International Journal of Soft Computing 12 (5-6): 329-336, 2017
ISSN: 1816-9503

Data Retention and Efficiency of Information Retrieval

Romani Farid Ibrahim
High Institute of Computer Science and Information,
City of Culture and Science, 6 October City, Egypt

Abstract: Over the years, the size of data is increasing in organizations information systems. The value of information/data depends on its accuracy and timeliness. When it becomes obsolete or there is no need to use it, it becomes a burden on the system. In this study, our concern is improving the performance of data retrieval from relational database systems and to determine how long data should be retained on the system. We classified transaction data into two classes: basic data class and historical data class. We proposed new properties for data item and an algorithm for data management that help in data analysis and transferring data among system components automatically until it removed from the system. We implemented a prototype for the algorithm and measured data retrieval performance, we found that by reducing a table size the data retrieval performance is increased approximately by the same ratio.

Key words: Database, transaction, query optimization, information retrieval, indexing, concurrency control, data warehouse, load balancing, cloud computing

INTRODUCTION

Information is defined as the processed data to make it meaningful and useful. The value of information depends on its accuracy, completeness and its availability on the required time. When this information become obsolete or there is no need for use it, it becomes a burden on the system and it costs space, increases processing time and retrieval time and complicates maintenance process. Logically, it is not acceptable that the system keeps historical transactions data and unused old data forever because there is available disk space and the increase of hardware speed and the performance of searching algorithms and indexing techniques.

Data are classified into three different kinds: fully structured, semi-structured and unstructured. Fully structured data follows a predefined schema. The schema defines the type and structure of data and its relations. A typical example for fully structured data is a relational database system. Unstructured refers to the fact that no identifiable structure within this kind of data is available. Unstructured data is also described as data that cannot be stored in rows and columns in a relational database. An example for unstructured data is a document that is archived in a file folder. Other examples are videos and images. Semi-structured data is often explained as schemaless or self-describing, terms that indicate that there is no separate description of the type or structure of the data, it does not require a schema definition. A typical example of semi-structured data is XML and JSON documents (Sint *et al*., 2009).

Information Retrieval (IR) deals with the representation, storage of and access to information items. The representation and organization of the information items should provide the user with easy access to the information in which he is interested (Yates and Neto, 1999). Another definition for information retrieval from the view of web search is finding material (usually documents) of unstructured nature (usually text) that satisfies an information need from within large collections (Manning *et al*., 2008).

Query optimization is the activity of choosing an efficient execution strategy for processing a query to minimizes resource usage. Query optimization tries to reduce the total execution time of the query which is the sum of the execution times of all individual operations that make up the query (Connolly and Begg, 2005). A query optimizer chooses whether or not to use indexes for a given query and which join techniques to use when joining multiple tables. These decisions have a tremendous effect on SQL performance (Anonymous, 2005).

DBMSs use indexing techniques for making the retrieval of data more efficient. An index is a data structure that allows the DBMS to locate particular records in a file more quickly and thereby speed response to user queries.

Transaction is defined as a means by which an application programmer can package together a sequence of database operations so that the database can provide a number of guarantees, known as the ACID (Atomicity, Consistency, Isolation and Durability) (O'neil and O'nell,

2001). We defined transaction as a program in execution in which each write-set satisfies the ACID properties. Simple transaction is a transaction that cannot be divided into subtransactions and all ACID properties are achieved.

Load balancing is a computer network method for distributing workloads across multiple computing resources to optimize resource use maximize throughput, minimize response time and evade overload of any one of the resources by the use of multiple components with load balancing instead of a single component may increase reliability through redundancy (Haryani and Jagli, 2014).

When a database table grows in size to the hundreds of gigabytes or more it can become more difficult to load new data, remove old data and maintain indexes. Just the sheer size of the table causes such operations to take much longer. Even the data that must be loaded or removed can be very sizable, making INSERT and DELETE operations on the table impractical. The Microsoft® SQL Server® 2008 database software provides table partitioning to make such operations more manageable (Talmage, 2009).

Based on the concepts that we are mentioned in the previous sections, we designed a data transfer algorithm that helps in managing data in the databases automatically and improve the data retrieval performance.

Literature review: Most researchers assume that data is retained in the system forever and it is the responsibility of database administrators to manage their organizations databases according to their business rules and their experiences but there is no one approach that manage this task automatically. To improve the performance of data retrieval, researches have done much work in many areas such as data partitioning, query optimization, indexing, etc.

Talmage (2009) introduce a partition-aware SQL query optimizer to generate efficient plans for SQL queries over partitioned tables. Alsultanny (2010) introduce a method to use RAID level1 with vertical partitioning to improve the database retrieval. Agrawal *et al.* (2004) introduce techniques to enable a scalable solution to the integrated physical design problem of indexes, materialized views, vertical and horizontal partitioning for both performance and manageability. Chu *et al.* (1999) introduce an algorithm for choosing the least expected cost plan for query execution. Gupta *et al.* (2015) present a summary of query optimisation techniques. Geng (2015) introduce a mechanism to achieve the purpose of optimizing the performance of massive data retrieval through the study of the Oracle database indexing technology. Bertino *et al.* (1997) present summary of indexing techniques for advanced database applications such as parallel and distributed databases, mobile computing, data warehousing and the web. Mamun *et al.* (2012) introduce a compression algorithm to compress the inverted file index by compressing the document number in the inverted file entries using a new coding technique based on run-length encoding.

MATERIALS AND METHODS

Transaction data classifications and data item properties: In this study, we present a classification of data that output from transaction processing and suggest new data item properties to help in the evaluation if the data item is needed to be retain on the system or not.

Transaction data classification: Transactional data can be classified into two classes basic data class and historical data class. Basic data class consists of the basic data of an organization information system and it is used frequently to perform update and insertion transactions on this data class. Example of this data class is a products table that stores data about the available quantities of products and update transactions are performed to increase or decrease the value of the available quantity of a product. Insertion transactions are performed to add new basic class data items for example, a new product is added to the product table. Historical, data class stores data about transactions on the basic data class and is usually stored in transactions tables. It stores data about daily transactions of an organization and data is usually added only to the transactions tables. This type of data class is usually retrieved for the purpose of analysis or as inputs for summary transactions for example a summary transaction that calculates the total sold amount of all products or the total sold amount for a specific customer. Usually, number of retrieval of the historical data is very small compared to number of retrieval of the basic data. After specific period of time historical data may be never retrieved but it is still stored in the database.

The values of historical data class are unchangeable after its transaction is committed because of the durability characteristic of transactions. Also, old basic data class items are rarely or never used after specific period of time and becomes a burden on the system. An example of the old basic data class is a university graduated students data that still stored in the database of the current undergraduate students, although students were graduated from long time such as 10 or 20 years or more. The system administrators solve the problem of increasing the size of data by increasing the sizes and

numbers of their hard disks but it is not logic to keep student data in the same database from the date of his or her registration in the university to forever. The same idea is applied on all old basic data class in information systems of different organizations such as old medical products that are not currently produced or old model of a device that currently not produced. Also, many organizations don't have and don't use a data warehouse subsystem to analysis old data and get the benefits of data warehousing.

Data items properties: All electronic data are usually stored as files of different types (text, image, web page, audio, video, database) and share common characteristics such as date and time of creation, last modified (save) date and time and last access (moving) date and time. Operating systems such as windows 10 does not store information about the last read time and the number of reads. In some MS office applications such as MS Word, it stores numbers of versions which means the number of modification of a document. All these characteristics don't give indication about the frequency of use of a document or if there is a need to keep it on the system or not.

We suggest to add new properties for data files that can give indication about the usage frequency of a document. For database applications, these properties can be added to records or tuples. It can be used for data analysis, data transferring among components of the system or removing it from the system. Every data item (record or file) can include the following properties:

- Last read time
- Last write time
- Number of read
- Number of write
- Lifetime

The first four properties give indication about the usage frequency of data item and can be used for different data analysis. These properties should be maintained automatically by the system. The fifth property is an expectation about the lifetime of the data item in a system. The default value of this property is determined by the system administrator according to the application type and the business rules. The lifetime property can be used for transferring data items among components of the system or removing it from the system.

Data Transfer Algorithm (DTA): In this study, we present suggested data classes and how DTA will transfer data among these data classes, then we present the data transfer algorithm and a flow chart as a graphical representation for it.

Data classes and description of the data transfer algorithm: As an example, we assume an university information system that stores students data such as registration data, grades data, payments fees data, books purchasing data, etc. The system administrator specifies the different data classes for the system which are: basic data class table, historical data class table, level 1 data class table, level 2 data class table, basic-archive data class table and historical-archive data class table. Every data item (record) has a lifetime period in its class before transferring to the next class. Default lifetime for each data class is assigned by the system administrator according to the application type and the business rules. Basic data are transferred among three level of tables before deleting. We assume basic data class contains the current data that are usually used, we call it basic active data. The level 1 data class contains semi-active data that are used from time to time but not frequently and are called semi-active data. Level 2 data class contains data that are rarely used and are called semi-passive data. Archive class contains data that approximately never used and are called passive data. Also, historical data class contains recently transaction data before transferring to archive and become passive data (Fig. 1).

The system checks the data item properties every period (for example daily) as set by the system administrator. It checks if the current date equals the end of lifetime date and the data item is never accessed (read or write) for 1 year (366 days) or more, if the evaluation of these conditions is true then the data item is transferred to the next level data class. Historical, data will be transferred to the historical data archive. The purpose of checking the data are never accessed for 1 year or more is to be sure that the lifetime period that is assigned by the system administrator is well expected and is suitable for the data item, otherwise the system will not transfer the data to next level data class. The data that is stored in the data archives for specific period of time according to law rules or organization rules before deleting it, the system requests a permission from the system administrator to delete it, if the permission is granted data will be deleted, otherwise data remains as it is in the archive.

Pseudo and flow chart for data transfer algorithm: The following algorithm can be put as a part of the database management system or as a separate program that can be used as a tool by database administrators to help them in managing their organizations databases. The system performs the check automatically and regularly as specified by the system administrator for example daily. The DBMS rebuilds indexes automatically after the transfer of data from the source table to the destination table.

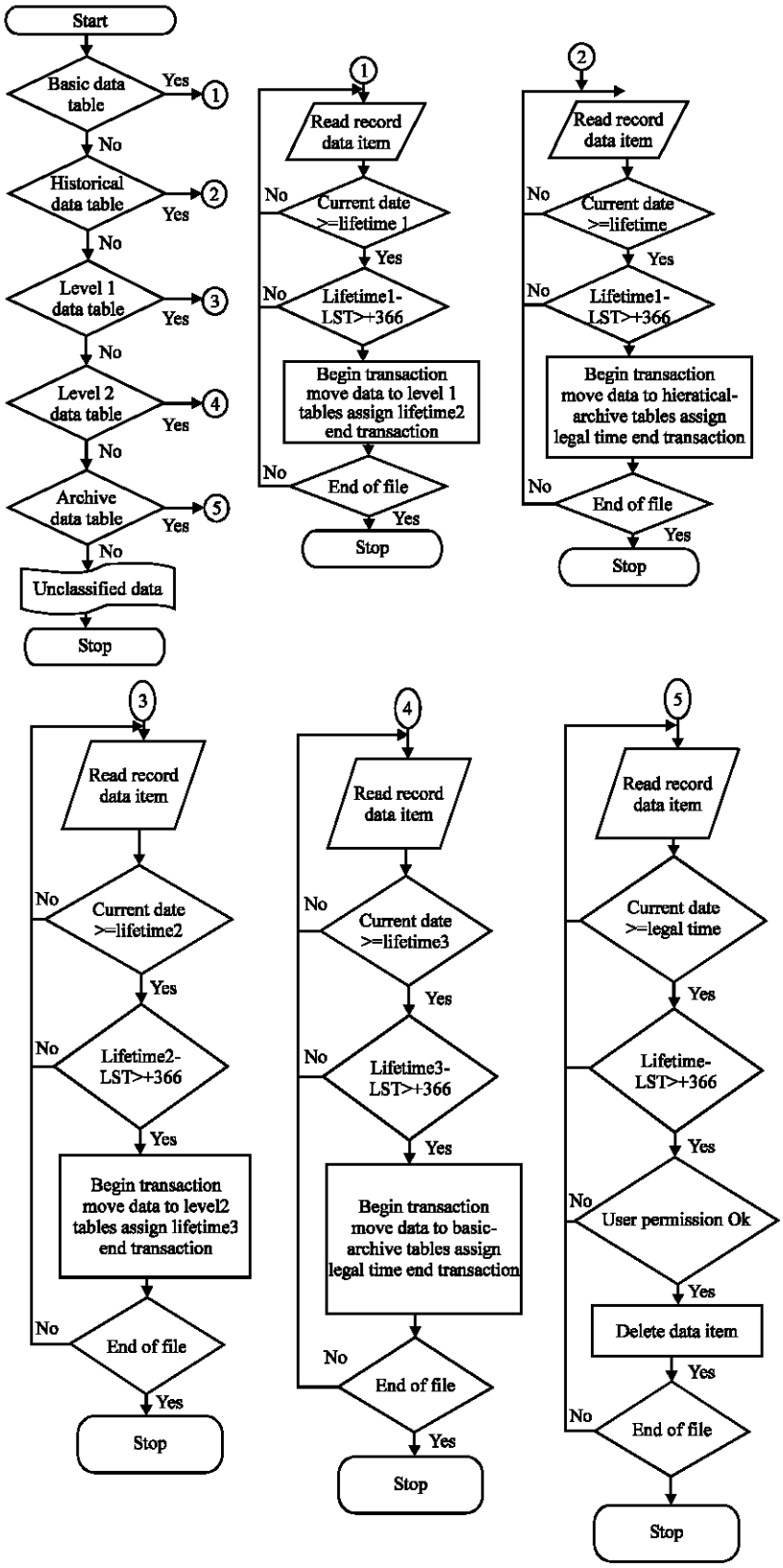

Fig. 1: Flow chart for data transfer algorithm

Algorithm:

```
      main()
       {
      If basic data class table
       {cal basic-class-transfer}
  else if historical data class table
   {call archive-class-transfer }
   else if level1 data class table
   {call level1-class-transfer }
   else if level2 data class table
    {call evel2-class-transfer }
   else archive data class table
   {call archive-handling}}
Basic-class-transfer()
{
If the current date > = the lifetime1-date
    {    If the lifetime1-date- last-access-time >= 366
begin transaction
get exclusive lock on data item
copy data item to level1 table
assign lifetime2-date to data item
delete data item from basic-class table
end transaction
 }
}
Historical-class-transfer()
{
If the current date > = the life-time1-date
    { If the life-time1-date-last access time > = 366
Begin transaction
Get exclusive lock on data item
Copy data item to historical-data archive table
Assign legal-period to data item
Delete data item from historical-data-class table
End transaction
}
}
Level1-class-transfer()
  {
If the current date >= the lifetime2-date
     { If the lifetime2-date-last-access-time > = 366
Begin transaction
Get exclusive lock on data item
Copy data item to level2 table
Assign lifetime3 to data item
Delete data item from level1-class table
End transaction
    }
 }
Level2-class-transfer()
{
If the current date > = the lifetime3-date
    { If the lifetime3-date-last-access-time > = 366
Begin transaction
Get exclusive lock on data item
Copy data item to basic-archive-class table
Assign legal-time to data item
Delete data item from level2-class table
End transaction
    }
  }
Archive-handling()
{
If the current date > = the legal-time-date
    {  If the legal-time-date last-access-time > = 366
 Request user permission for deletion
 If user accept deletion
   { Delete data items from archive-class table}
  }
}
```

RESULTS AND DISCUSSION

Experimental evaluation: The purpose of this study is to evaluate the performance of query processing when a table size is reduced by distributing its data to the levels tables and archive or remove old data from the system. We started by a database that includes a basic data table which contains 800000 records, then we distributed data into levels tables and remove old data so the number of records is decreased to 400000 records and the size of basic data table is reduced. We repeated the process by reducing the table size to the half to contain 200000 records and then 100000 records. The queries are used to retrieve or update the same 1000 records that are homogeneously distributed in the basic table. We measured the performance of the data retrieval and the data update because these processes don't change the number of records in a table. We used Microsoft SQL server 2008 and Idera software (SQL Check and SQL Statistics Aggregator). In order to measure the effect of hardware speed on the query performance we performed the tests in two different machines. The first machine is Intel Core I7 with processors speed 2.10 GHz and 8 GB RAM, the second machine is Intel Core 2 Duo with processors speed 1.6 GHz and 1 GB RAM. We performed the tests many times and excluded the outliers to give more accurate results. Our concern is the CPU time that the system is taken to execute the retrieval or the update queries. The data retrieval CPU time results from the first machine were summarized in Table 1 and Fig. 2 shows a graph representation for the results.

From Table 1, we found that when the number of records is decreased the CPU average time is decreased approximately by the same ratio (in this example 50%). But when the number of records is decreased by 50%, i.e., from 400000-200000 records, the CPU average time is decreased by 31%.

The data update CPU time results from the first machine were summarized in Table 2 and Fig. 3 shows a graph representation for the results.

From Table 2, we found that the CPU average time for the update operation is bigger than the data retrieval operation. When the number of records is decreased the CPU average time is decreased but not by the same ratio and in some cases the CPU average time is increased such

Table 1: Data retrieval

	Data retrieval CPU time (msec)			
Measure	800000 records	400000 records	200000 records	100000 records
Minimum	78.0	31.0	31.0	15.0
Maxium	94.0	47.0	32.0	16.0
Average	89.1	45.3	31.2	15.6

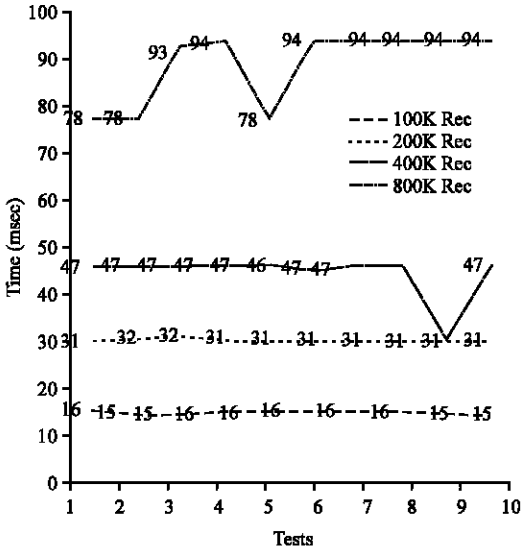

Fig. 2: Data retrieval CPU time (msec)

Table 2: Data update

	Data retrieval CPU time (msec)			
Measure	800000 records	400000 records	200000 records	100000 records
Minimum	93.0	47.0	62.0	31.0
Maxium	114.0	63.0	63.0	47.0
Average	98.9	53.2	62.3	42.3

Table 3: Machine 2 data retrieval

	Data retrieval CPU time (msec)			
Measure	800000 records	400000 records	200000 records	100000 records
Minimum	202.0	93.0	62.0	32.0
Maxium	234.0	187.0	109.0	62.0
Average	218.5	127.7	81.2	46.8

as when number of records is decreased from the 400000 records to 200000 records the CPU average time is increased from 53.2-62.3 msec.

The following tables and figures show the results of the tests that performed on the second machine. In general the retrieval time and the update time are bigger than the first machine. The data retrieval CPU time and the update CPU time are increased approximately by 250%.

The data retrieval CPU time results from the second machine were summarized in Table 3 and Fig. 4 shows a graph representation for the results.

From Table 3, we found that when the number of records is decreased the data retrieval CPU average time is decreased approximately by the same ratio (in this example 50%). But when the number of records is decreased by 50%, i.e., from 400000-200000 records, the CPU average time is decreased by 36%.

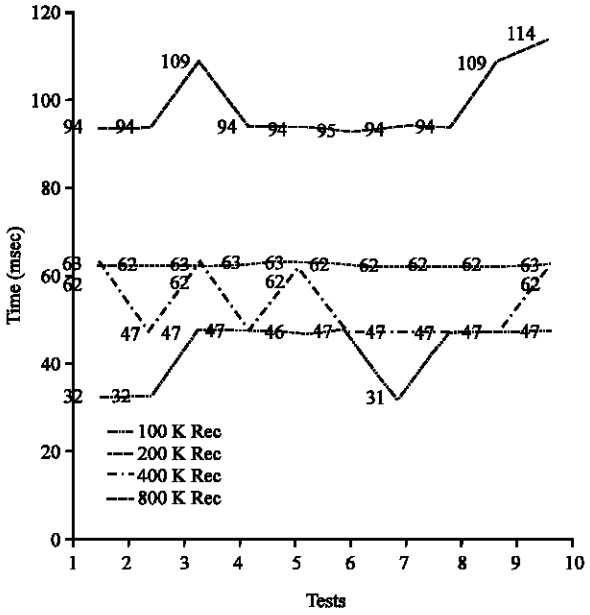

Fig. 3: Data update CPU time (msec)

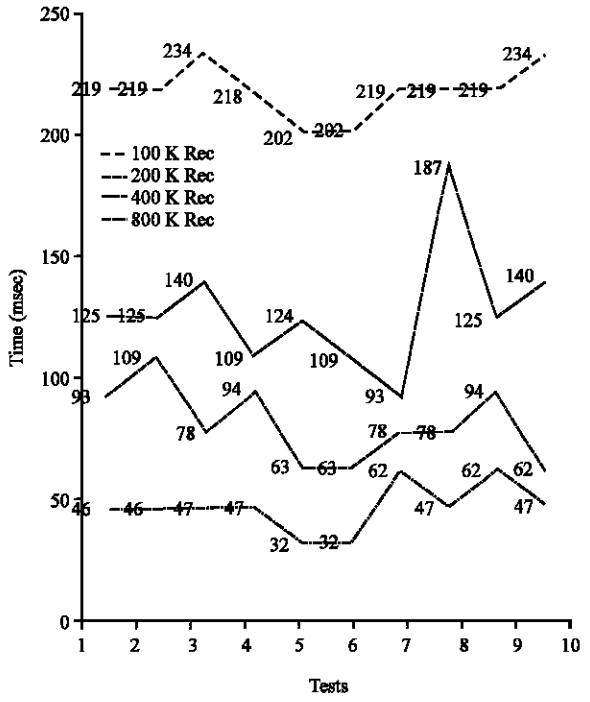

Fig. 4: Second machine data retrieval CPU time (msec)

The data update CPU time results from the second machine were summarized in Table 4 and Fig. 5 shows a graph representation for the results. From Table 4, we found that the CPU average time for the update operation is bigger than the data retrieval operation. When the number of records is decreased the CPU average time is

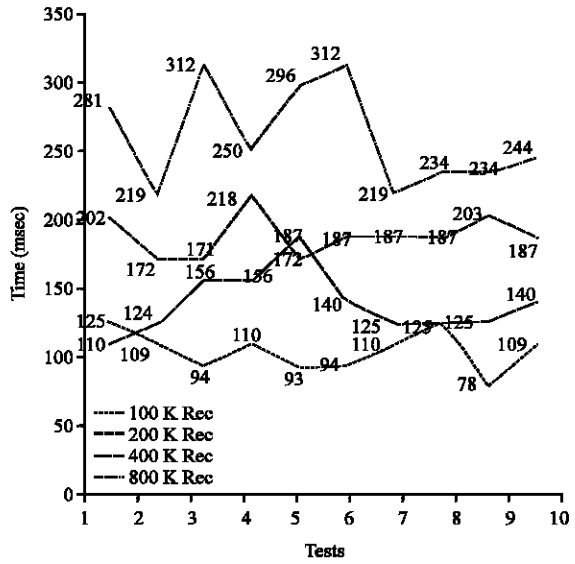

Fig. 5: Second machine data update CPU time (msec)

Table 4: Second machine data

	Data retrieval CPU time (msec)			
Measure	800000 records	400000 records	200000 records	100000 records
Minimum	219.0	110.0	171.0	78.0
Maxium	312.0	187.0	218.0	125.0
Average	260.1	138.6	188.6	104.7

decreased approximately by the same ratio. Also, as in machine 1 in some cases the CPU average time is increased when number of records is decreased such as when the number of records is decreased from the 400000-200000 records the CPU average time is increased from 138.6-188.6 msec. From the previous tests, we found that decreasing the number of records in a table will decrease the data retrieval CPU time approximately by the same ratio which increase the performance of the system. For update operations in most cases the CPU time is also decreased.

CONCLUSION

Everything on the system should have a lifetime such as internet web pages, documents on the cloud, records in a database, etc. In this study, we classified transaction data into two classes basic data class and historical data class. We proposed new properties for data item and an algorithm for data management that help in data analysis and transferring data among system component's until it removed from the system. We implemented a prototype for the algorithm and measured data retrieval performance, we found that by reducing a table size the data retrieval performance is increased approximately by the same ratio.

SUGGESTIONS

In future work, we will investigate how to apply the data transfer algorithm on unstructured data item such web pages and cloud documents to improve the web data retrieval performance.

REFERENCES

Agrawal, S., V. Narasayya and B. Yang, 2004. Integrating vertical and horizontal partitioning into automated physical database design. Proceedings of the ACM SIGMOD International Conference on Management of Data, June 13-18, 2004, Paris, France, pp: 359-370.

Alsultanny, Y., 2010. Database management and partitioning to improve database processing performance. J. Database Marketing Customer Strategy Manage., 17: 271-276.

Anonymous, 2005. Query optimization in oracle database 10g release 2: An oracle white paper. Oracle Corporation, Redwood City, California.

Bertino, E., B.C. Ooi, R.S. Davis, K.L. Tan and B.J. Zobel *et al*., 1997. Indexing Techniques for Advanced Database Systems. Kluwer Academic Publisher, Dordrecht, Netherlands, ISBN: 9780792399858, Pages: 250.

Chu, F., J.Y. Halpern and P. Seshadri, 1999. Least expected cost query optimization: An exercise in utility. Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, May 31-June 03, 1999, ACM, Philadelphia, Pennsylvania, USA., ISBN: 1-58113-062-7, pp: 138-147.

Connolly, T.M. and C.E. Begg, 2005. Database Systems: A Practical Approach to Design, Implementation and Management. Pearson, London, England, UK.,.

Geng, L., 2015. Optimization of massive data retrieval performance based B-tree index. Proceedings of the International Conference on Automation, Mechanical Control and Computational Engineering (AMCCE), October 24-25, 2015, Atlantis Press, Changsha, China, ISBN: 9781510803084, pp: 352-357.

Gupta, S., G.S. Tandel and U. Pandey, 2015. A survey on query processing and optimization in relational database management system. Intl. J. Latest Trends Eng. Technol., 6: 4094-4095.

Haryani, N. and D. Jagli, 2014. Dynamic method for load balancing in cloud computing. IOSR. J. Comput. Eng., 16: 23-28.

Mamun, M.A.A., M. Hanif, M.R. Uddin, T. Ahmed and M.M. Islam, 2012. A new compression based index structure for efficient information retrieval. Intl. J. Sci. Technol., 2: 10-14.

Manning, C.D., P. Raghavan and H. Schutze, 2008. Introduction to Information Retrieval. 1st Edn., Cambridge University Press, Cambridge, England, UK., ISBN-13:978-0521865715, Pages: 506.

O'neil, P. and E. O'nell, 2001. Database Principles Programming Performance. Morgan Kaufmann Publisher, Burlington, Massachusetts, ISBN: 9781558604384, Pages: 870.

Sint, R., S. Schaffert, S. Stroka and R. Ferstl, 2009. Combining unstructured, fully structured and semi-structured information in semantic wikis. Proceedings of the 4th and 6th Joint Conference on Semantic Wiki (SemWiki 2009) and European Semantic Web (ESWC 2009), June 1, 2009, Morgan Kaufmann Publishers, Hersonissos, Greece, pp: 1-15.

Talmage, R., 2009. Partitioned Table and Index Strategies using SQL Server 2008. MSDN Publisher, Woburn, Massachusetts,.

Yates, A.B. and B.A.R. Neto, 1999. Modern Information Retrieval. Pearson, London, England, UK., ISBN: 978-81-317-0977-1, Pages: 322.