ISBN 978-83-60810-22-4

Concurrency Control Techniques for a Multimedia Database System

Cosmin Stoica Spahiu
University of Craiova
Faculty of Automation, Computers and Electronics
Craiova - Dolj, Romania
Email: stoica.cosmin@software.ucv.ro

***Abstract*—The paper presents the concurrency control methods used to provide simultaneous access to databases, in a multimedia relational database management system. This is an original system that integrates methods for extracting visual characteristics (color and texture characteristics) from images and for executing content-based visual queries. In order to accomplish this, it was defined an original new data type called IMAGE. This data type is used to store the images along with the characteristics extracted and other important information. The problems that should be handled refers to process multiple requests and access the same set of data in a concurrent environment. The databases must be protected with a synchronization algorithm to ensure that the information doesn't get corrupted when multiple clients' requests access concurrently the same set of data.**

I. Introduction

THE VISUAL data along with other types of multimedia information is very complex. It needs a lot of storage space and it has to permit querying in order to be retrieved from large images collections. To solve all these demands in an efficient way, a multimedia database management system (MMDBMS) is needed.

Most of the systems existing on the market nowadays offer no support at all, or only partial support for the multimedia data. In [5][6] it is proposed an original solution: it is implemented a multimedia database relational system that includes all the algorithms needed to extract color and texture characteristics from images, store them inside the databases and executing visual-based queries.

It is a TCP/IP client-server system based on the SQL language. The system can be used to manage medium sized databases, containing up to several tens of thousands of records.

One of the major objectives of a every DBMS is to allow multiple users to access simultaneously the databases existing on the system.

If a single-user database system is taken into account, the active user can access all the information in the database without any concern that other users could modify the same set of data, at the same time. However, this kind of system has no use in real world. The biggest advantage is when multiple users, executing multiple operations in the same time, can access the same set of data. That is why in such systems it is vital to exist a module for managing concurrency and data consistency [12]. Its main role is to check that all the operations are executed in such a way that they appear to be executed one at a time (in a serializing mode).

This kind of data sharing implies the existence of specific algorithms for solving conflicts that can appear when the same set of data is accessed. The way these kind of conflicts are solved, depends on the type of the requests taken into account: retrieval or updates.

When discussing about retrieval, no other control is necessary, except the one that is provided by default by the operating system. This is due to the fact that the physical access to disk can be done only in a sequential manner. As long as no user makes any update to the data, it is not important in which order the data is accessed. The operating system will manage all the requests and will specify in each case the concurrency solving algorithm needed in order to minimize the total response time.

If the users' requests imply updating the data (insert, update or delete), it is necessary to have specific algorithms for dealing with concurrent writings. The problems that could appear refer to the cases when several users send requests for modifying the same set of data, or some of them send updating requests and the others send retrieval requests. The simplest way to solve these problems is to block the access to the database while each request is solved. In this way, each request is solved sequentially, but the performances of the system are highly affected.

The paper has the following structure: Section 2 presents the related work, Section 3 presents the MMDBMS and the adopted solution, and Section 4 presents the conclusion and future work in this project.

II. Related Work and Concurrency Methods

The interaction of two or several read/write operations can generate inconsistences into the database and/or non-valid results (the result obtained in a sequential execution can be different than the result of a concurrent execution). These problems might appear if the same set of data is accessed. Depending to the type of operation that is executed, several types of anomalies can be observed:

- loosing an update (write/write conflict): the update of one operation is lost because of an update of another operation which has not taken into account the result of the first one

- improper reading (write/read conflict): the data is read before the first operation has finished the update and modified the data
- non-repeatable reading (read/write conflict): It is also called "dirty reading". It is met when two consecutive readings of the same data return different results.

The simplest algorithm that can be used to avoid the problems presented above is to restrict the access to database to one operation at a time. The access of all the other operations will be restricted. Two methods are needed in order to implement this: lock(data) and unlock(data).

The types of locks that can be used are: SHARED and EXCLUSIVE locks [1][2]. An exclusive lock is the commonly used locking strategy that provides an exclusive control on the data set. A shared lock can be acquired when a command wants only to read a data set, and not to modify it. If it has already acquired a shared lock on the data set, no other operations can acquire an exclusive lock on that data [1][2].

In order to manage the active locks, a lock manager module should be implemented in order to maintain a list of records for each locked data. The locks will be stored in this list in the order in which they arrive.

It is presented next the strategies used by three well known database management systems for managing multimedia data (images files) and controlling the concurrency: MySQL, Microsoft SQL Server and Oracle Database Server.

A. MySQL Server

Most of the systems existing on the market nowadays offers only partial support for managing multimedia data, or no support at all. This is due to the fact that multimedia data needs a lot of disk space, making databases to become huge even for a relative small number of records [8]. In these cases it is recommended to have a special system file structure for the disc (NTFS recommended) and the free space to be carefully supervised.

MySQL does not contains any dedicated data type or methods for images management[9]. The only data type that can be used is BLOB. A BLOB is a binary large object that stores objects in an unstructured manner. BLOB attributes have no character set. The sorting and comparing operations are based on the numeric values of the bytes.

B. Microsoft SQL Server

MS SQL Server offers two special data types: image and text. Both data types are treated in a similar manner and no supplementary support is offered. The system does not include any methods for extracting visual characteristics from images or for executing special operations.

More than that, MS SQL Server 2008 recommends to void using these data types, as they will be removed in a future version of the system [11].

The multi-user environment is maintained using two concurrency control techniques: Pessimistic and Optimistic concurrency control techniques. Users specify the type of concurrency control by selecting transaction isolation levels for connections or concurrency options on cursors [11].

When the pessimistic concurrency control technique is used, the locks prevents users from modifying data in a way that could affect the other users. After a user performs an action that activates a lock, the other users cannot perform actions that would conflict with that lock, until it is deactivated by the owner. This is called pessimistic control because it is mainly used in environments where it is high contention for data, and where the cost of protecting data with locks is less than the cost of rolling back transactions when concurrency conflicts occur [10][11].

When optimistic concurrency control technique is used, users do not lock data when they read it. There are two ways for this method to be implemented: optimistic with values and optimistic with row versioning [10].

The optimistic with values method is used when there is only a slight chance that another user to update a row in the interval between when a cursor is opened and when the row is updated. When a user updates data, the system checks to see if another user changed the data after it was read. If another user updated the data, an error is raised. Typically, the user receiving the error rolls back the transaction and starts over. This method is mainly used in environments where there is low contention for data, and where the cost of occasionally rolling back a transaction is lower than the cost of locking data when read [10][11]. In this case the user can deal with occasional error indicating another user has modified the row.

The optimistic with versioning method is based on row versioning. The underlying table must have a version identifier of some type that the server can use to determine whether the row has been changed after it was read into the cursor. In SQL Server, that capability is provided by the timestamp data type, which is a binary number that indicates the relative sequence of modifications in a database. Each time a row with a timestamp column is modified in any way, SQL Server stores the current timestamp. If a table has a timestamp column, then the timestamps are taken down to the row level. The server can then compare the current timestamp value of a row with the timestamp value that was stored when the row was last fetched to determine whether the row has been updated. The server does not have to compare the values in all columns, only the timestamp column. If an application requests optimistic concurrency with row versioning on a table that does not have a timestamp column, the cursor defaults to values-based optimistic concurrency control [10] [11].

C. Oracle Database

The Oracle Database System provides the full solution for efficient management and retrieval of multimedia data (images, audio, and video), by using the Oracle Multimedia feature (formerly known as Oracle interMedia) [12].

The images are managed using the ORDImage object data type, which supports the storage, management, and manipulation of images. Each object includes all the attributes, methods, and SQL functions and procedures needed for management.

For concurrency control, The Oracle Database divides the locks types in three main categories [12]:

- DML locks (data locks): used to protect the data. The locks from this category can lock, either the entire table, or only specific rows in the table. Row-Level Locking are used by read committed and serializable transactions. A table lock can be held in any of several modes: row share, row exclusive, share, share row exclusive, and exclusive.
- DDL locks (dictionary locks): are used to protect the structure of objects (e.g.: the structure of the tables)
- Internal locks: are used to protect the datafiles (internal database structures). They are completely automatic and does not need any user interference.

The locks manager can upgrade the locks to a higher level when needed. If a user owns a shared lock (to execute SELECT operations) and at a certain step he executes an UPDATE, the lock will be converted to an exclusive lock [12].

III. Concurrency Management for a Multimedia Database Management System

A. General presentation of the system

The implemented system is a databases management system that can be used both for executing simple text-based queries, and more complex content-based visual queries. The content-based visual queries use the color and texture characteristics that were automatically extracted from images, in order to compute the images similarity [5][6][7][8].

This tool is easy to be used because it respects the SQL standard. It does not need advanced knowledge in informatics and has the advantage of low cost. It is a good alternative compared to a classical database management system, which would need higher costs for server acquisition and for designing the applications that execute content-based retrieval operations [5][6][20].

The MMDBMS permits databases and tables creation, constraints definition, inserting images and alphanumerical information, and executing simple text-based queries or complex content-based queries using color and texture characteristics.

An original element of the system is a new data type that was defined, called IMAGE. This type is used to store both the image itself and the vectors of characteristics (texture and color histogram) [5][6].

Another original aspect is that the system integrates all the algorithms needed to process the images, extract the characteristics and execute retrieval queries based on the content.

When discussing about multimedia data, especially images, it is not important to find an exact match between two images. It is more important to be able to find similar images. There are many algorithms that can be used for processing the images and extracting the color and texture characteristics, but there is not any certain method that can be considered to provide the best results in any situation. The quality of the results depends to the type of images taken into account [21].

Our system is designed to be used mainly in medical domain where the experiments indicated that the best results were obtained using Gabor filters [3][16][17] for texture characteristic and the histogram representation quantized to 166 values, for color characteristic [20][8].

The similarity was computed using Euclidian distance for the texture characteristic and histograms intersection for the color characteristic. The users have the possibility to choose for each executed query what characteristic to be used to compute the similarity: only texture, only color, or both of them (each with an weight of 50%)[5].

B. Concurrency management

The second important aspect of this MMDBMS is that it provides simultaneous access to information for many clients via TCP/IP network. The problems that should be handled refers to process multiple requests and access the same set of data in a concurrent environment.

The system must include a synchronization algorithm to ensure that the information doesn't get corrupted when multiple clients' requests access concurrently the same set of data. However, in most of the cases the information is frequently read and only occasionally written. It is far more efficient to allow all reading requests to be executed simultaneously and only write requests to be executed in an exclusive manner.

The locking mechanism that was chosen for the system is based on L. Lamport's bakery algorithm [2][22][23]. This algorithm was chosen because it offers a good balance between performances and implementation complexity.

There are two types of locks used: shared locks used for reading (e.g.: SELECT) and exclusive locks used for writing (e.g.: INSERT). These types of locks are used only at the table level of granularity. There are not defined row-level locks or others locks at a higher level of granularity.

If a SELECT command is retrieved (that implies reading from database), a read-lock will be enabled on the tables (files) involved in the operation. This lock will be active until the tables (files) will no longer be used. It is a non-exclusive lock, meaning that all other reading requests will be permitted, each of them activating their own read-lock [7].

If an INSERT command or other command that involves writing into database will be received meanwhile, it cannot be executed. No writes are permitted while any read-lock is active. Instead it will be put in a waiting queue for a random period of time. The write operation can be executed only when no other lock is active. After all locks are inactivated for a specific table, the write-lock can be activated. This type of lock is an exclusive one. No other request (read or write) can be accepted while this is active [7].

When an operation activates a lock, it can include one or several tables. If there is no foreign key defined on the requested table, only one table will be locked. If the table includes foreign keys, all the connected tables will be locked using the same type of lock for all of them.

In order to override the critical section when locks are activated or upgraded, it is used the Lamport's bakery synchronization algorithm [23]. This way it is not possible for two different users to lock accidentally the same resources.

When a lock is no longer needed, it will be deactivated directly without using any synchronization algorithm.

The basic idea for the Lamport's bakery algorithm is quite simple. Each user's request receives a serving number when a lock is needed. The holder of the lowest number is the next one that gets access to resources [23]. The implementation of the algorithm is presented next[24]:

```
Algorithm 1. The Bakery Algorithm
waiting[i] <- true;
No[i] <- max(No[0],...,No[n-1])+1;
waiting[i] <-  false;
for j <-  0 ... n-1 do {
     while waiting[j] do nothing;
     while (No[j] != 0) and
            (No[j] < No[i])
                   do nothing;
   }
*ENTER critical section:
       No[i] <- 0;
       * Activate requested lock
```

To implement this algorithm, there are need two lists. There is one entry in each list for every lock request. The first array stores the priority number. The other list contains a boolean value for each request specifying if that request is in line to receive a number.

When a new lock request arrives and needs to be enabled, first it sets its boolean value to true. Then it is assigned the next number available for waiting its turn. After it receives a number, its no longer waiting so it sets its waiting value to false. Next, the lock request goes through the first list and if there is a request with a lower number, or a request that's waiting for a number, it waits until that request is finished or assigned a higher number. After the lock manager traverses the list it searches for the request with the lowest number in order to be served and activate the lock [24].

IV. Conclusion

The paper presented the way concurrency is managed in an original implementation of a multimedia database management system. This system has integrated methods for extracting the color and texture characteristics from images and executing content-based visual queries. In order to accomplish this, it was defined a new data type called IMAGE that is used to store the images along with the extracted characteristics and other important information.

The problems that should be handled are: processing multiple requests and accessing the same set of data simultaneously. The system must include a synchronization algorithm to ensure that the information doesn't get corrupted when multiple clients' requests access concurrently the same set of data. The adopted solution uses a read/write locking mechanism that is based on Lamport's bakery algorithm for entering into critical section and activating the locks.When a lock is activated, the whole table is locked. There are not defined other levels of granularity.

In the future work, the system will include other types of locks defined at the row level. The DBMS will automatically chose what is the best type of lock that should be used for each request in part.

V. Acknowledgement

The support of the The National University Research Council under Grant CNCSIS IDEI 535 is gratefully acknowledged.

References

[1] ITL Education Solutions Limited, *Introduction to Database Systems*, Pearson Education India, 2008.
[2] A.S. Tanenbaum, *Modern Operating Systems (Second Edition)*,Prentice Hall, 2001.
[3] A. Del Bimbo, *Visual Information Retrieval*, Morgan Kaufmann Publishers. San Francisco USA, 2001.
[4] T. Gevers, *Image Search Engines: An Overview*, Emerging Topics in Computer Vision. Prentice Hall, 2004.
[5] C. Stoica Spahiu, *A Multimedia Database Server for information storage and querying*, International Symposium on Multimedia - Applications and Processing (MMAP'09), Vol. 4, pp. 517 - 522, 2009.
[6] C. Stoica Spahiu, L. Stanescu, D.D. Burdescu, and M. Brezovan, *File Storage for a Multimedia Database Server for Image Retrieval*, The Fourth International Multi-Conference on Computing in the Global Information Technology (ICCGI 2009), pp.35-40, 2009.
[7] C. Stoica Spahiu, *Testing the performances of a multimedia database system*, International Journal of Computer Science and Applications (IJCSA), 2011 (to be published).
[8] C. Stoica Spahiu, L. Stanescu, D.D. Burdescu, M. Brezovan, *Visual Interface for Content-based Query in Image Databases*, Intelligent Interactive Multimedia Systems and Services (KES 2009), Vol. 226, pp. 231 - 240, 2009.
[9] *MySQL 5.0 Reference Manual* http://dev.mysql.com/doc/refman/5.0/en/index.html
[10] K. Delaney, F. Guerrero, *Database Concurrency and Row Level Versioning in SQL Server 2005* http://msdn.microsoft.com/en-us/library/cc917674.aspx
[11] *SQL Server 2008 Books Online*, January 2009 http://msdn.microsoft.com/en-us/library/ms187875.aspx
[12] *Oracle9i Database Online Documentation (Release 2 (9.2))* http://download.oracle.com/docs/cd/B10501_01/server.920/a96524/c21cnsis.htm
[13] C. Carson, S. Belongie, H. Greenspan, J. Malik, *Blobworld: Image segmentation using expectation-maximization and its application to image querying*, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 24(8), pp. 1026-1038, 2002.
[14] D. Comaniciu, P. Meer, *Robust analysis of feature spaces: color image segmentation*, IEEE Conference on Computer Vision and Pattern Recognition, pp. 750-755, 2003.
[15] M. Cooper, *The tractibility of segmentation and scene analysis*, International Journal of Computer Vision, Vol. 30(1), pp. 27-42, 1998.
[16] J.J. Henriksen, *3D surface tracking and approximation using Gabor filters*, 2007.
[17] M. Lindenbaum, R. Sandler, *Gabor Filter Analysis for Texture Segmentation*, Technical Report CIS-2005-05, Technion - Computer Science Department, 2005.
[18] M. M. Martnez, *An introduction to content-based information retrieval by normalized compression distance*, Master Thesis, 2009.
[19] R. Yong, H. Thomas, S. Chang, *Image Retrieval: Current Techniques, Promising Directions, and Open Issues, Visual Communication and Image Representation*, Vol. 10, pp. 39-62, 1999.
[20] L. Stanescu, D.D. Burdescu, M. Brezovan, C. Stoica Spahiu, and A. Ion, *A New Software Tool For Managing and Querying the Personal Medical Digital Imagery*, International Conference on Health Informatics, pp. 199-204, 2009.
[21] H. Tamura, T. Mori and T. Yamawaki, *Textural Features Corresponding to Visual Perception* , SMC, Vol. 8, pp. 460-473, 1978.
[22] G.L. Peterson, *Myths About the Mutual Exclusion Problem*, Information Processing Letters, Vol. 12(3), pp. 115116, 1981.
[23] L. Lamport, *A New Solution of Dijkstra's Concurrent Programming Problem*, Communications of the ACM 17(8), pp. 453-455, 1974.
[24] J. Emerson, *Thread Synchronization and Critical Section Problem*, http://www.jonemerson.net/dev/articles/ThreadSynchronizationAndSemaphores.html