Vol. 35 No. 12 情報処理学会論文誌 Dec. 1994

一貫性情報を用いたデータベースの並行処理制御

徐 海燕[†] 古川哲也[††] 史 一華[†††]

並行処理制御の研究は，並行処理の正当性はスケジュールが直列可能であるということを前提にしてなされてきた．しかし，データベース（DB）の応用分野の拡大により，協同作業を必要としたり長時間の処理を含む高水準 DB では，直列可能性を満足しなければならないことによる問題点が広く認識されている．本論文では，論理式を用いて統一的な判定基準となる一貫性制約を表現することにより，DB が正しいと考えている正当なクラスを従来の直列可能なスケジュールから拡大できることを示す．設計 DB などの高水準 DB において，データの状態を利用することにより，作業手順は DB が一貫するための統一的な判定基準となることを示す．次に一貫性制約で表される統一的な判定基準の下で，独立化可能性という正当なクラスを導入し，それが直列可能性より大きいクラスであることを示す．独立化可能性の判定問題も直列可能性の判定問題と同様に NP 完全となるが，D-独立化可能という多項式時間で判定可能な部分クラスを導入し，それは高水準 DB が必要とする非直列可能なスケジュールを含む実用的なクラスであることを示す．

Database Concurrency Control Utilizing Consistency Knowledge

HAIYAN XU,[†] TETSUYA FURUKAWA[††] and YIHUA SHI[†††]

It is widely recognized that the concept of serializability is inadequate for advanced database systems. By associating status with each data, in this paper, we show the consistency of advanced databases can be validated by the integrity constraints expressed by logic programs. Based on the integrity constraints, we introduce a new criterion of schedules called equivalent-independent : " a schedule is correct if it is equivalent to any independent schedule of the set of transactions." We prove the correctness of our new criterion and indicate it is more generous than the serializability criterion. Since the problem of deciding whether a given schedule is equivalent-independent is also NP-complete, its polynomial time recognizable subclass D-equivalent-independent is introduced and we show that it includes schedules which are non-serializable but required by advanced database systems.

1. はじめに

データベース（DB）システムの全体の利用効率を向上させるため，複数個の処理単位を同時に並行して扱う必要がある．しかし，各々の処理単位に用いられるデータに重なりがあれば，共有データによる処理単位間の干渉によって種々の問題が生じることになる．それを防止するために，並行処理制御が研究されてきている．

特に変更操作を含む処理単位の並行実行は，DB の一貫性を破壊するかもしれない．DB が一貫しているかどうかを判断するのに，次の二つの方法が考えられる．

（1）処理単位が正しいものと仮定する方法．すなわち，各処理単位を一つずつ逐次的に実行する直列実行の結果が DB の一貫性を保持しているなら，直列可能である（実行結果がある順序での直列実行と等価となる）実行の結果も必ず DB の一貫性を保持できる．

（2）統一的な判定基準を定め，データベース管理システム（DBMS）が判定を行う方法．

従来の事務 DB においては，統一的な判定基準を定めにくい[1]ことと，処理単位が短いので方法（2）の判定のためのオーバヘッドをなるべく避ける必要がある[5]ことから，方法（1）が用いられてきた．

しかし，計算機技術の進歩により，様々な分野で DB が利用されるようになり，協同作業を必要とする

† 福岡工業大学工学部電子工学科
Department of Electronics, Fukuoka Institute of Technology
†† 九州大学経済学部経済工学科
Department Economic Engineering, Faculty of Economics, Kyushu University
††† 福岡工業短期大学
Fukuoka Junior College of Technology

応用分野や長時間の処理を含むような場合においては，直列可能性を満足しなければならないことによる問題点が指摘されている[1),5),7),10)]．これは，長大な処理単位に対しても直列可能なスケジュールしか許可できなければ，並行性の低下もしくは後退復帰の増加を招くことになることによる．特に，協同作業などの特徴を持つ設計DBにおいては，さらに問題が深刻である．この問題は広く認識されており，様々な角度から研究されてきている[1),6),8)~10)]．しかし，作業手順によって定義される統一的な判定基準というDBの一貫性に関する情報を利用する立場から，非直列可能なスケジュールにも正当なものが存在するかどうかに関する研究はまだ知られていない．各々の処理単位の個別の意味論を利用することによる研究[6),8),9)]はなされているが，それには処理単位の意味論を定義するための利用者の負担やDBMSがその意味論を利用するためのオーバヘッドが大きいなどの問題がある．

本論文では，設計DBなどの高水準DBについて，作業の順序と作業における制約を表す作業手順によって定義される状態の変化規則が，DBが一貫するための統一的な判定基準となることを示す．そして，次のように，一貫性制約で表現される統一的な判定基準の下でスケジュールの正当性問題を検討する．

(1) 統一的な判定基準の下でのスケジュールの正当性基準である独立化可能性を提案する．

(2) 独立化可能性は，従来の直列可能性より大きいクラスであることを示す．

(3) D-独立化可能という多項式時間で判定可能な部分クラスを導入し，それが高水準DBにおいて指摘された問題を解決している実用的なクラスであることを示す．

本論文は，次のように構成される．2章は，設計DBという高水準DBを例にして，設計手順によって定義される状態の変化規則が，設計DBが一貫しているかどうかを判定するための統一的な基準となることを示す．3章では，並行処理制御に関する基本的事項を定義する．4章では，独立化可能性という統一的な判定基準の下でのスケジュールの正当性基準を提案し，直列可能なスケジュールより範囲を拡大したことを示す．また，独立化可能性の判定問題の計算複雑さなどの問題も検討する．5章では，D-独立化可能という多項式時間で判定可能な部分クラスを導入し，D-独立化可能性の実用上の意義を示す．6章は，全体のまとめである．

2. データの状態に基づく統一的な判定基準

本章では，設計DBを例として，データの状態を利用することにより，高水準DBにおいてDBの一貫性を統一的な判定基準で制御できることを示す．さらに，一貫性制約という統一的な判定基準の一つの表現方法を導入し，一貫性制約上で関連するデータ集合を分析する．

定義 1 DBは，オブジェクトの集合Oと関連集合Rの組$\langle O, R\rangle$である．

- オブジェクト集合Oで，各オブジェクトは〈識別子，実体，属性の列〉の組で表される．IをOに対する識別子の集合とすると，各$i\in I$に対して，o_iでそのオブジェクトを識別し，$e(o_i)$でo_iの実体，$at_k(o_i)$ $(k=1,\cdots,m)$でo_iのk番目の属性を表す．
- 関連集合Rで，

$$R=\left\{r(o_i,o_j)\,\middle|\,\begin{array}{l}o_i\text{ から }o_j\text{ への種類 }r\\ \text{の関連が存在する}\end{array}\right\}$$

とする． □

議論を簡単にするため，属性の種類をm以内としている．また，すべての関連を2項関連で表す．n項関連も変数を導入することにより，2項関連で表せるため，一般性は失われない．

例えば，設計DBにおいては，異なる設計段階で生成される設計結果が含まれ，それらの設計結果はさまざまなテストを受けなければならないし，複数種類の制約を満たさなければならない．また，部品階層などに関する情報を記述する関連も存在する．従って，設計DB内の各オブジェクトo_iに対して，実体$e(o_i)$は設計結果を，属性$at_k(o_i)$は実体$e(o_i)$がk $(k\in\{1,\cdots,m\})$番目のテストを通過しているかどうかや，k番目の制約を満たしているかどうかというような情報を記述している．

さらに，設計作業は一定の手順に従って行わなければならない．例えば，

(1) 実体xが単体テストを通過しなければ，xを含む実体zが同種類の複合テストを通過できない，

(2) 1番目のテストを通過しなければ，2，3番目のテストを通過できない，

(3) i番目とj番目のテストが同時に通過できないテストである，

などがある．定義1のようにオブジェクトの属性を明記することにより，作業手順は，論理式によって表現できる．

定義 2 節とは，次のような論理式である．

$B_1, \cdots, B_m \leftarrow A_1, \cdots, A_n$

$(\equiv \neg(A_1 \wedge \cdots \wedge A_n) \vee (B_1 \vee \cdots \vee B_m))$

ここで，$B_1, \cdots, B_m, A_1, \cdots, A_n$ は素論理式 (atomic formula) であり，$n \geq 0$ かつ $m \geq 0$ である ($B \leftarrow$ のような節を簡単に B で書き，事実という)．素論理式 $A_1, \cdots, A_n$ はこの節の積条件といい，$B_1, \cdots, B_m$ を和結論という．素論理式とは，$p(t_1, \cdots, t_l)$ という形をした表現である．ここで，p は l 変数の述語記号，$t_1, \cdots, t_l (l \geq 1)$ は次のように構成される項である．

- 変数記号，または定数記号 $o_j \in O$ は項である．
- 1変数の関数記号 f に対して，s が項であれば，$f(s)$ も項である． □

作業手順を記述するのに，オブジェクトからその k 番目の属性への関数 at_k ($k \in \{1, \cdots, m\}$) を関数記号として用いる必要がある．ただし，属性の値域が異なる場合，素論理式に対して一般的な議論をするのは非常に困難である．さらに，どのような，どれくらいの属性を明記すればよいのかという問題も存在する．そこで，議論を簡単化するため，本論文では，属性の中から，作業手順と最も密接に関連している属性，状態だけに注目する．状態とは，値域を真理値とする属性のことをいう．ただし，否定情報の取り扱いは閉世界仮説を利用する．すなわち，O の状態集合を M とすると，真理値が真の状態しか M に記述しない．さらに，状態 $at_k(x)$ の値は真理値であるため，定義2の素論理式 $p(at_k(x))$ のかわりに，直接 $at_k(x)$ を素論理式として利用する．

例えば，上記で例示した設計 DB において，

$$M = \left\{ at_k(o_i) \middle| \begin{array}{l} o_i \in O, k \in \{1, \cdots, m\}, \\ e(o_i) \text{ は } k \text{ 番目のテストを通過} \end{array} \right\}$$

とすると，設計手順(1)，(2)，(3)は，それぞれ次のように表現できる．

(1)：$at_k(x) \leftarrow at_k(z), in(x, z). \ (k = 1, 2, \cdots, n)$

(2)：$at_1(x) \leftarrow at_2(x)$；$at_1(x) \leftarrow at_3(x)$.

(3)：$\leftarrow at_i(x), at_j(x)$.

ただし，$in(x, y)$ は y が x を含むという部品階層に関する述語である．さらに，

$$gt(t_1, t_2) = \begin{cases} \text{真} & t_1 < t_2 \\ \text{偽} & \text{そうでない} \end{cases}$$

という素論理式を利用すると，各種のテストを行う時にテスト結果が入力されるまでテスト対象が変更されなければ，各 $at_k(x) \in M$ が現時点の $e(x) (x \in O)$ の状態を記録している要求は，$gt(t_c(x), t_c(at_k(x)))$ $(k = 1, 2, \cdots, n)$ という事実で表現できる．ただし，$t_c(x)$/$t_c(at_k(x))$ は実体 $e(x)$/状態 $at_k(x)$ の作成時刻を記述する関数である．

作業手順は通常，(a)関連する実体の状態間の満たすべき性質と，(b)同一実体の異なる種類の状態間の満たすべき性質を要求する．例えば，上記の(1)は(a)，(2)と(3)は(b)の要求を表している．

定義 3 DB$\langle O, R \rangle$ における一貫性制約 C は，次のような(a)関連する実体の状態間，(b)同一実体の異なる種類の状態間，(c)実体とその状態間の満たすべき性質を表す節の集合で定義される $(k, n, l, w \geq 0)$.

(a) $at_{i_1}(x), \cdots, at_{i_k}(x), at_{i_{k+1}}(y), \cdots, at_{i_{k+l}}(y) \leftarrow$
$at_{j_1}(y), \cdots, at_{j_n}(y), at_{j_{n+1}}(x), \cdots, at_{j_{n+w}}(x), r(x, y).$
$(at_{i_1}(x), \cdots, at_{i_k}(x), at_{i_{k+1}}(y), \cdots, at_{i_{k+l}}(y) \leftarrow$
$at_{j_1}(y), \cdots, at_{j_n}(y), at_{j_{n+1}}(x), \cdots, at_{j_{n+w}}(x), r(y, x).)$

(b) $at_{i_1}(x), \cdots, at_{i_k}(x) \leftarrow at_{j_1}(x), \cdots, at_{j_n}(x).$

(c) $p(f(x), g(at_i(x)))$. □

節 $B_1, \cdots, B_m \leftarrow A_1, \cdots, A_n \equiv \neg(A_1 \wedge \cdots \wedge A_n) \vee (B_1 \vee \cdots \vee B_m) \equiv \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$ であるため，1階述語論理の表現力を持つ和積標準形は，節集合でも表現できる．従って，上記のような一貫性制約(a)と(b)によって，1階述語論理で表現できる関連する実体の状態間または同一実体の異なる種類の状態間の性質を表現できる．本論文では，このような1階述語論理で表現できる一貫性制約を持つ高水準 DB におけるスケジュールの正当性問題を検討する．

このため，設計 DB を例として，設計手順を表している一貫性制約と DB の一貫性の関係を考察する．設計作業の目的は，一定の設計手順に沿って各種のテストを通過でき，各種の制約を満足できる最終的な結果を作成することである．設計 DB は設計過程で生成される様々な中間データを統一的に管理し，設計作業の目的の達成を支援する．このため，設計 DB 内のデータに対して，最終的な設計結果の満たすべき性質は，(i)各種のテストの通過や制約の満足というような作業の進行にともなって徐々に満たしてゆく必要のある性質と，(ii)設計手順のように常時満たす必要のある性質，という2種類に分けられる．設計データが(i)に分類される性質を満たしているかどうかは，データの状態としてデータの持つ属性の値で記述することができる．そのような属性値が正しくデータの状態を表しているかどうかは，データの名称，作成年月日等の他の一般の属性値が正しい値であるかと同様，設計

DB の一貫性管理から切り離して考えるべきである．従って，設計データが設計手順によって要求される各種の状態間の制約という(ii)に分類される性質を満たせば，設計 DB の一貫性管理の役割を果たしたことになる．すなわち，設計手順を満たすことが設計 DB の一貫性と考えることができ，設計手順が統一的な判定基準となる．また，設計手順とその記述の長さは，通常作成されるデータの量とは直接関係しない．

作業手順を，DB における一貫性制約 C と定義することにより，DB$\langle O, R\rangle$ が一貫していることを，$\langle O, R\rangle$ が C を満たすことで表現できる．$\langle O, R\rangle$ が C を満たすことは，C の各節中の変数 x, y について O 中の任意のオブジェクト o_i, o_j で置き換えて得られた節が $M\cup R$ において真となることである．$\langle O, R\rangle$ の部分集合 $\langle O', R'\rangle(O'\subseteq O, R'\subseteq R)$ が一貫していることも，同様に $\langle O', R'\rangle$ が C を満たすことで表される．また，一貫している $\langle O, R\rangle$ の任意の部分集合 $\langle O', R'\rangle$ $(O'\subseteq O, R'\subseteq R)$ は C に違反しないという．

DB $\langle O, R\rangle$ に対して，本論文では，各オブジェクト $o_i\in O$ の実体 $e(o_i)$ と属性 $at_k(o_i)$ を利用者による操作対象とする．$U=\{e(o_i)\mid o_i\in O\}$ とすると，関連 $r(o_i, o_j)\in R$ は $e(o_i)$, $e(o_j)\in U$ の生成／削除に伴って生成／削除されるので，本論文では，関連に対する操作を陽に扱わず，$V=U\cup M$ を利用者によって操作されるデータ項目集合とする．従って，DB$\langle O, R\rangle$ に対して，$V'\subseteq V$ が一貫性制約 C を満たす（C に違反しない）ことは，$V'\cup R$ が C を満たす（C に違反しない）ことを意味する．

次では，$V'\subseteq V$ が一貫性制約 C を満たす（C に違反しない）ことを検査するために，$V'\cap V_1, \cdots, V'\cap V_n$ が C を満たすかどうか（C に違反しないかどうか）のみを検査すればよいような V の部分集合 $V_1, \cdots, V_n$ を求める．そのために，まず，M を $M_1, \cdots, M_n$ に分割する．V' が一貫性制約 C 中の種類(a)と(b)の節を満たすかどうかの検査は，各節に現れる状態集合に対して行っているため，C 内の同じ節に現れる状態は同じ M_i に分割する必要がある．そのため，同じ節の和結論と積条件に現れる状態間に関連 IS (Influence Set)，同じ節の和結論または積条件に現れる状態間に関連 RS(Related Set) を定義する．

定義 4　状態間の関連 IS，RS は次のものである．

$$
\begin{aligned}
IS=\{&(at_{i_u}(o_s), at_{j_v}(o_t)), (at_{i_\eta}(o_t), at_{j_\xi}(o_s)),\\
&(at_{i_u}(o_s), at_{j_\xi}(o_s)), (at_{i_\eta}(o_t), at_{j_v}(o_t))\mid\\
&at_{i_1}(x), \cdots, at_{i_k}(x), at_{i_{k+1}}(y), \cdots, at_{i_{k+l}}(y)\leftarrow\\
&at_{j_1}(y), \cdots, at_{j_n}(y), at_{j_{n+1}}(x), \cdots, at_{j_{n+w}}(x), r(x, y).\\
&\in C, 1\le u\le k, k+1\le v\le k+n, 1\le\eta\le l,\\
&n+1\le\xi\le n+w, o_s, o_t\in O, r(o_s, o_t)\in R\}\\
\cup\{&(at_{i_u}(o_s), at_{j_v}(o_s))\mid at_{i_1}(x), \cdots, at_{i_k}(x)\\
&\leftarrow at_{j_1}(x), \cdots, at_{j_n}(x).\in C,\\
&1\le u\le k, 1\le v\le n, o_s\in O\}
\end{aligned}
$$

$$
\begin{aligned}
RS=\{&(at_{i_\eta}(o_s), at_{i_\xi}(o_s)), (at_{i_\sigma}(o_t), at_{i_\tau}(o_t)),\\
&(at_{i_\eta}(o_s), at_{i_\sigma}(o_t)), (at_{j_u}(o_t), at_{j_v}(o_t)),\\
&(at_{i_\sigma}(o_t), at_{i_\eta}(o_s)), (at_{j_\lambda}(o_s), at_{j_u}(o_t)),\\
&(at_{j_\lambda}(o_s), at_{j_\mu}(o_s)), (at_{j_u}(o_t), at_{j_\lambda}(o_s)),\mid\\
&at_{i_1}(x), \cdots, at_{i_k}(x), at_{i_{k+1}}(y), \cdots, at_{i_{k+l}}(y)\leftarrow\\
&at_{j_1}(y), \cdots, at_{j_n}(y), at_{j_{n+1}}(x), \cdots, at_{j_{n+w}}(x), r(x, y).\\
&\in C, 1\le\eta, \xi\le k, 1\le u, v\le n, r(o_s, o_t)\in R,\\
&k+1\le\sigma, \tau\le k+l, n+1\le\lambda, \mu\le n+w, o_s, o_t\in O\}\\
\cup\{&(at_{i_\eta}(o_s), at_{i_\xi}(o_s)), (at_{j_u}(o_s), at_{j_v}(o_s))\mid\\
&at_{i_1}(x), \cdots, at_{i_k}(x)\leftarrow at_{j_1}(y), \cdots, at_{j_n}(y).\in C,\\
&1\le\eta, \xi\le k, 1\le u, v\le n, o_s\in O\}
\end{aligned}
$$

□

V' が一貫性制約 C 中の種類(a)と(b)の節に違反しない検査は，V' と V' において積条件が真となる節の和結論部分も真であるという推論結果の全体が各節を偽にしないことの検査であるので，$\alpha\in M$ と α からの推論結果も同じ M_i に分類する必要がある．α から推論できる $\beta\in M$ は $(\beta, \alpha)\in IS^*$ (IS^* は IS の反射的推移閉包) に含まれるので，$(\alpha, \beta)\in IS^*$ あるいは $(\beta, \alpha)\in IS^*$ のような α, β を同じ M_i に分割すればよい．このように M を分割しておけば，C 中の種類(3)の節を利用することによって，$V=U\cup M$ まで分割できる．

定義 5　V, M の C に対する被覆 V_i, $M_i(i=1, \cdots, n)$ とは，次のように構成されるものである．

- M_i' は，$\forall\alpha, \beta\in M_i'$, s.t. $(\alpha, \beta\in M)\wedge(((\alpha, \beta)\in IS^*)\vee((\beta, \alpha)\in IS^*))$ となる M の極大部分集合である．
- $M_i=M_i'\cup\{\beta\mid\beta\in M, \exists\alpha\in M_i'((\alpha, \beta)\in RS)\}$
- $V_i=M_i\cup\{e(x)\mid x\in O, \exists\alpha\in M_i((\alpha, x)\in MU)\}$.

ただし，$MU=\{(at_k(o_j), o_j)\mid p(f(x), g(at_k(x)).\in C, o_j\in O\}$ である．□

例 1　実体と状態間の一貫性，単体テストと複合テスト間の一貫性を要求する一貫性制約を，

$$
C=\left\{\begin{array}{l} gt(t_c(x), t_c(at_k(x))).\\ at_k(x)\leftarrow at_k(y), in(x, y).\ (k=1, \cdots, 4)\end{array}\right\}
$$

とし，

$$
\begin{aligned}
&O=\{o_1, o_2, o_3\},\\
&M=\{at_k(o_i)\mid o_i\in O, k=1, 2, 3, 4\},\\
&R=\{in(o_2, o_1), in(o_3, o_1)\}
\end{aligned}
$$

とすると，

$RS=\phi$；

$MU=\{(at_k(o_i), o_i) \mid o_i \in O, k=1, 2, 3, 4\}$；

$$IS=\left\{(at_k(o_i), at_k(o_j)) \middle| \begin{array}{l} o_i, o_j \in O, \\ at_k(o_i), at_k(o_j) \in M, \\ in(o_i, o_j) \in R, \ k=1, 2, 3, 4 \end{array}\right\}$$

となる．このため，$\{e(o_1), at_k(o_1), e(o_2), at_k(o_2)\}$ $(k=1, \cdots, 4)$，$\{e(o_1), at_k(o_1), e(o_3), at_k(o_3)\}$ $(k=1, \cdots 4)$ が V の C に対する被覆 $V_1, \cdots, V_8$ となる． □

部品関連 *in* を持つ実体の状態間の一貫性が要求される場合に，IS は関連 *in* を持つ実体の状態間の関連集合となる．オブジェクトは通常複数の部品階層に分けられるので，この例から分かるように，$V=U \cup M$ は複数個の C に対する被覆 $V_1, \cdots, V_n$ に分けられる．

補題 1 $V_1, \cdots, V_n$ を $V=U \cup M$ の C に対する被覆とする．$V' \subseteq V$ が一貫性制約 C を満たす（C に違反しない）ための必要十分条件は，$V' \cap V_1, \cdots, V' \cap V_n$ が一貫性制約 C を満たす（C に違反しない）ことである． □

証明：必要条件は明らかなので，十分条件を証明する．

（a）一貫性制約を満たす証明：

V_i の定義により，C 内の同じ節に現れる $U \cup M$ の要素は同じ V_i に分割されている．V' が C を満たすかどうかの検査は C の各節に現れる $U \cup M$ の要素に対して行うため，結論が成り立つ．

（b）一貫性制約に違反しない証明：

V_i の定義により，V_i の任意の部分集合による推論結果も同じ V_i に分割される．このため，$V_i \cap (V' \cup R$ による推論結果$) = ((V_i \cap V') \cup R$ による推論結果$)$ $(i=1, \cdots, n)$ である．従って，上記の（a）の証明を利用すると，結論が成り立つ． □

3. 並行処理制御に関する基本的事項

本章では，並行処理制御に関する基本的事項を定義する．ただし，本論文では，DB$\langle O, R \rangle$ に対して，関連 R に対する操作を陽に扱わないため，データ項目集合 $V=U \cup M (U=\{e(o_i) \mid o_i \in O\}$，$M=\{at_k(o_i) \mid o_i \in O, k \in \{1, \cdots, m\}\})$ で識別される集合に対する操作のみを考慮する．

定義 6 $R_i(X)/W_i(X)$ で T_i のデータ X に対する読み出し/書き込み操作を表すと，処理単位 T_i は，$T_i \subseteq \{R_i(X), W_i(X) \mid X \subseteq V\}$，かつ任意の p，$q \in T_i$ に対して，$p <_i q$ か $q <_i p$ の全順序関連 $<_i$ を持つ操作集合である．ただし，$(R_i(X), R_i(Y) \in T_i) \vee (W_i(X), W_i(Y) \in T_i)$ なら，$X \cap Y=\phi$ であり，$W_i(X) <_i R_i(Y)$ $(X \cap Y \neq \phi)$ のような $R_i(Y)$ は存在しない． □

以降，f_i で T_i の $<_i$ の下での最終操作を記述する．

定義 7 $T=\{T_1, T_2, \cdots, T_n\}$ を処理単位の集合とすると，T 上のスケジュール H は，下記のような推移的全順序関連 $<_H$ を持つ操作集合である．

（1） $H=\bigcup_{i=1}^{n} T_i$，

（2） $<_H \supseteq \bigcup_{i=1}^{n} <_i$． □

一般に，$X \cap Y \neq \phi$，$i \neq j$ の $R_i(X)$ と $W_j(Y)$，$W_i(X)$ と $W_j(Y)$ を競合操作という．

定義 8 $T=\{T_1, T_2, \cdots, T_n\}$ 上の二つのスケジュール H と H' が等価（$H \equiv H'$ と記する）とは，次の二つの条件を満たした時にいう[3),11)]．

（1）任意の $T_j \in T$ に対して，H において T_j によって読み出されるデータ項目 $(\cup R_j(X) \in H)$ と，H' における T_j によって読み出されるデータ項目 $(\cup R_j(X) \in H')$ の値が同一である．

（2）任意の T_0 と任意の計算を行う各 $T_j \in T$ に対して，$R_0W_0(V)HR_f(V)W_f$ において T_f によって読み出されるデータ項目と，$R_0W_0(V)H'R_f(V)W_f$ において T_f によって読み出されるデータ項目の値が同一である． □

すなわち，H と H' が各 $T_i \in T$ の利用者からもDBMSからも同一と見られれば，等価である．与えられた H が，ある直列スケジュール H' と等価であれば，H は直列可能であるという．しかし，直列可能性の判定問題はNP完全である[2)~4),11)]．そのため，実際に多項式時間で判定できる部分クラスである D-直列可能性が利用されている[11)]．

定義 9 T 上のスケジュール H が，次のような隣合う操作の入替え（D-等価交換）で H' に変換される場合に，H と H' は D-等価であるといい，$H \sim H'$ で記述する．

（1） $R_i(X)R_j(Y)$ 操作の場合，

（2） $R_i(X)W_j(Y)$ が $i \neq j$，$X \cap Y=\phi$ を満たす場合，

（3） $W_i(X)W_j(Y)$ や $W_i(X)R_j(Y)$ で $X \cap Y=\phi$ を満たす場合． □

$\sim^*$ を $\sim$ の反射的推移閉包とすると，与えられた T 上の H が，ある直列スケジュール H' に対して $H \sim^* H'$ であれば，H は D-直列可能であるという．

D-直列可能性は競合保存直列可能性 (conflict serializability) とも呼ばれている[2]. D-直列可能であれば直列可能であるのは自明であるが, 直列可能で D-直列可能でないものが存在することが知られている. D-直列可能性の判定は, 次の判定グラフによって行われる.

D-直列可能性判定グラフ $D(H)$ は, 各 $T_i \in T$ に対応して一つの節点 T_i を持ち, T_i から T_j $(i \neq j)$ への枝は, 次の条件が満たされる時に作られる.

(1) $R_i(X) <_H W_j(Y), X \cap Y \neq \phi$,

(2) $(W_i(X) <_H R_j(Y)) \vee (W_i(X) <_H W_j(Y)), X \cap Y \neq \phi$.

定理 1[11] スケジュール H が D-直列可能であるための必要十分条件は, H に対応する D-直列可能性判定グラフ $D(H)$ が閉路を持たないことである. これにより, D-直列可能性の検査は多項式時間で可能となる. □

ただし, どの $W_i(X) \in T_i \in T$ に対しても, 必ず $\exists R_i(X)(R_i(X) <_i W_i(X))$ ならば, T 上のスケジュールの直列可能性問題と D-直列可能性問題は等価となる[11].

4. 統一的な判定基準の下でのスケジュールの正当性

本章では, 一貫性制約という統一的な判定基準の下におけるスケジュールの正当性基準を提案し, その性質を検討する.

4.1 独立化可能性

本節では, 利用者と DBMS という二つの立場からスケジュールの正当性基準を検討する.

定義 10 一貫性制約 C という統一的な判定基準の下で T 上のスケジュール H が独立であるとは, 各 $T_i \in T$ が次の三つの性質を満たした時に限る.

- 検索一貫性: T_i によって読み出されるデータ項目が, 一貫性制約 C に違反しない.
- 結果一貫性: T_i の終了時に, T_i によって変更されたデータ項目と, T_i によって読み出され変更されていないデータ項目の全体が, 一貫性制約 C を満たす.
- 論理性: T_i によって操作されたデータ項目は, T_i が終了するまで, 他の T_j $(j \neq i)$ によって変更されていない. すなわち, $\forall (R_i(X) <_H W_j(Y)) \vee (W_i(X) <_H W_j(Y))$ $(i \neq j, X \cap Y \neq \phi)$ に対して, $f_i <_H W_j(Y)$ である. □

T_i によって変更されたデータ項目が DB の一貫性を破壊してないことを保証するためには, T_i はその変更結果が一貫性制約を満たす検査に必要なデータ項目を読み出す必要がある. 結果一貫性は, そのようなデータ項目の読み出しを要求している. 論理性は, 従来の処理単位の論理単位 (logical unit) 概念そのものであり[1),9)], 各 T_i が検索された DB から結果 DB への独立遷移であることを要求する. 従って, 論理性に検索一貫性と結果一貫性を加えると, 独立スケジュールにおける各 T_i は, 一貫した DB から一貫した DB への独立遷移となる. このため, 独立スケジュールは利用者に対する正当なスケジュールである.

従来の直列可能の定義と同様に, 独立化可能なスケジュールを定義する.

定義 11 $H \equiv H'$ である独立スケジュール H' が存在すれば, H は**独立化可能**である. □

等価となる利用者に対する正当な H' が存在するため, 独立化可能な H も利用者に対する正当なスケジュールである.

定義 12 独立化可能なスケジュールは, 利用者に対する正当なスケジュールという. □

次に, DBMS に対する正当なスケジュールという概念を定義する.

定義 13 DBMS に対する正当なスケジュール H とは, $R_0W_0(V)HR_f(V)W_f$ において, 結果一貫性を満たす T_0 に対して, T_f が必ず検索一貫性を満たすスケジュールである. □

すなわち, 一貫した DB を一貫した DB へ遷移させたスケジュール H が, DBMS に対する正当なスケジュールである. 利用者と DBMS に対する正当なスケジュールという二つの概念を合わせると, 一般的な意味での正当なスケジュールの概念を定義できる.

定義 14 **正当なスケジュール**とは, 利用者と DBMS の両方に対する正当なスケジュールである. □

DBMS に対する正当なスケジュールは次の性質を持つ.

補題 2 独立化可能なスケジュール H は, *DBMS* に対する正当なスケジュールである. □

証明: H' を $H \equiv H'$ である独立スケジュールとする. H が DBMS に対する正当なスケジュールでないとすると, スケジュール $R_0W_0(V)H'R_f(V)W_f$ において, T_f は検索一貫性を満たさない, すなわち, H' の終了時の DB が一貫性制約 C 中のある節を満たさないことになる. T_0 が結果一貫性を満たすため, その節に

現れたデータ項目を変更した T 中の処理単位 $T_{i_1}, \cdots, T_{i_m} \in T$ が存在する．$T_{i_1}, \cdots, T_{i_m} \in T$ は論理性を満たすため，その節を偽にした $T_{i_j} \in \{T_{i_1}, \cdots, T_{i_m}\}$ が存在することになる．しかし，これは $T_{i_j} \in T$ が結果一貫性を満たさないことになるので，H' の定義と矛盾する． □

補題2は，利用者に対する正当なスケジュールは，必ず DBMS に対する正当なスケジュールでもあることを意味する．従って，次の定理が成り立つ．

定理 2 スケジュール H が**正当**であるための必要十分条件は，H が独立化可能であることである． □

証明：(必要性) H が正当であれば，定義 14 より，H は利用者と DBMS の両方に対する正当なスケジュールである．利用者に対する正当なスケジュールは独立化可能なので，H は独立化可能である．

(十分性) H が独立化可能であれば，定義12より，利用者に対する正当なスケジュールである．また，補題2より，独立化可能な H は必ず DBMS に対する正当なスケジュールでもある．従って，定義 14 より，H は正当なスケジュールである． □

4.2 独立化可能性の性質

本節では，独立化可能性の性質を検討する．ただし，これからの議論は，各 $T_i \in T$ が単独実行されるならば必ず検索一貫性と結果一貫性を満たすという仮定の下で行う．まず，独立化可能なスケジュールと直列可能なスケジュールの範囲を比較する．

補題 3 T 上のスケジュール H は，直列可能であれば，独立化可能である． □

証明：H が直列可能であれば，$H \equiv H'$ となる直列スケジュール H' が存在する．H' において，各 $T_i \in T$ は論理性，結果一貫性，検索一貫性を満たすので，H' は独立スケジュールである．従って，H は独立スケジュール H' と等価なので，独立化可能である． □

次に，一貫性制約 C と独立化可能性の関わりを取り扱う．

定義 15 $V_1, \cdots, V_n$ を V の C に対する被覆とすると，スケジュール H の C に対する部分スケジュール H^j $(j=1, \cdots, n)$ は，次の T^j 上の推移的全順序関連 $<_{H^j}$ を持つ操作集合である．

- $T^j = \cup_i T_i^j$．ただし，T_i^j は $T_i \in T$ の C に対する部分処理単位である：

$$T_i^j = \left\{ A_i(X) \,\middle|\, \begin{array}{l} A \in \{R, W\}, A_i(X) \in T_i, \\ X \cap V_j \neq \phi, \end{array} \right\},$$

$$<_i^j = \{p_s <_i^j q_t \mid p_s, q_t \in T_i^j, p_s <_i q_t\}.$$

- $<_{H^j} = \{p_i <_{H^j} q_k \mid p_i, q_k \in T^j, p_i <_{H} q_k\}$． □

定義15と補題1により，次の結論が得られる．

補題 4 T 上の H において各 $T_i \in T$ が検索一貫性，結果一貫性を満たすための必要十分条件は，H の C に対する部分スケジュール H^j $(j=1, \cdots, n)$ において各 $T_i^j \in T^j$ とも検索一貫性，結果一貫性を満たすことである． □

例 2 $X \cap Y = \phi$, $(X \cup Y) \cap Z = \phi$ とする．$X \cup Z$ と $X \cup Y$ が V の C に対する被覆 V_1 と V_2 に分割される．すなわち，$X \cup Z \subset V_1$, $X \cup Y \subset V_2$, $Y \not\subset V_1$, $Z \not\subset V_2$ であるとすると，スケジュール $H = R_i(X)R_i(Y)W_i(Y)R_j(X)R_j(Y)R_j(Z)W_j(Z)R_i(Z)W_i(Z)$ において，$T_i^1 = R_i(X)R_i(Z)W_i(Z)$, $T_i^2 = R_i(X)R_i(Y)W_i(Y)$, $T_j^1 = R_j(X)R_j(Z)W_j(Z)$, $T_j^2 = R_j(X)R_j(Y)$ となり，$H^1 = R_i(X)R_j(X)R_j(Z)W_j(Z)R_i(Z)W_i(Z)$, $H^2 = R_i(X)R_i(Y)W_i(Y)R_j(X)R_j(Y)$ となる．明らかに，T_j は論理性を満たし，$(X \cup Y) \cap Z = \phi$ であるため，T_i も論理性を満たす．このため，$T_i^1, T_i^2, T_j^1, T_j^2$ がそれぞれ検索一貫性と結果一貫性を満たせば，H は独立化可能である．しかし，$W_j(Z) <_H R_i(Z)$ かつ $W_i(Y) <_H R_j(Y)$ であるため，H は D-直列可能でない．また，どの $W_i(X) \in T_i \in T$ に対しても，$\exists R_i(X)(R_i(X) <_i W_i(X))$ なので，H は直列可能でもない． □

従って，DBMS が各々の処理単位の個別の意味論を把握していなくても，一貫性制約という DB の一貫性の統一的な判定基準を利用することにより，次のように独立化可能性を制御できる．

定理 3 T 上の H の C に対する部分スケジュール H^j $(j=1, \cdots, n)$ が直列可能で，かつ $\exists H' \equiv H$ (H' において各 $T_i \in T$ が論理性を満たす) ならば，H は独立化可能である． □

証明：補題3により，直列可能な H^j $(j=1, \cdots, n)$ は独立化可能なので，H^j における各処理単位は検索一貫性と結果一貫性を満たすことになる．$H' \equiv H$ より $H'^j \equiv H^j$ であるため，H'^j $(j=1, \cdots, n)$ における各処理単位も検索一貫性と結果一貫性を満たす．従って，補題4により，H' における各処理単位は検索一貫性と結果一貫性を満たす．

まとめると，H^j $(j=1, \cdots, n)$ が直列可能で，かつ $\exists H' \equiv H$ (H' において各 $T_i \in T$ が論理性を満たす) 時，H' において各 $T_i \in T$ とも検索一貫性，結果一貫性，論理性を満たすので，H' は独立スケジュールである．$H' \equiv H$ であるため，H は独立化可能である． □

例 3 例2の $H=R_i(X)R_i(Y)W_i(Y)R_j(X)R_j(Y)R_j(Z)W_j(Z)R_i(Z)W_i(Z)$ において，C に対する部分スケジュール $H^1=R_i(X)R_j(X)R_j(Z)W_j(Z)R_i(Z)W_i(Z)$, $H^2=R_i(X)R_i(Y)W_i(Y)R_j(X)R_j(Y)$ はともに直列可能である．例2で示したように，H において T_i, T_j とも論理性を満たすので，この非直列可能な H は，独立化可能である．すなわち，例2の H は，独立化可能なスケジュールが直列可能なスケジュールより範囲を拡大している実例である． □

直列可能性の判定問題がNP完全であることはよく知られており，同様に独立化可能性の判定問題を考察する．$L(T)=\{H|T$ 上の H に対して，各 $T_i\in T$ が論理性を満たす $H'\equiv H$ が存在する$\}$ とする．

定理 4 与えられた T 上のスケジュール H に対して，$H\in L(T)$ かどうかの判定問題は，NP 完全である． □

証明：$T_i\in T$ が $R_i(X)W_i(Y)$ という2ステップモデル[11]に限定される場合に，$H\in L(T)$ の必要十分条件は，H が直列可能であることを証明する．

（必要性） $T_i\in T$ が $R_i(X)W_i(Y)$ という2ステップモデルに限定される場合に，T 上の H に対して，各 $T_i\in T$ が論理性を満たす $H'\equiv H$ が存在するなら，H' においては，$R_i(X)$ と $W_i(Y)$ の間に，$X\cap Z\neq\phi$ の $W_j(Z)$ は存在しない．従って，D-等価交換の繰り返しで $R_i(X)$ と $W_i(Y)$ 間の操作を $R_i(X)$ の前に移動できる．このため，各 $T_i\in T$ に対して同様な操作を行っていれば，H' は直列スケジュール H'' に変換できる．$H\equiv H'$, $H'\sim^* H''$ であるため，H は直列可能である．

（十分性） H が直列可能ならば，補題3より，独立化可能である．独立化可能な H なら，$H\in L(T)$ である．

与えられた T 上のスケジュール H が直列可能であるかどうかの判定問題は，NP 完全であるため[11]，2ステップモデルにおいて $H\in L(T)$ かどうかの判定問題も NP 完全である．従って，一般の $H\in L(T)$ かどうかの判定問題も NP 完全である． □

与えられた T 上のスケジュール H の独立化可能性を判定するためには，まず，各 $T_i\in T$ が論理性を満たす $H'\equiv H$ を見つけ，そして，H' における各 T_i が検索一貫性と結果一貫性を満たすかどうかを判定する必要がある．前者の判定問題が NP 完全ならば，独立化可能性の判定問題も NP 完全である．

5. D-独立化可能性

独立化可能性の判定問題がNP完全であるため，本章では，D-独立可能性という多項式時間で判定可能な部分クラスを導入し，その必要性と有効性を示す．

定義 16 T 上のスケジュール H は，H の C に対する部分スケジュール H^j $(j=1,\cdots,n)$ が D-直列可能で，かつ H' において各 $T_i\in T$ が論理性を満たすというような $H'\sim^* H$ が存在する時，D-独立化可能という． □

定理3により，D-独立化可能な H は，必ず独立化可能である．各 $T_i\in T$ が論理性を満たす $H'\sim^* H$ が存在するかどうかの判定方法を検討するために，判定グラフを導入し，$L_D(T)=\{H|T$ 上の H に対して，各 $T_i\in T$ が論理性を満たす $H'\sim^* H$ が存在する$\}$ とする．

定義 17 $H\in L_D(T)$ かどうかの判定グラフ $DG(H)$ は，各 $A_i(X)\in T_i\in T$ を節点とし，枝は次の条件を満たす時に作られる．

(1) $R_i(X)<_H W_j(Y), (X\cap Y\neq\phi)\vee(i=j)$ 時に，$R_i(X)$ から $W_j(Y)$ への枝を持つ．

(2) $W_i(X)<_H R_j(Y)$, $X\cap Y\neq\phi$ 時に，$W_i(X)$ から $R_j(Y)$ への枝を持つ．

(3) $W_i(X)<_H W_j(Y)$, $X\cap Y\neq\phi$ 時に，$W_i(X)$ から $W_j(Y)$ への枝を持つ．

(4) $(R_i(X)<_H W_j(Y))\vee(W_i(X)<_H W_j(Y))$, $X\cap Y\neq\phi, i\neq j$ 時に，T_i 内のすべての操作から $W_j(Y)$ への枝を持つ． □

例 4 例2の H における $DG(H)$ は，**図1**のようになる． □

定理 5 与えられた T 上のスケジュール H に対して，$H\in L_D(T)$ の必要十分条件は，判定グラフ $DG(H)$ に閉路を持たないことである． □

必要性：$H\in L_D(T)$, すなわち，T 上の H に対して，

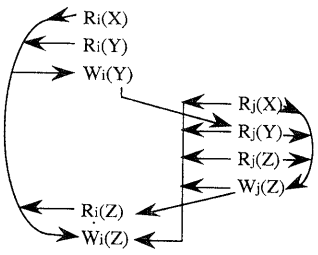


図 1 L_DT 判定グラフ $DG(H)$
Fig. 1 Decision graph $DG(H)$ for L_DT.

各 $T_i \in T$ が論理性を満たす $H_1 \sim^* H$ が存在する．H_1 において，各 $A_j(X) <_{H_1} W_k(Y) (k \neq j, X \cap Y \neq \phi)$ に対して，$T_j \in T$ が論理性を満たすため，$(\forall A_j(Z) \in T_j)(A_j(Z) <_{H_1} W_k(Y))$ である．すなわち，$DG(H_1)$ において種類(4)の枝はすべて $<_{H_1}$ の順に沿っている．また，定義17により $DG(H_1)$ における種類(1)-(3)の枝もすべて $<_{H_1}$ の順に沿ってできている．このため，全順序 $<_{H_1}$ に沿って生成されたグラフ $DG(H_1)$ には閉路は存在しえない．

次に，$H_1 \sim^* H$ なら，$DG(H) = DG(H_1)$ であることを証明する．まず，判定グラフの構成法と D-等価の定義から，$DG(H)$, $DG(H_1)$ の両グラフにおける節点と種類(1)から(3)の枝が同じであることが分る．また，D-等価の定義により，H_1 において，$k \neq j, X \cap Y \neq \phi$ の $A_j(X) <_{H_1} W_k(Y)$ が存在するなら，H においても $A_j(X) <_H W_k(Y)$ である．すなわち，両グラフの種類(4)の枝も同じである．従って，$DG(H)$ にも閉路を持たない．

十分性：与えられる H に対して，判定グラフ $DG(H)$ に閉路を持たないなら，H 内の操作を $DG(H)$ の枝の順で全順序の H' に展開できる．H' において，種類(1)-(3)の枝より，$H' \sim^* H$ であることが保証され，種類(4)の枝より，各 $T_i \in T$ が論理性を満たすことが保証される．従って，$H \in L_D(T)$ である． □

$H \in L_D(T)$ かどうかの判定が多項式時間で行えるので，D-独立化可能性の判定も多項式時間でできる．例2において，H^1, H^2 の D-直列可能性判定グラフは**図2**となり，閉路を持たない．このため，例2の非直列可能な H は D-独立化可能である．

次に，例2以外で D-独立化可能で非直列可能なスケジュール H のいくつかの場合を示す．

例5 $X \cap Y = \phi$ という**図3**(a)で示す $DG(H)$ を持つスケジュールには，$W_i(X) \rightarrow R_j(X), R_j(Y) \rightarrow W_i(Y)$ が存在し，$X \cap Y = \phi$, $(X \cup Y) \cap Z = \phi$ という図3(b)で示す $DG(H)$ を持つスケジュールには，$W_i(X) \rightarrow R_j(X)$, $W_j(Y) \rightarrow R_i(Y)$ が存在するため，両方のスケジュールとも D-直列可能でない．しかし，どちらの場合とも判定グラフ $DG(H)$ に閉路を持たないので，$H \in L_D(T)$ である．このため，V の C に対する被覆 $V_1, \cdots, V_n$ に対して，$X \subseteq V_1, Y \subseteq V_2, X \not\subset V_2, Y \not\subset V_1$ なら，H^1, H^2 における D-直列可能性の判定グラフはそれぞれ**図4**(a)と(b)のようになる．閉路を持たないので，図3(a)と(b)のスケジュールはそれぞれが D-独立化可能である． □

すなわち，D-独立化可能なクラスは，図3(a)というような中間結果を参照する非直列可能な H と，例2や図3(b)というような互いに相手の変更結果を参照する非直列可能な H を含むように範囲を拡大している．また，与えられる T 上のスケジュール H の D-独立化可能性を判定するためには，$T_i \in T$ 内の操作の実行順序 $<_i$ や H 内の実行順序 $<_H$ とも全順序である必要のないことも分る．$<_i$ は T_i 内の R 操作と W 操作間の順序関係を指定し，$<_H$ は $\cup_1^n <_i$ を含むかつ競合する操作間の順序関係を指定していればよい．

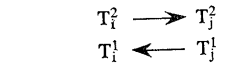
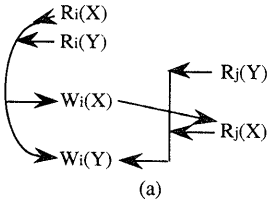
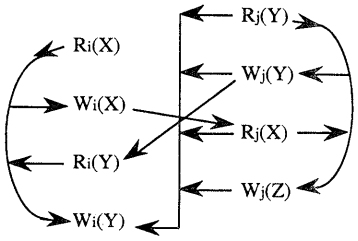


図2 H^1 と H^2 の D-直列可能性判定グラフ $D(H^1)$, $D(H^2)$
Fig. 2 D-serialization graph $D(H^1)$, $D(H^2)$ for H^1, H^2.



(a)



(b)

図3 非直列可能で D-独立化可能な H の $DG(H)$
Fig. 3 $DG(H)$ of Non-serializable but D-equivalent-independent schedule H.

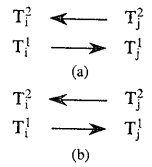


図4 図3の H^1 と H^2 の D-直列可能性判定グラフ $D(H^1)$, $D(H^2)$
Fig. 4 D-serialization graph $D(H^1)$, $D(H^2)$ for H^1, H^2 of Fig. 3.

6. おわりに

本論文では，文献 1) に指摘された高水準 DB における並行処理制御の新たな課題に取り組み，スケジュールの正当性という根本的な問題から見直しを行った．設計 DB などの高水準 DB が統一な判定基準で一貫性制御を行える性質を利用することにより，正当なスケジュールの範囲を従来の直列可能から独立化可能に広げられた結果を示した．また，D-独立化可能性という実用上で重要な部分クラスも与えた．

本論文の最も重要な貢献は，作業手順によって定義される DB の一貫性に関する情報を利用することにより，従来の直列可能性から脱出する方法を提案したことである．本論文では，独立化可能性を判定するための十分条件しか与えていない（定理 3）が，処理単位の意味論を同時に利用すると，判定条件を一層緩めることができる．

また，本論文の主な目的は，状態間の満たすべき性質を DB の一貫性の統一的な判定基準とした場合におけるスケジュールの正当性問題を検討することである．このため，テストが開始されてからテスト結果が入力されるまでテスト対象が変更されない，というような並行実行における状態の正しさを保証する実現方法については触れていない．実現方法はごく簡単ではあるが，必要であることを最後に一言触れておく．

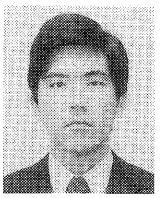
参考文献

1) Barghouti, N. S. and Kaiser, G. E.: Concurrency Control in Advanced Database Applications, *ACM Comput. Surv.*, Vol. 23, No. 3, pp. 269-317 (1991).
2) Bernstein, P. A., Hadzilacos, V. and Goodman, N.: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Publishing Company (1987).
3) Bernstein, P. A., Shipman, D. W. and Wong, W. S.: Formal Aspects of Serializability in Database Concurrency Control, *IEEE Trans. Softw. Eng.*, Vol. 5, No. 5, pp. 203-216 (1979).
4) Cellary, W., Gelenbe, E. and Morzy, T.: *Concurrency Control in Distributed Database Systems*, North-Holland (1988).
5) Elmagarmid, A. K.: *Database Transaction Models for Advanced Applications*, Morgan Kaufmann Publishers (1990).
6) Frrg, A. A. and Ozsu, M. T.: Using Semantic Knowledge of Transactions to Increase Concurrency, *ACM Trans. Database Syst.*, Vol. 14, No. 4, pp. 503-525 (1989).
7) Garcia-Molina, H. and Salem, K.: Sagas, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pp. 249-259 (1987).
8) Garcia-Molina, H.: Using Semantic Knowledge for Transaction Processing in Distributed Database, *ACM Trans. Database Syst.*, Vol. 8, No. 2, pp. 186-213 (1983).
9) Lynch, N. A.: Multilevel Atomicity-A New Correctness Criterion for Database Concurrency Control, *ACM Trans. Database Syst.*, Vol. 8, No. 4, pp. 484-502 (1983).
10) Pu, C., Kaiser, G. E. and Hutchinson, N.: Split-Transactions for Open-Ended Activities, *Proc. 14-th Int. Conf. on Very Large Data Bases*, pp. 26-37 (1988).
11) Papadimitriou, C. H.: The Serializability of Concurrent Database Updates, *J. ACM*, Vol. 26, No. 4, pp. 631-653 (1979).

(平成 5 年 11 月 15 日受付)
(平成 6 年 9 月 6 日採録)

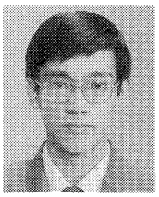
徐 海燕（正会員）

1962 年生．1983 年中国復旦大学理学部計算機科学科卒業．1990 年九州大学工学研究科情報処理工学専攻博士課程修了．工学博士．同年福岡工業大学電子工学科講師，現在同助教授．高水準データベースの質問処理論，並行処理制御などの研究に従事．電子情報通信学会，日本ソフトウェア科学会，ACM 各会員．



古川 哲也（正会員）

1959 年生．1983 年京都大学工学部情報工学科卒業．1985 年同大大学院修士課程修了．1988 年九州大学大学院博士後期課程修了．工学博士．九州大学工学部助手，同大大型計算機センター講師，同助教授を経て，現在同大経済学部助教授．データベースの設計理論，質問処理論などの研究に従事．電子情報通信学会，日本ソフトウェア科学会，日本 OR 学会，ACM, IEEE 各会員．



史 一華

1961 年生．1982 年中国復旦大学理学部計算機科学科卒業．1992 年九州大学総合理工学研究科情報システム学専攻博士課程修了．理学博士．同年福岡工業短期大学電子情報学科助教授．時間限定推論，真理保全，知識ベースシステムなどの研究に従事．電子情報通信学会，人工知能学会各会員．