アウトオブオーダ型データベースエンジン OoODE による構造劣化軽減効果に関する実験的考察

合田　和生†　　喜連川　優†

† 東京大学 生産技術研究所　〒 153-8503 東京都目黒区駒場 4-6-1
E-mail: †{kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

あらまし　一般にディスクアクセスの応答時間はシーク距離に影響される．問合せの種類にも依るが，データベースの問合せ処理にかかる実行時間は，当該応答時間の蓄積によって支配されることが多く，即ち，ディスク上の物理レイアウトによる影響が無視できない．データベースに更新が行われるにつれ，データベースの初期ロード時に期待されたデータ構造の効率性が失われ，クエリ処理性能が低下していくという構造劣化現象が見られる場合がある．これまで，再編成なる手法が，構造劣化の解消を目指して研究され，多くの DBMS に組み込まれるに到っている．本論文では，当該問題に対する新たなアプローチとして，問合せ処理に係るディスクアクセスを**非同期化**することにより，軽減を図る．著者らが考案した**アウトオブオーダ型データベースエンジン（OoODE）**は，高多重の非同期ディスクアクセスを行う点に特徴がある．ディスクアクセスの非同期化によって，問合せの実行時間は個々のディスクアクセスの応答時間からより影響されにくくなり，むしろ，ディスクアクセスのスループットによって決定される比重が高まり，一部の構造劣化現象の軽減が期待される．本論文では，著者らが実装したアウトオブオーダ型データベースエンジンの試作機を用いた実験を行い，当該データベースエンジンによる構造劣化現象の軽減効果を検証する．

キーワード　OoODE，アウトオブオーダ型実行，非同期ディスクアクセス，構造劣化

Kazuo GODA† and Masaru KITSUREGAWA†

† Institute of Industrial Science, The University of Tokyo　Komaba 4-6-1, Meguro-ku, Tokyo, 153-8505 Japan
E-mail: †{kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

1. はじめに

一般にディスクアクセスの応答時間はシーク距離に影響される．データベースの問合せ処理にかかる実行時間は，通常，応答時間の蓄積に影響され，即ち，ディスク上の物理レイアウトによって影響されることが多い．データベースに更新が行われるにつれ，データベースの初期ロード時に期待されたデータ構造の効率性が失われ，問合せ処理性能が低下していくという現象が見られる場合があり，当該現象は構造劣化と呼ばれる [31]．例えば，B+木 [2,27] においては，論理的に隣接したページは，同一のディスクトラック上において，更には物理的に隣接していることが望ましい．しかし，B+木の特定のページに偏って多数のレコード挿入を実施すると，当該ページは満杯となり分割され，物理的に離れた位置に新たなページを獲得し，レコードが格納されることになる．このようなページ分割を繰り返すと，徐々にページの物理位置と論理的なキー値との相関は低下することとなり，B+木の走査はディスク上ではランダムなアクセスとなるため，検索性能が大幅に低下する [25]．構造劣化による性能低下は，膨大なデータをディスクドライブをはじめとするストレージ上に格納するデータベース管理システム（DBMS）にとって，最も深刻な問題の一つである．

当該問題を解決すべく，これまでディスク上のデータを再配置することにより，劣化した構造を回復し性能を改善するデータベース再編成 [21] が研究されてきた．データベース再編成は，今日，多くの DBMS に組み込まれ，データベースの性能管理に不可欠な機能と言えよう．オフライン再編成は最も単純な再編成の方式である．一般に，再編成はデータの再配置のために膨大なディスクアクセスを要するため，負荷は極めて大きく長時間の処理を必要とする．このためデータベースの構造劣化と再編成のコストを慎重に見極めて，再編成を実施するかどうかの判断をすることがデータベース管理者には求められる．構造劣化による性能悪化と再編成コストをモデル化することにより，長期的なデータベース運用に於いて必要な再編成の間隔や時間を見積り，最適化を試みる研究が行われて来た [1,3,4,6–8,10,16,19,28]．また，近年ではデータベースには極めて高い可用性が要請されていることから，サービス継続中に再編成を実行可能なオンラ

イン再編成 [5, 9, 11–13, 17, 20, 23, 29–31] に関する研究が盛んに行われて来た．しかし，昨今のハードウェア環境においては，プロセッサならびに主記憶と比較して，ディスクストレージの帯域は相対的に限られており，フロントエンドの問合せ処理とバックエンドのデータベース再編成を同時に実行する場合，当該帯域がボトルネックとなりやすい．ディスクアクセスの競合は避けずらく，また同時にその高速化も容易ではないという問題が存在している．

本論文では，当該問題に対する新たなアプローチとして，問合せ処理に係るディスクアクセスを非同期化することにより，構造劣化の軽減を図りたい．著者らが考案したアウトオブオーダ型データベースエンジン [33] は，高多重のディスクアクセス発行を行う点に特徴がある．ディスクアクセスの非同期化によって，問合せの実行時間は個々のディスクアクセスの応答時間からより影響されにくくなる．ストレージ装置の有するディスクアクセス制御機構は，高度なスケジューリング機能 [18, 26] を具備するに至っており，シーク等の処理オーバヘッドを最小限に抑えるべくアクセス要求の順序を入れ替え，全体としてのスループットを向上させることが可能となっている．例えば，連続的ではない複数のディスクアクセス要求を受けた場合，制御機構はアドレス順に並べ替えて実行することにより，全体のシーク距離を最小化し，スループットを向上させることが可能である．ディスクアクセスを非同期化することにより，ストレージの持つスケジューリング機能を活用し，構造劣化による性能低下を回避することを狙う．

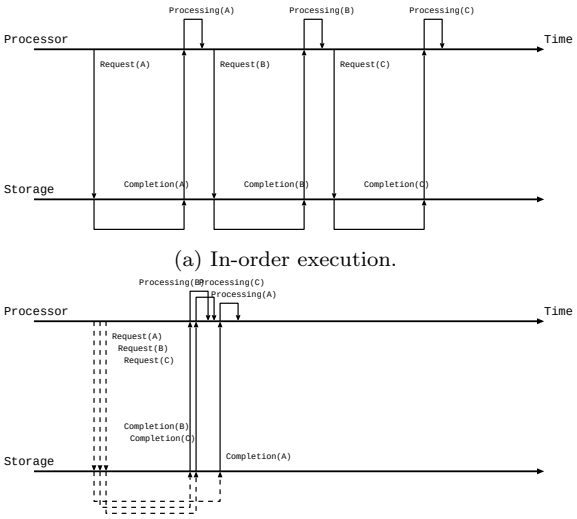
本論文では，これまで著者らが進めてきたオープンソース DBMS をベースとするアウトオブオーダ型データベースエンジンの試作機の実装 [32] を用いて実験を行い，当該データベースエンジンによる構造劣化現象の軽減効果を検証し，そのロバスト性を明らかにする．

本論文の構成は以下の通りである．第 2 章では，アウトオブオーダ型データベースエンジンの仕組みを簡単に紹介するとともに，著者らが進めている試作機の実装を示す．第 3 章では，当該試作機において TPC-H ベンチマークデータセットを用いて行った構造劣化の軽減効果の検証実験を報告し，第 4 章において本論文を纏める．

2. アウトオブオーダ型データベースエンジン

今日の多くの DBMS ではデータベースエンジンにインオーダ型の実行方式が採用されている．即ち，問合せ処理の実行時に，問合せ実行木の演算ノードにおいてディスクアクセスと演算を逐次的に繰り返し，実行時間としては，ディスクアクセスにかかる応答時間が蓄積することとなる．これに対して，著者らの考案したアウトオブオーダ型の実行方式では，問合せ処理の実行時に，演算ノードにおいて新たなディスクアクセスを発行する必要が生じると，都度にタスク分解を行い，分解された並行実行可能なタスク上でディスクアクセスと関連する演算を行う．すなわち，ディスクアクセスを非同期化する．ソフトウェアとしての実装は多様な方式が考えられるが，問合せ処理の実行時に，実行論理の許す限りにおいて，必要に応じてタスク分解が行われ，多数の非同期ディスクアクセスがストレージに発行されることとなる．図 1 に，インオーダ型の実行方式と著者らの提案するアウトオブオーダ型の実行方式の相違を，模式的に示す．現実的には，資源量は有限であるため，利用可能な主記憶の容量によって並行実行するタスクの数の上限は制約され，また，ディスクドライブやディスクアレイ，ホストバスアダプタや OS カーネルの処理能力によって同時処理可能な非同期ディスクアクセスの数の上限が制約されるものの，最近のサーバにおいては数百程度の主記憶容量は珍しくなく，またストレージシステムも数百のディスクドライブを搭載するに到っており [22]，従来に対してより多くのタスクとディスクアクセスの同時処理が可能となってきている．また，ディスクドライブ内，ディスクアレイコントローラ内さらには OS 内の高度なスケジューリング機構 [18, 26] により，論理的なディスクアクセス発行順序とは異なる順序でディスクアクセスの処理がなされるのが通例である．即ち，アウトオブオーダ型の実行方式では，ディスクアクセスの完了によって演算が駆動されるべく制御がなされることとなる．従来のインオーダ型の実行方式は，プロセッサ性能と主記憶容量を節約するべく，極めて少量のディスクアクセスのみを同時に発行していた．これにに対し，アウトオブオーダ型の実行方式は，ソフトウェアの非同期化によって，実行論理が許す限り大量のディスクアクセスの同時発行を可能とするものであって，本来は，これにより，極めて高速化の進んでいるプロセッサ資源の活用と，膨大な数のディスクドライブの並列アクセスによって，効果的な性能バランスの達成を狙うものであるが，加えて，本論文では，ディスクアクセスを非同期化することにより，ストレージの持つスケジューリング機能を活用し，構造劣化による性能低下を軽減することを狙いたい．今日の情報システム構成の中で，ストレージが性能に寄与する割合は高まっており，その効果は大きいと考えられる．

著者らは，これまで，オープンソースのデータベースソフトウェアをベースとしたアウトオブオーダ型データベースエン



(a) In-order execution.

(b) Out-of-order execution.

図 1　データベースエンジンの実行方式の比較.
Fig. 1　Comparison of execution architectures in database engines.

ジンの試作実装を行ってきた．本論文では，その一実装である MySQL [14] をベースとする試作実装を用いた実験を行う．なお，試作実装の詳細に関しては [32] を参考にされたい．

3. アウトオブオーダ型データベースエンジンの試作実装を用いた構造劣化の軽減硬化の検証

データベースの更新によって生じる構造劣化には，ストレージエンジンの実装や問合せの種別によって，多様な類型が考えられ，また，存在するが，本論文では，多くのデータベースに見られ，また，性能上の影響の大きいページ格納順序の劣化なる現象に焦点を絞りたい．例えば，B+木の葉ページの格納順序の効率性が低下すると，本来シーケンシャルアクセスとなることを期待している全表検索などに伴う走査は，ランダムアクセスを伴うこととなり，大幅な性能低下を招くことになる．本論文では，当該現象を軽減することをめざし，アウトオブオーダ型の実行方式の有効性を検証する．

この際，MySQL における InnoDB ストレージエンジンを用いた問合せ処理を想定したい．ストレージエンジンによるデータベースへのアクセス方式にはいくつかの方式があるが，ここでは，上記の構造劣化の影響を被りやすい全表検索ならびにクラスタ化キーによる範囲検索について検証する．

全表検索　MySQL の表は，B+木によって構成されており，レコードは葉ページに格納されている．全表検索は，まず，B+木の左側端の葉ページを探索した後，カーソルを用いてレコードフェッチを繰り返す．この際，葉ページの右側水平ポインタを用いて葉ページが走査される．葉ページの格納順序が効率的であれば，走査に伴うディスクアクセスはシーケンシャルとなるが，格納順序が劣化している場合，走査はランダムアクセスを伴い，走査の性能は低下する．

クラスタ化キーによる範囲検索　MySQL のクラスタ化機能は，Oracle における索引構成表 [15] と同様の構成により表の格納する．即ち，一次索引と表とが同一の B+木によって構成される．B+木のキーはクラスタ化キーであり，レコードは葉ページに格納される．範囲検索は，まず，範囲条件の左側境界条件に基づいて葉ページを探索した後，カーソルを用いてレコードフェッチを行い，範囲検索の右側境界条件に至るまで繰り返す．この際，葉ページの右側水平ポインタを用いて葉ページが走査される．全表検索と同様に，格納順序が劣化している場合，走査はランダムアクセスを伴い，走査の性能は低下する．

アウトオブオーダ型の実行方式による構造劣化の軽減効果を検証するために，MySQL ならびに当該 MySQL をベースとするアウトオブオーダ型データベースエンジンの試作実装を用い，代表的なデータベースベンチマークである TPC-H [24] を利用して性能評価を行った．なお，上述のとおり，MySQL では葉ページを走査する際に水平ポインタを用いるが，アウトオブオーダ型データベースエンジンの試作実装では，親ページからの垂直ポインタを用いることとしている．

表 1 に著者らが構築した実験システムの諸元を示す．4 台のファイバチャネルディスクドライブを用いてパリティなしストライピング（RAID-0，セグメントサイズ 64KB）編成のボリュームを構成し，当該ボリュームからページ長 16KB，空間長 256GB の InnoDB 表空間を構築し，当該表空間には 512MB のデータベースバッファを割り当てた．

表 1　実験システムの諸元.
Table 1　Experimental setup.

Server hardware (Dell PowerEdge)	
Processors	2x Intel Xeon 3.2GHz (2p/2c)
Memory	2048MB
HBA	Emulex LP10000DC
Storage JBOD hardware (JMR Fortra 2G6)	
Disk drives	4x 15,000rpm 146GB FC HDDs (database)
Software	
OS	RedHat Linux 3.0 (customized kernel)
DBMS	OoODE (prototype, MySQL based) MySQL 4.1.1

TPC-H は典型的な意志決定支援データウェアハウス応用のベンチマークであり，23 個の問合せと 2 個の更新操作が定義されている．RF1(New Sales Refresh Function) は新しい売り上げ記録を `ORDERS` 表及び `LINEITEM` 表に追加し，RF2(Old Sales Refresh Function) は古い売り上げ記録を削除する．RF1 と RF2 の組合せによりデータウェアハウスの一部が更新されるが，データウェアハウスの論理的な大きさは殆ど変化しない．実験は以下の通り行った．図 2 に示す通り，クラスタ化表と索引をスキーマとして定義し，初期データを生成した．RF1 及び RF2 の組合せを 100 回実行することにより徐々にデータウェアハウスを更新し，定期的に図 2 に示す 6 つの問い合わせを実行し，その応答時間の変化を測定した．以下に測定した 6 つの問い合わせを示す．

- ORDERS 表の全表検索
- LINEITEM 表の全表検索
- ORDERS 表の主キー (クラスタ化キー) による範囲検索
- LINEITEM 表の主キー (クラスタ化キー) による範囲検索

この際，RF1 及び RF2 の更新率は 1%とし，DBMS のキャッシュ効果を除くため，問い合わせ毎に共有バッファをクリアした．

問合せ実行時間の変化を図 3，図 4 に纏める．まず，全表検索に関しては，通常の MySQL，即ち，インオーダ型の実行方式では，`ORDERS`，`LINEITEM` とも更新を行うにつれ，実行時間は長くなり，100 回更新を行なった時点では，4-5 倍程度まで悪化している．一方，アウトオブオーダ型データベースエンジンの試作実装では，アウトオブオーダ型の実行方式によって，更新に伴う悪化は大きく抑えられており，100 回更新を行なった時点では，悪化を比較的抑制できており，平均的に 1.5 倍程度となっていることが見て取れる．

次に，クラスタ化キーによる範囲検索についても，全表検索と同様であり，アウトオブオーダ型の実行方式により構造劣化の軽減効果があることが確認できる．

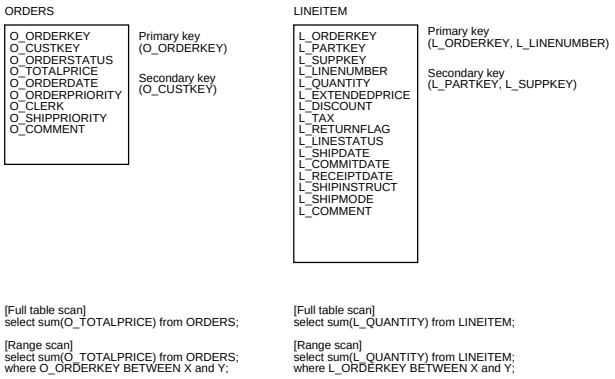
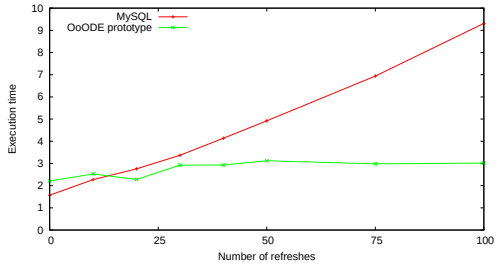
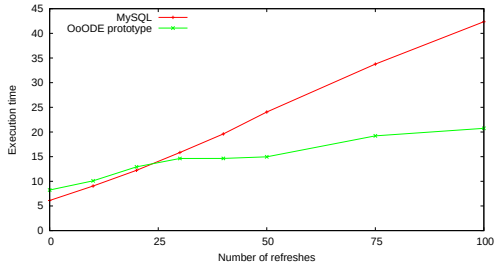


図 2　スキーマと問い合わせ.

Fig. 2　Database schema and queries.



(a) ORDERS.



(b) LINEITEM.

図 3　全表検索にかかる実行時間.

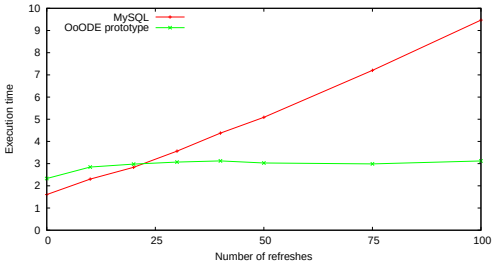
Fig. 3　Execution time of full table scan.

最後に，全表検索におけるディスクアクセスをトレースし，著者らの開発したディスクアクセスの可視化ツールで可視化した図を図 5 に示す．これより，MySQL，即ち，インオーダ型の実行方式では，構造劣化によってディスクアクセスが著しくランダム化しているのに対して，アウトオブオーダ型データベースエンジンの試作実装では，アウトオブオーダ型の実行方式によって，非同期的に発行されたディスクアクセス要求がスケジューリングされ，実際のディスクアクセスがシーケンシャル化され，構造劣化の影響を軽減していることが分かる．

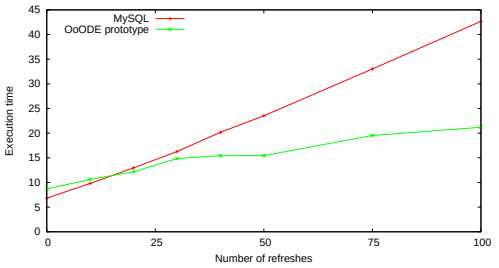
以上のことから，アウトオブオーダ型の実行方式によって，構造劣化の一種である格納順序の劣化が軽減されたことが判る．

4. おわりに

本論文では，構造劣化なるデータベースの性能低下問題に対する新たな解決アプローチとして，データベースの問合せ処理に係るディスクアクセスを**非同期化**することにより，当該問題の軽減を図る試みを示した．著者らが考案したアウトオブオーダ型データベースエンジンは，高多重の非同期ディスクアクセス発行を行う点に特徴があり，ディスクアクセスの非同期化によって，クエリの実行時間を個々のディスクアクセスの応答時間からより影響されにくくし，さらに，ストレージシステムが備えるスケジューリング能力を活用することを可能とし，これによって，構造劣化現象の軽減が期待される．本論文では，オープンソース DBMS である MySQL を対象とした試作機を実装し，TPC-H ベンチマークデータセットを用いてその有効性を検証した．この結果，全表検索を例に取ると，従来的なインオーダ型問合せ処理を行うデータベース管理システムで，4-5 倍程度の構造劣化による性能低下が見られるのに対して，著者のアプローチによると，平均的には 1.5 倍程度まで性能低下を抑えることが可能となり，構造劣化現象に対するロバスト性を確認することができた．



(a) ORDERS.



(b) LINEITEM.

図 4　範囲検索にかかる実行時間.

Fig. 4　Execution time of range scan.

本論文の検証実験は小規模かつ荒削りなものに留まっており，実験規模を拡大し，また，より広範な問合せへの影響を確認してゆきたい．

謝　辞

本研究の一部は，内閣府最先端研究開発支援プログラム「超巨大データベース時代に向けた最高速データベースエンジンの開発と当該エンジンを核とする戦略的社会サービスの実証・評価」の助成により行われた．

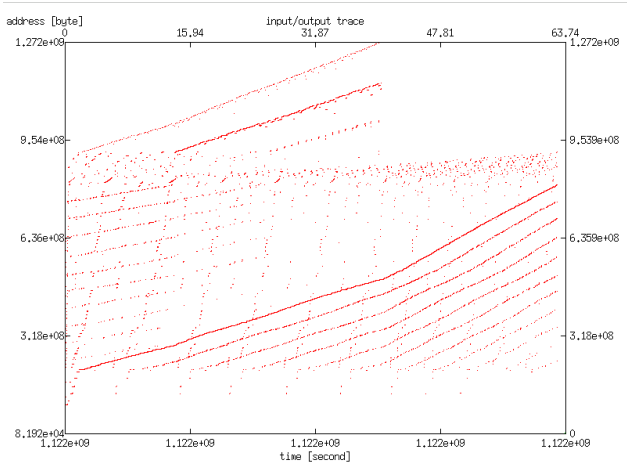
文　献

[1] D. S. Batory. Optimal file designs and reorganization points. *ACM Trans. Database Syst.*, Vol. 7, No. 1, pp. 60–81, 1982.

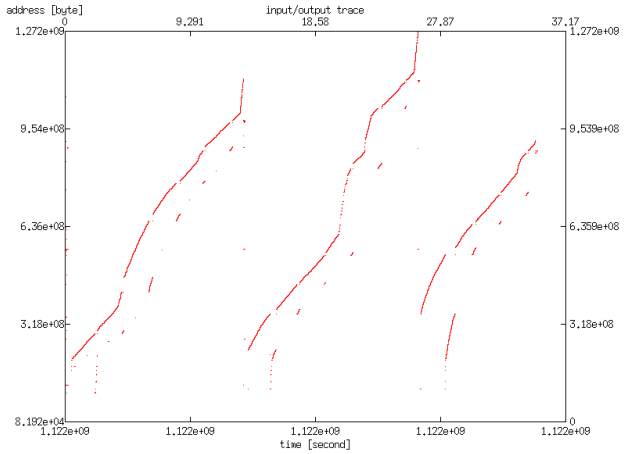
[2] D. Comer. The ubiquitous B-tree. *ACM Comput. Surv.*, Vol. 11, pp. 121–137, 1979.

[3] K. T. Fung. A Reorganization Model Based on the Database Entropy Concept. *Comput. J.*, Vol. 27, No. 1, pp. 67–71, 1984.

[4] Carsten A. Gerlhof, Alfons Kemper, and Guido Moerkotte.



(a) Full table scan on `LINEITEM` (MySQL).



(b) Full table scan on `LINEITEM` (OoODE prototype).

図 5　ディスクアクセス挙動の比較.

Fig. 5　Comparison of disk accesses.

On the Cost of Monitoring and Reorganization of Object Bases for Clustering. *SIGMOD Record*, Vol. 25, No. 3, pp. 22–27, 1996.

[5] S. Ghandeharizadeh and D. Kim. On-line Reorganization of Data in Scalable Continuous Media Servers. In *Proc. Int'l. Conf. on Database and Expert Syst. Applications*, pp. 751–768, 1996.

[6] Takashi Hoshino, Kazuo Goda, and Masaru Kitsuregawa. Online monitoring and visualisation of database structural deterioration. *Int'l J. on Autonomic Comput.*, Vol. 1, No. 3, pp. 297–323, 2010.

[7] William G. Tuel Jr. Optimum Reorganization Points for Linearly Growing Files. *ACM Trans. Database Syst.*, Vol. 3, No. 1, pp. 32–40, 1978.

[8] G. M. Lohman and J. A. Muckstadt. Optimal Policy for Batch Operations: Backup, Checkpointing, Reorganization, and Updating. *ACM Trans. Database Syst.*, Vol. 2, No. 3, pp. 209–222, 1977.

[9] David Lomet, editor. *IEEE Data Eng. Bull.: Special Issue on Online Reorganization.*, Vol. 19. IEEE Computer Society, 1996.

[10] K. Maruyama and S. E. Smith. Optimal Reorganization of Distributed Space Disk Files. *Comm. ACM*, Vol. 19, No. 11, pp. 634–642, 1976.

[11] A. Mohamed, G. Candia, and D. Sherwin. Comparing Architectures of Online Reorganization. White Paper, Quest Software, 2002.

[12] E. Omiecinski, L. Lee, and P. Scheuermann. Performance Analysis of a Concurrent File Reorganization Algorithm for Record Clustering. *IEEE Trans. Knowl. Data Eng.*, Vol. 6, No. 2, pp. 248–257, 1994.

[13] E. Omiecinski and P. Scheuermann. A Global Approach to Record Clustering and File Reorganization. In *Proc. ACM SIGIR Conf.*, pp. 201–219, 1984.

[14] Oracle Corp. MySQL: The World's Most Popular Open Source Database. `http://www.mysql.com/`.

[15] Oracle Corp. Oracle 9i Index-Organized Tables Technical Whitepaper. White Paper, 2001.

[16] June S. Park and V. Sridhar. Probabilistic Model and Optimal Reorganization of B+-Tree with Physical Clustering. *IEEE Trans. Knowl. Data Eng.*, Vol. 9, No. 5, pp. 826–832, 1997.

[17] Betty Salzberg and Allyn Dimock. Principles of Transaction-Based On-Line Reorganization. In *Proc. Int'l. Conf. on Very Large Data Base*, pp. 511–520, 1992.

[18] Margo I. Seltzer, Peter M. Chen, and John K. Ousterout. Disk Scheduling Revisited. In *Proc. USENIX Tech. Conf.*, pp. 313–323, 1990.

[19] B. Shneiderman. Optimum Data Base Reorganization Points. *Comm. ACM*, Vol. 16, No. 6, pp. 362–365, 1973.

[20] G. H. Sockut, T. A. Beavin, and C-C. Chang. A Method for On-Line Reorganization of a Database. *IBM Syst. J.*, Vol. 36, No. 3, pp. 411–436, 1997.

[21] Gary H. Sockut and Robert P. Goldberg. Database Reorganization - Principles and Practice. *ACM Comput. Surv.*, Vol. 11, No. 4, pp. 371–395, 1979.

[22] N. Takahashi and H. Yoshida. Hitachi TagmaStore Universal Storage Platform: Virtualization without Limits. White Paper, Hitachi Ltd., 2004.

[23] E. Thereska, J. Schindler, J. S. Bucky, B. Salmon, C. R. Lumb, and G. R. Ganger. A framework for building unobtrusive disk maintenance applications. In *Proc. USENIX Conf. on File and Storage Tech.*, pp. 213–226, 2004.

[24] Transaction Processing Performance Council. TPC-H, an ad-doc, decision support benchmark. `http://www.tpc.org/tpch/`.

[25] S. Watanabe and T. Miura. Reordering B-tree Files. In *Proc. ACM Symp. on Applied Comput.*, pp. 681–686, 2002.

[26] Bruce L. Worthington, Gregory R. Ganger, and Yale N. Patt. Scheduling Algorithms for Modern Disk Drives. In *Proc. ACM SIGMETRICS Conf.*, pp. 241–251, 1994.

[27] A. Chi-Chih Yao. On random 2-3 trees. *Acta Informatica*, Vol. 9, pp. 159–170, 1978.

[28] S. B. Yao, K. S. Das, and T. J. Teorey. A Dynamic Database Reorganization Algorithm. *ACM Trans. Database Syst.*, Vol. 1, No. 2, pp. 159–174, 1976.

[29] P. Zabback, I. Onyksel, P. Scheuermann, and G. Weikum. Database Reorganization in Parallel Disk Arrays with I/O Service Stealing. *IEEE Trans. Knowl. Data Eng.*, Vol. 10, No. 5, pp. 855–858, 1998.

[30] Chendong Zou and Betty Salzberg. On-line Reorganization of Sparsely-populated B+-trees. In *Proc. ACM SIGMOD Conf.*, pp. 115–124, 1996.

[31] 合田 和生, 喜連川 優. データベース再編成機構を有するストレージシステム. 情報処理学会論文誌：データベース, Vol. 46, No. SIG 8(TOD 26), pp. 130–147, 2005.

[32] 合田和生, 豊田正史, 喜連川優. アウトオブオーダ型データベースエンジン OoODE の試作とその実行挙動. 電子情報通信学会／日本データベース学会データ工学と情報マネジメントに関するフォーラム, 2013 (to appear).

[33] 合田和生, 喜連川優. アウトオブオーダ型データベースエンジン OoODE の構想と初期実験. 日本データベース学会論文誌, Vol. 8, No. 1, pp. 131–136, 2009.