INFORMATION AND CONTROL **60**, 103–108 (1984)

On the Complexity of Concurrency Control by Locking in Distributed Database Systems

ELJAS SOISALON-SOININEN* AND PETER WIDMAYER†

Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, Postfach 6380, 7500 Karlsruhe 1, West Germany

Given a pair of locked transactions, accessing a distributed database, the problem is studied of whether this pair is safe, i.e., guaranteed to produce only serializable schedules. It is shown that an easy-to-test graph condition, which characterizes safety for a pair of locked transactions in a centralized database, also applies when the database has been distributed among at most three sites. © 1984 Academic Press, Inc.

1. INTRODUCTION

Database concurrency control deals with the problems arising when several transactions access and update a database concurrently. A widely accepted method for implementing concurrency control is locking (e.g., Eswaran *et al.*, 1976). That is, *lock* and *unlock* steps exclusively controlling the access to the shared data are inserted to the transactions. The insertion should be done so that all incorrect interleavings of transactions are forbidden; then the resulting set of locked transactions is *safe.*

Theory of locking includes, e.g., the analysis of the amount of parallelism supported by locking (Papadimitriou, 1982), and the analysis of the computational complexity of such fundamental questions as the safety and deadlock-freedom of a given set of locked transactions (Lipski and Papadimitriou, 1981; Papadimitriou, 1983b; Yannakakis, 1982a, 1982b; Soisalon-Soininen and Wood, 1984). The theory of database concurrency control is surveyed in Papadimitriou (1983a).

The complexity of the safety question when the database has been distributed among many sites is studied in Kanellakis and Papadimitriou (1984). In their model transactions are partial orders of actions accessing the

* Present address: Department of Computer Science, University of Helsinki, Tukholmankatu 2, SF-00250 Helsinki 25, Finland. The work of this author was partly carried out while visiting the University of Karlsruhe under the support of the Alexander von Hulmboldt Foundation.

† The work of this author was supported by a grant from the Deutsche Forschungsgemeinschaft.

0019-9958/84 $3.00

database such that a total order is required within each site. It turned out that a difference between centralized and distributed locking can be seen when considering pairs of transactions: the safety problem for transaction pairs distributed among many sites is coNP-complete, whereas there is a low polynomial bound in the centralized case.

Further, Kanellakis and Papadimitriou (1984) show that the safety question for a transaction pair can be tested efficiently when the database has been distributed among two sites only. In this note we extend this result to the case in which the distribution of the database has been done among three sites at most. We show that an easy-to-test graph condition which is completely independent of the number of sites is enough to characterize safety up to three sites. This condition has already been shown to be insufficient for four sites (Kanellakis and Papadimitriou, 1984). The problem of whether the safety question can be tested in polynomial time for a fixed number $\geqslant 4$ of sites remains open.

2. The Model

In this section we shortly review the model of Kanellakis and Papadimitriou (1984) for distributed locking. A *distributed database* is a triple $D = (E, c, g)$, where E is a set of *entities*, $c > 0$ is the number of *sites*, and $g: E \to \{1,..., c\}$ is a mapping assigning a site to each entity.

Let S be a set the elements of which are of the forms LOCK(x) and UNLOCK(x) (or Lx and Ux, for short), where x is in E, and let $<$ be a partial order on S. Further, S and $<$ satisfy the following conditions:

(1) $<$ is a total order at each site: $<$ is total when restricted to the set $S(i)$ containing all elements Lx and Ux of S such that $g(x) = i$, for all $i = 1,..., c$;

(2) if Lx or Ux is in S, then both Lx and Ux are in S and $Lx < Ux$.

We say that pair $T = (S, <)$ is a *locked transaction* with respect to the distributed database D.

Safety of a set of locked transactions is defined here only informally (see Kanellakis and Papadimitriou, 1984, for details). A set τ of locked transactions is safe if all legal interleavings of the operations of τ (Lx cannot be followed by another Lx without a preceding Ux) can be "serialized." That is, the effect of the interleaving, when updates are put between Lx–Ux pairs, is the same as a serial execution of the transactions.

3. Safety in the Case of Three Sites

The following definition is from Kanellakis and Papadimitriou (1984).

DEFINITION. Let $T_1 = (S_1, <_1)$ and $T_2 = (S_2, <_2)$ be two locked transactions. $D(T_1, T_2)$ is defined as the directed graph (V, A), where

(1) $V = \{x \mid Lx, Ux \in S_1 \cap S_2\}$,

(2) (x, y) is in A, if $Lx <_1 Uy$ and $Ly <_2 Ux$.

PROPOSITION 1 (Kanellakis and Papadimitriou, 1984). *Let T_1 and T_2 be locked transactions. If $D(T_1, T_2)$ is strongly connected, then $\{T_1, T_2\}$ is safe. On the other hand, if T_1 and T_2 are distributed among two sites only, then the safety of $\{T_1, T_2\}$ implies that $D(T_1, T_2)$ is strongly connected.*

In the following we prove that also when *three* sites are involved Proposition 1 provides a characterization of safety.

THEOREM 2. *Let $\tau = \{T_1, T_2\}$, where $T_1 = (S_1, <_1)$ and $T_2 = (S_2, <_2)$ are distributed among at most three sites. If τ is safe, then $D(T_1, T_2)$ is strongly connected.*

Proof. In the case of two sites Kanellakis and Papadimitriou (1984) prove that if τ is safe, then $D(T_1, T_2)$ is strongly connected. They prove a central lemma (Lemma 3 in Kanellakis and Papadimitriou, 1984) which allows the derivation of an unsafe situation whenever $D(T_1, T_2)$ is not strongly connected. We shall prove the corresponding lemma in the case of three sites. We assume that $D(T_1, T_2)$ is not strongly connected and obtain a contradiction to the safety of τ.

Let X be a dominator of $D(T_1, T_2)$, i.e., a subset of the set V of nodes of $D(T_1, T_2)$ such that there is no arc from a node in $V \backslash X$ to a node in X. Because $D(T_1, T_2)$ is not strongly connected, we know that both X and $V \backslash X$ must be nonempty. If for all x, y in X and z in $V \backslash X$ satisfying the conditions $Lz <_1 Ux$ and $Ly <_2 Uz$, conditions $x \neq y$, $Uy <_1 Ux$, and $Ly <_2 Lx$ hold, then τ can easily been shown to be unsafe (see Kanellakis and Papadimitriou, 1984). However, we can deduce from $Lz <_1 Ux$ and $Ly <_2 Uz$ only that $x \neq y$, $Ux \nless_1 Uy$, and $Lx \nless_2 Ly$ ($\nless$ denotes the negation of $<$). It is sufficient to show that adding the precedences $Uy <_1 Ux$ and $Ly <_2 Lx$ for all x, y, and z as above leads to a transaction pair which is safe if τ is safe and for which X is still a dominator. To be precise, let us define this (new) transaction pair $\tau' = \{T_1', T_2'\}$ as follows:

$$T_1' = (S_1', <_1') = (S_1, (<_1 \cup \{(Uy, Ux) \mid x, y \in X, Lz <_1 Ux, Ly <_2 Uz \text{ for some } z \in V \backslash X\})^*),$$

$$T_2' = (S_2', <_2') = (S_2, (<_2 \cup \{(Ly, Lx) \mid x, y \in X, Lz <_1 Ux, Ly <_2 Uz \text{ for some } z \in V \backslash X\})^*).$$

(By R^* we mean the reflexive transitive closure of R.) Clearly, $<_1'$ and $<_2'$ are partial orders, and if τ' is unsafe then so is τ. It remains to show that X is a dominator of $D(T_1', T_2')$:

Claim. X is a dominator of $D(T_1', T_2')$.

Proof of Claim. Assume for the contrary that $x' \in X$ and $z' \in V \backslash X$ are entities for which $Lz' <_1' Ux'$ and $Lx' <_2' Uz'$. Because X is a dominator of $D(T_1, T_2)$, this can only be the case if at least one of the conditions $Lz' <_1' Ux'$ and $Lx' <_2' Uz'$ holds because of the newly added precedences. Let us now distinguish various cases, according to the distribution of the entities among the sites.

(1) x and y are stored at the same site. In this case $<_1 = <_1'$ and $<_2 = <_2'$, because Lx, Ux, Ly, and Uy are totally ordered. Hence the existence of z' and x' such that $(z', x') \in D(T_1', T_2')$ implies $(z', x') \in D(T_1, T_2)$, which contradicts the fact that X is a dominator of $D(T_1, T_2)$.

(2) x and y are stored at different sites and z is stored at the site of x. In this case, $Lz <_1 Ux$ implies $Uz <_2 Lx$ because otherwise $(z, x) \in D(T_1, T_2)$. $Ly <_2' Lx$ cannot contribute because $Ly <_2 Uz$ and $Uz <_2 Lx$ already hold. Hence, $Lx' <_2 Uz'$ must hold. For $Uy <_1' Ux$ to contribute, $Lz' <_1 Uy$ and $Ux <_1 Ux'$ must hold. $Lz <_1 Ux$ and $Ux <_1 Ux'$ imply $Lz <_1 Ux'$. Let us now further distinguish.

(2.1) x' is stored at the site of x. As $(z, x') \notin D(T_1, T_2)$, $Lz <_1 Ux'$ implies $Uz <_2 Lx'$. $Ly <_2 Uz$, $Uz <_2 Lx'$, and $Lx' <_2 Uz'$ imply $Ly <_2 Uz'$. With $Lz' <_1 Uy$ this yields $(z', y) \in D(T_1, T_2)$, a contradiction.

(2.2) x' is stored at the site of y. As $Lz <_1 Ux'$, $Ly <_2 Lx'$ must hold, because $Lx' <_2 Ly$ would imply $Lx' <_2 Uz$ and hence $(z, x') \in D(T_1, T_2)$. $Ly <_2 Lx'$ and $Lx' <_2 Uz'$ imply, however, $Ly <_2 Uz'$, which together with $Lz' <_1 Uy$ implies $(z', y) \in D(T_1, T_2)$, a contradiction.

(2.3) x' is stored at the third site. Let us further distinguish.

(2.3.1) z' is stored at the site of x. As $Lz <_1 Ux'$ and $Lx' <_2 Uz'$, $Uz <_2 Uz'$ must hold, because otherwise $(z, x') \in D(T_1, T_2)$. But this implies $Ly <_2 Uz'$, hence with $Lz' <_1 Uy$ we conclude $(z', y) \in D(T_1, T_2)$, a contradiction.

(2.3.2) z' is stored at the site of y. As $Lz <_1 Ux'$ and $Lx' <_2 Uz'$, $Ly <_2 Uz'$ must hold, because otherwise $(z, x') \in D(T_1, T_2)$. But this implies, together with $Lz' <_1 Uy$, that $(z', y) \in D(T_1, T_2)$, a contradiction.

(2.3.3) z' is stored at the site of x'. Trivial: Lx', Ux', Lz', and Uz' are totally ordered in T_1 and T_2 (see (1)).

(3) x and y are stored at different sites, and z is stored at the site of y. Symmetric with case (2) in the following sense:

(a) exchange x with y;

(b) exchange U with L;

(c) exchange T_1 with T_2;

(d) invert all precedences.

The corresponding arguments from (2) establish that also in (3) X is a dominator of $D(T_1', T_2')$.

(4) x, y, and z are all stored at different sites. Let us distinguish according to the site of x', then according to the site of z'.

(4.1) x' is stored at the site of x.

(4.1.1) z' is stored at the site of x. Analogous to (2.3.3).

(4.1.2) z' is stored at the site of y. In $T_2', Ly <_2' Lx$ cannot contribute to $Lx' <_2' Uz'$, because $Lx' <_2 Ly$ implies $Lx' <_2 Lx$ and $Lx <_2 Uz'$ leads to $Lx' <_2 Uz'$. For $Uy <_1' Ux$ to contribute, $Lx' <_2 Uz'$ and $Lz' <_1 Uy$ and $Ux <_1 Ux'$ must hold. Hence, $Uz' <_2 Ly$ must hold, otherwise $(z', y) \in D(T_1, T_2)$. From $Lz <_1 Ux$ and $Ux <_1 Ux', Lz <_1 Ux'$ holds. From $Lx' <_2 Uz', Uz' <_2 Ly$, and $Ly <_2 Uz$ we conclude $Lx' <_2 Uz$, which means $(z, x') \in D(T_1, T_2)$, a contradiction.

(4.1.3) z' is stored at the site of z. As in (4.1.2), $Lx' <_2 Uz'$, $Lz' <_1 Uy$, and $Ux <_1 Ux'$ must hold. Thus, $Lz' <_1 Lz$ implies $(z', x') \in D(T_1, T_2)$, and $Lz <_1 Lz'$ implies $(z, y) \in D(T_1, T_2)$, a contradiction.

(4.2) x' is stored at the site of y.

(4.2.1) z' is stored at the site of x. $Uy <_1' Ux$ cannot contribute to $Lz' <_1' Ux'$. Hence $Lz' <_1 Ux'$ must hold. For $Ly <_2' Lx$ to contribute, $Lx' <_2 Ly$ and $Lx <_2 Uz'$ must hold. This implies that $Ux <_1 Lz'$, because otherwise $(z', x) \in D(T_1, T_2)$. With $Lz <_1 Ux$ and $Lz' <_1 Ux'$ this yields $Lz <_1 Ux'$, implying $(z, x') \in D(T_1, T_2)$, a contradiction ($Lx' <_2 Uz$, because $Lx' <_2 Ly <_2 Uz$).

(4.2.2) z' is stored at the site of y. Analogous to (2.3.3).

(4.2.3) z' is stored at the site of z. $Uy <_1' Ux$ cannot contribute to $Lz' <_1' Ux'$. Hence $Lz' <_1 Ux'$ must hold. $Ly <_2' Lx$ can only contribute if $Lx' <_2 Ly$ and $Lx <_2 Uz'$. $Lx' <_2 Ly$ and $Ly <_2 Uz$ imply $Lz \nless_1 Ux'$, because otherwise $(z, x') \in D(T_1, T_2)$. $Lz' <_1 Ux'$ implies that $Lz' <_1 Lz$, because otherwise $Lz <_1 Ux'$ would hold. $Lz' <_1 Lz, Lz <_1 Ux$, and $Lx <_2 Uz'$ imply $(z', x) \in D(T_1, T_2)$, a contradiction.

(4.3) x' is stored at the site of z.

(4.3.1) z' is stored at the site of x. $Uy <_1' Ux$ cannot contribute to $Lz' <_1' Ux'$. Hence $Lz' <_1 Ux'$ must hold. $Ly <_2' Lx$ can only contribute if $Lx' <_2 Ly$ and $Lx <_2 Uz'$. $Lx <_2 Uz'$ implies that $Ux <_1 Lz'$, and as $Lz <_1 Ux$ and $Lz' <_1 Ux'$ hold, $Lz <_1 Ux'$ holds. Hence $Uz <_2 Lx'$ must hold, because otherwise $(z, x') \in D(T_1, T_2)$. This, however, would imply a cycle of Uz, Lx', and Ly in T_2, a contradiction.

(4.3.2) z' is stored at the site of y. $Ly <_2' Lx$ cannot contribute to $Lx' <_2' Uz'$. Hence $Lx' <_2 Uz'$ must hold. For $Uy <_1' Ux$ to contribute to $Lz' <_1' Ux, Lz' <_1 Uy$ and $Ux <_1 Ux'$ must hold. As $Lz <_1 Ux$ holds, $Lz <_1 Ux'$ holds, implying that $Uz <_2 Lx'$ must hold, because otherwise $(z, x') \in D(T_1, T_2)$. As $Lz' <_1 Uy$, and $(z', y) \notin D(T_1, T_2)$, $Uz' <_2 Ly$ must hold. Hence $Uz <_2 Lx', Lx' <_2 Uz', Uz' <_2 Ly$, and $Ly <_2 Uz$ yield a cycle in T_2, a contradiction.

(4.3.3) z' is stored at the site of z. Analogous to (2.3.3).

This completes the proof of the claim, and hence the proof of our theorem is complete. ▮

By Proposition 1 and Theorem 2 we get

COROLLARY 3. *A transaction pair $\{T_1, T_2\}$ distributed among at most three sites is safe if and only if $D(T_1, T_2)$ is strongly connected.*

If S_1 in $T_1 = (S_1, <_1)$ and S_2 in $T_2 = (S_2, <_2)$ have n elements altogether, we have

COROLLARY 4. *A transaction pair $\{T_1, T_2\}$ distributed among at most three sites can be tested for safety in $O(n^2)$ time.*

RECEIVED: February 28, 1983; ACCEPTED: January 27, 1984

REFERENCES

ESWARAN, K. P., GRAY, J. N., LORIE, R. A., AND TRAIGER, I. L. (1976), The notions of consistency and predicate locks in a database system, *Comm. ACM* **19**, 624–633.

KANELLAKIS, P. C. AND PAPADIMITRIOU, C. H. (1984), Is distributed locking harder?, *J. Comput. System Sci.* **28**, 103–120.

LIPSKI, W. AND PAPADIMITRIOU, C. H. (1981), A fast algorithm for testing for safety and detecting deadlocks in locked transaction systems, *J. Algorithms* **2**, 211–226.

PAPADIMITRIOU, C. H. (1982), A theorem in database concurrency control, *J. Assoc. Comput. Mach.* **29**, 998–1006.

PAPADIMITRIOU, C. H. (1983a), Theory of concurrency control, *in* "Theoretical Computer Science, 6th GI-Conference, January 1983," Lecture Notes in Computer Science No. 145, pp. 35–47, Springer-Verlag, Berlin/Heidelberg/New York.

PAPADIMITRIOU, C. H. (1983b), Concurrency control by locking, *SIAM J. Comput.* **12**, 215–226.

SOISALON-SOININEN, E. AND WOOD, D. (1984), An optimal algorithm to compute the closure of a set of iso-rectangles, *J. Algorithms* **5**, 199–214.

YANNAKAKIS, M. (1982a), A theory of safe locking policies in database systems, *J. Assoc. Comput. Mach.* **29**, 718–740.

YANNAKAKIS, M. (1982b), Freedom from deadlock of safe locking policies, *SIAM J. Comput.* **11**, 391–408.