In: *Proc. of the 6th Workshop on Transaction Concepts*, Bad Honnef, Germany, Jan. 1995. GI Datenbank Runbdbrief 15, May 1995.

Concurrency Control in KBMSs - The LARS Protocol

Fernando de Ferreira Rezende[1]

Department of Computer Science - University of Kaiserslautern
P.O.Box 3049 - 67653 Kaiserslautern - Germany
Phone: +49 (0631) 205 3274 - Fax: +49 (0631) 205 3558
E-Mail: rezende@informatik.uni-kl.de

Knowledge Base Management Systems (KBMSs) are a growing research area finding applicability in several different domains. On behalf of this increasing applicability, the demand for ever-larger knowledge bases (KBs) is growing more and more. Inside this context, knowledge sharing turns out to be a crucial point to be supported by KBMSs. In turn, it is exactly in this point that concurrency control (CC) techniques for KBMSs play a crucial role, because they are among the most important means for allowing large, multi-user KBs to be widespread. We present our approach for CC in KBMSs, the LARS (Locks using Abstraction Relationships' Semantics) protocol [RH94, RH94a]. The main objective we have in mind is the provision of serializability [GLPT76] for ACID transactions [HR83]. In particular, we have chosen to develop our CC technique for KBs based on locking, because the class of locking-based algorithms has shown its practicality and performance, and additionally, locking-based algorithms have special solutions for graph structures, the abstractions for KBs that appear to be the most appealing [CHM92, Ch94, CHMS94].

Our protocol considers the KB graph as a structure with different granules of locks, and applies explicit locks on its nodes. In addition, it states that a lock on a node implicitly comprises locks on its descendants, what minimizes the number of locks to be acquired and managed. However, to the correct behavior of such implicit locks, it needs to mark the ascendants with some tag indicating that objects are being locked at a lower level. Such a sign represents an intention to set locks at a finer granularity and prevents implicit or explicit conflicting locks on the ancestors. In other words, our protocol is based on the *granular locks protocol* (GLP, for short) of Gray et al. [GLPT76]. Therefore, we use the power and elegance of granular locks also in the KBMS environment. Granular locks are known to be meaningful, because they provide transactions the possibility of choosing, among different locking granules, the most appropriate one to accomplish their tasks. In addition, the use of implicit locks significantly minimizes the number of locks to be set by transactions [GR93].

Notwithstanding, a protocol like the GLP is designed for a single organization hierarchy, extended to DAGs in case of index structures [Gr78]. If we directly apply the GLP to a structure rich in semantics like KB graphs, we are not at all able to interpret their edges. To put it another way, using the GLP, when a shared/ exclusive lock on a node is granted to a transaction, all descendants of this node are implicitly locked in the same mode, independently of the relationship the descendants have to the ascendant. With such a behavior, many objects may be locked unnecessarily, because it is not possible to precisely specify which kind of descendants should be implicitly locked, and thus the overall concurrency may be affected negatively.

As a matter of fact, a KB graph is built through the superposition of the classification/generalization, association, and aggregation hierarchies (or in fact directed acyclic graphs (DAGs)). However, many accesses in a KB are directed to a particular hierarchy, and not to the KB graph as a whole. Due to that, we logically partition the KB graph into those three main hierarchies. As a result, we obtain a combination of different abstraction hierarchies, and we apply hierarchical lock schemes on each one of them. By such a way, on one hand we acquire a minimization of the locks in comparison with for example a conventional approach

1. Financially supported by the CNPq (National Council for the Scientific and Technological Development) of the Secretary for Science and Technology of Brazil.

with shared and exclusive lock modes, where every touched object must be locked. On the other hand, we define more precisely the granule of lock to be accessed by a transaction, allowing it to lock just the objects it really needs to access.

Thus, we create three different logical views from the whole KB graph. These are called the *classification* (which includes also generalization), *association*, and *aggregation* graphs. Obviously, these logical views are based on the abstraction relationships that are (or at least should be) provided by KBMSs. By this way, we provide users with the possibility of looking at a KB, and abstracting from it just the viewpoint to be worked out. In addition, this logical division is mirrored in each object of the KB. Following these logical partitions, we have created three distinct sets of lock types. Hence, similar to the GLP, we have a *basic set* of lock modes, named: IR (Intention Read), IW (Intention Write), R (Read), RIW (Read Intention Write), and W (Write). However, we have this basic set of lock modes to each one of our logical partitions, i.e., to the classification, association, and aggregation graphs, and not to the whole structure as in the GLP. In this manner, we capture more of the semantics contained in the KB graph in the sense that we do not consider descendants of an object as being simply descendants of it, but, on the contrary, descendants with special characteristics and significance, which are based on the abstraction relationships of generalization, classification, association, and aggregation. This is the most important point of our technique, by means of which we can really obtain a high degree of concurrency, with a full exploitation of all inherent parallelism in a knowledge representation approach.

Finally, the main important characteristics of our protocol are: First, the LARS protocol offers different granules of locks. Second, it considers implicit locks, alleviating the task of managing too many locks due to the high number of objects in real world applications. Thirdly, it copes well with multiple abstraction relationships to objects, by means of a requirement of explicitly locking bastards (objects with multiple parents), which, in turn, relaxes the necessity of covering all paths to the root with intentions for writing an object (as required by the GLP), reducing it to only one path. Fourth, it interprets the relationships between objects with respect to their semantics, providing typed locks for all abstraction concepts. At last, we are going to use the KBMS prototype KRISYS [Ma89], developed at the University of Kaiserslautern, as a practical vehicle for the implementation.

References

[Ch94] Chaudhri, V.K.: *Transaction Synchronization in Knowledge Bases: Concepts, Realization and Quantitative Evaluation*. Doctor Thesis, University of Toronto, Toronto, Canada, 1994.

[CHM92] Chaudhri, V.K., Hadzilacos, V., Mylopoulos, J.: Concurrency Control for Knowledge Bases. In: *Proc. of the 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, Cambridge, USA, 1992.

[CHMS94] Chaudhri, V.K., Hadzilacos, V., Mylopoulos, J., Sevcik, K.C.: Quantitative Evaluation of a Transaction Facility for a Knowledge Base Management System. In: *Proc. of the 3rd Int. Conf. on Information and Knowledge Management (CIKM'94)*, Gaithersburg, USA, Nov. 1994.

[GLPT76] Gray, J.N., Lorie, R.A., Putzolu, G.R., Traiger, I.L.: Granularity of Locks and Degrees of Consistency in a Shared Data Base. In: *Proc. of the IFIP Working Conference on Modeling in Data Base Management Systems*, Freudenstadt, Germany, Jan. 1976. pp. 365-394.

[Gr78] Gray, J.N.: *Notes on Database Operating Systems*. In: Operating Systems: An Advanced Course, Springer Verlag, Berlin, 1978. (Lecture Notes in Computer Science No. 60).

[GR93] Gray, J.N., Reuter, A.: *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers, San Mateo, CA, USA, 1993.

[HR83] Härder, T., Reuter, A.: Principles of Transaction-Oriented Database Recovery. *ACM Computing Surveys*, Vol. 15, No. 4, Dec. 1983. pp. 287-317.

[Ma89] Mattos, N.M.: *An Approach to Knowledge Base Management - Requirements, Knowledge Representation, and Design Issues*. Doctor Thesis, University of Kaiserslautern, Kaiserslautern, April 1989.

[RH94] Rezende, F.F., Härder, T.: Capturing Abstraction Relationships' Semantics for Concurrency Control in KBMSs. *ZRI Report No. 6/94*, University of Kaiserslautern, Kaiserslautern, Nov. 1994.

[RH94a] Rezende, F.F., Härder, T.: A Lock Method for KBMSs Using Abstraction Relationships' Semantics. In: *Proc. of the 3rd Int. Conf. on Information and Knowledge Management (CIKM'94)*, Gaithersburg, USA, Nov. 1994.