as range, round robin algorithm and hashing technique. Range partitioning technique allocates the record to each partition based on some range values. Round robin technique allocates consecutive record to each partition in round robin fashion. Hash partition allocates the records based on the hash values to each partition. In this paper the ranges used for partitioning is of variable size. Some partition may contain more records and some may contain fewer records. This technique balances the load in database systems.

Author [5] proposed a new type of partition algorithm namely near uniform range partition algorithm. This type of partition algorithm can be used in real time data warehousing. There are different types of partition algorithms. Among those the author focused on range partitioning. In order to support the real time data and to improve the performance of the system, near uniform range partition algorithm is used. In traditional range partition approach, the table data is divided based on some range values. Using range partition, each partition will have same number of records. In the proposed near uniform range partition algorithm, the range values are not exactly same. Nearer values are taken for partitioning. Each partition do not contain do not contain the same number of records.

Author [6] proposed a new partition algorithm for large databases. The most common operation done in large databases is querying for data. Partitioning is the most often used approach for improving the query performance. If the database is too large then the response time for search option will be higher. In order to reduce the response time partitioning is done. In this paper horizontal partitioning method is used. Horizontal partitioning partitions the table horizontally. The number of records in the table is divided among partitions in horizontal method. Range partition is the traditional method used for partitioning. In range partitioning approach records are allocated to partitions based on some range values. In traditional method all the partitions contains same amount of data. In this paper near uniform range partitioning is done where ranges used are not same. The partition contains varying amounts of data.

PROPOSED WORK

Data retrieval and search is the most common task performed in databases. If the database is of very large size then the data retrieval time will be high and if the database is of small size then the data retrieval time will be less comparatively. Usually for retrieving any data select queries will be given. In the proposed system dynamic
search queries are used. There is no need to type the query each and every time when data has to be retrieved. Instead list box and text boxes are used. All the attributes are listed in list box. In order to perform search first the specified attribute has to be selected from the list box and the values to be searched from the table should be specified in the text box. The different like queries used are ‘$key_word’, ‘$key_word%’, and '%$key_word$’.

Using these like queries data can be retrieved dynamically from single database and parallel databases. The data retrieved is produced in a table format. The data retrieved can also be generated as a report. Generation of report is done dynamically. Any of the records from the produced output can be generated as report and also can be printed. Databases are of usually very large size and if the size of the database table is too high the partitioning can be done. Using partitioning techniques database tables can be split into smaller parts which can be stored, accessed and retrieved independent of one another. Partitioning supports parallel query processing. In parallel query processing if a query is issued then the query will be processed in parallel in all partitioned tables. Some of the partitioning techniques are range partitioning, round robin partitioning and hash partitioning. In range partitioning technique range values are used for partitioning and based on the range values records will be stored in partitions. If the number of records in particular range is more than that partition will contain more records than other partitions. In round robin technique the first record from the table to be partitioned will be placed in first partition and the second record will be placed in second partition and the third record will be placed in third partition and if the last partition is reached then the records will be placed in clockwise manner from first partition. In round robin technique all the partitions will contain almost same number of records. For efficient data retrieval exact match and different like queries are used. In hash partition specific records say employees of age 50 will be in one partition. Each partition contains specific values. Even in hash partition each partition will contain different number of records. The like queries can also be used with data partitioning technique for efficient data retrieval.

**METHODOLOGY**

PHP and mysql are used for implementing round robin partition technique. Database by name employee is created and employee details table is created under employee database. The attributes are empid, emp name, school name, designation, age, gender, salary, experience, subjects handled.

**EXPERIMENTAL QUERIES**

The three like queries are used with select statement for data retrieval or search performance from single table and partitioned tables. Employee details table contains the attributes are empid, emp name, school name, designation, age, gender, salary, experience, subjects handled.

```
select * from emp_details where emp_details.school LIKE '%s%' ORDER BY emp_id
select * from emp_details where emp_details.school LIKE '%si%' ORDER BY emp_id
select * from emp_details where emp_details.school LIKE '%se%' ORDER BY emp_id
select * from emp_details where emp_details.emp_desgn LIKE '%junior%' ORDER BY emp_id
select * from emp_details where emp_details.emp_desgn LIKE '%senior%' ORDER BY emp_id
select * from emp_details where emp_details.emp_desgn LIKE '%asst prof%' ORDER BY emp_id
select * from emp_details where emp_details.emp_desgn LIKE '%prof%' ORDER BY emp_id
select * from emp_details where emp_details.emp_id='1010' ORDER BY emp_id
select * from emp_details where emp_details.salary_det='20000' ORDER BY emp_id
select * from emp_details where emp_details.empname LIKE 'a%' ORDER BY empname
```

Based on the above queries the records are retrieved dynamically. Based on the retrieved data the precision of the like query is determined. "$key_word%" Like command retrieves only the exact data from database. "$key_word%" like command retrieves all the records whose name attribute starts with particular character. '%$key_word%' like command fetches both exact and irrelevant records. Based on the values specified in the text box the values may be exact or irrelevant. Graph is plotted for precision details. Precision is the fraction of the records retrieved that are relevant to the user's information need.

**RESULTS AND DISCUSSION**

Graph 1: Dynamic search on employee id attribute

The attribute to be retrieved is selected from list box and the specific value to be searched is specified in text box. The result of search is produced is exact.

Graph 2: Search on employee name attribute

In the above fig: 2 the attribute selected is employee name and the value specified is d. The resulting table contains all the names starting with alphabet d.
For the attribute designation the command 

```
%.$key_word."%'
```

is used. If the value specified in text box is junior then only asst professor junior designation records will be produced as result. If value is senior then only asst professor senior will be produced as result. Where as if asst professor is specied in text box then the result will contain asst professor, asst prof professor junior and asst professor senior.

The result of search is produced in a table and the result can also be generated as report dynamically. For each of the records retrieved as result report can be generated and can be printed. The report generation is done dynamically.

In the above graph the result of commands "".$key_word."" and 

```
%.$key_word."%'
```

do not retrieve any irrelevant data where as the command "%.$.key_word."%'" retrieves some irrelevant data.

**CONCLUSION**

Using dynamic like commands data can be retrieved very efficiently from single table as well as partitioned tables. The query response is also very less using dynamic like queries. Dynamic like queries also improve query performance. Parallelism is achieved through data partitioning techniques.

**REFERENCES**


**AUTHOR PROFILE**

J.VIJAYASHREE has received B.TECH in INFORMATION TECHNOLOGY from ANNA University, India 2009 and is currently pursuing master's in Networking in VIT University Vellore and area of interest is MOBILE COMPUTING. Has published three papers in International journals.

C.RANICHANDRA has received B.TECH and M.TECH from VIT university, India. The author is currently working as Assistant Professor Senior in VIT and doing research work in Grid Databases.