5AA－2

グラフ判定方式に基づく一貫性情報を用いた並行処理制御の実現法

徐 海燕　古川哲也　史一華
福岡工業大学　九州大学　福岡工業短期大学

1 まえがき

データベースの並行処理制御方式として，２相施錠方式，時刻印方式などの直列可能性基準に基づく方式がある．競合率が高い場合には２相施錠方式の方が，高くない場合には時刻印方式の方が並行性が高いという特徴が知られている[1]．また，時刻印方式より常に並行性が高いグラフ判定方式SGTでは実現コストが高いという問題がある[1]．

一方，データベースの応用分野が協同作業や長時間の処理を支援する分野に拡大するにつれ，直列可能なスケジュールのみでは，満足できる並行実行を提供できない．このため，様々な角度から研究が行われてきており，データベースの持つ一貫性情報を利用することにより，独立化可能という直列可能より範囲を拡大した正当なクラスが提案されている[2]．スケジュールHの独立化可能性を判定する問題は，次の２つの問題に分けられる．

- 並行処理制御に関する問題：各処理単位Tによって操作されたデータ項目は，Tが終了するまで，他の処理単位によって変更されていないような等価なスケジュールが存在するかどうかの問題．
- 一貫性に関する問題：各処理単位によって検索されるのは一貫したデータベースの部分集合であるかどうかの問題と変更結果に対する一貫性検証問題．

前者の問題に対しては，２相施錠方式の拡張版2PLE，時刻印方式の拡張版TOE，グラフ判定方式IGTという３種類の方式が提案されている[3]．高水準データベースにおいて最も重要なのは並行性であるため，常にTOEより並行性が高いIGT方式の効率化に関する研究が必要である．本稿では，グラフ判定方式IGTの効率的な実現方法について検討する．

2 論理性クラス

[定理 1] [2] 与えられた処理単位集合$\boldsymbol{T}$上のスケジュールHに対して，論理性判定グラフ$DG(H)$に閉路が存在しなければ，各$T_i \in \boldsymbol{T}$によって操作されたデータ項目は終了するまで他の処理単位によって変更されていないようなHと等価なスケジュールH'が存在する．ただし，$DG(H)$は，各$T_i \in \boldsymbol{T}$内の操作$A_i(x)$ $(A \in \{R, W\})$を節点とし，枝は次の条件を満たす時に作られる．

- 処理単位内枝：$A_i(x) <_i A_i(y)$ ならば，$A_i(x)$ から $A_i(y)$ への枝．
- 処理単位間R枝：$W_i(x) <_H R_j(x)(i \neq j)$ ならば，$W_i(x)$ から $R_j(x)$ への枝．
- 処理単位間W枝：$A_i(x) <_H W_j(x)(i \neq j)$ ならば，T_i内の全ての操作から $W_j(x)$ への枝． □

グラフ判定方式IGTとは，$DG(H)$に閉路が存在しないスケジュールのみを許可する方式である．従来の処理単位を節点とする直列可能判定グラフによるSGTにおいても効率の問題があるので，操作を節点とする$DG(H)$によるIGTの効率化は一層必要となる．

[例 1] 次のHについて考察する．

$$H = R_3(y)R_1(x)W_2(x)W_4(y)R_3(x)R_1(y)$$

$R_1(x) <_H W_2(x)$ と $R_3(y) <_H W_4(y)$ より処理単位間W枝 $(R_1(y), W_2(x))$, $(R_3(x), W_4(y))$ ができる．さらに $W_2(x) <_H R_3(x)$ よりその２つの枝が接続され，経路 $R_1(y)$, $W_2(x)$, $R_3(x)$, $W_4(y)$ となる．枝 $(R_3(x), W_4(y))$ の立場からは，これが始点からの延長ということになる．最後に $W_4(y) <_H R_1(y)$ より，閉路 $R_1(y)$, $W_2(x)$, $R_3(x)$, $W_4(y)$, $R_1(y)$ が形成される． □

閉路中の作成時刻が一番遅い節点 $A_j(y)$ を閉路の始点，そこから出る枝を最初の枝とすると，$DG(H)$には次の特徴がある．

[補題 1] [3] $DG(H)$に閉路があれば，その最初の枝は時間順と逆順の処理単位間W枝である． □

3 グラフ判定方式の効率化

補題1により，$DG(H)$における閉路は，時間順と逆順の処理単位間W枝を最初の枝とする経路によって構成される．このため，$DG(H)$中の処理単位間W枝を最初の枝とする経路のみを検査の対象とする．さらに，$A_i(x) <_H W_j(x)$ に対して，$T_i \in \boldsymbol{T}$内の全ての操作から $W_j(x)$ への枝が生成されるので，効率化をはかるため，各操作に対して先行処理単位集合FTSを設け，iというT_iのIDを$FTS[W_j(x)]$に記録し，かつ経路上のその後の操作のFTSにも記録していく．経路が枝 $(A_k(y), A_i(z))$

An Implementation Method of Graph Testing Concurrency Control Protocol with Consistency Information
Haiyan XU†, Tetsuya FURUKAWA‡, Yihua SHI ††
†Fukuoka Institute of Technology, ‡Kyushu University, ††Fukuoka Junior College of Technology

による延長時に，$FTS[A_k(y)]$ に属する ID i の処理単位 T_i の操作 $A_i(z)$ があれば閉路 $A_i(z), W_j(x), \cdots, A_i(z)$ が検出される．ただし，時間順と逆順の処理単位間 W 枝を最初の枝とする経路は，時間に伴って始点と終点との両方向から延長する．終点からの延長は，時間順に沿っている枝によるもので，始点からの延長は，2本の時間順と逆順の処理単位間 W 枝を最初の枝とする経路が接続する時に起こる．

経路の始点からの延長という特徴に効率よく対処するため，$FTS[A_i(x)]$ に属する各要素 k に対して $k \prec i$ という推移的な関連 $\prec$ を導入する．このため，処理単位内枝と処理単位間 R 枝で延長される経路による閉路は，$i \prec i$ が生じることによって検出できる．また，時間順と逆順の処理単位間 W 枝を最初の枝とする経路 1 と，時間順と逆順のを最初の枝とする経路 2 が，経路 2,1 という形式で接続されることによる複合経路（例 1）は，$\prec$ の推移性により対処できる．

[定義 1] グラフ判定方式 IGT の効率版とは，次のように各操作 $A_i(x) \in H$ の $FTS[A_i(x)]$ と処理単位間の関連 $\prec$ を管理し，閉路を検出する方式である．ただし，各 FTS の初期値は空である．

a) $A_i(x) <_H W_j(x)$, $i \neq j$ の場合：
$FTS[W_j(x)] := FTS[W_j(x)] \cup \{i\}$；
$i \prec j$ を登録；
if $j \prec j$ then 閉路発見；

b) $W_i(x) <_H R_j(x)$, $i \neq j$ の場合：
$FTS[R_j(x)] := FTS[R_j(x)] \cup FTS[W_i(x)]$；
For every $k \in FTS[W_i(x)]$ DO
$k \prec j$ を登録；
if $j \prec j$ then 閉路発見；

c) $A_i(x) <_i A_i(y)$ の場合：
$FTS[A_i(y)] := FTS[A_i(y)] \cup FTS[A_i(x)]$； □

例 1 の H の実行結果は次のようになり，推移的な関連 $1 \prec 1$, $3 \prec 3$ が生じ，閉路が検出される．

$R_1(x) <_H W_2(x)$：$FTS[W_2(x)] := \{1\}$；$1 \prec 2$ を登録；
$R_3(y) <_H W_4(y)$：$FTS[W_4(y)] := \{3\}$；$3 \prec 4$ を登録；
$W_2(x) <_H R_3(x)$：$FTS[R_3(x)] := \{1\}$；$1 \prec 3$ を登録；
$W_4(y) <_H R_1(y)$：$FTS[R_1(y)] := \{3\}$；$3 \prec 1$ を登録；
$1 \prec 1$, $3 \prec 3$ より閉路発見

[定理 2] H が論理性クラスに属するための必要十分条件は，IGT の効率版によって閉路が発見されないことである． □

証明： 必要性はすでに説明したので，十分性のみを証明する．IGT の効率版では，$DG(H)$ において T_i から T_j 内のある操作へ処理単位間 W 枝を最初の枝とする他の 2 種類の枝によって延長される経路が存在する時に，$i \prec j$ を定義する．また，$\prec$ の推移性により，そのような 2 つの経路の接続によって得られる複合経路に対してもこの結論が成り立つ．このため，$i \prec i$ が生じた時に $DG(H)$ に時間順と逆順の処理単位間 W 枝 $(A_i(z), W_j(y))$ を最初の枝とする経路による閉路 $A_i(z), W_j(y), \cdots, A_i(z)$ が存在する． □

一方，SGT については，各処理単位に関する情報がどの時点で削除できるかという問題がある[1]．例 1 の H において $R_1(y)$ が実行される時点で関連 $3 \prec 1$ が登録されることにより閉路が検出されたが，処理単位 T_3 はその時点ですでに終了している．このため，従来の時刻印方式のようにデータ項目ごとに情報を記録するように改良する．

[定義 2] 終了した処理単位情報の即時削除を考慮したグラフ判定方式 IGT の効率版は，次のようにデータ項目 x ごとに，$FTSR(x)$, $FTSW(x)$, $TNO(x)$（x に対する検索，変更操作を実行した処理単位の ID の集合）を設けた方式である．ただし，初期値は空である．

a) $R_j(x)$ の場合：
$FTSR(x) := FTSR(x) \cup FTSW(x)$；
For each $k \in FTSW(x)$ do $k \prec j$ を登録；
if $j \prec j$ then 閉路発見；$TNO(x) = TNO(x) \cup \{j\}$；
for each $R_j(y)(W_j(y)) <_i R_j(x)$ do
$FTSR(x) := FTSR(x) \cup FTSR(y)(FTSW(y))$；

b) $W_j(x)$ の場合：
$FTSW(x) := FTSW(x) \cup TNO(x)$；
for each $i \in TNO(x)$ do $i \prec j$ を登録；
if $j \prec j$ then 閉路発見；$TNO(x) = TNO(x) \cup \{j\}$；
for each $R_j(y)(W_j(y)) <_j W_j(x)$ do
$FTSW(x) := FTSW(x) \cup FTSR(y)(FTSW(y))$；

4 まとめ

IGT の効率版では従来の時刻印方式と同程度の速度で高水準データベースの並行性要求に答えている．先行処理単位集合を従来の最大値のみを記憶する方法からすべてを記憶する方法に変更しているので，記憶容量はその分必要であるが，大容量メモリ技術の今日では問題となるものではない．

参考文献

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

[2] 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，情処論，No. 12 (1994).

[3] 徐　海燕，古川哲也，史 一華：並行処理制御方式による独立化可能クラスと直列可能クラスの比較，情処論，Vol. 37, No. 8 (1996).