Technical Article

Database management and partitioning to improve database processing performance

Received (in revised form): 6th July 2010

Yas Alsultanny

was an assistant professor at Baghdad University, and the head of the Remote Sensing department at the scientific research council in Baghdad from 1997 to 2000. From 2000 he was an associate professor at the Applied Science University, in Jordan, and in 2007 he became an associate professor at the Arabian Gulf University, in Bahrain. He has spoken at many conferences.

ABSTRACT The technologies of searching methods are very important in database management systems, in that they must be a simple and practical way of managing problems in database systems. These methods must allow databases to be more easily deployed with optimal performance. The vertical and horizontal database partitioning are used to improve database performance. The horizontal method is more efficient than the vertical method in a large database, and it will be more efficient if distributed in more than one computer. Redundant Arrays of Inexpensive Disks (RAID) is one of the important technologies used in processing database. The proposed method was implemented to get optimal utilization of RAID level 1, by partitioning vertically the original database before storing them on RAID level 1, in order to get faster database retrieving while at the same time keeping the mirror technique of the RAID level 1 without any changes.

Journal of Database Marketing & Customer Strategy Management (2010) 17, 271–276.

doi:10.1057/dbm.2010.14; published online 11 October 2010

Keywords: data partitioning; vertical partitioning; horizontal partitioning; RAID processing

INTRODUCTION

Database technology is having a major effect on the increased use of computers in business, engineering, medicine, law, education and library science. Databases are a form of indispensable, effective information management because they provide a range of services for effective storage and retrieval of data. The most important measure is the time required to perform the basic operations of retrieval. Methods for data storage organization and searching aim to minimize the time spent to retrieve data.

The first general-purpose Database Management System (DBMS) was designed by Charles Bachman at General Electric in the early 1960s. In the late 1960s, IBM developed the Information Management System DBMS. In 1970, Edgar Codd, at IBM, proposed the relational data model based on the relational model. In the 1980s, the SQL was developed as part of IBM's System project. In the 1990s, several vendors (for example, IBM's DB2, Oracle 8 and Informix UDS) extended their systems with the ability to store new data types such as images and text.¹ The

Correspondence:

Yas Alsultanny
College of Graduate Studies,
Arabian Gulf University,
Manama, PO Box 26671,
Kingdom of Bahrain
E-mail: alsultanny@hotmail
.com

primary function of a database is to provide timely and reliable information that supports the daily operations of an organization.²

Knowledge discovery and data mining is an emerging field whose goal is to make sense out of large amounts of collected data.³

The rapid development of Internet applications and more complex and complicated application systems bring a major challenge related to query optimization. The continuous growth of database size could slow down query execution dramatically.⁴

Database system performance is usually measured in terms of query and transaction response time: the major indicator of a system capacity problem. After a database system exhibits a performance problem, the main effort of post-deployment performance tuning is concentrated on the revision of the design of the database and the transactions running against the database.¹

One of the approaches used to evaluate database performance is by designing the database by using queuing networks; this method differs from other methods of database system performance evaluation in that the performance assessment is specifically targeted at the database design, not at the database system software architecture.⁵

DATABASE TIME (DT) MEASURING

Database time would be similar to a MIPS (Million of Instructions per Second) rating, which is used in computer processor evaluation.⁶ In database processing, a query is a question or task a user asks of a database, and the function of the query manager is to turn a user's high-level database access or manipulation command into SQL.² Figure 1 shows that every time an SQL statement is processed, the DBMS must be precompiled, bind utility and then execute the query.

Figure 2 illustrates a simple scenario of a single user submitting a request. User response time is the time interval between the instant of request sent and the instant of response received. The DT involved in the user request is only a portion of that user's response time that is spent inside the database.⁷

For tables with total tuples ranging from 0 to 1000 tuples, there is a direct relation between time and Tuples read (Tr). Tables with 400 tuples have a Tr twice as large, as the 200 tuple base case, tables with 800 tuples have a Tr four times as large, and so on. Tr increases as the total attributes bytes

Figure 1: State SQL processing by DBMS.

Figure 2: Database time from a single user's perspective.

for a tuple goes up. This is also a linear increase, and can be calculated from extrapolation of the T_r values for the 200 and 1000 tuple relation tables. Database transaction to access tables is calculated by;⁵

Total response time of a transaction=

$$\sum_{i=1}^n (\text{total time to wait for access table}_i + \text{total time to access the data of table}_i + \text{total time to return data to the client from table}_i + \text{total time to process procedural statements on the client}) \quad (1)$$

where tables 1, 2, ..., n are the tables accessed by the transaction.

Compression is typically used for databases that have grown large enough to create a strain on system storage capacity. Many database management systems provide support to a limited extent for users to store data in compressed form.⁸

Good compression techniques allowed transferring more data for a given bandwidth. It also allowed better utilization of bandwidth for data transfer. A large number of databases exist that do not require frequent updates, and need to focus more on storage costs and transfer costs.⁹

Database compression is widely used in database management to save storage space and network bandwidth. The advantageous effects of data compression on Input/Output performance in database systems are rather obvious, that is, its effects on disk space, bandwidth and throughput.¹⁰

People had to wait a long time for data transfers. When sitting at the computer, waiting for a web page to come in or for a file to download, we naturally feel that anything longer than a few seconds is a long time to wait. Using data compression is useful in this situation.¹¹

Compression is a heavily used technique in many of today's computer systems. To name just a few applications, compression is used for audio, image and video data in

multimedia systems, to carry out backups. Data compression is the process of encoding a given data set by applying some algorithms to produce output that requires fewer bytes than required by the original data.¹²

DATABASE PARTITIONING

System performance is one of the problems where a significant amount of query processing time is spent on full scans of large relational tables. Partitioning of the relational tables either through vertical or horizontal partitioning or a combination of both is a reliable way of solving this problem.¹³

Horizontal partitioning of database

Horizontal database partitioning breaks down a relational table into smaller pieces, called partitions, according to the range of values of a given attribute or combination of attributes, called a partition key. Horizontal partitioning reduces the total number of read operations needed to access the rows determined by the values of the partition key provided in a query. Figure 3 shows how horizontal partitioning works, depending on one of the attributes that is the *year*.

It has four values: 2009, 2008, 2007 and 2006. Therefore, for queries only the student information retrieving to each year will be accessed.

Vertical partitioning of database

A database system quite frequently has to access large amounts of data in order to retrieve or update a relatively small number of values determined by a user. This is mainly due to the average row length being significantly greater than the amount of data retrieved or modified in a row. A typical example is a projection of a single attribute of a relational table whose schema consists of many attributes with the values consuming a larger amount of storage.

Vertical partitioning splits a relational table into a number of pieces, also called

Figure 3: Principle of horizontal partitioning.

partitions, and replicates a primary key in each partition. This reduces the average row length, and in consequence minimizes the total number of read and write operations.

Figure 4 shows an example of the vertical partitioning principle. When scanning the student table to retrieve basic student information, other non-relevant student information such as *Blood* and *social* will also appear. In such a situation, we can partition the student table into two parts: *student1* for basic information, *student2* for advanced information. Then the queries need only the student information and this

can be improved by avoiding access to the other student information.

BEST UTILIZATION OF RAID

A major advance in secondary storage technology is represented by the development of RAID, which originally stood for Redundant Arrays of Inexpensive Disks. Lately, the 'I' in RAID is said to stand for Independent.¹⁴

Different RAID organizations were defined based on different combinations of the two factors of granularity of data interleaving (striping) and pattern used to compute redundant information. The levels

Figure 4: Principle of vertical partitioning.

Figure 5: Suggested RAID utilization.

of RAID are from 0 to 6. Our concern is with RAID level 1, which uses mirrored disks. The original database (*Student*) was partitioned into two lists; *Student 1* and *Student 2* in order to speed up data retrieval. Where the storage media formatted into two parts, the upper data *Student1* will be stored in the upper parts of storage media of *Group 1*, and a copy of the same data will be stored in the lower parts of storage media in *Group 2* (see Figure 5).

For list *Student 2*, the data will be stored in the upper parts of *Group 2*, and the copy of the same data will be stored in the lower parts of storage media in *Group 1*.

By this organization, the RAID level 1 technology will have two functions; the first function is partitioning vertically the original database into two lists to improve database retrieving: the first list will be stored at the upper parts of storage media of *Group 1* and at the lower parts of storage media of *Group 2*, the second list will be stored at the lower parts of storage media of *Group 1* and at the upper parts of storage media of *Group 2* (see Figure 5). The second function of RAID level 1 is database mirroring, which is still keeping the same function.

CONCLUSION

Database organization is very important because it helps to reduce the consumption of resources such as hard disk space and minimize data retrieval time. Data organization technique has important application in the areas of data transmission and data storage. Many data processing applications require storage of large volumes of data, and the number of such applications is constantly increasing as the use of computers extends to new disciplines.

The vertical and horizontal partitioning methods can both be used to speed up database retrieving. RAID technology was originally used to improve database reliability; the proposed method of using

RAID level 1 with vertical partitioning, to keep the reliability and improve the database retrieval.

REFERENCES

- 1 Ramakrishnan, R. and Gehrke, J. (2004) *Database Management Systems*. Boston, MA: McGraw-Hill.
- 2 Yeung, A. and Hall, G. (2007) *Spatial Database Systems: Design, Implementation and Project Management*. Dordrecht, The Netherlands: Springer.
- 3 Zaki, M.J., Parthasarathy, S., Ogihara, M. and Li, W. (1997) New algorithms for fast discovery of association rules. In: Heckerman D, Mannila H, Pregoibon D and Uthurusamy R. (eds.) *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, 14–17 August 1997; Newport Beach, Menlo Park: AAAI Press, pp. 283–286. <http://www.aaai.org/Papers/KDD/1997/KDD97-060.pdf>.
- 4 Zhenjie, L. (2007) Adaptive reorganization of database structures through dynamic vertical partitioning of relational tables. Unpublished Master thesis of computer science from the University of Wollongong.
- 5 Osman, R., Awan, I. and Woodward, M. (2008) *Queuing networks for the performance evaluation of database designs*, *Electronic Notes in Theoretical Computer Science*, Vol. 232, Amsterdam, the Netherlands: Elsevier Science Publishers B.V., pp. 172–183.
- 6 Hennessy, J. and Patterson, D. (2003) *Computer Architecture: A Quantitative Approach*, 3rd edn. San Francisco, CA: Morgan Kaufmann Publishers.
- 7 Dias, K., Ramacher, M., Shaft, U., Venkataramani, V. and Wood, G. (2005) *Automatic Performance Diagnosis and Tuning in Oracle*, *Proceedings of the 2005 CIDR Conference*, CA, USA, pp. 84–94.
- 8 Ray, G., Haritsa, J. and Seshadri, S. (1995) *Database Compression: A Performance Enhancement Tool*, *Proceedings of the 7th International Conference on Management of Data*, New York: ACM.
- 9 Baid, A. and Krishnan, S. (1995) *Binary Encoded Attribute-pairing Technique for Database*, *Proceedings of the 11th International Conference*, New York, pp. 540–549.
- 10 Chen, Z., Gehrke, J. and Korn, F. (2001) *Query Optimization in Compressed Database Systems*, *Proceedings of the SIGMOD International Conference on Management of Data*, Vol. 30, No. 2, New York: ACM, pp. 271–282.
- 11 Salomon, D. (2004) *Data Compression: The Complete Reference*, 3rd edn. London: Springer.
- 12 Mengyi, I. (2006) *Fundamental Data Compression*, 12th edn. Burlington, MA: Butterworth-Heinemann.
- 13 Kabra, N. and Dewitt, D. (1998) *Efficient Mid-query Re-optimization of Sub-optimal Query Execution Plans*, *Proceedings ACM Sigmoid International Conference on Management of Data*, ACM Press, pp. 106–117.
- 14 Elmasri, R. and Navathe, S. (2007) *Fundamentals of Database Systems*, 7th edn. Greg Tobin.