マルチバージョンデータベースにおけるワークフローに基づく並行処理制御

徐 海燕[†] 古川哲也[††] 史 一華[†††]

ワークフローによって表される作業手順が，協調型データベースの並行処理制御で果たす役割について検討する．協同作業では，修正前のデータも残す，各処理単位の検索できるバージョンに対する制限を設けない，処理単位の後退復帰を避けるといった要求がある．本論文では，データのバージョンを記憶するマルチバージョンデータベースにおいて，ワークフローに従って作成されたデータ集合は一貫していると定義し，そのようなマルチバージョンデータベースにおける並行処理制御は，処理単位を後退復帰させることなく実用的な方法で実現できることを示す．さらに，ワークフローで表される一貫性情報は，利用者の各種の検索に対する要求を支援するためにも役立つことを示す．

Concurrency Control of Multiversion Databases Based on Workflows

Haiyan Xu,[†] Tetsuya Furukawa[††] and Yihua Shi[†††]

In this paper, we discuss the effect on concurrency control in cooperative databases using the consistency information expressed by workflows. Cooperative works require to record multiversions, to make users possible read any necessary versions, and to avoid rollback. In a multiversion database, which has versions of data, a set of data is consistent if and only if it is created following the workflow. We show that the concurrency control of multiversion databases can be carried out efficiently without rollback. Such consistency information is also useful to satisfy the requirement of reading the latest versions.

1. はじめに

原子性，整合性，隔離性，耐久性という ACID 特性を満たす処理単位は，OLTP などの応用分野で重要な役割を果たしてきた[3]．一方データベースの応用分野の拡大にともなって，処理単位を協同作業支援に用いる必要性が高まっている[5),9)]．それを実現するためには，新しい応用分野の特徴に対処できなければならない．

複数の利用者による協同作業においても，ACID 特性の整合性と隔離性で表される並行実行時の一貫性の管理は必要となる．しかし，協同作業を支援するためには，一貫性管理による処理単位の後退復帰は，極力避けなければならない．また，知的設計作業では，修正前のデータも記憶，管理，利用できなければならない．すなわち，利用者が必要に応じて各種のバージョンを利用した場合でも，同様な一貫性管理を行えることが必要である．さらに，最新の作業結果を参照できるなど，協同作業に対する制限は行わないことが望まれる．

OLTP における一貫性管理は，各々の処理単位はデータベースの一貫性を保持するという仮定の下で，直列可能性基準に基づく並行処理制御方式によって行われている[3]．データベースの新しい応用分野の特徴に対応するため，様々な研究がなされているが，どの方式もデータベースの一貫性管理と，利用者の協同作業に対する要求を同時に満たすことはできていない[5),9)]．

一方，オフィスにおける高度なビジネスプロセスをモデル化し，実行または制御するための有効な技術として，ワークフローが注目されている[4]．商用，経営管理，CAD/CAM 等の分野では，多くの実用的なワークフロー管理システムが開発されている．ワークフローは協調作業においては作業手順となり，正しくデータが生成されていることの指標とすることがで

† 福岡工業大学情報工学部情報工学科
Department of Information and Engineering, Fukuoka Institute of Technology
†† 九州大学大学院経済学研究院
Departmemt of Economic Engineering, Kyushu University
††† 西南学院大学商学部
Faculty of Commerce, Seinan Gakuin University

きる．

本論文では，ワークフローで表される作業手順をデータベースが一貫するための統一的な判断基準と見なし，一貫性情報の利用が複数のバージョンを管理するデータベースの並行処理制御で果たす役割について検討する．

処理単位は変更操作と検索操作の系列からなるので，一貫性情報を利用できれば，並行実行の一貫性は具体的に次のように表現できる．

- 変更結果の一貫性：各々の処理単位が単独および並行に実行される場合，データベースの一貫性は保持される．
- 検索結果の一貫性：各々の処理単位が検索するデータ集合は一貫性を満たす．

本論文では，ワークフローに従って作成されたデータ集合は一貫性を満足していると定義し，次のような3つの部分からなる制御法を導入することによって並行実行時の一貫性を管理する．

(1) ワークフローに従う変更操作の実行
(2) 検索結果の一貫性の判定
(3) 検索結果の一貫性が満たされない場合の対処
  (a) 実行中の検索操作に対する調整
  (b) 既存の検索操作に対する調整

データベースの一貫性がワークフローで表されるならば，変更結果の一貫性は能動的な方法で保持でき，検索結果の一貫性は実用的なコストで判定できることを示す．また，実行中の検索操作に対する調整によって，検索結果の一貫性を満たすようにできることを示す．このため，処理単位を後退復帰させることなく，実用的な方法で並行実行の一貫性管理を実現できる．一方で，実際の作業では，利用者は一貫性を満たす古いバージョンよりも最新のバージョンを利用したい場合が多い．検索結果の一貫性のために新しいバージョンを利用できないことは，協同作業では望ましくない．既存の検索操作に対する調整は，このような要求に対処する．

本論文は，次のように構成される．2章では，データの複数のバージョンを管理するデータベースをマルチバージョンデータベースとして定義し，ワークフローに基づく一貫性を示す．3章では，ワークフローに基づく一貫性情報の利用がどのように変更結果の一貫性管理に役立つかについて検討する．4章では，検索結果の一貫性の実用的な判定方法を提案する．5章では，検索結果の一貫性が満たされない場合の対処法を示す．6章は従来の研究との比較であり，7章は全体のまとめである．

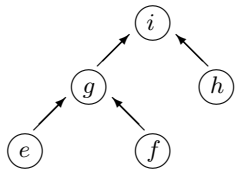

図1 ワークフロー $WF_1(D_1, E_1)$
Fig. 1 Workflow $WF_1(D_1, E_1)$.

2. データベースの一貫性

本章では，ワークフローで表される作業手順を形式的に定義する．また，過去のデータを保持するマルチバージョンデータベースを導入し，ワークフローによる一貫性についてまとめる．

定義1 ワークフローは，作業手順を示す非巡回有向グラフ $WF(D, E)$ である．D は作業過程で作成，変更，利用されるデータ項目に対応する節点の集合である．E は節点間の有向枝の集合であり，枝 (x, y) は，子データ項目 y のデータは親データ項目 x の完成されたデータを用いて作成されることを表す． □

ワークフローのモデル化にはいくつかの方法が存在するが，本論文では，データに焦点を当てる方法を用いている．また，作業手順で関連しないデータ項目間は一貫性管理を必要としないため，$WF(D, E)$ は連結グラフであると仮定する．

例1 ソフトウェア開発において，作業は部品階層に従って子部品から順に作成されるものとする．**図1** は，そのような作業を表すワークフローの例 $WF_1(D_1, E_1)$ で，$D_1 = \{e, f, g, h, i\}$，$E_1 = \{(e, g), (f, g), (g, i), (h, i)\}$ である．$WF_1(D_1, E_1)$ によって，たとえば，g の部品は完成された e と f の部品を用いて作成されなければならないことが示されている． □

1つの作業を完成するために，利用者は繰り返して結果を修正することがある．古いバージョンは重要な履歴情報であり，その一部は継続的に利用されることもある．このような利用では，データベースはデータ項目の各々のバージョンを記憶しておかなければならない．データ項目 x のバージョンをそのバージョンを作成した処理単位のインデクス k，j などを用いて $x_k, x_j, \cdots$ と記述する．処理単位 T_i が同じデータ項目 x を複数回変更した場合は，$x_{i_1}, x_{i_2}, \cdots$ とする．

すべての作業は1回で成功するとは限らない．たとえば，目的プログラムを生成するためのコンパイルは失敗に終わることがあり，目的プログラムが仕様で定義されるとおりに動作するとも限らない．このような作業情報を記憶するために，各バージョンは，状態部

分を持つものとする．

定義 2 マルチバージョンデータベース $MDB(MV, MR)$ は，バージョン集合 MV と関連集合 MR から構成される．MV のバージョン x_i は，インデクスが i である処理単位 T_i によって作成されたデータ項目 x のバージョンである．x_i が完成したデータであればその状態の値は真であり，それ以外の場合は偽である．MR の関連 (x_i, y_j) は，子バージョン y_j が親バージョン x_i に対応して作成されていることを記録している． □

例 2 ワークフロー $WF_1(D_1, E_1)$ に従う作業が処理単位 T_1，T_2，T_3，T_4 によって行われ，**図 2** で示されたような作業結果が生成されたとする．T_1 と T_2 はそれぞれ e_1，f_1，g_1 と e_2，f_2，g_2 を作成し，T_3 と T_4 はそれぞれ h_3，i_3 と h_4，i_4 を作成したとする．それらの結果および関連を記憶する $MDB_1(MV_1, MR_1)$ は次のようになる．

$$MV_1 = \{e_1, e_2, f_1, f_2, g_1, g_2, h_3, h_4, i_3, i_4\}$$

$$MR_1 = \left\{ \begin{array}{l} (e_1, g_1), (e_2, g_2), (f_1, g_1), (f_2, g_2), \\ (g_2, i_3), (h_3, i_3), (g_2, i_4), (h_4, i_4) \end{array} \right\}.$$

ここで，g_1 と i_3 の状態の値のみが偽であり，その他のバージョンの状態はすべて真であるとする． □

集合内のバージョンがすべて異なるデータ項目に属する場合，そのバージョン集合を単一バージョン集合という．ワークフローに従って生成された単一バージョン集合を記述するために，ワークフローインスタンスの概念を導入する．

定義 3 マルチバージョンデータベース $MDB(MV, MR)$ におけるワークフロー $WF(D, E)$ のワークフローインスタンス $WFI(V, R)$ は，バージョンを節点とする有向グラフである．節点集合 V は単一バージョン集合であり，V 中の各バージョン y_j に対し，$WF(D, E)$ に枝 (x, y) が存在するならば $MDB(MV, MR)$ で y_j の親であり状態が真である x のバージョンも V に存在する．枝集合 R は，バージョン集合 V 上の MR の部分集合 $R = \{(x_i, y_j) \mid (x_i, y_j) \in MR, x_i \in V, y_j \in V\}$ である． □

ワークフローインスタンスは，ワークフローに従って作成された単一バージョンの集合である．

例 3 $MDB_1(MV_1, MR_1)$ における $WF_1(D_1, E_1)$ のワークフローインスタンスは，g_2 を含めば f_2 と e_2 も含まなければならない．一方，i_3 または i_4 を含むワークフローインスタンスは，それぞれ g_2 と h_3 または h_4 を含まなければならない．**図 3** は，そのようなワークフローインスタンスを示している． □

本論文では，ワークフローで表される作業手順をデータベースが一貫するための統一的な基準と見なしている．ワークフローに従って作成されたデータ集合かどうかという情報はワークフローインスタンスから得られるので，マルチバージョンデータベースの一貫性を次のように定義できる[10]．

定義 4 マルチバージョンデータベース $MDB(MV, MR)$ は，MV のすべてのバージョンがいずれかのワークフローインスタンスに属するとき，$WF(D, E)$ に対して一貫しているという．また，単一バージョン集合 V は，V が V' の部分集合となるワークフローインスタンス $WFI(V', R')$ が存在するとき，$WF(D, E)$ に対して一貫しているという． □

すなわち，ワークフローに従って作成されたマルチバージョンデータベースや単一バージョン集合は一貫している．たとえば，$MDB_1(MV_1, MR_1)$ において，$\{e_2, f_2, g_2, h_3, i_4\}$ は，$WF_1(D_1, E_1)$ 中の枝 (h, i) に対して i_4 は h_3 から作成されておらず，一貫していない．

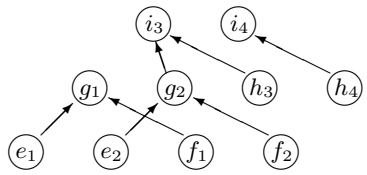

図 2 マルチバージョンデータベース $MDB_1(MV_1, MR_1)$
Fig. 2 Multiversion database $MDB_1(MV_1, MR_1)$.

3. 変更結果の一貫性

本章では，変更結果の一貫性，すなわち，処理単位

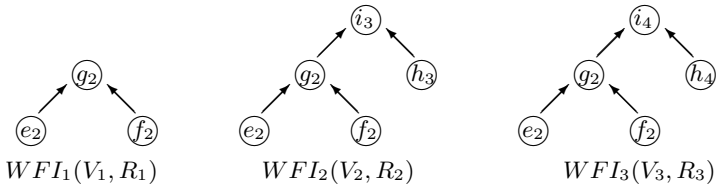

図 3 MDB_1 における WF_1 のワークフローインスタンス
Fig. 3 Workflow instances of MDB_1 for WF_1.

の実行によるデータベースの一貫性保持について検討する．

従来の並行処理制御では，各処理単位が単独実行される場合のデータベースの一貫性を利用者の責任において保証している．データベースの一貫性がワークフローによって定義されるならば，変更結果の一貫性管理は，変更操作のワークフローに従った実行の管理となる[7]．

定義 5 ワークフロー $WF(D,E)$ に対して一貫しているマルチバージョンデータベース $MDB(MV,MR)$ において，変更操作 $W_j(y)$ は，次の条件を満たすとき $WF(D,E)$ に従っているという．

(1) $WF(D,E)$ に枝 (x,y) が存在するデータ項目 x に対して，$W_j(y)$ は MV 中の状態が真である x のバージョンを参照している．さらに，参照しているバージョンからなる単一バージョン集合は一貫している．

(2) y_j の書き込み時に，その状態と y_j に関する関連も同時に $MDB(MV,MR)$ に記録される．

□

利用者の変更操作は，定義 5 の (1) の条件の検査と (2) のトリガよる実現によって，能動的にワークフローに従う変更操作に変換することができる．

ワークフローに従う変更操作で生成されたマルチバージョンデータベースには，次の性質がある．

補題 1 ワークフロー $WF(D,E)$ に従う変更操作によって作成されたマルチバージョンデータベース $MDB(MV,MR)$ において，その単一バージョン集合 V は，その中の各々の x_i，y_j が次の条件を満たすとき，またこのときに限り $WF(D,E)$ に対して一貫している．ここで，E^+ と MR^+ は，それぞれ E と MR の推移的閉包である．

(1) $(x,y) \in E^+$ ならば，$(x_i,y_j) \in MR^+$ である．

(2) $(y,x) \in E^+$ ならば，$(y_j,x_i) \in MR^+$ である．

(3) $\exists z((z,x) \in E^+ \wedge (z,y) \in E^+)$ ならば，$\exists z_k \in MV((z_k,x_i) \in MR^+ \wedge (z_k,y_j) \in MR^+)$ である．

□

証明： $WF(D,E)$ に従う作業の結果を記憶する $MDB(MV,MR)$ において，$(x,y) \in E$ に対して，子データ項目 y のバージョンに対応する親データ項目 x のバージョンは 1 つである．逆に，親データ項目 x の 1 つのバージョンに対してその子データ項目 y の対応するバージョンは複数存在しうる．すなわち，子データ項目のバージョンが決まれば親データ項目の対応するバージョンは一意に決められるが，その逆が成り立つとは限らない．

したがって，V 中のバージョンに対応する祖先のバージョンは一意に定められる．V のバージョンに対し，その祖先バージョンを加える．補題の条件 (1)〜(3) より，結果のバージョン集合は単一バージョン集合である．$MDB(MV,MR)$ のバージョンは $WF(D,E)$ に従って作成されるので，親バージョンの状態は真である．このため，結果のバージョン集合は定義 3 よりワークフローインスタンスである．V は，それを含むワークフローインスタンスが存在するので，一貫している．

逆に，いずれかの条件を満たさなければ，V とそれらの祖先のバージョンの集合は単一バージョン集合とはならず，V を含むワークフローインスタンスは存在しないので $WF(D,E)$ に対して一貫していない． □

定理 1 ワークフロー $WF(D,E)$ に対して一貫しているマルチバージョンデータベース $MDB(MV,MR)$ において，検索操作と $WF(D,E)$ に従う変更操作からなる処理単位を単独実行または並行実行した結果のデータベースは，$WF(D,E)$ に対して一貫している．

□

証明： ワークフロー $WF(D,E)$ に従う変更操作 $W_j(y)$ は一貫している単一バージョン集合を参照するので，それらを含むワークフローインスタンスが存在する．そのワークフローインスタンスに y のバージョンが含まれていれば，y のバージョンおよびそのすべての子孫バージョンをワークフローインスタンスから削除する．$WF(D,E)$ は非巡回グラフなので，結果のバージョン集合は y_j によって参照されているバージョンとそれらの祖先バージョンからなるワークフローインスタンスである．$W_j(y)$ は $WF(D,E)$ に従う変更操作なので，それに y_j を追加したものもワークフローインスタンスである．処理単位の並行実行でも生成されたバージョンは同様にいずれかのワークフローインスタンスに含まれる．すなわち，結果のデータベースは $WF(D,E)$ に対して一貫している． □

定理 1 は，ワークフローに従う変更操作は，マルチバージョンデータベースの一貫性を保持できることを意味する．このため，変更結果の一貫性管理のために，処理単位を後退復帰させることはない．

例 4 **図 4** は例 2 の処理単位 T_3，T_4 の実行を示している．$W_3(h)$ は $W_4(h)$ より先に実行されているが，$W_3(i)$ は $W_4(i)$ より後であることより，図 4 の実行は直列可能ではない．しかし，それらが生成したバージョンは例 3 で示したワークフローインスタンス $WFI_2(V_2,R_2)$ と $WFI_3(V_3,R_3)$ に含まれる． □

以降，変更操作はすべてワークフローに従っているものとする．処理単位の変更操作がワークフローに

T_3	値	T_4	値
$W_3(h)$	h_3		
$R_3(g)$	g_2		
		$W_4(h)$	h_4
		$R_4(g)$	g_2
		$W_4(i)$	i_4
$W_3(i)$	i_3		

図 4 直列可能でない変更操作
Fig. 4 Non-serializable update.

従っていれば，データベースシステムの故障によるリカバリによる後退復帰も，従来の処理単位全体ではなく実行中の変更操作のみとなる[7]．

4. 検索結果の一貫性の判定

変更結果の一貫性は，各変更操作がワークフローに従って実行されれば保証できる．本章では，処理単位の検索結果の一貫性の判定方法について検討する．満たさない場合の対応などについては，5 章で検討する．

処理単位の k 番目までの検索操作の結果からなる単一バージョン集合を TR^k とする．同じデータ項目が複数回検索された場合は，最後に検索されたバージョンとする．TR^k がワークフローに対して一貫しているかどうかは，補題 1 によって判定できるが，バージョン数の増大につれ，判定に要する時間も急速に長くなる．また，この判定は検索操作を行うたびに実行しなければならないため，より効率的で実用的な判定方法が必要である．検索結果の一貫性判定のため，TR^k をグループ化する．

単一バージョン集合 V に対し，共通の子孫を持つバージョンを同一のグループに分類する．また，その共通の子孫をグループ代表元とし，グループ代表元の集合を $rep(V)$ で表す．

k 番目の検索結果が TR^{k-1} と一貫しているかどうかの判定は，TR^{k-1} 中の各バージョンとの判定ではなく，$rep(TR^{k-1})$ 中の各バージョンとの判定とすることができる．

Algorithm 1： k 番目 ($k \geq 2$) の検索結果の一貫性判定アルゴリズム
入力： $rep(TR^{k-1})$，k 番目の検索結果 y_j．
出力： $rep(TR^k)$，判定結果 NON．
begin
  $NON = \phi$; $rep(TR^k) = rep(TR^{k-1})$;
  $group$=**false**;
  foreach $x_i \in rep(TR^{k-1})$ **do**
    begin
      if $(x, y) \in E^+$ **then** ／＊場合 (1) ＊／
        if $(x_i, y_j) \in MR^+$ **then**
          begin
            $rep(TR^k)$=$(rep(TR^k) - \{x_i\}) \cup \{y_j\}$;
            $group$=**true**;
          end
        else $NON = NON \cup \{x_i\}$
      else if $(y, x) \in E^+$ **then** ／＊場合 (2) ＊／
            if $(y_j, x_i) \in MR^+$ **then** $group$=**true**
            else $NON = NON \cup \{x_i\}$
    end
  if $NON = \phi \wedge group$=**false then**
    foreach $x_i \in rep(TR^{k-1})$ **do**
      begin
        $rel = \{z_k \mid (z_k, y_j) \in MR\}$;
        while $rel \neq \phi$ **do**
          begin
            select z_k from rel; $rel = rel - \{z_k\}$;
            if $(z, x) \in E^+$ **then** ／＊場合 (3) ＊／
              begin
                if $(z_k, x_i) \notin MR^+$ **then**
                  $NON = NON \cup \{x_i\}$
              end
            else $rel = rel \cup \{w_p \mid (w_p, z_k) \in MR\}$
          end
      end
  if $NON = \phi \wedge group$=**false then**
    $rep(TR^k)$=$rep(TR^k) \cup \{y_j\}$
end □

例 5 **図 5** は，$MDB_1(MV_1, MR_1)$ を検索する処理単位 T が 2 種類の並行実行における Algorithm 1 を実行した結果である．簡単のため，T_4 中の $R_4(g)$ を略している．(a) と (b) のどちらの場合も $R(i)$ が実行された時点で NON の値は空でなくなる． □

補題 2 k 番目 ($k \geq 2$) の検索操作に対する Algorithm 1 の出力 NON の値が空であれば，$rep(TR^k)$ は TR^k のグループ代表元の集合である． □

証明： $rep(TR^1)$ には TR^1 のグループ代表元が記録されている．$k \geq 2$ の場合，$rep(TR^{k-1})$ 中の各 x_i に対して，場合 (1) においては，NON の値が空のとき，y_j は x_i をグループ代表元とするグループ内のすべてのバージョンの子孫である．Algorithm 1 は y_j を x_i と同じグループに分類し，グループ代表元を y_j に変更している．場合 (2) においては，NON の値が空のとき y_j は x_i の祖先である．Algorithm 1 は y_j

T_4	値	T	値	$rep(TR^k)$	NON
$W_4(h)$	h_4				
		$R(h)$	h_4	ϕ	ϕ
		$R(i)$	i_3	$\{h_4\}$	$\{h_4\}$
$W_4(i)$	i_4				

(a)

T_4	値	T	値	$rep(TR^k)$	NON
		$R(h)$	h_3	ϕ	ϕ
$W_4(h)$	h_4				
$W_4(i)$	i_4				
		$R(i)$	i_4	$\{h_3\}$	$\{h_3\}$

(b)

図 5 判定アルゴリズムの実行結果
Fig. 5 The result of Algorithm 1.

を x_i と同じグループに分類し，グループ代表元は x_i のままにしている．それ以外の場合は，x_i と y_j は互いに他方の祖先ではないので，y_j は x_i をグループ代表元とするグループには属さない．

NON の値が空であれば，Algorithm 1 はどのグループにも分類されてない y_j を新たなグループに分類するとともに，そのグループの代表元としている．□

補題 2 より Algorithm 1 はグループ代表元を正しく計算している．次に，グループ代表元を用いた判定方法の正しさについて証明する．

定理 2 ワークフロー $WF(D,E)$ に対して一貫しているマルチバージョンデータベース $MDB(MV,MR)$ において，処理単位の k 番目 ($k \geq 2$) までの検索操作が検索結果の一貫性を満たすための必要十分条件は，Algorithm 1 の出力 NON の値が空であることである．□

証明： Algorithm 1 の場合 (1)〜場合 (3) はそれぞれ補題 1 の条件の (1)〜(3) に対応している．NON が空のとき，$rep(TR^{k-1}) \cup \{y_j\}$ は補題 1 の条件を満たす単一バージョン集合なので，補題 1 よりそれを含むワークフローインスタンスが存在する．TR^{k-1} は $rep(TR^{k-1})$ より一意に定められるので，$rep(TR^{k-1})$ と同じワークフローインスタンスに属する．したがって，$TR^{k-1} \cup \{y_j\}$ は一貫する．NON が空でないとき，y_j と NON 中のバージョンは同一のワークフローインスタンスに属することはできないので，$rep(TR^{k-1}) \cup \{y_j\}$ は $WF(D,E)$ に対して一貫しない．□

次に，判定コストについて議論する．Algorithm 1 は，グループに分けられている TR^{k-1} のバージョンに対して，各グループ代表元と y_j の一貫性を判定している．場合 (1) と (2) における判定は，y_j と各グループ代表元間の関連する経路の長さに比例する．$WF(D,E)$ によって検査すべき経路を事前に判断でき，親バージョンも一意に定められるので，y_j と特定のグループ代表元間の経路の検査はデータ項目の数 n に対して $O(n)$ である．グループの数は定数なので，場合 (1) と (2) における判定コストは $O(n)$ である．

場合 (3) では，各グループ代表元 x_i に対して，x と y の最も近い共通祖先であるデータ項目のみに対して共通祖先バージョンの存在を検査している．これは祖先バージョンは一意に定められるので，最も近い共通祖先であるデータ項目に対して共通祖先バージョンが存在すれば他の項目の祖先バージョンも存在する，存在しなければ他項目の祖先バージョンも存在しないためである．最も近い共通祖先バージョンへの経路上のバージョンはすべて異なり，グループの数は定数であるため，場合 (3) におけるコストも $O(n)$ である．

通常，処理単位は短い経路で関連するデータ項目について検索することが多いので，実用的なコストで判定できる．

5. 検索結果の一貫性の調整

本章では，検索結果の一貫性が満たされないときの対応方法とどのようにして利用者の検索要求に対応するかについて検討する．また，ワークフローで表された一貫性情報がそのような場合の対策にも役立つことを示す．

検索結果の一貫性は，同じ処理単位内の検索操作がワークフローに従って作成されたマルチバージョンデータベースの一貫している単一バージョン集合を検索することを意味する．利用者は複数の処理単位を利用することによって同一データ項目の異なる単一バージョン集合を同時に利用できる．このため，どのようにして利用者に必要な一貫している単一バージョン集合を提供するかが要点になる．まず，どのようにしてこれまでの検索結果と一貫するバージョンを提供するかについて検討する．

定義 6 処理単位の k 番目の検索操作 $R(y)$ の実行に対して Algorithm 1 を適用し，検索するバージョンが TR^{k-1} と一貫している場合は $R(y)$ を実行し，一貫していない場合は次の処理を行う機能を実行中の検索操作の調整という．

(1) Algorithm 1 を y の他のバージョンに対して適用する．一貫しているバージョン y_p が存在すれば，$R(y)$ の検索結果を y_p とする．

T_4	値	T	値
$W_4(h)$	h_4		
		$R(h)$	h_4
$W_4(i)$	i_4		
		$\underline{R(i)}$	i_4

図 6 実行中の検索操作に対する調整機能（2）
Fig. 6 The result of the adjustment function (2).

T_4	値	T	値
		$R(h)$	h_3
$W_4(h)$	h_4		
		$\underline{R(i)}$	i_3
$W_4(i)$	i_4		

図 7 実行中の検索操作に対する調整機能（1）
Fig. 7 The result of the adjustment function (1).

(2) 存在しなければ，生成されるまで $R(y)$ の実行を待機する． □

例 6 例 5 で示したように，図 5 (a) において $R(i)$ の実行時に処理単位 T は検索結果の一貫性を満たさない．データ項目 i にはその時点でバージョン i_3 しか存在しないので，調整機能 (2) が適用される．結果は**図 6** の実行となり，$R(i)$ は待機させられた後 i_4 を検索する．一方，例 5 (b) の場合は調整機能 (1) が適用され，$R(i)$ はそれまでの検索結果 $\{h_3\}$ と一貫している古いバージョン i_3 を検索する（**図 7**）． □

実行中の検索操作に対する調整には，次の性質がある．

定理 3 実行中の検索操作に対する調整によって，処理単位の検索結果の一貫性を保証できる． □

証明： 定理 2 より，Algorithm 1 の判定結果が一貫している場合は定理が成り立つ．調整機能 (1) では TR^{k-1} と一貫している y のバージョンを検索し，(2) では一貫するバージョンが生成されるまで $R(y)$ の実行を延期させているので，検索結果の一貫性が満足される． □

$R(y)$ の実行が待機になる場合には，TR^{k-1} と一貫する y のバージョンの作成を待たずに処理単位を終了しても定理 1 より変更結果の一貫性を満たす．したがって，互いに待機するすくみになることによる後退復帰はない．

定理 3 は，検索結果の一貫性は各検索操作の実行時に判断でき，それに応じて対策をとれることを意味する．すなわち，検索結果の一貫性管理のために処理単位を後退復帰させる必要はない．定理 1 とまとめると，ワークフロー $WF(D, E)$ に従って作成された結果を記憶するマルチバージョンデータベース $MDB(MV, MR)$ において，一貫性管理のために処理単位を後退復帰させる必要はない．

実行中の検索操作に対する調整を用いれば，最新バージョンというようなデフォルト指定方法が用いられてもつねに検索結果の一貫性は満たされる．これは最新バージョンが検索結果の一貫性を満たさなければ，調整機能によって一貫する古いバージョンを検索させるように調整されるためである．しかし，この調整は必ずしも利用者の要求を満足しているとは限らない．ワークフローを用いれば，すでに実行された検索操作から問題となる部分を検出できる．

定義 7 実行中の検索操作に対する調整において，調整機能 (1) の代わりに次のような手順で y_j と同じワークフローインスタンスに属することのできない TR^{k-1} 中の部分集合を特定し，TR^{k-1} から削除する方法を，既存の検索操作に対する調整という．

- NON のバージョンを TR^{k-1} から削除する．
- NON 中の x_i をグループ代表とするグループ中のバージョンを，子孫から祖先の順で y_j と同じワークフローインスタンスに属すものが見つかるまで繰り返し削除する． □

定理 4 既存の検索操作に対する調整は，処理単位の検索結果の一貫性を保証する． □

証明： 定理 2 より NON 中のバージョンは y_j と同じワークフローインスタンスに属さない．NON に含まれないバージョンが代表元となるグループのバージョンは y_j と同じワークフローインスタンスに属する．NON 中のバージョンを代表元とするグループでも，y_j と同じワークフローインスタンスに属すバージョンが存在すれば，その祖先のバージョンはすべて同じワークフローインスタンスに属する．既存の操作に対する調整は，そのような y_j と同じワークフローインスタンスに属さないバージョンを削除している． □

TR^{k-1} で k 番目の検索結果 y_j と一貫しない原因となるバージョンを正しく特定できれば，それらを検索した操作を再実行することによって y_j とワークフローに従って作成されたバージョンを検索することができる．例 5 (b) で既存の検索操作に対する調整を適用した結果は，h_3 が特定され TR^2 から削除される．$R(h)$ の再実行は**図 8** となり，$\{i_4\}$ と一貫しているバージョン h_4 を検索する．

6. 従来の研究との比較

協同作業を支援するデータベースとして，高度な処

T_4	値	T	値
		$R(h)$	h_3
$W_4(h)$	h_4		
$W_4(i)$	i_4		
		$R(i)$	i_4
		$\underline{R(h)}$	h_4

図 8 既存の検索操作に対する調整
Fig. 8 Adjusting existing read operations.

理単位モデルの研究がなされている．

(1) 直列実行と等価となるスケジュールの範囲を拡大する方式．具体的に処理単位の意味論を利用することによって，並行実行が等価であることの定義を拡張する方法[1]や，処理単位間の干渉を限定することによる方法[6]がある．

(2) 処理単位を階層的に分割し，直列可能性を満たす単位や後退復帰の単位を縮小する処理単位の階層モデルを用いる方式[8]．

(3) 応用分野の一貫性情報を利用することによって，独立化可能性という新たな正当な並行実行のクラスを用いる方式[12]．

本論文で提案している方式は，作業手順という一貫性情報を用いることによって，変更結果の一貫性を能動的に保証している．並行実行の正当性をスケジュールの直列可能性に求めていない点で (1) の研究と異なる．また，検索結果の一貫性は処理単位のレベルで対処しているので，(2) のような直列可能性を満たす単位を縮小する立場からの研究とは異なる．

一方，本論文は検索結果の一貫性に対して現実的に受け入れられるコストで判定する方式を採用している．一貫していなければ，問題の原因を利用者に示したうえで，解決方法を利用者に選択させる．(3) の独立化可能性と比べると，具体化された一貫性情報を用いて実現可能性まで示した．

データの複数のバージョンを用いた並行処理制御法として，マルチバージョン並行処理制御[3]がある．この方式は，スケジュールの直列可能性を保証するために過去の値を用いている．すなわち，利用者はデータのバージョンを意識しておらず，明示的に利用することもできない．この点で本論文のマルチバージョンデータに対する並行処理制御とは本質的に異なる．本論文の方法では，一貫性情報を用いて変更結果の一貫性を満たすように制御する．検索結果の一貫性は検索操作ごとに判定しているので，一貫性上の問題があれば即時に対処する．従来のマルチバージョン並行処理制御方式は直列可能性に基づいて制御するので，ワークフロー上で関連のないデータ項目に対する検索操作からなる処理単位に対しても，直列可能でなければ過去の値を読ませたり，処理単位を後退復帰させたりする．

データの複数のバージョンを扱うデータベースにおける並行処理制御の研究もなされている．論文 2) は，線形に順序づけられたデータ項目に対して処理単位をその順序に従って実行させることによってスケジュールの直列可能性を保証し，複数のバージョンを扱う場合にも処理単位の満たすべき実行順序に従うバージョンの順序でその方法を適用できることを示している．一貫性を直列可能性に求めている点でマルチバージョン並行処理制御と同等であり，ワークフローで示された作業の順序を一貫性の規準とする本論文とは本質的に異なる．

ワークフロー管理システムは理論的な基盤に欠け，並行処理制御基準が明確に定義されておらず，並行処理制御に関する機能が弱いなどの問題点をかかえている[11]．本論文では，協同作業環境の意味論とワークフローを用いた手法をデータベースの処理単位技術に導入することによって，問題の総合解決を図っている．

7. おわりに

本論文では，ワークフローで表された一貫性情報に基づく，マルチバージョンデータベースにおける並行処理制御について検討した．変更結果の一貫性と検索結果の一貫性に分離した並行実行時の一貫性の管理問題に対して，それぞれ一貫性情報を用いることによって，処理単位を後退復帰させることなく解決できることを示した．これは一貫性情報を利用できれば，操作を行う前に一貫性上の問題を把握でき，対策を講じられるためである．一方，協同作業における最新バージョンを検索したいという要求に対しても，一貫性情報の利用が役立つことを示した．

本論文では，非巡回有向グラフで表されるワークフローを対象として議論した．しかし，一般には再帰構造を持つソフトウェア開発や開発グループ内で閉路を構成するような場合もあり，ワークフローを巡回有向グラフで表現する方が適当な場合もある．枝の始点と終点が同一節点である巡回は，本論文の結果の簡単な拡張で対処できる．一般の巡回に対しては，巡回部分を 1 つの節点にまとめ，まとめられた節点のデータ集合をその節点のデータとすることで，全体の制御は可能となる．本論文の手法を巡回部分の直接的な制御に適用することは今後の課題である．

本論文の結果は，ワークフローに基づく協同作業支援等に応用できる．協同作業では，互いに同じデータ

項目を変更していることを承知していれば，一方が変更を遅延させるまたは互いに参照しながら変更していくことがある．また，祖先となるデータ項目のバージョンアップの予定を知っていれば，最新の結果を利用して子孫のデータ項目に関する作業を進めていくことがある．このような作業を可能とするようなワークフローによって表された一貫性情報を利用した処理単位の構築法も重要な課題である．

謝辞 本研究の一部は文部省科学研究費補助金基盤研究（C)(2)(課題番号：11680429）の援助を受けている．

参 考 文 献

1) Agrawal, D., Abbadi, A.E. and Singh, A.K.: Consistency and Orderability: Semantics-Based Correctness Criteria for Databases, *ACM Trans. Database Syst.*, Vol.18, No.3, pp.460–486 (1993).
2) Ahuja, M. and Browne, J.C.: Concurrency Control by Pre-Ordering Entities in Databases with Multi-Versioned Entities, *Proc. Int. Conf. on Data Engineering*, pp.312–321 (1987).
3) Bernstein, P.A., Hadzilacos, V. and Goodman, N.: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley (1987).
4) Cichocki, A., Helal, A., Rusinkiewicz, M. and Woelk, D.: *Workflow and Process Automation: Concepts and Technology*, Kluwer (1998).
5) Elmagarmid, A.K. (Ed.): *Database Transaction Models for Advanced Applications*, Morgan Kaufmann (1992).
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) Furukawa, T., Xu, H. and Shi, Y.: Supporting Collaborative Work by Process-based Transaction Model, *Lecture Note in Coumpter Science*, Vol.1552, pp.421–433, Springer Verlag (1999).
8) Nodine, M.H., Ramaswamy, S. and Zdonik, S.B.: A Cooperative Transaction Model for Design Databases, in 5), pp.53–83 (1992).
9) Ramamritham, K. and Chrysanthis, P.K.: *Advances in Concurrency Control and Transaction Processing*, IEEE Computer Society Executive Briefing (1996).
10) Xu, H., Furukawa, T. and Shi, Y.: Supporting Cooperative Work Based on the Semantics of Workflows, *Database Applications in Non-Traditional Environments*, pp.366–369, IEEE Computer Soceity Press (2000).
11) Worah, D. and Sheth, A.: Transactions in Transactional Workflows, *Advanced Transaction Models and Architectures*, Jajodia, S. and Kerschberg, L. (Eds.), Kluwer (1997).
12) 徐 海燕，古川哲也，史 一華：一貫性情報を用いたデータベースの並行処理制御，情報処理論文誌，Vol.35, No.12, pp.2752–2761 (1994).

(平成 12 年 9 月 20 日受付)
(平成 12 年 12 月 27 日採録)

（担当編集委員 石川 博）

徐 海燕（正会員）

昭和 58 年中国復旦大学理学部計算機科学科卒業．平成 2 年九州大学大学院博士後期課程修了．工学博士．福岡工業大学電子工学科講師，同助教授を経て，現在同大学情報工学部情報工学科助教授．平成 12 年度米国カリフォルニア大学サンタバーバラ校客員研究員．並行処理制御，時空間データベース等の研究に従事．ACM，IEEE，電子情報通信学会，日本ソフトウェア科学会各会員．

古川 哲也（正会員）

昭和 58 年京都大学工学部卒業．昭和 60 年京都大学大学院修士課程修了．昭和 63 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，大型計算機センター講師，同助教授，経済学部助教授を経て，現在同大学大学院経済学研究院助教授．この間，平成 7 年米国パデュ大学客員研究員．データベースの設計論・質問処理論，情報システムの研究に従事．電子情報通信学会，ACM，IEEE，日本 OR 学会等会員．

史 一華

昭和 57 年中国復旦大学理学部計算機科学科卒業．平成 4 年九州大学大学院博士後期課程修了．理学博士．福岡工業短期大学電子情報システム学科助教授を経て，平成 12 年西南学院大学商学部経営学科教授．データベース，知識ベースシステム，WEB 型教育支援システム，真理保全等の研究に従事．人工知能学会，統計科学研究会，電子情報通信学会各会員．