J.Sci.Soc.Thailand, 21(1995) 323-329

SHORT REPORT

TIME AND A DUPLICATED DATABASE IN A SHARABLE DATA SYSTEM

PATTARASINEE BHATTARAKOSOL[a], PETER NICKOLAS[b]

[a] *Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok 10330, Thailand.*

[b] *Department of Computer Science, Faculty of Informatics, Wollongong University, Wollongong, NSW 2522, Australia.*

(Received January 10, 1995)

ABSTRACT

In the present world, the need to share data over a heterogeneous distributed database system (HDDS) is widespread. The disconnection of links between a client and a server under the HDDS is an unavoidable situation. Implementation of duplicated databases has been proposed as a solution, but it is not cost effective when users are interested only in some pieces of data. Therefore, we propose here the implementation of a local database storing only the data of interest to a client, and the use of time values to control the maintenance and query processes of the local database. Results are proved which give theoretical justification to the proposal.

INTRODUCTION

In the present day, the need to share data between different organizations has increased. There are many techniques developed to grant services and control sharable systems. Various software designs have been introduced and implemented. Most of the designs focus on the standard problems: semantic heterogeneity, syntactic heterogeneity, accessing language, and concurrency control, including recovery methods at the server sites. A system that contains all of these constraints is called a heterogeneous distributed database system (HDDS) [Parazoglou,1991].

The problems stated above are normally considered in the context of servers and databases in order to maintain consistency of data, and most system designs always concern about the reliabilities of servers and database systems. However, the fact that the system consists of users or clients and servers cannot be avoided. Since the design methods set up the fundamental constraints on the clients that the clients will be available and the process delivered from clients will rely on the main control at server sites. Therefore, the efficiency of the client program is limited.

This paper will not focus on the methods that resolve the difficulties mentioned in the first paragraph; but it will propose a method that grants services to users without seriously being affected by the unavailability of the connection between the client and servers. Many HDDS contained duplicated databases in their environment. This paper will present the concept of using time values in cooperation with a local duplicated database in order that the duplicated database can replace a remote source under controlled conditions.

The Problem

In a complicated system that contains links to various computers and databases, the connection lines between clients and servers are not permanently available on the network system. Besides, the disconnection of the entire system can occur at any time without warning. The consequence of this situation is that users at the client sites must wait until the connection returns. These situations have been considered in [Bhattarakosol,1993]. Furthermore, a design solution has been presented.

From [Bhattarkosol,1993], the designed system consists of three subsystems installed at the client site: an Information Server System, a Query Generater System, and a Preserve Data System. According to the design, a local database named a Preserve Database (PDB) is used to serve users in the case that disconnection occurs. The use of time to confirm the consistency of data from various databases was the presented solution in the design, based on [Kim *et al.*,1990] and [Klahold *et al.*,1986]. Besides, the PDB will store only the data of interest to the user.

Given suitably maintained time values, it is possible that the PDB can be used as an original data resource instead of a remote resource. However, there are two processes that have to be specified and shown to be sound. The first process is the method that maintains data in the PDB, while the second process is the method that considers a PDB as a resource even when the connection is available.

The Solution

According to [Chung,1990], the relational data model is suitable to represent every data model from different databases; and from [Bhattarakosol,1993], the PDB is implemented as a relational database. Therefore, the maintenance process on PDB will not change the characteristics of data available in the PDB. The PDB does not store only data but also records time values of data from every required database. The data in the PDB will normally be updated when links between the client and servers are established. However, the update process will be done only under the condition that a time value recorded in the PDB and the time value from a server are not equal. The change of data in the PDB is performed after the retrieval process from the remote sources has finished. The following paragraph will discuss the maintenance of the PDB.

Maintaining the PDB

As the retrieved content from a remote database is kept as a table in the PDB, and applying relational notations consistent with those in [Maier,1983], we assume the characteristics of an original database at a remote site are as follows.

1. The database has a single relation T.
2. The (designated) key K of T is singleton.
3. The relation T has no nulls.
4. All queries made on T (or at least those which are used to maintain the PDB) are of the form

   SELECT *Fields*
   FROM T
   WHERE *Cond*

where *Fields* is a subset of the attributes of T which includes the key K, and where *Cond* is as usual a Boolean combination of simple comparisons of attributes (of T) with attributes, or of attributes with values. That is, more compactly, each query Q applied to T is of the form $\pi_{Fields} \circ \sigma_{Cond}$.

We model the PDB as a relation T' over the same relation scheme as T. The table T' can be initialised in two different ways: initialise T' as an empty table (over the scheme of T); or initialise T' with one row for each tuple of T, with the key values copied from T, and with all other values set to the null value $\perp$.

Let Q be a query $\pi_{Fields} \circ \sigma_{Cond}$ applied to T, where *Fields* is $KF_1F_2...F_n$. Note that for each tuple $\sigma \in Q(T)$, we have $s(K), s(F_1),..., s(F_n)$ non-null. For each such σ, we maintain T' by proceeding as follows.

(a) If there exists $t' \in T'$ such that $T'(K) = s(K)$, perform an update CH(T' ; $K = s(K)$; $F_1 = s(F_1)$, ... , $F_n = s(F_n)$).

(b) If there does not exist $t' \in T'$ such that $T'(K) = s(K)$, perform an addition ADD (T'; $K = s(K)$, $F_1 = s(F_1)$, ..., $F_n = s(F_n)$, $G_1 = \perp$,....,$G_m = \perp$), where G_1,....,G_m are the attributes of T (or T') other than K and F_1,....,F_n.

Lemma 1 *Let K be the key attribute of both T and T'. Suppose that $T' \in T'$. Then*

i) $t'(K) \neq \perp$;

ii) *there exists a unique $t \in T$ such that $t'(K) = t(K)$; and*

iii) *if $t'(A) \neq \perp$, for some attribute A, then $t'(A) = t(A)$, for the above t.*

Proof

We prove the result by induction on the number of queries (and subsequent maintenance steps) performed. At the start, before any queries have been made, there are two cases, depending on which initialising step was used: the first method or the second method.

If T' was initialised as an empty table, the required statements hold vacuously. Consider the other case, the initialisation of T' with one row for each row of T, with key values copied from T, and with all other values set to $\perp$. Then for $t' \in T'$, (i) and (ii) above are immediate, and since $T'(A) \neq \perp$ implies that $A = K$, (iii) holds because of (ii). Thus (i), (ii) and (iii) hold initially.

Now suppose inductively that (i), (ii) and (iii) hold for all $t' \in T'$ before the execution of a query $Q = \pi_{Field^S} \circ \sigma_{Cond}$ and the subsequent maintenance step. Suppose also that *Fields* $= KF_1F_2...F_n$. Let $t' \in T'$ and prove (i),(ii) and (iii).

First note that if t' was present in T' before Q, and is unaffected by Q (and the maintenance step), then (i), (ii) and (iii) are automatic. Now, consider the case when t' is either created or updated by Q.

(i) Suppose first that t' is updated by step (a) of the maintenance procedure. Note that only one such update occurs, since there can be only be one $s \in Q(T)$ with $s(K) = t'(K)$. Now t' before the update satisfied $t'(K) = s(K)$, and t' is updated by setting $t'(K)$ to $s(K)$, so $t'(K)$ in fact unchanged. Since $t'(K) \perp$ before the step, this also holds afterwards. Second, if t' is created by step (b) of the maintenance procedure, we have $t'(K) = s(K) \neq \perp$. Thus (i) holds.

(i) Suppose first that t' is updated by step (a) of the maintenance procedure. Note that only one such update occurs, since there can be only be one $s \in Q(T)$ with $s(K) = t'(K)$. Now t' before the update satisfied $t'(K) = s(K)$, and t' is updated by setting $t'(K)$ to $s(K)$, so $t'(K)$ in fact unchanged. Since $t'(K) \neq \perp$ before the step, this also holds afterwards. Second, if t' is created by step (b) of the maintenance procedure, we have $t'(K) = s(K) \neq \perp$. Thus (i) holds.

(ii) If t' is updated by step (a), then $t'(K)$ is not changed, so (ii) holds by the inductive assumption. If t' is added by step (b), then by the definition of a key, there is a unique $t \in T$ such that $t'(K) = s(K) = t(K)$. Thus (ii) holds.

(iii) Let A be an attribute of T such that $t'(A) \neq \perp$. First suppose that t' is updated by step (a). If A is none of $K, F_1, F_2...F_n$, then $t'(A)$ is unaltered by the update, so since $t'(K)$ is not changed by the update either, $t'(A) = t(A)$ holds by the inductive assumption. If $A = K$, then $t'(A) = t(A)$ holds by (ii). Suppose that $A = Fi$ for some i. Now $t'(F_i)$ becomes $s(F_i)$ in the update, and we want to show that $s(Fi) = t(Fi)$. But t is the unique tuple in T such that $t(K) = t'(K) = s(K)$, and therefore $s = t(KF_1F_2...F_n)$, giving $s(F_i) = t(F_i)$, as required. Second, suppose that t' is created by step (b). Then $t'(A) \neq \perp$ implies that A is K or one of the F_i. If $A = K$, we have $t'(A) = t(A)$ by (ii). If $A = F_i$, then $t'(A) = t'(F_i) = s(F_i)$, which as before is $t(F_i) = t(A)$. Thus (iii) holds.

Therefore by induction the lemma is proved.

As a consequence of Lemma 1, the data in the PDB can be retrieved under two situations. Firstly, the retrieval process occurs when disconnection arises. This performance has been mentioned in [Bhattarakosol,1993]. Secondly, the retrieval system chooses to read data from PDB if the time values between the PDB and a remote database are equal. In the second situation, the problem of the equivalence between data from the PDB and the remote database has to be considered, and is discussed below.

Querying the PDB

According to the result of storing some pieces of data from the original database, the PDB can be used as a data source when the time value of the remote database is equal to the time value of PDB. Therefore, we will say that a query Q of the form $\pi_{Field^S} \circ \sigma_{Cond}$

(with *Fields* and *Cond* as earlier) is permitted (on T) if for every tuple $t' \in T$, all fields of t' whose attributes names occur in *Field* or *Cond* have non-null values. Only permitted queries are applied to T.

Theorem 1 *If Q is a permitted query, then $Q(T) \subseteq Q(T)$.*

Proof

As before, write Q as $\pi_{Fields} \circ \sigma_{Cond}$, where $Fields = KF_1F_2...F_n$. Consider any $t' \in T$ such that t' is selected by σ_{Cond}. By Lemma 1, there is a unique $t \in T$ such that $t'(K) = t(K) \neq \perp$. For each such attribute A occurring in *Fields* or *Cond*, the fact that Q is permitted shows that $t'(A) \neq \perp$, and so Lemma 1 shows that $t'(A) = t(A)$ for each such A. It follows in particular that σ_{Cond} also selects t from T (strictly by induction over the structure of *Cond*). Now $\pi_{KF1F2...Fn}(t') = t'(KF_1F_2...F_n)$ and $t'(K)$ and each $t'(Fi)$ is non-null. Therefore $t'(Fi) = t(Fi)$, and so $t'(KF_1F_2...F_n) = t(KF_1F_2...F_n)$. Therefore $\pi_{KF1F2...Fn}(t') = \pi_{KF1F2...Fn}(t)$, and so $\pi_{KF1F2...Fn} \circ \sigma_{Cond}$ produces all the tuples when applied to T that it produces when applied to T.

The reverse of Theorem 1 does not hold for the first initialisation scheme, but if the second scheme is applied, the reverse inclusion does hold.

Theorem 2 *Suppose that T is initialised with one row for each tuple of T, with the key values copied from T, and with all other values set to $\perp$. If Q is a permitted query, then $Q(T) \subseteq Q(T)$.*

Proof

Take Q to be $\pi_{Fields} \circ \sigma_{Cond}$ as earlier. Let $t \in T$ be selected by σ_{Cond}. Now there is a $t' \in T$ such that $t'(K) = t(K)$, because of the initialisation step, and because no key values are changed in the maintenance process. By Lemma 1, there is a unique $t'' \in T$ such that $t'(K) = t''(K)$ and such that $t'(A) \neq \perp$ implies $t'(A) = t''(A)$ for all attributes A of T. Since $t(K) = t'(K) = t''(K)$, we have $t = t''$. Now as Q is permitted, $t'(A) \neq \perp$ for all attributes A of *Cond* and *Fields*, so we have $t'(A) = t(A)$ for all such A. Hence σ_{Cond} selects t' from T, and by an argument similar to that of Theorem 1, we have $\pi_{Fields}(t) = \pi_{Fields}(t')$, giving the result.

Theorem 3 *Suppose that T is initialised with one row for each tuple of T, with the key values copied from T, and with all other values set to $\perp$. If Q is a permitted query, then $Q(T) = Q(T)$.*

Proof

From Theorem 1 and Theorem 2, $Q(T) \subseteq Q(T)$ and $Q(T) \subseteq Q(T)$. Therefore $Q(T) = Q(T)$.

As a consequence of Theorem 1, Theorem 2 and Theorem 3, reading data from the PDB using permitted queries is safe.

SUMMARY

Since heterogeneous distributed database systems (HDDS) are highly needed in the present world, many constraints are considered and solved in order that they will be user friendly and efficient. Data consistency is one important factor that must be maintainable. Furthermore, the requirement of providing services when disconnection arises was also included. The duplicated database is a solution that most of the HDDS use to serve users when disconnection occurred. Moreover, consistency of data in the system is maintainable. On the other hand, it is also not possible that every HDDS will implement a duplicated database. For example, the cost of maintaining the duplicated repository is not reasonable if the users use only a subset of the data. Therefore, making duplication of data should be according to the users's interests and stored at the local area to avoid the disconnection problem.

Using time values in conjunction with implementing a local database can be a good solution to providing consistent data to users when the connection is closed. Moreover, the local repository, the PDB, implemented in the relational model can be maintainable, as proved in Lemma 1. A consequence of using time values is that the PDB can represent an original repository when the comparison of times between the PDB is equal to the remote database. The data obtained from the local area can be a subset of the data obtained from the real remote resource. However, it also can be the same set of data if all keys of interested data are initialised in the local database (see Theorem 1, Theorem 2 and Theorem 3).

REFERENCES

[BHATTARAKOSOL,1993], Bhattarakosol, P., "Presentation of Consistent Information from Independent Databases," Proc.of the Int. Conf. on Information Systems and Management of Data, CISMOD'93, pp.15-26.

[GARCIA,1982] Garcia-Molina, J., "Reliability Issues For Fully Replicated Distributed Databases," IEEE Computer, Vol.16, No.9, September 1982, pp.34-42.

[KIM *et al.*,1990], Kim, J., *et al.*, "Design and Implementation of a Temporal Query Language with Abstract Time," Information System, Vol.15, No.3, pp. 349-357.

[KLAHOLD *et al.*,1986], Klahold, P., *et al.*, "A General Model for Version Management in Databases," Proc.12th Int. Conf. on Very Large Databases, pp. 319-327.

[MAIER,1983], Maier, D., The Theory of Relational Databases, London:Pitman Publishing Limited, 1983.

[PAPAZOGLOU,1991], Papazoglou, M.P., "Framework for Interconnecting Distributed Information Systems." DBIS'91: Database and Information System Conference, Sydney, February 1991.

[RAM,1991] Ram, S., "Heterogeneous Distributed Database Systems," IEEE Computer, Vol.24, No.12, December 1991, pp. 7-10.

[STANKOVIC,1985] Stankovic, J., "A Reliable Distributed System Software," IEEE Computer Society order No.570, IEEE Catalog No.EH0230-3, 1985, pg. 3-5.

บทคัดย่อ

ในปัจจุบันการใช้ระบบฐานข้อมูลหลากหลายแบบกระจาย (HDDS) เป็นสิ่งจำเป็นมากอย่างหนึ่ง ที่ช่วยส่งเสริมให้การทำงานในสาขาต่างๆ ที่เกี่ยวข้องกันเป็นไปได้อย่างมีประสิทธิภาพ อย่างไรก็ดี การทำงานบนระบบติดต่อสื่อสาร อาจมี อุปสรรคอันไม่สามารถจะหลีกเลี่ยงได้ เช่น การเชื่อมโยงระหว่างระบบ client และ server สามารถขาดการติดต่อได้โดย ไม่มีการคาดหมายมาก่อน การแก้ไขป้องกันที่นิยมใช้กันนั้น คือ การจัดทำสำเนาของฐานข้อมูลทั้งหลายกระจายอยู่บนระบบเครือข่ายที่ client สามารถเปลี่ยนไปใช้งานได้โดยทันที เมื่อการเชื่อมโยงกับ server เกิดปัญหาขึ้น แต่วิธีการนี้ จะใช้ค่าใช้จ่ายที่สิ้นเปลืองไม่คุ้มค่ากับกรณีที่ผู้ใช้ข้อมูลต้องการใช้ข้อมูลเพียงบางส่วน บทความนี้จะเสนอวิธีการใช้ระบบฐานข้อมูลที่เก็บสำเนาเฉพาะข้อมูลที่ต้องการไว้เพื่อเรียกใช้ในยามที่ระบบการติดต่อขาดหายไป โดยทำการติดตั้งระบบสำเนาฐานข้อมูลนี้ที่ส่วนของระบบ client เพื่อใช้งานในเวลาที่การเชื่อมโยงขัดข้อง พร้อมทั้งนำเสนอแนะการนำค่าของเวลามาใช้เพื่อช่วยให้ผู้ใช้ข้อมูลเกิดความมั่นใจว่าข้อมูลที่ได้จากแหล่งข้อมูลต่างๆ มีความสอดคล้องกันและทันสมัย เพราะสำเนาที่ได้ทำการเก็บไว้ภายใต้ระบบ client ไม่สามารถรับรองได้ว่ามีข้อมูลที่ทันสมัย เนื่องจากการแก้ไขข้อมูลที่ฐานข้อมูล ณ server สามารถมีการเปลี่ยนแปลงแก้ไขตลอดเวลา ดังนั้น การนำค่าของเวลามาใช้งานในรูปแบบที่เหมาะสมโดยมิก่อให้เกิดความยุ่งยากต่อการเปลี่ยนแปลงแก้ไขรูปแบบของข้อมูลในระบบที่มีอยู่เดิม จะช่วยให้ผู้ใช้ข้อมูลเกิดความมั่นใจในข้อมูลที่ได้จากการใช้ระบบฐานข้อมูลหลากหลายแบบกระจายที่มีสำเนาข้อมูลบางส่วน ณ client