SEMANTIC CONCURRENCY CONTROL

6H-6

Takuma OUCHI and *Makoto TAKIZAWA*
Tokyo Denki University

1. INTRODUCTION

In database systems and distributed database systems, multiple transactions access the same objects. In order to keep the databases and distributed databases consistent in the presence of multiple transactions, the concurrency controls are required. There are two approaches. In the first approach, based on the object concept, the permutation relations on the objects among higher level operations, e.g. deposit and withdrawal on the accounts, than read and write operations are semantically given [LYNC]. In the second approach, based on the semantics of the transactions, a set of transactions which can be executed concurrently is defined [GARC]. In this paper, we take the second approach.

2. DEFINITIONS

A database system S is composed of the database D and a set O of operations for manipulating the objects in D. A database D is a set of objects. Each object has its identifier and value. An object is an abstraction of conventional data elements like record, file, tuple relation, page. A transaction T is a sequence of operations. Formally, the transaction T is written as an ordered set of operations, $(\Sigma_T, \rightarrow_T)$, where Σ_T is a set of operations and $\rightarrow_T$ is an ordering relation on Σ_T. For every two operations o_1 and o_2 in Σ_T, $o_1 \rightarrow_T o_2$ iff o_2 is executed after o_1 completes. For each transaction T, operations in T are grouped into steps, $T=(\Omega_T, \Rightarrow_T)$. Here, Ω_T is a set of steps $\{\sigma_{T1}, \ldots, \sigma_{Tm}\}$. That is to say, each step σ_{Tj} is a subset of Σ_T, $\Sigma_T = \sigma_{T1} \cup \ldots \cup \sigma_{Tm}$. Next, let us consider the interleaved execution of multiple transactions. Let TT be a set $\{T_1, \ldots, T_n\}$ of transactions in the system S. A step log SL_{TT} for TT is a totally ordered set $(\Omega, \Rightarrow)$ where Ω is set of steps and $\Rightarrow \subseteq \Omega^2$. Here, $\Omega = \Omega_{T1} \cup \ldots \cup \Omega_{Tn}$, and for every two steps σ and σ' in Ω, if σ and σ' belong to the same transaction T_j and $\sigma \Rightarrow_{Tj} \sigma'$, then $\sigma \Rightarrow \sigma'$. A log L_{TT} for TT is a totally ordered set $(\Sigma, \rightarrow)$ where $\Sigma = \Sigma_{T1} \cup \ldots \cup \Sigma_{Tn}$ and $\rightarrow \subseteq \Sigma^2$. For every two operations o and o' in Σ, if o and o' belong to the same transaction T_j and $o \rightarrow_{Tj} o'$, then $o \rightarrow o'$.

[Def.]A log L_{TT} is said to be a step-wise serial iff there exists some step log SL_{TT} such that for every two steps σ and σ' in SL_{TT}, if $\sigma \Rightarrow \sigma'$, then for every operation o in σ and o' in σ', $o \rightarrow o'$.□

If a log L_{TT} is not a step-wise serial, it is said to be concurrent or interleaved.

[Def.]A log L_{TT} is said to be a step-wise serializable iff there exists some step log SL_{TT} such that for every two steps σ and σ' in SL_{TT}, if $\sigma \Rightarrow \sigma'$, then for every operation o in σ and o' in σ', if o and o' conflict, then $o \rightarrow o'$.□

For every transaction T, a semantic Y is assigned. Let Π be a set of semantic in the system S. Let Type(T) be a semantic type of T. Here, $Type(T) \subseteq 2^{\Pi}$. The transaction type have the following constraints and meanings.

[Constraint](1) Each $h \in Type(T)$ is $h \in 2^{\Pi}$.
(2) If $g \in Type(T)$ and $h \in Type(T)$ then $g \not\subseteq h$.
(3) Let Y_1 and Y_2 be semantics of T_1 and T_2, respectively. If $h \in Type(T_1)$ and $Y_2 \in h$ then $h \in Type(T_2)$.□

[Def.]Let T_1 and T_2 be transactions in TT. If $Type(T_1) \cap Type(T_2) \neq \phi$, then T_1 and T_2 can be executed step-wise serializable.□

Let T_1 and T_2 be transactions which are composed of steps σ_{11}, σ_{12} and σ_{21}, σ_{22}, σ_{23}, respetively. $Type(T_1)=\{\{a,b\}\}$ and $Type(T_2)=\{\{a,b\}\}$. Since $Type(T_1) \cap Type(T_2)=\{a,b\} \neq \phi$, T_1 and T_2 are executed step-wise serializable. For example, $SL=\sigma_{21}\sigma_{11}\sigma_{22}\sigma_{23}\sigma_{12}$ is a step-wise serializable log. If $Type(T_1)=\{\{a,b\}\}$ and $Type(T_2)=\{\{c\}\}$, T_1 and T_2 cannot be executed concurrently. Suppose that $\sigma_{11}=\langle o_{111}, o_{112}\rangle$, $\sigma_{12}=\langle o_{121}, o_{122}, o_{123}\rangle$, $\sigma_{21}=\langle o_{211}, o_{212}\rangle$, $\sigma_{22}=\langle o_{221}\rangle$, $\sigma_{23}=\langle o_{231}, o_{232}, o_{233}\rangle$. Suppose that o_{111} and o_{211} conflict, and o_{121} and o_{221} conflict. For a step log SL, the following operation log L is a step-wise serializable one. $SL=\langle o_{211}, o_{111}, o_{112}, o_{212}, o_{121}, o_{221}, o_{122}, o_{231}, o_{123}, o_{232}, o_{233}\rangle$

If $Type(T)=\Pi$, T can be executed with every transaction in TT concurrently.

3. SYNCHRONIZATION METHOD

Next, we consider how to get the correct serializable schedule from a set of transactions $TT=\{T_1, \ldots, T_n\}$. Since most database systems adopt a locking method as the synchronization method of interleaved execution of transactions, we try to realize the semantic concurrency control by using the locking methods provided by the underlying database system. There are two kinds of synchronization like [GARC], i.e global and local synchronization.

SEMANTIC CONCURRENCY CONTROL
Takuma OUCHI and Makoto TAKIZAWA
Tokyo Denki University

3.1 Global Synchronization

A global synchronization aims at executing only transactions which are correctly executable.

Let Act be a subset of transactions in TT which are executed at present. Act is said to be an active transaction set.

[Def.]For an active transaction set Act= {T_{A1}, ...,T_{Am}} ($m \leqq n$), a concurrency level Con_{Act} is defined to be a intersection of semantic types of $T_{A1},...,T_{Am}$, i.e. $Con_{Act}=Type(T_{A1}) \cap ... \cap Type(T_{Am})$.□

For example, suppose that there are three transaction T_1, T_2, and T_3 in the active transaction set A. Let $Type(T_1)$, $Type(T_2)$, and $Type(T_3)$ be {{a,b},{a,c}}, {{a,c}}, and {{d}}, respectively. Con_A is {a,c}.

[Def.]An active transaction set Act is said to be correct iff the concurrency set Con_{Act} is not bottom, i.e. $Con_{Act} \neq \phi$.□

The active transaction set A as stated before is correct, because $Con_A=\{a,c\} \neq \phi$.

[Def.]For an active transaction set Act and a transaction T, if $Con_{Act} \cap Type(T) \neq \phi$, T is said to be correctly executable.□

Suppose that the active transaction set A is {a,b}. Suppose that a transaction T_1 has a type {{a,b}} and T_2 {{c}}. T_1 is correctly executable but T_2 is not.

Let us show us our global synchronization method. Let T be a transaction which wants to start the execution. Let Act be an active transaction set. The data manager takes the requirements from T and behaves as follows.
[Execution of a transaction T]
(1) If $Type(T) \cap Con_{Act} \neq \phi$, then T is executed, i.e. Act:=Act$\cup$ {T} and $Con_{Act}:=Con_{Act} \cap Type(T)$,
(2) else T blocks, i.e. T is enqueued in AQUEUE.
□
Suppose that A is an active transaction set where Con_A is {a,b}, and T is a transaction which wants be executed and whose semantic type is {{a,b},{a,c}}. Since $Con_A \cap Type(T)$ is {a,b}, T is allowed to be executed. {T} is added to A and Con_A is not change. Here, another transaction S where Type(S) is {{a,c}} wants to be executed. But, since $Con_A \cap type(S)=\phi$, S in enqueued into AQUEUE.

Next, let us consider a case that a transaction T in Act completes to execute. The manager takes the request informing of the completion of T's execution and behaves as follows.
[Termination of an active transaction T]
(1) Act:=Act-{T}, and $Con_{Act}=s \in Act$ Type(S).
(2) AQUEUE is searched from the top and a transaction S which is executable with Act is tried to be found. If such a transaction S is found, then S is executed.□
For example, an active transaction set A if {T_1, T_2,T_3} where $Type(T_1)$={{a,b}}, $Type(T_2)$={{a,b}, {a,c}}, $Type(T_3)$={{a,b}}. Here, Con_A={a,b}. Suppose that T_1 and T_3 completes to execute. {T_1, T_3} is removed from A. Con_A is changed to {a,c}. Suppose that AQUEUE includes transactions T_4, T_5 in this sequence, where $Type(T_4)$={{d}}, $Type(T_5)$ ={{a,c}}. AQUEUE is searched from the top, i.e. T_4. Since T_5 is correctly executable, T_5 is removed from AQUEUE and executed. Now A={T_2,T_5} and Con_A is {a,c}.

3.2 Local Synchronization

Next, let us consider the local synchronization. The local synchronization aims at realizing the step-wise serializability of the log. In our method, the conventional locking mechanism is used. A transaction is composed of one or more steps. Each step σ_{Tj} is composed of operations on the objects. In our local synchronization, each step obeys the two-phase locking protocol [ESWA].

3.3 Correctness

Now, we show the correctness of our synchronization method.

[Poposition 1] The local synchronization method assures the step-wise serializability.
[Proof] It is clear from [ESWA].■
[Proposition 2] The global synchronization method assures that only correctly executable transactions are executed.
[Proof] Form the meaning of the semantic types, it is clear.■
[Theorem 3] Our synchronization method is correct.
[Proof] From proposition 1 and proposition 2, it is clear.■

4. CONCLUDING REMARKS

In this paper, we have discussed the semantic concurrency control based on a set of transactions which are concurrently executed. This set is defined by users on the basis of the semantics of transactions.

REFERENCES

[ESWA] Eswaran, K. P. and Gray, J. N., "The Notions of Consistency and Predicate Locks in a Database System," CACM, Vol.19, No.11, 1976.
[GARC] Garcia-molina, H., "Using Semantic Knowledge for Transaction Processing in a Distributed Database," ACM TODS, Vol.8, No.2, 1983, pp.186-213.
[LYNC] Lynch, A. N., "Multilevel Atomicity-A New Correctness Criterion for Database Concurrency Control," ACM TODS, Vol.8, No.4, 1983, pp.484-502.