Published in: Proc. of the 11th Int. Conf. on Database and Expert Systems Applications (DEXA 2000), 4.-8. Sept. 2000, London, UK, Springer, LNCS 1873, 2000, pp. 203-211

Awareness in Interactive Database Applications

Günther Specht[1], Patrick Freiherr Harsdorf von Enderndorf[2], Thomas Kahabka[3], and Franz Peterander[2]

[1] Institut für Informatik
Technische Universität München
D–80290 München, Germany

[2] Institut für Pädagogische Psychologie und Empirische Pädagogik
Ludwig Maximilian Universität München
D–80802 München, Germany

[3] Exolution GmbH
Barerstr. 60, D–80799 München, Germany

`specht@in.tum.de, patrick@harsdorf.de, kahabka@exolution.de, peterander@lrz.uni-muenchen.de`

Abstract. This paper proposes a way to overcome locking problems in interactive database applications by using awareness concepts. Parallel long running editing sessions in interactive database applications often cause locking conflicts. The occurrence of conflicts can be drastically decreased by giving users means to be aware of each other and to communicate. This paper explains how this is done in database applications for the social area, which are prone to locking conflicts due to their use of relations with a very large number of attributes and long running transactions. Additionally it shows how high scalability can be achieved with the help of dynamic partitioning schemes. We present MAL, a development system for interactive database applications, which allows to develop applications that automatically include awareness-based locking in a network environment.

1 Introduction

In multiuser database applications used for long-running editing sessions severe locking problems are often encountered. The probability of these conflicts increases with the number of attributes per tuple and the duration of editing times.

We faced such problems in the development of applications for *early childhood intervention centers*, institutions providing help to children with development problems. Such database applications assisting in social diagnostics use relations containing up to 3000 attributes, all functionally dependent on the patient-identifier. Since editing sessions for diagnostic reports often take over one hour, locking conflicts are a problem.

We offer a new awareness-based approach to overcome these locking problems and have implemented it in the MAL system, a database application development system successfully used in the social area. MAL was developed by the MAL Team, an interdisciplinary research group.

The rest of the paper is organized as follows. In the next section we introduce the awareness-based locking approach of MAL. In section 3 we present techniques to maintain high scalability. Section 4 is devoted to a closer description of the MAL system. Conclusions are discussed in section 5.

2 Locking and Awareness

2.1 The Gallery

Instead of using strict isolation and anonymity between different users the concept of making users aware of each other gives them a chance to cooperate. The central element of the awareness interface is a floating window called "the gallery" (figure 1). It informs users about which of their co-workers are currently logged in, and which co-workers are using the same record as they do. It can also be used to communicate with others by exchanging text messages.

Each co-worker is identified by a unique color in the gallery which is used to draw the border of his or her image. This color is used to identify screen elements locked by different users working on the same record.

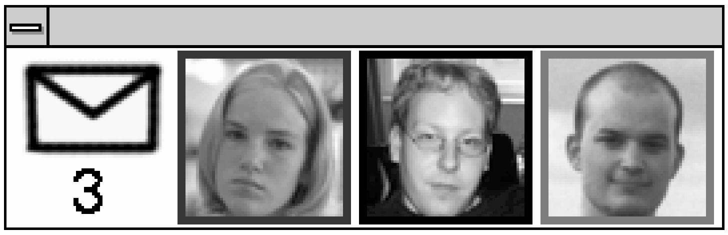


Fig. 1. The "Gallery"

Making users aware of the virtual presence of their co-workers not only facilitates work but also creates a feeling of community. The MAL system uses pictures instead of names – which would consume less screen real estate – , because pictures are recognized faster than words and can be perceived even if not focused directly [7]. Also in a cooperative multiuser environment subjective satisfaction, inclination to cooperate and efficiency increase as the users establish a mental connection between the virtual representation of their co-workers and the actual persons [1]. For that effect, pictures – being more personal and direct – are more appropriate [6].

2.2 Locking and Awareness

In MAL applications locking has multiple functions: It is not just seen as a necessary evil to gain consistency and prevent data loss. Rather, the visual representation of locks represents awareness information about the presence of co-workers.

While classical database applications are based on ACID transactions using strict isolation and anonymity between different users, MAL's awareness concept surrenders the anonymity of locks. Whenever a user encounters a locked attribute its value is visible along with the information about who is working on that attribute. In addition we surrender the concept of isolation. If a user stays on a user interface (UI) page with locked attributes, he can see the values change with a certain delay as his team-mate modifies them. Thus we allow informed dirty reads. The person currently working on that data can recognize the virtual presence of his colleague, who just "stepped into the room". The gallery is extended and the fields in the form are colored correspondingly. Two MAL users working on the same database record are very close to what is a shared desktop in *relaxed WYSIWIS (what you see is what i see)* mode [4]. The MAL concept takes advantage of the fact that people tend to transfer behaviour patterns and emotions concerning physical rooms and spaces over to virtual spaces. This encourages a form of natural and intuitive cooperation, comparable to the one existing between athletes sharing training machines. An athlete who is just about to finish his exercises or is just resting will step back as soon as possible to free the machine for an arriving colleague. On the other hand, if he is still using the machine and the other one stays waiting, the former will give some informal information like "just a moment" or "give me ten more minutes".

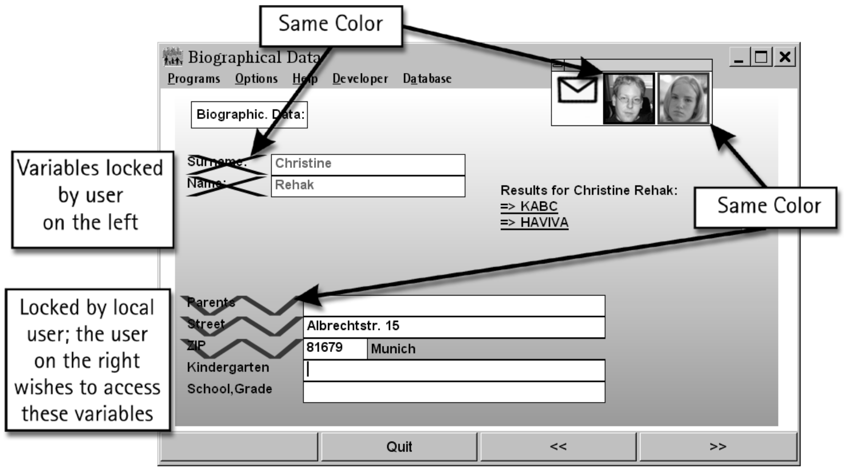
Whenever two MAL users virtually meet in a MAL application, both have the option to initiate a conversation by clicking on the picture of the co-worker. In most occasions awareness of the presence of the other and maybe a personal communication via the gallery will solve a locking problem. A polite and cooperative conduct is encouraged through contextual awareness [2]. Awareness about co-workers helps to reduce frustration and increase the efficiency of cooperative work [3]: "Seeing" co-workers often prevents potential conflicts from occurring, since it is possible for a user to intuitively understand his co-worker's task and to correctly predict their next steps.

2.3 Intuitive User Interface supporting Awareness-based Locking

One of the primary goals in creating the MAL system was to keep the user interface of the MAL applications as intuitive as possible. An interface like that of many multiuser text editors, where a user has to claim temporary ownership over a chunk of data before he can modify it [5] would consume too much time and distract the users from their tasks. Instead we propose a straightforward and intuitive user interface which does not require additional buttons, commands and procedures to remember.

Therefore we introduce the new concept of wish locks, with the intended meaning to hold a (probably dirty) read lock and to wish (and to wait for) a write lock. Wish locks are always immediately obtainable. When a user enters a UI page for editing, the MAL application automatically locks all unlocked attributes on this UI page with write locks and all locked variables with wish locks. In either case the page can be displayed immediately.

If user A enters a UI page containing attributes locked by user B, the MAL system behaves as follows. On user A's side the editing fields of the corresponding attributes are disabled and their labels are crossed out in user B's color. On user B's side the labels of the same editing fields are marked with a "W" ("wish") in user A's color, to inform user B that he is blocking those attributes for user A (figure 2). To keep the user interface simple there are no dedicated buttons or commands for releasing locked attributes. To do this, users "step away" from them, i.e. move to another UI page. Clients always release all wish- and write-locks whenever the user changes from one page to another, even if both pages use the same variables. Users waiting for a lock to be released form a queue (FIFO) and are guaranteed to be granted access in the correct order – independent of network lag. Likewise, if a user quickly moves one screen back and forth while another user is waiting for locks on the the first screen to be released, the access rights are guaranteed to be granted to the second user and not be reclaimed by the first one.


Same Color
Biographical Data
Programs Options Help Developer Database
Biographic. Data:
Variables locked by user on the left
Surname:
Name:
Results for Christine Rehak:
=> KABC
=> HAVIVA
Same Color
Locked by local user; the user on the right wishes to access these variables
Parents
Street
ZIP
Kindergarten
School,Grade
Quit
<<
>>

Surname:	Christine
Name:	Rehak
Parents	
Street	Albrechtstr. 15
ZIP	81679 Munich
Kindergarten	
School,Grade	

Fig. 2. User Interface

2.4 Implementation

For the implementation of our awareness-based locking concept MAL uses its own locking scheme on top of the locking mechanism of the underlying database. This is done with special database tables containing locking information and messages sent directly from client to client. Notification mechanisms are tightly integrated into the MAL system. If a user changes the value of an attribute the change of this attribute will not only be transmitted to all other MAL clients working on the same record, but also all results that depend on this value will be recomputed. In this way displayed results – even diagrams – react in near real-time to changes made by other users.

The relations used by MAL applications often contain very large numbers of attributes. On the database server these relations get transformed to name-value pairs, mapping every attribute of the relation to a row in the database. This allows the minimum lockable unit to be a single attribute.

For optimal efficiency the database server should support row level locking. If a database supports only page level locking it may occur that MAL clients cannot get write access to attributes even though no other MAL client deliberately locked them, since they reside on the same database page as a locked attribute. Since locking information is held in MAL in separate lock tables, physical database locks are only held very shortly. In this case short delays can result.

MAL applications are by their nature deadlock free. Input elements in an MAL application are always tied to exactly one attribute. Cyclic deadlocks on different UI pages cannot occur, since all write- and wish-locks of one UI page are claimed in an atomic step (preclaiming) and completely released on each page change. Thus locks cannot be subsequently demanded. Lifelocks are avoided since wish locks form a FIFO queue.

3 Scalability

Since MAL-based applications may be used in larger institutions such as hospitals, the network database subsystem was built with scalability in mind. A large part of the awareness functionality and the propagation of changed values is done through direct client to client connections, thereby reducing load on the central server. Nevertheless the tables containing locking information and the tables containing the actual data reside on the database server and are shared by all clients. These tables may become heavily used and could become potential hotspots. Therefore these tables can be horizontally partitioned, reducing lock contention and increasing performance.

For the partitioning to have optimal benefit, the goal is not to have partitions of equal size, but to have the amount of insert and delete operations fairly equally distributed. Therefore the number of insert and delete operations is counted for each attribute.

The balancing mechanisms of the MAL system are as follows:

- New attributes are always created in the least-used partition.
- The first MAL client to connect to a given MAL database on a given day moves attributes from more heavily used partitions to lesser-used ones to balance them. This takes only fractions of a second and goes unnoticed by the user. Since MAL databases only change incrementally, the balance

between partitions does not deteriorate rapidly and thus balancing – akin to file system defragmentation – does not have to be performed often. As no data needs to be moved among partitions while the databases are in use, the clients can cache the information about which attributes resides in which partition locally. The initial number of partitions per table is freely configurable. The system administrator can instruct the MAL system to add more partitions to an existing table at any time. Data will then be moved automatically from the original partitions to the new ones until once again all partitions are balanced. On networks with several database servers, each partition can reside on a different database server, significantly increasing performance. Since real-time bookkeeping of the insert and delete operations could become a performance issue in itself, every MAL client logs its insert and delete operations and updates the information just once per session.

The effect of partitioning on performance (figure 3) is as follows: The performance at first increases in a near logarithmic fashion with each added partition then reaches a plateau. Finally as more and more partitions are added, a slow linear decrease in performance is measured, which can be attributed to administrative overhead.

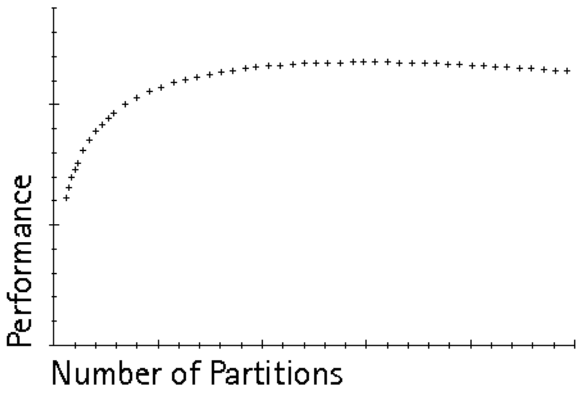


Fig. 3. Effect of partitioning on Performace

Since the effectiveness of partitioning depends on the number of occurring conflicts, the optimal number of partitions varies with the number of clients typically connecting, access patterns of the users, the underlying hardware, and the database management system.

Partitioning significantly reduces lock contention and allows a large number of MAL clients to share a common database and still maintain high response times.

4 Description of the MAL System

MAL was initially created to develop applications for *early childhood intervention centers*, and its major use is still in that area.

4.1 Early Childhood Intervention

Early childhood intervention is a service provided to children and their families in many countries. The goal is to detect potential problems in the development of children at an early stage. By administering special care to these children it is then possible to prevent these development problems from occurring, remedy existing problems or at least lessen their impact. In early childhood intervention centers, specialists of different medical and therapeutical professions are working together. MAL applications are mostly used to assist in diagnosis, and to pool and structure knowledge of different experts over several sessions and to help for example in the creation of reports. Additionally many standardized tests like KABC have been implemented as MAL applications.

4.2 Database Application Development using MAL

The MAL applications are created in a different setting than those in the classic commercial, administrative or technical fields. We decided to build a new software development system because of the different circumstances in the social area:

– It is not possible for the users to give in advance a clean definition of what a program that is developed for them shall do: The software development process is highly iterative with a lot of trial-and-error between the developers and the users. This leads to a high change frequency in the application's data scheme and program logic.
– The software developer has to understand the needs of the users. He has to be a specialist in the social area to be able to communicate about the wishes of the users: We can not assume the programmer to have classic software-design and development skills. The system has to give the developer as much freedom as he needs but also protect him from faults and inconsistencies that might be easily introduced because of the high change frequency.
– The programs are written for computer-illiterates who might even fear using a computer. The software that is developed with MAL has to be very easy to use and present a consistent user interface. The programs have to allow the user instant access to the data and let him interactively "play" and analyze the stored data. This requires a seamless but generic integration of the data analysis functions with the user interface elements.

To cope with these requirements we decided to create a rapid software development system based on a newly designed language for the definition of the data-processing structure, the form-sets and the form-transition-graph. The MAL language shares many concepts and features with functional languages such as prevention of side effects, lazy evaluation, declarative structure.

To write a MAL program the author has to define the user interface and the data structure with the functional data processing structure. The definition of the user interface is done by designing each page by placing predefined user interface elements on a virtual grid on the screen. Each input element is bound to a attribute in the database, where the value entered by the user is almost immediately stored. Each output element is bound to either a database attribute or a MAL function, whose results it displays. A MAL function is defined by combining values of database attributes or other MAL functions with the help of predefined functions or operators. Database attributes and functions can store or respectively return only values of two types: numbers and texts. There is no way of assigning a value to a database attribute other than binding it to a user interface input element. That means, the value of all functions is at all times a direct function of the user-entered values: the internal state of the program is entirely bound to the values entered by the user. A change of state can only happen by the user altering a value in a user interface input element.

The user interface and the functional structure are internally tightly coupled and kept consistent. That means, if an output-element is on the same form as an input element, and the function that is bound to the output-element uses (directly or indirectly via multiple levels of functions) the database attribute bound to an input element on the same form, the output-element updates automatically whenever the user alters the value of the input element. The programmer only defines the function-structure behind it – the MAL system takes care of screen-updates.

5 Conclusions

In this paper we presented awareness concepts for avoiding locking conflicts in interactive database applications with long running transactions. While classical database applications are based on ACID transactions using strict isolation and anonymity between different users, our awareness concept surrenders isolation and the anonymity of locks. Making concurrent users aware of each other gives them a means to cooperate on a meta level about restricted resources like long-locked tuples. We introduced wish locks to express write wishes to the owner of write locks and to gain intermediate results. An intuitive user interface supporting awareness-based locking was introduced. Scalability can be reached by the use of vertical and dynamic horizontal partitioning. We implemented these concepts in the MAL system, a rapid database application development system. Several applications in the area of *early childhood intervention* using MAL are already in use.

References

1. Elena Rocco, University of Michigan: *Trust Breaks Down in Electronic Contexts but Can Be Repaired by Some Initial Face-to-Face Contact.* Conference on Human

Factors and Computing Systems (CHI), 1998, Los Angeles, pp. 496 – 502

2. Gloria Mark, Ludwig Fuchs, Markus Sohlenkamp: *Supporting Groupware Conventions through Contextual Awareness.* Proceedings of the Fifth European Conference on Computer Supported Cooperative Work (ECSCW), 1997, Lancaster, pp. 184 – 193
3. Carl Gutwin, University of Saskatchewan: *Effects of Awareness Support on Groupware Usability.* Human factors and Computing Systems (CHI), 1998, pp. 511 – 518
4. Johann Schlichter, Michael Koch, Chengmao Xu: *Awareness – The Common Link Between Groupware and Community Support Systems.* Lecture Notes in Computer Science 1519, Community Computing and Support Systems, T. Ishida (ed.), Springer Verlag, 1998, Berlin, pp. 77 – 93
5. Johann Schlichter, Michael Koch, Martin Bürger: *Workspace Awareness for Distributed Teams.* Proc. Coordination Technology for Collaborative Applications - Organizations, Processes, and Agents, Singapore, Lecture Notes on Computer Science 1364, W. Conen, G. Neumann (eds.), Springer Verlag, 1997, Berlin, pp. 199 – 218
6. Carl Gutwin, Saul Greenberg, Mark Roseman: *Supporting Awareness of Others in Groupware.* Conference on Human Factors and Computing Systems (CHI), Vancouver, 1999, pp. 205 ff.
7. Paul Dourish, Victoria Bellotti: *Awareness and Coordination in Shared Workspaces.* Conference proceedings on Computer-supported cooperative work, 1992, Toronto, pp. 107 – 114