並列 Btree 構造における SMO 発生時の処理性能を改善する並行性制御方式

Parallel Btree Concurrency Control Method to Improve Performance for Wide SMOs

吉原 朋宏♡ 小林 大♡ 田口 亮♢ 上原 年博♢ 横田 治夫♠

Tomohiro YOSHIHARA Dai KOBAYASHI
Ryo TAGUCHI Toshihiro UEHARA
Haruo YOKOTA

並列 Btree 構造に適した並行性制御手法として INC-OPT 方式が提案されているが，広範囲に SMO(structure modification operation) が発生する時に，X ラッチ拡大に伴うコストから，処理性能が低下するという問題点がある．本稿では，楽観的なラッチカップリング中に SMO が発生するポイントのマーキングを行うことで，広範囲 SMO 発生時にも高い処理性能を維持することが可能な新手法を提案する．また，並列 Btree 構造である Fat-Btree を採用している自律ディスクに提案手法を実装し，従来手法との比較から，提案手法の有効性を示す．

As an effective concurrency control method for parallel Btree structures, the INC-OPT method has been proposed. However, the costs of spreading an X latch in the method are still high when structure modification operations (SMOs) occurred widely, which decreases the total performance. In this paper, a new concurrency control method marking a point of SMO occurrence during optimistic latch couplings is proposed to improve the performance even when SMOs occurred widely. To compare the performance of the proposed method and INC-OPT method, we implemented them on an autonomous-disk system adopting the Fat-Btree, a parallel Btree structure. The experimental results indicate that the proposed method is effective.

1. はじめに

データベース用無共有並列計算機における検索，更新処理は，参照されるデータが配置されている各 PE(Processing Element) 上で並列に実行されることが望ましい．アクセス集中による負荷の偏りが存在する場合，負荷が大きい PE がボトルネックとなり，全体の処理性能が低下してしまう．したがって，各 PE で負荷を均等化することは処理性能の向上につながり，負荷を均等にするためのデータ分配方式が重要になる [1, 2].

従来のデータ分配方式にはハッシュ分配方式や値域分配方式 [3] などがあるが，ハッシュ分配方式では領域指定された問い合わせや，連続したアクセスの I/O 回数を削減するクラスタ化に対応できないという欠点がある．一方，値域分配方式では，静的決定された分割境界に沿って分割するため，データ更新によってデータ配置の偏りが生じたときに均一化するコストが非常に大きくなる欠点がある．

♡ 学生会員 東京工業大学 大学院 情報理工学研究科 計算工学専攻 {yoshihara,daik}@de.cs.titech.ac.jp

♢ NHK 放送技術研究所 {taguchi.r-cs,uehara.t-jy}@nhk.or.jp

♠ 正会員 東京工業大学 学術国際情報センター yokota@cs.titech.ac.jp

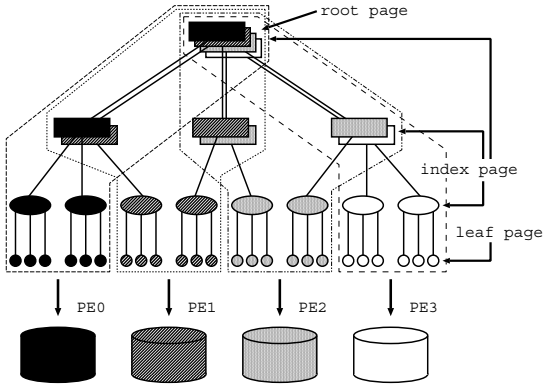

図 1: Fat-Btree
Fig.1 Fat-Btree

データ分配方式として値域分配方式を採用した上で，インデックス構造に並列 Btree を用いる研究がある．並列 Btree を利用することによって，両方式の欠点が解消でき，同時に高速アクセスが可能になる．しかし，従来の並列 Btree ではディレクトリ更新によるスループットの低下や，少数の PE へのアクセスの集中といった問題が生じる．

これらの問題を解決するために，新しい並列 Btree 構造として Fat-Btree が提案されている [4, 5, 6, 7, 8, 9, 10, 11, 12]．ディレクトリ構成として Fat-Btree を用いることで，単一キー問い合わせ，レンジ問い合わせが並列に高速実行できることが確率モデルの検証 [4], および nCUBE3[6] や LAN 環境での PC クラスタ [12] 上への実装による実験により明らかにされている．

Fat-Btree を含めた並列 Btree に適した並行性制御方式として INC-OPT 方式が提案されている [7, 10]．無共有並列計算機向けに設計されている INC-OPT は，従来の Btree の並行性制御方式である B-OPT[13] や ARIES/IM[14] より優れたパフォーマンスを示すことが明らかにされている [7, 10]．しかし，INC-OPT は Btree の構造変化を起こす操作 (SMO:Structure Modification Operation) が広範囲で行われるとき，ルートページからの木降下リスタートが多数必要となり，システムの処理性能の低下をもたらす．

本稿では，INC-OPT と比べて，SMO 発生時のリスタート回数を少なく抑えることが可能である新しい Fat-Btree の並行性制御方式を提案する．さらに，分散ディレクトリとして Fat-Btree を採用している自律ディスク [15, 16, 17] 上に INC-OPT および提案手法を実装し，台数を変化させた場合のスループットを測定し，提案手法の効果を示す．

2. Fat-Btree と並行性制御

2.1 Fat-Btree 構造

Fat-Btree は，B^+-Tree 全体をページ単位で PE 間で分配する並列 Btree の一種であり，データページである Btree の葉ページを各 PE に均等に分配し，格納している葉ページから見て再帰的に上位にあたるインデックスページにのみを各 PE に配置する (図 1)．更新頻度が高い下位のインデックスページほどそのコピーを持つ PE が減少していくため，ディレクトリ更新時に同期が必要となる PE 数が少なくなり，PE 間の局所的な通信によって更新処理を行うことができる．

また，各 PE では格納している葉ページの探索に必要のないインデックスページを持たないため，各 PE でインデックスページのキャッシュを行った場合にキャッシュのヒット率を高く保つことができる．高キャッシュヒット率により，更新処理だけでなく，探索処理も従来の並列 Btree 構造と比較して高速に行うことができる．

表 1: ラッチマトリクス
Table 1 Latch matrix

mode	IS	IX	S	SIX	X
IS	○	○	○	○	×
IX	○	○	×	×	×
S	○	×	○	×	×
SIX	○	×	×	×	×
X	×	×	×	×	×

2.2 並行性制御

木構造の一貫性を保証するために，Btree に並行性制御は必須である．通常，Btree や他のアクセスパスには，ロックの代わりにデッドロック検出機構を持たない高速かつ単純なラッチが用いられる．ラッチはセマフォの一種である．従って，アクセスパスの並行性制御はデッドロックフリーでなければならない．

2.2.1 ラッチモード

本稿では，5 種類のモードを持つラッチを仮定する．各モードは IS，IX，S，SIX，X であり，これらのモードの適合性は表 1 に示される [18]．表中の“ ○ ”は同時に複数のラッチが獲得可能なモードである．

あるインデックスページの複製が複数の PE に存在する場合を考える．もしラッチ処理が各 PE で分散して行われるならば，表 1 より，IS および IX モードはローカル PE のみで獲得するだけでよい．しかし，S，SIX および X モードは，コピーを持つすべての PE でラッチを獲得する必要がある．すなわち，S，SIX および X モードのラッチはデータのコピーが存在するとき，PE 間の同期オーバーヘッドが生じる．さらに各 PE に対してランダムにラッチを獲得しようとすればデッドロックの可能性がある．デッドロックを回避するためには PE 番号の昇順または他の順序に従いラッチを獲得する方法が有効である．しかし，これはラッチに関わる PE 数に比例して同期完了までの時間が長くなることを意味している．

2.2.2 Fat-Btree の並行性制御

前述のとおり，アクセスパスの並行性制御はデッドロックフリーでなければならない．

また，多数の PE にコピーを持つ Fat-Btree の上位ページにおいて，S，SIX および X ラッチを獲得することはできる限り避けなければならない．なぜなら，並列 Btree における S，SIX および X ラッチの獲得は PE 間の同期が必要で，多くの PE にコピーを持つページでのそれらのラッチの獲得は，PE 間の同期オーバーヘッドが増大させてしまうからである．

表 1 のラッチを用いる方式と異なる並行性制御方式として，木ラッチと呼ばれる Btree 全体のラッチを用いて，SMO 発生時の Btree の並行性制御を行う方式が存在する [14]．その方式を分散環境における並列 Btree で実現するためには，ある特定の PE で木ラッチを管理するのが最も簡単であるが，PE 数が増加すればその PE にラッチリクエストが集中するため，その PE がボトルネックとなる．また，木全体をラッチするため，PE 間の同期オーバーヘッドも大きい．

以上のことを要約すると，Fat-Btree を含む並列 Btree の並行性制御は，一般に次の条件を満たすことが望まれる [7, 10].

条件 2.1 並列 Btree の並行性制御には以下の三点を満たすことが要求される．

1. デッドロックフリーである
2. ルートおよび木の上位ページにできる限り S，SIX および X ラッチを獲得しない
3. 短時間であっても木全体をラッチすべきではない

優れた Btree の並行性制御方式として，B-OPT[13]，OPT-DLOCK[19] および ARIES/IM[14] などがあるが，これらは条件 2.1 を満たしていない．B-OPT は 2 を，OPT-DLOCK は 1 を満たさず，ARIES/IM は 3 を満たさない．よって，これらの並行性制御は並列 Btree である Fat-Btree に適さない．

2.2.3 INC-OPT 方式

Fat-Btree を含めた並列 Btree 向けの並行性制御方式として INC-OPT 方式は提案された [7, 10]．INC-OPT は条件 2.1 を満たしており，並列 Btree の性質に合致している．

INC-OPT の検索時のプロトコルは，IS モードによるラッチカップリングが使用される．まず，ルートページを IS モードでラッチした後，以下の処理を繰り返す．

1. 親ページのキーを比較して，子ページのポインタを取得
2. 子ページの IS ラッチを取り，親ページのラッチを解放

葉ページまで辿り着けば，葉ページに S ラッチを獲得しデータを読む．

INC-OPT での更新処理時のプロトコルは，二つのフェーズで行われる．

第 1 フェーズ: IX ラッチカップリングで葉ページまで辿る (葉ページは X ラッチが獲得される)．もし，SMO が起きないならば葉ページを更新して終了．もし葉ページがスプリットを起こし SMO が起こるならば，葉ページの X ラッチを解放し，第 2 フェーズに移行する．

第 2 フェーズ: 葉ページとその親ページを X モードでラッチする．もし親ページもスプリットするならば，同様にして X ラッチの範囲を拡大していく．十分な X ラッチが獲得されたならば更新操作を実行する．

この手続きの厳密な定義は [7, 10] を参照されたい．

INC-OPT は，更新処理のプロトコルからわかるように，SMO 発生時のリスタートが複数回必要になることがある．ルートページまで更新が及ぶ場合には，木の高さと同じフェーズ数が必要となる．このことは更新命令のレスポンスタイムを増加させるとともに，上位ページで複数回 X ラッチを獲得することで，システム全体のスループットも低下させる．

3. 提案手法

並列 Btree である Fat-Btree 向けの並行性制御方式を提案する．従来の INC-OPT は，広範囲に SMO が発生するときに，X ラッチ拡大のために複数回のリスタートが必要だった．それに対し提案手法は，SMO が発生する木の高さをマークすることにより，広範囲の SMO 発生時も，ほとんどの場合 1 回のリスタートで更新フェーズに移ることができるので，リスタート回数を少なく抑えることが可能である．よって，リスタート回数を少なく抑えることによるレスポンスタイムの向上と，中間の X ラッチ拡大フェーズの除去による不必要な X ラッチの獲得の減少から，高更新環境において従来手法より高いスループットを得ることが期待できる．提案手法は INC-OPT の拡張であるため，検索処理時のプロトコルは同様であるが，更新処理時のプロトコルが異なる．

提案手法での更新処理時のプロトコルは次のように行われる．

第 1 フェーズ: IX ラッチカップリングで葉ページまで辿る (葉ページは X ラッチが獲得される)．フルエントリー (次にエントリーが増えたときにスプリットする状態のこと) ではないインデックスページが存在したら，そのページのルートページからの高さをマークする．マークする高さは，フルエントリーではないページに遭遇するたびに逐次更新される．もし，SMO が起きないならば葉ページを更新して終了．もし葉ページがスプリットを起こし SMO が起こるならば，葉ページの X ラッチを解放し，第 2 フェーズに移行する．

第 2 フェーズ: 第 1 フェーズと同様に木の高さのマークは行う．前フェーズでマークした高さ以下のインデックスページと葉ページを X モードでラッチする．もし獲得した X ラッチの範囲が SMO の範囲に対して十分でなければ，同様にマークを用い

```
↑   1   l := H;
    2   Parent := null; Child := ROOT; h := 1; m := 1;
    3   while h < l do begin
    4       Child に IX ラッチ，Parent のラッチ解放;
第  5       if Child がフルエントリーではない then
1   6           m := h;              /* マーキング */
フ  7       NewChild を決定;
ェ  8       Parent := Child; Child := NewChild; h := h + 1;
ー  9   end;
ズ  10  Child とそのコピーに X ラッチ，Parent のラッチ解放;
    11  if Child がフルエントリー then
    12      すべてのラッチを解放; l := m;
    13  else
↓   14      更新操作を実行; 全ラッチ解放し終了;
↑   15  Parent := null; Child := ROOT; h := 1; m := 1;
    16  while h < l do begin
    17      Child に IX ラッチ，Parent のラッチ解放;
    18      if Child がフルエントリーではない then
    19          m := h;              /* マーキング */
    20      NewChild を決定;
    21      Parent := Child; Child := NewChild; h := h + 1;
第  22  end;
2   23  Child とそのコピーに X ラッチ，Parent のラッチ解放;
フ  24  NewChild を決定;
ェ  25  Parent := Child; Child := NewChild; h := h + 1;
ー  26  while h ≤ H do begin
ズ  27      Child とそのコピーに X ラッチ;
    28      NewChild を決定;
    29      Parent := Child; Child := NewChild; h := h + 1;
    30  end;
    31  if 獲得した X ラッチが SMO に不十分 then
    32      すべてのラッチを解放; l := m; goto 15;
    33  else
↓   34      更新操作 (SMO) を実行; 全ラッチ解放し終了;
```

図 2: 提案プロトコル
Fig.2 Proposed protocol

て X ラッチの範囲を変化させていく．十分な X ラッチが獲得されたならば更新操作を実行する．

上記の手続きを厳密に定義する．木の高さを H とすれば，ページのレベル (h) はルートが 1，葉ページが H となる．また，l を X ラッチを獲得し始めるページのレベルとすれば，l は初め H にセットされる．そして，木の降下時にマークする値は m にセットされる．このとき，提案プロトコルは図 2 で定義される．

提案手法はマークした前フェーズの状態をもとにラッチ範囲を決定している．そのため，前フェーズからの木構造の変化による SMO 範囲の拡大により，複数回リスタートが必要となる場合もあるが，INC-OPT 同様に最大フェーズ数は高々H に抑えられる．SMO 範囲が複数のレベルに渡って同時に拡大することは極めて稀であり，多くの場合は 1 回のリスタートで処理を完了できる．よって，上位ページまで SMO が及ぶときでも，INC-OPT のように多くのリスタートを必要としない．

また，提案手法は次の 3 点からわかるように条件 2.1 を満たしている．

1. トップダウンにラッチを獲得するためデッドロックフリーである
2. S，SIX ラッチをインデックスページには使用せず，SMO に関係しない不必要な X ラッチも獲得しない
3. 木全体のラッチは存在しない

すなわち，提案手法は並列 Btree の更新の性質に合致している．

4. 実験

提案する新たな並行性制御の有効性を示すために，Fat-Btree を採用している自律ディスク [15, 16, 17] に提案手法を実装し実験を行う．

4.1 実験環境

実験は，我々の提案する分散ストレージ技術である自律ディスクの模擬実装上で行う．これは Linux クラスタ上に Java を用いて模擬実装されている．今回の実験では表 2 に示す構成の PC と十分なバックボーン性能を持つネットワークスイッチを用いて，実験環境を構成した．

表 2: 実験システムの性能諸元
Table 2 Spec of PCs used for experiments

ノード	32-128 台 (Storage)，32 台 (Clients)
CPU	AMD Athlon XP-M1800+ (1.53GHz)
メモリ	PC2100 DDR SDRAM 1GB
ディスク	TOSHIBA MK3019GAX (30GB，5400rpm，2.5inch)
OS	Linux 2.4.20
Java VM	Sun J2SE SDK 1.4.2_04 Server VM

表 3: 実験システムのパラメータ
Table 3 Parameters used for experiments

ページサイズ	4KB
タプルサイズ	400B
インデックスページのキー数	最大 64
葉ページのタプル数	最大 8

初期 Fat-Btree の構築

本来自律ディスクにはデータマイグレーションによる負荷分散機能 [16, 17] を備えているが，今回の実験用の自律ディスクには実装されていない．よって，Fat-Btree の葉ページを移動させることによる動的な負荷分散 [11] を行うことはできない．この場合，どのようにして挿入するタプルを均等に PE に分散させるかが問題になってくる．そこで今回は初期 Fat-Btree として各 PE に葉ページを 1 ページずつ作成し，これらの葉ページのキーをその PE に格納されるキーの下限値とする方法をとった．PE 番号の昇順に，初期葉ページのキーの値が大きくなるようにしておくことにより，以後の挿入処理では各 PE に格納される葉ページが静的に分割される．この初期 Fat-Btree に対してランダムな要素を繰り返し挿入したものを実験に用いる．今回の実験における Fat-Btree の構成パラメータはを表 3 に示す．このようなパラメータに設定したのは，提案手法の効果を明確にするためである．

4.2 実験方法

このような環境下の自律ディスク上に上記の手法で初期 Fat-Btree を構築した後，各クライアント PC から同時にリクエストを送信する．キーはランダムに選び，検索と挿入操作を実行したときのスループットから性能を評価する．操作を送るのは，ランダムに選択した PE(自律ディスク) に対してである．

4.3 自律ディスクの台数を変化させたときの比較

32 台のクライアント PC(1 台あたり並行して 4 スレッド) からリクエストを送信し，10 秒間操作したときのスループットを測定した．更新比率 20%，自律ディスク 1 台数あたりのデータベースサイズ 3200 タプルのときの Fat-Btree を，自律ディスクの台数を 8 から 128 に変化させ，INC-OPT と提案手法のスループットを測定した結果が表 4 である．

INC-OPT は，台数増加に伴うスループットの上昇が小さい．これは，上位ページに SMO が及ぶ場合複数回のリスタートが発生し，毎フェーズ X ラッチを獲得することによる PE 間同期オーバーヘッドが大きくなるからである．それに対し，提案手法は上位ページに SMO が及ぶときでも，ほとんどの場合 1 回のリスタートのみで済み，X ラッチの獲得は最小限になるので，PE 間同期オーバーヘッドが小さい．よって，提案手法は台数増加に伴うスループットの上昇が大きい．

5. まとめと今後の課題

本稿では，無共有並列計算機に適したディレクトリ構造である並列 Btree の並行性制御として新手法を提案した．これは，SMO 発生によりリスタートが必要となったとき，前フェーズの木の状態に基づき X ラッチの獲得開始位置をマークし，リスタート後の SMO に必要な X ラッチの範囲を決定する並行性制御方式である．さらに自律ディスク上での実験により，提案手法と従来手法との比較を行い，提案手法の有効性を確認した．

本稿ではアクセスパターンが各 PE に対して均一であると仮定

表 4: 自律ディスクの台数を変化させたときの
並行性制御方式の比較
Table 4 Comparison of concurrency control protocols with changing number of Autonomous Disks

Number of Autonomous Disks	Throughput (operations/sec)	
	INC-OPT	Proposed method
8	3098.26	3246.41
16	3167.05	3262.67
32	5239.43	5386.33
64	8993.58	9164.48
128	13620.61	14591.57

したが，実際は偏りがある場合が多い．そのときには，自律ディスクの機能である自律的なデータ移動による負荷分散機能も用いて実験を行う必要がある．また，負荷分散時の並行性制御についても検討しなければならない．

さらに，優れた並行性制御を実現できることが知られている B-link[20, 21] の Fat-Btree への適用を検討している．B-link は，サイドポインタにより隣のインデックスノードにリンクをもっている．サイドポインタがあることにより，ラッチカップリングを用いず，単一ノードラッチによる並行性制御を行うことができる．また，リスタートを行うことはなく，すべての処理が 1 フェーズで行われる．B-link の Fat-Btree への適用方法を検討し，適用したものと従来手法との比較が必要である．

[謝辞]

Fat-Btree の並行性制御に関して奈良先端科学技術大学の宮崎純助教授に助言を頂いた．ここに感謝の意を表します．また本研究の一部は，独立行政法人科学技術振興機構戦略的創造研究推進事業 CREST，情報ストレージ研究推進機構 (SRC)，文部科学省科学研究費補助金特定領域研究 (16016232) および東京工業大学 21 世紀 COE プログラム「大規模知識資源の体系化と活用基盤構築」の助成により行なわれた．

[文献]

[1] G. Copeland, W. Alexander, E. Boughter, and T. Keller. Data Placement in Bubba. In *Proc. of ACM SIGMOD conf. '88*.

[2] S. Ghandeharizadeh and D. J. DeWitt. HybridRange Partitioning Strategy: A New Declustering Strategy for Multiprocessor Database Machines. In *Proc. of VLDB Conf. '90*, 1990.

[3] D. DeWitt and J. Gray. Parallel Database Systems: The Future of High Performance Database Systems. In *Communications of the ACM*, Vol. 35, pp. 85–98, Jun. 1992.

[4] 金政泰彦 , 宮崎純 , 横田治夫. 並列データベースシステムにおける更新を考慮したディレクトリ構成. 信学技報，DE97-77(AI97-44). 電子情報通信学会, 1997.

[5] Haruo YOKOTA, Yasuhiko KANEMASA, and Jun MIYAZAKI. Fat-Btree: An Update-Conscious Parallel Directory Structure. In *Proc. of IEEE ICDE Conf. '99*, pp. 448–457, Mar. 1999.

[6] 宮崎純 , 横田治夫. 無共有並列計算機向けディレクトリ構造 Fat-Btree の実装とその評価. 情処学会研究会報告，データベースシステム DBS-119-68, pp. 407–412. 情報処理学会, Jul. 1999.

[7] 宮崎純 , 横田治夫. 並列ディレクトリ構造 Fat-Btree の並行性制御とその評価. 情処学会研究会報告，データベースシステム DBS-124-69, pp. 37–44. 情報処理学会, Jul. 2001.

[8] 宮崎純 , 横田治夫. 並列ディレクトリ Fat-Btree のリカバリについて．信学技報，DE2001-107, pp. 17–24. 電子情報通信学会, Oct. 2001.

[9] 宮崎純 , 横田治夫. 高信頼 Fat-Btree 構成への neighbor-WAL プロトコルの適用. 第 13 回データ工学ワークショップ論文集，DEWS2002，C1-2. 電子情報通信学会, Mar. 2002.

[10] Jun MIYAZAKI and Haruo YOKOTA. Concurrency Control and Performance Evaluation of Parallel B-tree Structures. In *IEICE Transactions on Information and Systems*, Vol. E85-D, pp. 1269–1283. The Institute of Electronics, Information and Communication Engineers, Aug. 2002.

[11] 鈴木裕通 , 横田治夫. 並列ディレクトリ構造 Fat-Btree における負荷分散の手法とその実装. 第 11 回データ工学ワークショップ論文集，DEWS2000，4B-4. 電子情報通信学会, Mar. 2000.

[12] 風戸広史 , 横田治夫. 並列ディレクトリ構造 Fat-Btree におけるレンジ問い合わせの取り扱い．第 12 回データ工学ワークショップ論文集，DEWS2001，7A-7. 電子情報通信学会, Mar. 2001.

[13] R. Bayer and M. Schkolnick. Concurrency of Operations on B-trees. In *Acta Informatica*, Vol. 9, pp. 1–21, 1977.

[14] C. Mohan and F. Levine. ARIES/IM: an Efficient and High Concurrency Index Management Method Using Write-Ahead Logging. In *Proc. of ACM SIGMOD Conf. '92*, pp. 371–381, 1992.

[15] Haruo Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pp. 441–448, Nov. 1999.

[16] 風戸広史 , 横田治夫. 自律ディスクへの Fat-Btree の実装. 情処学会研究会報告，データベースシステム DBS-125-71, pp. 45–52. 情報処理学会, Jul. 2001.

[17] 伊藤大輔 , 風戸広史 , 横田治夫. 負荷分散機構と組み合わせた自律ディスクのクラスタ再構築. 信学技報，FTS2001-20, pp. 119–126. 電子情報通信学会, Jul. 2001.

[18] J. Gray and A. Reuter. *Transaction Proceeding: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 1993.

[19] V. Srinivasan and M. J. Carey. Performance of B-Tree Concurrency Control Algorithms. In *Proc. of ACM SIGMOD Conf. '91*, pp. 416–425, 1991.

[20] P. L. Lehman and S. B. Yao. Efficient Locking for Concurrent Operations on B-trees. In *ACM Trans. Database Syst.*, Vol. 6, pp. 650–670, 1981.

[21] V. Lanin and D. Shasha. A Symmetric Concurrent B-tree Algorithm. In *Proc. FJCC*, pp. 380–389, 1986.

吉原 朋宏　Tomohiro YOSHIHARA
平 17 東工大・工・情工卒．同大大学院・情報理工・計算工・修士課程在学中．日本データベース学会学生会員．

小林 大　Dai KOBAYASHI
平 17 東工大大学院・情報理工・計算工・修士課程了．同大大学院・情報理工・計算工・博士課程在学中．日本データベース学会学生会員．

田口 亮　Ryo TAGUCHI
平 6 慶應義塾大大学院・理工・計測工・修士課程了．同年より NHK 放送技術研究所．映像情報メディア学会会員．

上原 年博　Toshihiro UEHARA
昭 56 慶應義塾大・工・電気工・修士課程了．昭 59 より NHK 放送技術研究所．電子情報通信学会，映像情報メディア学会各会員．

横田 治夫　Haruo YOKOTA
昭 55 東工大・工・電物卒．昭 57 同大大学院・情報・修士課程了．同年富士通 (株)．同年 6 月 (財) 新世代コンピュータ技術開発機構研究所．昭 61(株) 富士通研究所．平 4 北陸先端大・情報・助教授．平 10 東工大・情報理工・助教授．平 13 東工大・学術国際情報センター・教授．工博．日本データベース学会, 電子情報通信学会, 情報処理学会，人工知能学会，IEEE，ACM 各会員．