Concurrency Control Based on Order Constraints in Advanced Database Systems

Haiyan Xu † Tetsuya Furukawa ‡ Yihua Shi ††

† Dept. Electronics, Fukuoka Institute of Technology
‡ Dept. Economic Engineering, Kyushu University
†† Fukuoka Junior College of Technology

Abstract

It is widely recognized that the serializability correctness criterion of concurrency control in typical database systems, is unacceptable for cooperative database applications. By associating status with each data, in this paper, we show the consistency information of advanced database applications can be expressed declaratively by logical formula. Utilizing the consistency information of databases, we introduce a new correctness criterion called equivalent-independent, which is more generous than serializability. A subclass of equivalent-independent schedules, which is a superset of serializable schedules, includes many non-serializable schedules required by cooperative database applications.

1 Introduction

In database systems, transactions are expected to be consistent units. If executed alone, a transaction takes the database from a consistent state to another. When a set of transactions are executed concurrently, the execution also has to keep the consistency of the database and each transaction observes a consistent database.

There are two approaches to assure the database consistency. The first one assumes the correctness of individual transactions. In this approach, each transaction is supposed to result in a consistent database if executed alone in a consistent database. The second one requires the database to validate that no transaction violates the consistency of the database. In traditional databases, the first approach has been used because it is accepted that the consistency information is unknown. The consistency problem is reduced to the problem of testing serializability of executions. An execution of concurrent transactions is called serializable if it is equivalent to a serial execution. The consistency of a database without information about application specific consistency is guaranteed by allowing only serializable executions of concurrent transactions.

In these days, advanced databases have been used in such new areas as computer-aided design and manufacturing (CAD/CAM), office automation, computer-aided software engineering, and so on. These new applications require long duration, interactive transactions, and cooperation among users. The serializability correctness is too restrictive to satisfy these new requirements [2]. These problems are widely recognized. Current research on dealing with cooperative database applications can fall into the following three basic categories.

- Supporting user control [6].
- Extending serializability-based techniques or relaxing serializability by extending transaction model [4].
- Using the semantics defined by each transaction [1, 5].

The problem of the first category is that users are not adept at managing concurrent activity. Although users know their work, they need not understand the implication of their concurrent activities. The second one is still based on serializability so that it cannot settle the problems of serializability completely. The last one is still in research level and also has the problem of the burden of users.

In this paper, we show that the consistency information of applications can be expressed declaratively by introducing status of each data. Using the consistency information of databases represented by logical formula, we propose a new correctness criterion called equivalent-independence. A subclass of the equivalent-independent class, which treats the most popular consistency information, is introduced. The subclass is more generous than serializable class and allows non-serializable executions by relaxing the restriction on read operations.

2 Preliminaries

An advanced database $DB\langle O, R\rangle$ is a pair of a set of objects O and a set of relationships R. Each object $o \in O$ has two parts, an entity part $e(o)$ and a status part $at_1(o), at_2(o), \cdots, at_m(o)$. The value $e(o)$ is the data of o. Status $at_k(o)$ $(1 \leq k \leq m)$ has truth value expressing the information of o such as whether o has passed the k-th test, whether o satisfies the k-th constraint condition, and so on. $e(o)$ and $at_k(o)$ are data items, which are the units of database access. $\mathcal{DB}$ is the set of all data items in the database. $I(d)$

denotes the value of data item d. For a set of data items $D \subseteq \mathcal{DB}$, $I(D) = \{I(d) \mid d \in D\}$.

R is a set of relationship $r(o_i, o_j)$ which expresses the relationship of type r between objects o_i and o_j. Although we assume that the relationships in this paper are binary relationships for the simplicity, the results of this paper can be extended to n-ary relationships easily.

An execution of user's work is abstracted as a transaction. A transaction T is a set of read and write operations on data items with a total order $<$, $T = \{R(x), W(x) \mid x \in \mathcal{DB}\}$. We assume that $x \neq y$ if $R(x), R(y) \in T$ or $W(x), W(y) \in T$.

A concurrent execution of a set of transactions is modeled as a schedule. Let $\boldsymbol{T} = \{T_1, T_2, \cdots, T_n\}$ be a set of transactions. A schedule H over $\boldsymbol{T}$ is the set operations of T_i, $\bigcup_{i=1}^{n} T_i$, with a total order $<_H \supseteq \bigcup_{i=1}^{n} <_i$ where $<_i$ is the total order of T_i.

Definition 1 Let $\boldsymbol{T}$ be a set of transactions. For a schedule H over $\boldsymbol{T}$, $D_R(T_i) = \{x \mid R_i(x) \in T_i\}$ and $D_W(T_i) = \{x \mid W_i(x) \in T_i\}$ are the sets of data items read and written by T_i, respectively. $D(T_i) = D_R(T_i) \cup D_W(T_i)$ is the set of data items accessed by T_i. Let $I(R_i(x))$ and $I(W_i(x))$ be the value of x read and written by T_i, respectively. Then $I(D_R(T_i)) = \{I(R_i(x)) \mid R_i(x) \in T_i\}$ and $I(D_W(T_i)) = \{I(W_i(x)) \mid W_i(x) \in T_i\}$. The resulting value of T_i is $I(D(T_i)) = \{I(R_i(x)) \mid R_i(x) \in T_i, W_i(x) \notin T_i\} \cup \{I(W_i(x)) \mid W_i(x) \in T_i\}$. □

Let $\boldsymbol{T}$ be a set of transactions. Schedules H and H' over $\boldsymbol{T}$ is equivalent (denoted by $H \equiv H'$) if $I(D_R(T_i))$ of H is equal to $I(D_R(T_i))$ of H' $(1 \leq i \leq n)$ and $I(D_R(T_f))$ of $W_0(\mathcal{DB})HR_f(\mathcal{DB})$ is equal to $I(D_R(T_f))$ of $W_0(\mathcal{DB})H'R_f(\mathcal{DB})$ for any $T_0 = W_0(\mathcal{DB})$. If H is equivalent to a serial schedule H', H is called serializable.

Theorem 1[3] A schedule H is serializable if its serialization graph is acyclic. The serialization graph of H, denoted by $SR(H)$, is the directed graph whose nodes corresponds to the transactions in $\boldsymbol{T}$ and whose edges are $T_i \rightarrow T_j$ $(i \neq j)$ such that one of T_i's operations precedes and conflicts with one of T_j's operations, that is $W_i(x) <_H R_j(x)$, $R_i(x) <_H W_j(x)$, or $W_i(x) <_H W_j(x)$. □

3 Consistency of Databases

A large project is often divided into a collection of smaller tasks. Some tasks must be operated step by step according to certain specified orders while some tasks can be operated in parallel. Utilizing status of data, specific order among tasks can be expressed naturally by directed acyclic graphs of status.

Example 1 Let $DB_1\langle O_1, R_1\rangle$ be such a database as $O_1=\{o_1, o_2, o_3\}$ and $R_1=\{comp(o_1, o_2), comp(o_1, o_3)\}$, where $comp(x, y)$ is the relationship that y is a component of x. Suppose the objects have to pass Tests 1, 2, and 3 which are ordered as follows.

1. Test 3 of an object requires that the object has already passed Tests 1 and 2.

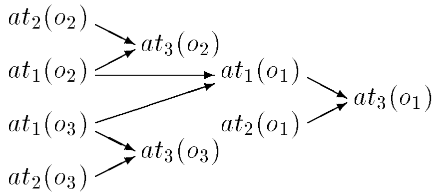

Figure 1: Specific orders among tasks.

2. Test 1 of an object requires the test of its components as a pretask.

The specific orders among the nine tasks can be expressed by the order of $at_k(o_i)$ $(i = 1, 2, 3, k = 1, 2, 3)$ changing its value from $false$ to $true$ as illustrated in Figure 1. □

Since status takes truth value, each edge $at_k(o_i) \rightarrow at_l(o_j)$ can be expressed by horn clause $at_k(o_i) \leftarrow at_l(o_j)$ $(\equiv at_k(o_i) \vee \neg at_l(o_j))$. Using first order logic variables to represent $o \in O$, we can express the orders of the tasks formally.

Definition 2 The order constraints of a database are the following kinds of rules represented by horn clauses where variables x and y are universally quantified.

1. Status constraints, which require the consistency of status of objects:
   $at_i(x) \leftarrow at_j(x).$
   $at_i(x) \leftarrow at_i(y), r(x, y).$
2. Entity constrains, which require the consistency between the entity and the status of an object:
   $p(x, at_i(x)) \leftarrow at_i(x).$
   p is a predicate and $p(x, at_i(x))$ is supposed to change its value only when $e(x)$ is modified. □

Example 2 The orders among tasks of the project with database DB_1 can be represented by the following order constraints declaratively.
$at_1(x) \leftarrow at_3(x).$
$at_2(x) \leftarrow at_3(x).$
$at_1(x) \leftarrow at_1(y), comp(x, y).$ □

To discuss whether $I(\mathcal{DB})$ satisfies the constraints, we use the ground extension of the constraints.

Definition 3 For a database $DB\langle O, R\rangle$, the ground extension of a set of constraints $\boldsymbol{C}$ by a set of data item D $(D \subseteq \mathcal{DB})$, denoted by $G(\boldsymbol{C}, D)$, is the following set of ground clauses, where $\boldsymbol{C}'$ is the set of ground clauses obtained by substituting an element of O for each variable of $\boldsymbol{C}$.

$$G(\boldsymbol{C}, D) = \{p \leftarrow q \mid p \in D \vee q \in D, p \leftarrow q \in \boldsymbol{C}'\} \cup \left\{ p \leftarrow q \;\middle|\; \begin{array}{l} p \in D \vee q \in D, r(o_i, o_j) \in R, \\ p \leftarrow q, r(o_i, o_j) \in \boldsymbol{C}' \end{array} \right\} \cup \left\{ p(o, at_i(o)) \leftarrow at_i(o) \;\middle|\; \begin{array}{l} e(o) \in D \vee at_i(o) \in D, \\ p(o, at_i(o)) \leftarrow at_i(o) \in \boldsymbol{C}' \end{array} \right\}$$
□

For a database DB, we say that $I(\mathcal{DB})$ satisfies $\boldsymbol{C}$ if every ground clause of $\boldsymbol{C}$ is true on $I(\mathcal{DB})$. Database DB is consistent if $I(\mathcal{DB})$ satisfies $\boldsymbol{C}$.

For a set of data items $D \subset \mathcal{DB}$, however, the truth value of a ground clause of $G(\boldsymbol{C}, D)$ may not be determined by $I(D)$ when $at_k(o_i)$ in the ground clause is not in D. We say that $I(D)$ is satisfiable if every clause of $G(\boldsymbol{C}, D)$ is satisfiable. It is evident that $I(D)$ ($D \subseteq \mathcal{DB}$) is satisfiable if $I(\mathcal{DB})$ satisfies $\boldsymbol{C}$.

4 Equivalent-Independence

When the available information for concurrency control is only syntactic one of transactions, the serializability must be the correctness criterion of concurrency control for the database consistency. If we can specify the database consistency as constraints, the consistency problem can be dealt with in another way.

Definition 4 For a database with a set of constraints $\boldsymbol{C}$, a schedule H over $\boldsymbol{T}$ is independent if every $T_i \in \boldsymbol{T}$ satisfies the following three properties.

1. Logical unit: Execution of T_i is not interrupted by other transactions. Data items accessed by T_i will not be updated by any other transaction $T_j \in \boldsymbol{T}$ ($j \neq i$) until T_i finishes.
2. View consistency unit: T_i views a consistent database. For each $u \in T_i$, there is an $I(D_V^u(T_i))$ satisfiable for $\boldsymbol{C}$, where
 $D_V^u(T_i) = \{x \mid R_i(x) <_i u\}$ and
 $I(D_V^u(T_i)) =$
 $$\left\{ I(x) \left| \begin{array}{l} x \in D_V^u(T_i), I(x) = I(R_i(x)) \ or \\ I(W_i(x)) \ (W_i(x) <_i u) \end{array} \right. \right\}.$$
3. Result consistency unit: T_i keeps the consistency of the database if executed alone. Since $G(\boldsymbol{C}, D_W(T_i))$ includes all of the ground clause affected by T_i, it can be translated to that every ground clause of $G(\boldsymbol{C}, D_W(T_i))$ is true on $I(D(T_i))$.

A schedule H is equivalent-independent (EI for short) if there is an independent schedule H' which is equivalent to H. $\Box$

For a data item x both read, wrote by T_i, the value of x in $I(D_V^u(T_i))$ can be selected from $I(R_i(X))$ or $I(W_i(x))$. The concept of view consistency unit is more generous than the concept that T_i retrieves a consistent database (i.e. $I(D_R(T_i))$ is satisfiable), because $\{I(R_i(x)) \mid R_i(x) <_i u\}$ is no more than one of the choices of $I(D_V^u(T_i))$.

Theorem 2 The execution of an EI schedule preserves the consistency of the database. $\Box$

Proof: For an EI schedule H, there is an independent schedule H' such that $H \equiv H'$. Suppose the resulting database of execution of H is not consistent, that is the resulting database does not satisfy the constraints. There is a ground clause of $\boldsymbol{C}$ which is false on the database $I(\mathcal{DB})$. According to the definition of equivalence of schedules, the ground clause is also false after execution of H'. Since each $T_i \in \boldsymbol{T}$ is a logical unit in H', there must be a $T_i \in \boldsymbol{T}$ which is not a result consistency unit, that is H' is not an independent. *Q.E.D.*

Theorem 3[8] For a schedule H over a set of transactions $\boldsymbol{T}$, there is a schedule H' equivalent to H such that each $T_i \in \boldsymbol{T}$ is a logical unit in H' if and only if the decision graph $DG(H)$ of H is acyclic. The decision graph $DG(H)$ of H is a directed graph, whose nodes corresponds to the operations of $T_i \in \boldsymbol{T}$ and whose edges are constructed as follows.

- If $u <_i v (u, v \in T_i)$, there is an edge $u \rightarrow v$.
- If $W_i(x) <_H R_j(x)$ $(i \neq j)$, there is an edge $W_i(x) \rightarrow R_j(x)$.
- If $R_i(x) <_H W_j(x)$ or $W_i(x) <_H W_j(x)$ $(i \neq j)$, there are edges from all of the operations of T_i to $W_j(x)$. $\Box$

Proof: (if) Since $DG(H)$ is acyclic, it can be topologically sorted. We claim that any topological sort of $DG(H)$ is equivalent to H and there is a topological sort $H' \in$ EL, in which each $T_i \in \boldsymbol{T}$ is a logical unit.
(only if) Let H' be a schedule in which each $T_i \in \boldsymbol{T}$ is a logical unit. Then $DG(H')$ is acyclic. Suppose H is equivalent to H'. For any conflict operations p_i of T_i and q_j of T_j, $p_i <_H q_j$ assures $p_i <_{H'} q_j$. Then $DG(H) = DG(H')$. *Q.E.D.*

In this paper, we use EL class to refer to the set of schedules whose decision graph $DG(H)$ are acyclic.

Example 3 Let us consider the following two cooperative schedules.
View intermediate status H_1:
$W_1(d_1)R_2(d_1)R_2(d_2)W_1(d_2)$
Cooperative execution H_2:
$W_3(d_2)R_4(d_2)R_4(d_3)W_4(d_3)R_3(d_3)W_3(d_1)$
The decision graph $DG(H_1)$ is acyclic, because the edges between operations in different transactions are $W_1(d_1) \rightarrow R_2(d_1)$ and $R_2(d_2) \rightarrow W_1(d_2)$. The serialization graph $SR(H_1)$ of H_1, however, is cyclic due to $W_1(d_1) <_{H_1} R_2(d_1)$ and $R_2(d_2) <_{H_1} W_1(d_2)$. In the same way, the decision graph $DG(H_2)$ is acyclic and the serialization graph $SR(H_2)$ is cyclic. Thus, both H_1 and H_2 are non-serializable but belong to the EL class. $\Box$

The non-serializable schedules such as H_1 and H_2 should be accepted in cooperative database applications [2, 4]. Schedule H_1 was the motivating example in the survey [2] to show the necessity of relaxing serializability correctness criterion.

For a given schedule H, the serialization graph $SG(H)$ is acyclic if and only if the resulting graph after appending the following edges to the decision graph $DG(H)$ is acyclic [8]. *The appended edges are the edges from all of the operations of* T_i *to* $R_j(x)$ *if* $W_i(x) <_H R_j(x)$ $(i \neq j)$. That is, H belongs to the EL class if it is serializable. The converse, however, is not true as shown in Example 3. In a summary, the EL class is more generous than the serializable class by relaxing the restriction made on read operations.

5 Properties of EI Schedules

Let IS be the set of relationships between status,

$$IS = \left\{ (p, q) \,\middle|\, \begin{array}{l} p \leftarrow q \in G(\boldsymbol{C}, \mathcal{DB}), \\ p, q \in \{at_k(o) \mid at_k(o) \in \mathcal{DB}\} \end{array} \right\},$$

and IS^* be the transitive closure of IS. In a consistent database, $I(p)$ is true if $I(q)$ is true and $I(q)$ is false if $I(p)$ is false ($(p, q) \in IS^*$). This fact indicates that a write operation $W_i(at_k(o))$ determines what the other write operations have to do since transaction T_i is a result consistency unit for status constraints.

- Suppose $W_i(at_k(o))$ writes $true$ to $at_k(o)$. For data item p ($(p, at_i(o)) \in IS^*$), there have to be $W_i(p)$ in T_i which writes $true$ if $R_i(p)$ reads $false$.
- Suppose $W_i(at_k(o))$ writes $false$ to $at_k(o)$. For data item p ($(at_i(o), p) \in IS^*$), there have to be $W_i(p)$ in T_i which writes $false$ if $R_i(p)$ reads $true$.

Let T_i^u be such subset of T_i as $\{q \mid u \in T_i, q <_i u\}$. T_i^u is a result consistency unit if the following conditions are satisfied.

- Suppose $W_i(p) \in T_i$ ($R_i(p) \in T_i$) writes (reads) $true$ on data item p. For data item q ($(p, q) \in IS^*$), $W_i(p)$ ($R_i(p)$, respectively) $<_i$ $W_i(q)$ if $W_i(q)$ ($\in T_i$) writes $true$ on q.
- Suppose $W_i(p) \in T_i$ ($R_i(p) \in T_i$) writes (reads) $false$ on data item p. For data item q ($(q, p) \in IS^*$), $W_i(p)$ ($R_i(p)$, respectively) $<_i$ $W_i(q)$ if $W_i(q)$ ($\in T_i$) writes $false$ on q.

For entity constraints, every $T_i^u = \{q \mid u \in T_i, q <_i u\}$ is assured of being a result consistency unit by the following way. Suppose $W_i(e(x))$ requires that values of $at_{k1}(x)$, $at_{k2}(x)$, $\cdots$, $at_{kl}(x)$ have to be changed to satisfy the entity constraints. $W_i(at_{k1}(x))$, $W_i(at_{k2}(x))$, $\cdots$, $W_i(at_{kl}(x))$ may need the change on $at_{h1}(x_1)$, $at_{h2}(x_2)$, $\cdots$, because of status constraints. If we put the write operations of the status into single operation $W_i(\{e(x), at_{k1}(x), at_{k2}(x), \cdots, at_{kl}(x), at_{h1}(x_1), at_{h2}(x_2), \cdots\})$, T_i^u is a result consistency unit.

Definition 5 Let H be a schedule over $\boldsymbol{T}$. In a database with constraints $\boldsymbol{C}$, the schedule H is ordered equivalent-independent (O-EI for short) if $H \in$ EL and each $T_i \in \boldsymbol{T}$ is a view consistency unit and a result consistency unit, where the view of T_i is defined as follows.

$$I(D_V^u(T_i)) = \left\{ I(x) \,\middle|\, \begin{array}{l} I(x) = I(R_i(x)) \ (if\ W_i(x) \not<_i u \vee \\ T_i^{W_i(x)} \ is\ not\ a\ result\ consistency\ unit) \\ or\ I(x) = I(W_i(x)) \ (if\ W_i(x) <_i u, \\ \quad T_i^{W_i(x)} \ is\ a\ result\ consistency\ unit) \end{array} \right\}.$$

□

Note that $I(D_V^u(T_i))$ in the O-EI schedule definition is uniquely determined while $I(D_V^u(T_i))$ in the EI schedule definition has several choices. If $R_i(x), W_i(x) \in T_i$ and $T_i^{W_i(x)}$ is a result consistency unit, $I(x) = I(W_i(x))$ must be used in the $I(D_V^u(T_i))$ after $W_i(x)$ is carried out. Obviously, an O-EI schedule is also an EI schedule.

Theorem 4 For a schedule H over $\boldsymbol{T}$, $T_i \in \boldsymbol{T}$ is a view consistency unit, that is, H is an O-EI schedule if $H \in$ EL and each T_i^u ($T_i \in \boldsymbol{T}, u \in T_i$) is a result consistency unit. □

Proof: Since $H \in$ EL and each $T_i^u (T_i \in \boldsymbol{T}, u \in T_i)$ is a result consistency unit, each $R_i(x) (R_i(x) \in T_i)$ retrieves a consistent database by Theorem 2. If there is a $T_i \in \boldsymbol{T}$ which is not a view consistency unit, there must be $W_j(x) (R_i(x) <_H W_j(x))$ such that T_i also reads the data items which are consistent with the result of $W_j(x)$. However, it contradicts the assumption $H \in$ EL if $i \neq j$. If $i = j$, $W_i(x)$ does not destroy view consistency itself. Hence each $T_j \in \boldsymbol{T}$ is a view consistency unit. *Q.E.D.*

As the result, an EL schedule is also an O-EI one if the transactions are well defined. Since EL class has the pretty feature for cooperative work, the result is a very desirable.

6 Conclusions

Since serializability is unnecessarily strong for cooperative database applications, in this paper, we proposed a new criterion of schedules called equivalent-independence by using the semantics of advanced database applications. Using one of its subclass, we have shown the features of the new criterion: if users work according to the orders of tasks, it allows non-serializable executions by relaxing the restriction on read operations.

References

[1] Agrawal, D. et al.: Consistency and Orderability: Semantics-Based Correctness Criteria for Databases, *ACM Trans. Database Syst.*, Vol. 18, No. 3 pp. 460–486 (1993).

[2] Barghouti, N. S. and Kaiser, G. E.: Concurrency Control in Advanced Database Applications, *ACM Comput. Surv.*, Vol. 23, No. 3, pp. 269–317 (1991).

[3] Bernstein, P. A. et al.: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Publishing Company (1987).

[4] Elmagarmid, A. K.: *Database Transaction Models for Advanced Applications*, Morgan Kaufmann (1992).

[5] Frrg, A. A. et al.: Using Semantic Knowledge of Transactions to Increase Concurrency, *ACM Trans. Database Syst.*, Vol. 14, No. 4 (1989).

[6] Kim, W. et al.: A Transaction Mechanism for Engineering Design Database, *Proc. 10-th Int. Conf. on Very Large Data Bases,* pp. 355–362 (1984).

[7] Xu, H., Furukawa, T. and Shi, Y.: Database Concurrency Control Utilizing Consistency Knowledge (in Japanese), *Trans. IPS Japan*, Vol. 35, No. 12, pp. 2752–2761 (1994).

[8] Xu, H., Furukawa, T. and Shi, Y.: Comparing Equivalent-independent and Serializable Classes by Concurrency Control protocols (in Japanese), *Trans. IPS Japan*, Vol. 37, No. 8, pp. 1600–1609 (1996).