構造劣化の局所性を活かしたデータベース部分再編成の提案

合田　和生† 　喜連川　優†

† 東京大学 生産技術研究所 〒 153-8505 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　データベースにおいては一般に更新操作が繰り返されるにつれて，例えば，レコードの並びが乱雑化し，連続したキー値を有するレコードの走査に要するコストが大きく増加するなど，格納構造が劣化することによりアクセス性能が低下する現象，即ち構造劣化が発生する．多くのデータベースにおける更新は局所性を有していることから，構造劣化は記憶空間のうち一部の限られた領域に発生する場合が多い．本論文では上記の特性に着目し，構造が劣化した局所空間に対して再編成を行うことにより，再編成実行時間を大きく削減し，空間全体を再編成する場合とほぼ同程度の構造回復効果を実現する部分再編成手法を提案する．構造劣化が局所性を有する２つの代表的なケーススタディによって，構造劣化空間の同定方式，並びに当該同定空間の再編成方式を示す．また，試作機において実装した部分再編成機構の性能評価を示すことにより，その有効性を明らかにする．
キーワード　データベース再編成，データベース管理，構造劣化，物理データ構造

Database Partial Reorganization for Exploiting Spatial Locality of Structural Deterioration

Kazuo GODA† and Masaru KITSUREGAWA†

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

Abstract　As updates are repeatedly performed in database, the data structure may gradually deteriorate and then the data access performance may also degrade. Specifically speaking, records are usually expected to be physically placed in order in many types of data structures. However, a number of record manipulations may lead to scrambled placement, resulting in much larger cost of record scanning. Such a phenomenon is called *structural deterioration*. As database has usually access locality, its data structure may also deteriorate in limited parts of the storage space. Exploiting this characteristic, the paper proposes a new reorganization method, *partial reorganization*. The method is able to reorganize *only* locally structurally deteriorated space in the database. Partial reorganization need not reorganize all the database, but can remove most of the structural deterioration. Thus the reorganization time can be significantly reduced, while the structural efficiency can be recovered similarly to the conventional method. This paper presents two typical case studies for investigating how to identify locally structurally deteriorated space and how to reorganize the identified deteriorated space. In addition, the paper presents experimental evaluations with an implemented prototype, which confirm the effectivity of the proposal.
Key words　Database Reorganization, Database Administration, Structural Deterioration, Physical Data Structure

1. はじめに

記憶装置上のデータは，更新によってデータ構造の編成が乱れ，データアクセスの性能が著しく低下する恐れがあり，当該現象を構造劣化（structural deterioration）と呼ぶ [37]．例えば，B+木 [3,32] においては，論理的に隣接したページが記憶装置上においても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鍵値との相関は低下することとなり，B+木の走査は記憶装置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 [30]．とりわけ，二次記憶装置のアクセス性能はディスクドライブの機械的特性に依存するため，構造劣化による性能低下は，膨大なデータを二次記憶装置上に格納するデータベー

スシステム [8, 15, 19, 22, 39] 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，従来より，記憶装置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 (database reorganization) [26] をデータベースシステムの一機能として実現する試みが行われて来ており，データベース再編成は今日では，データベースシステムに不可欠な機能となっている．一般に，再編成はデータベース空間中の全データの再配置を行うため，膨大な IO を発行する必要があり，その負荷は極めて大きく，処理は長時間に渡る．このため，データベース再編成の高度化，並びにデータベース再編成の管理に関して，多くの様々な研究がこれまで行われて来た [1, 2, 4, 6, 7, 11–14, 16–18, 20, 23–25, 28, 33–38].

一方で，一般に，データベースの更新はデータベース空間全体に対して一様に行われることはなく，局所性を有していることが広く知られている．即ち，データベース空間の一部の限られた領域に多くの更新操作が集中するアクセス特性がしばしば見られる．このため，データベースの更新操作によって発生する構造劣化に関しても，同様に，データベース空間の一部の限られた領域において発生する場合が多い．本論文では，上記のような構造劣化の局所性に着目し，構造が劣化した局所空間に対して再編成を行うことにより，再編成実行時間を大きく削減し，空間全体を再編成する場合とほぼ同程度の構造回復効果を実現する部分再編成手法を提案する．

更に，本論文では，構造劣化が局所性を有する２つの代表的なケーススタディを用いて，構造劣化空間の同定方式，並びに当該同定空間の再編成方式に関して，詳しく論じる．また，自己再編成ストレージ [37] の試作機において実装した部分再編成機構の性能評価結果を示すことにより，その有効性を明らかにする．提案手法はとりわけ大規模なデータベースに対して極めて有効な方式といえる．著者の知る限り，これまでに，構造劣化の局所性に着目し，局所的に発生した構造劣化空間の同定方式，並びに当該空間の再編成方式を提案し，その有効性を明らかにする提案はなされていない．

本論文の構成は以下の通りである．2. においては，関連研究をまとめる．3. においては，本論文の提案する部分再編成に関して一般的な説明を行い，4. においては，局所的な構造劣化を２例示し，これをケーススタディとして，より具体的な部分再編成の検討を行う．5. においては，自己再編成ストレージにおける部分再編成の設計，及び実装を示し，これを用いたケーススタディの検証を行う．6. においては，本論文をまとめるとともに，今後の課題を示す．

2. 関 連 研 究

データベース再編成の高度化を目指す研究は，サービス継続中に再編成を実行可能とするオンライン再編成 [12] に関する研究を中心として行われて来た．その方式は主に同時実行再編成 [6, 16–18, 23, 28, 34–36] と分離再編成 [2, 7, 14, 20, 25] の 2 つに分類することができる [37].

これらの研究は，トランザクション処理とデータベース再編成処理を並行，若しくは並列に動作させ，トランザクション処理性能への副作用を最小限としながら再編成時間を短縮させる目的を以って，主に，同時実行制御方式，並びに IO 最適化方式に関する議論を中心として行われて来た．対して，本論文で提案する部分再編成は，データベースの構造劣化の局所性に着目し，構造劣化空間を同定し，当該空間のみの再編成を可能とする再編成方式を導入することにより，再編成時間の短縮を目的としており，着眼点は大きく異なる．

構造劣化の定量的な把握方式を明らかにする試みに関しては，これまで，データベース再編成の最適化，若しくは自律化を目指す研究の中心的課題として行われて来た．例えば，B. Shneiderman らによる文献 [24] では，構造劣化を考慮して検索コストと再編成コストを線形モデル化し，それに基づき，検索コストと再編成コストの総和を最小化するべく，最適な再編成間隔を導出する解析的手法が提案されている．当該手法は，S. B. Yao により拡張され，文献 [33] において非線型のモデルを扱うことのできるより一般的なアルゴリズムと，ヒューリスティックな解法が示されている．その他にも，ディスクファイルに焦点を当てた研究 [13] [1]，再編成以外にバックアップ等を含む各種のバッチ処理をスケジュールする研究 [11]，特定のデータベースに特化したより詳細なモデル化を提案する研究 [4, 5, 10, 21, 31]，ディスクアクセスコストモデルに基づく構造劣化の定量的な指標を提案する研究 [38] が行われてきた．

これらの研究は，本研究と同様に構造劣化の定量的な把握に基づく再編成の高度な管理を目指しているものの，再編成を時間軸上で制御しようとしている．対して，本研究の提案する部分再編成は，構造劣化の空間的な局所性を活用し，再編成を空間軸上で制御することを目指しており，目的を大きく異にする．

既に幾つかのデータベースベンダは，そのデータベースシステム製品において，類似の名称を有する部分再編成なる機能を実装しているが [9, 27]，これらの機能が予め，データベース空間を複数の区画に分割して，再編成を区画単位で実施するものであるであるのに対し，本論文は，データベース空間内で構造が劣化した任意の部分空間を同定し，当該空間の再編成を可能とする手法を提案していることから，その学術的意義は大きい．

3. データベース部分再編成

部分再編成 (partial reorganization) は，データベースにおいて，構造が劣化した局所空間に対して再編成を行うことにより，再編成実行時間を大きく削減し，空間全体を再編成する従来的な手法と比較してほぼ同程度の構造回復効果を実現することを目指す．

本章では，部分再編成の方式に関して，一般的にその枠組を述べ，実現のために必要な要素技術を明らかにする．次章以降においては，ケーススタディとして，具体的に構造劣化が局所的に発生する典型的な例を示し，より詳細な議論を行う．

一般に，従来的な再編成においては，データベース空間全体を再編成する，若しくはユーザが指定したオブジェクトや空間を再編成する．対して，本論文の提案する部分再編成は，データベース空間のうち構造が劣化した部分空間，即ち構造劣化空

間 (structurally deteriorated space) を再編成することから，その処理は以下の二手順によって実施される．

- **構造劣化空間の同定**
- **構造劣化空間の再編成**

第一手順によって，部分再編成の対象空間とするべき構造劣化空間を同定し，第二手順によって，当該同定空間の再編成を実施し，構造劣化の解消を行う．本論文では，上記 2 つの手順の手法に関して焦点を当てた議論を行う．

いずれの手順とも，その具体的な方式は，想定するデータ構造と問い合わせ操作に依存することから，詳細な議論は次章以降に譲り，本章では，その一般的な性質を述べる．

3.1 構造劣化空間の同定

構造劣化空間の同定に関しては，データベースの部分空間に対する構造劣化を定量的に把握することが必須要件である．また，部分再編成は再編成に係る実行時間の低減を目的としていることから，その計測コストを極力低減することが強く望まれる．上記の要件を満たす構造劣化の定量的な指標を用意することが重要である．

本論文では，後述のケーススタディにおいて，それぞれ構造劣化の定量的な指標，即ち構造劣化度 (structural deterioration degree) を導入する．ここでは，部分再編成を効率的に実施するために求められる構造劣化度の特徴を以下にまとめる(注1)．

- あるデータ構造と問い合わせ操作を想定した下で，正しく構造劣化空間を同定するためには，データベース空間の構造劣化度は，当該問い合わせ処理に係るデータ構造への近似的なアクセスコストを表す必要がある．
- データベース空間において，その部分空間である構造劣化空間を同定するためには，構造劣化度はデータベース空間の部分空間に対しても定義可能である必要がある．即ち，データベース空間中の特定のアドレス範囲や，鍵範囲などの部分空間に対して，構造劣化度は計測可能でなければならない．
- 構造劣化空間の同定のために，データベース空間全体を走査することは効率的でない．構造劣化度はデータベースの更新に応じて，インクリメンタルに計測可能である必要がある．

構造劣化を定量的に把握することにより，一定の閾値を用いて，それを上回った部分空間を構造劣化空間として同定し，部分再編成の対象とすることが可能となる．

3.2 構造劣化空間の再編成

同定された構造劣化空間の再編成に関しては，一般的な再編成と同様に，アンロード・ロード方式 [26] の再編成を行うことが可能である．即ち，データベースからレコードをアンロードし，索引等の鍵順序性のある構造に関しては整列を行い，再度，データベースにロードする．

なお，部分再編成においては，構造劣化は局所的に発生しており，その特徴がある程度判明していることから，構造劣化のパターンに応じて再編成方式を工夫することができる．後述のケーススタディでは，それぞれの場合に於ける再編成方式の工夫を併せて述べる．

(注1)：既に，著者らのグループは文献 [38] において，構造劣化の定量的な指標に関して詳細な議論を行っている．本論文では，部分再編成をケーススタディを用いて検討し，有効性を検証することに焦点を当てる．

4. ケーススタディによる部分再編成の検討

本章では，構造劣化が局所的に発生する典型的な例を 2 つ示し，ケーススタディとして，部分再編成における構造劣化空間の同定方式，並びに当該空間の再編成方式について，より具体的な議論を行う．

4.1 ケーススタディ 1: 局所的疎空間の縮退

通常，非クラスタ化表，即ち，レコード格納順序を意識しない関係表においてレコードが削除されると，格納構造においては当該レコードに削除フラグ(注2)が立てられるものの，その空間は即座には回収されない．これは，削除操作の応答性能を高めるとともに，ロールバック時のレコード回復を高速化する目的による．

上記の実装を有する多くのデータベースシステムにおいては，図 1 に示すように，非クラスタ化表に対してレコードの挿入と削除が繰り返される場合，表の一部の空間の充填率が低下し，著しくデータの格納効率の悪い疎な空間が発生し，全表検索の性能を低下させる．このようなアクセス特性は，一般に，Web 上の電子商取引のバスケットシステムなどに頻繁に見られる．電子商取引サイトの訪問者は商品を購入するため，一度は商品をバスケットに入れるが，その後，ある確率で商品の購入を止めバスケットから出す．バスケットに相当する表にはレコードの挿入と削除が繰り返されるため，当該表の挿入ポインタ(注3)近郊には他の空間に比べ著しく疎な空間が発生する．即ち，検索等に係る走査の効率が低下することから，全表検索などを伴う問い合わせ処理の性能が低下する恐れがある．

このような局所的に疎な空間が発生する構造劣化を解消するためには，当該疎空間中の空き空間を回収し，縮退させることにより，空間配置の効率性を高める必要がある．従来的には，非クラスタ化表の再編成は，データベース中の表から全レコードをアンロードし，再度レコードをロードすることにより，上記の縮退を実施する．しかし，一般的に疎空間は局所的に発生することから，上記のアンロード・ロード戦略による再編成は必ずしも効率的でない．むしろ，局所的な疎空間を把握し，当該空間のみをアンロード・ロードすることにより，全体をアンロード・ロードする場合と比較して，大幅に再編成の実行時間を削減するとともに，同程度の構造劣化の解消効果を達成することが可能である．

4.1.1 疎空間の同定

当該ケーススタディにおける構造劣化空間の同定は，構造劣化度として，データベース空間を構成する基本空間単位 (ページ，若しくはエクステント(注4)) 毎の空き率を計測することによ

(注2)：多くのデータベースシステムの実装では，スロットと呼ばれるページ内ポインタ列が存在し，削除フラグはスロット上に存在する．

(注3)：幾つかのデータベースシステムでは，非クラスタ化表への挿入ポインタとして，高水準位 (HWM: high-water mark) なるポインタを利用する．

(注4)：データベース構造中のオブジェクトに対する割り当て単位であって，一般には，隣接したページ群を意味する．

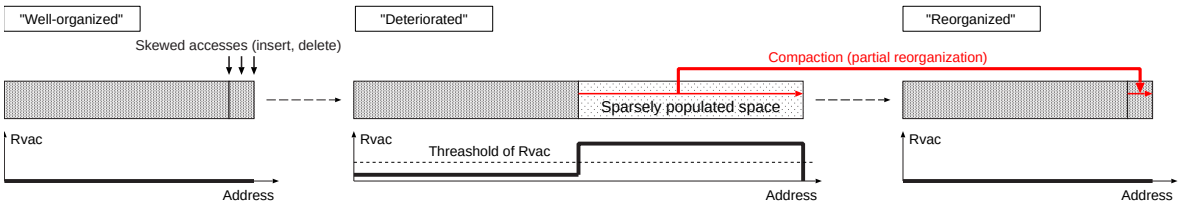


図 1　Identification and reorganization of structurally deteriorated space (case study #1.)

り，可能となる．例えば，基本空間単位をページとする場合，アドレス p によって示されるページの空き率 $R_{vac}(p)$ は以下の式により算出可能である．

$$R_{vac}(p) = \frac{L_{pag} - \sum_j L_{rec_j}}{L_{pag}} \quad (1)$$

ここに，L_{pag} はページ p の有効ページ長(注5)，$\sum_j L_{rec_j}$ はページ p 中の有効な全レコード長の総和を意味する．データベース空間中の各ページに対して空き率を計測することにより，図 1 に示すように，空き率閾値を下回った部分空間に関して，これを構造劣化空間と同定し，部分再編成の対象とすることが可能となる．

4.1.2　疎空間の縮退

同定された構造劣化空間に関しては，当該空間のアンロード・ロードにより，空間の縮退を行う．即ち，まず，構造劣化空間を走査して有効なレコードをアンロードし，その後，アンロードされたレコードを同じ空間にロードする．この際，レコードは詰めて配置される．

一般に，同じ空間へのアンロードとロードは並行して実行することができないが，この場合，空間を縮退させる操作であることが予め判明していることから，再編成を更に高速化されるためのテクニックとして，アンロード操作とロード操作の間に適切な処理バッファを置くことにより，アンロードとロードをパイプライン的に実行することが可能となり，これにより，高速に空間の縮退を行うことが期待される．

4.2　ケーススタディ 2: オーバーフローレコードの解消

多くのデータベースシステムはクラスタ化表，即ち，指定されたクラスタ鍵によって格納順序を決定する関係表を有しているが，その実装は，データベースシステムにより多岐に渡る．ここでは，図 2 に示すような，クラスタ索引ベースのクラスタ化表に焦点を当てて述べる．クラスタ化表へのレコード挿入については，クラスタ鍵によりクラスタ索引を辿り，挿入先ページが決定され，当該ページへレコードが格納される．この際，当該ページが満杯で挿入が失敗する場合，近傍ページへレコードの格納が試みられるが，それも失敗した場合，データベース空間内に新たなページが確保され，当該ページにレコードが格納される．この際，新たに割り当てられるページは，本来格納されるべきページから物理的に離れている可能性がある．本論文では，上記において，新規に割り当てられたページをオーバーフローページ，当該ページへ格納されたレコードをオーバーフローレコード，オーバーフローレコードが本来格納されるべきであったページをオーバーフロー元ページと呼ぶ．

(注5)：ページ内にはページ内データ構造管理のためのメタデータ領域が存在するため，有効ページ長は実ページ長より小さい．

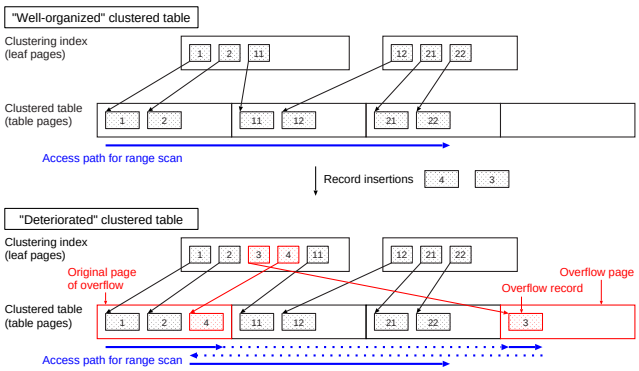


図 2　Example of clustered table.

クラスタ鍵による範囲検索を行う場合，鍵順にレコードを走査する必要があり，一般にクラスタ化表では当該走査がデータベースの物理空間のシーケンシャルアクセスとなることが期待されるが，オーバーフローレコードに関しては，本来格納されるべきページから離れたページに格納されることから，走査がランダムアクセスとなり，範囲検索の性能が低下する恐れがある．特に，一部の限られたクラスタ鍵範囲に挿入が繰り返される場合，上記のオーバーフローレコードが多発し，著しい性能低下を招く．このようなアクセス特性は，日付や時間をクラスタ鍵として利用する財務会計アプリケーション等に頻繁に見られる．年度や四半期の境界などの特定の日付に帳票が集中するため，偏った鍵分布の挿入が実施される．

オーバーフローレコードを解消するには，再編成が必要であるが，大きなクラスタ化表を再編成する場合，多数のランを伴う外部ソートが必須となり，再編成の時間が長くなる．このような場合，オーバーフローレコード以外の空間は殆ど構造は劣化していないため，部分再編成として，オーバーフローレコードの解消のみを行うことにより，効率的な再編成を実施することができる．

4.2.1　構造劣化空間の同定

当該ケーススタディにおける構造劣化空間の同定は，クラスタ鍵に対するレコード格納順序不正率の変動と，データベース空間の物理アドレスに対するクラスタ鍵の分布を計測することにより，可能となる．

前者は，クラスタ化索引の葉ページから計測され，当該葉ページに対応するクラスタ鍵範囲のレコードの格納順序の不正率を意味する．即ち，クラスタ索引の葉ページがクラスタ化鍵範囲 $[k_0, k_1)$ の索引レコードを有する場合，対応するクラスタ化表

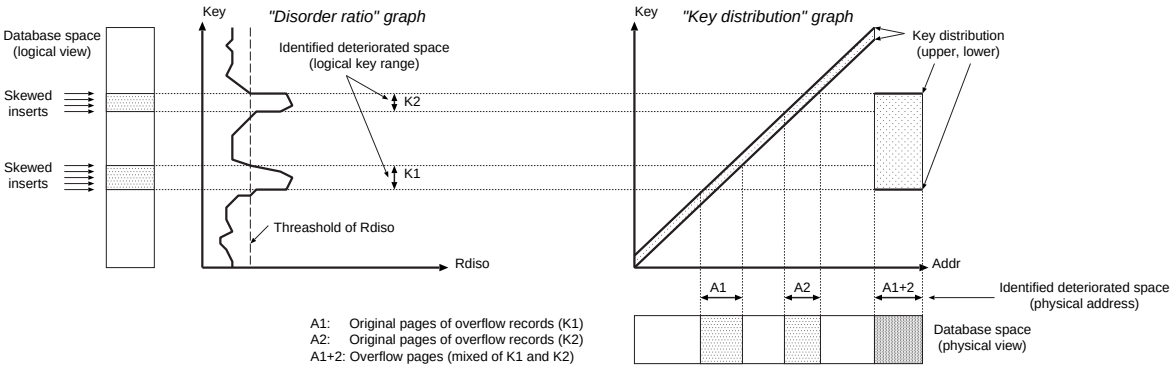


図 3　Identification of structurally deteriorated space (case study #2.)

の部分空間 $[k_0, k_1)$ のレコード格納順序不正率 $R_{diso}([k_0, k_1))$ は以下の通りとなる．

$$R_{diso}([k_0, k_1)) = \begin{cases} \frac{P^-}{P^+ + P^-} & (P^+ + P^- > 0) \\ 0 & (otherwise) \end{cases} \quad (2)$$

ここに，P^+ はクラスタ化表内で鍵範囲 $[k_0, k_1)$ のレコードを走査する際に，昇順にページを遷移する回数，P^- は降順にページを遷移する回数を意味する．P^+ 並びに P^- はクラスタ索引の葉ページから容易に算出することが可能である．

例えば，図 2 の下部に示す例では，レコード格納順序不正率をクラスタ化索引の 2 つの葉ページ毎に算出することができ，鍵範囲 [1, 12) の場合は $R_{diso}([1, 12)) = \frac{1}{2+1} \sim 0.33$，鍵範囲 [12,23) の場合は $R_{diso}([12, 23)) = \frac{0}{1+0} = 0.00$ となる．

一方，後者は，クラスタ化表のページから，当該ページに格納されているレコード群の鍵の平均値，並びに標準偏差を計測し，これに基づき，データベース空間におけるページに対するレコードの鍵分布を求める．

以上，レコード格納順序不正率の変動と，クラスタ鍵の分布を計測することにより，図 3 に示すように，以下の手順により，構造劣化空間を同定することが可能となる．まず，レコード格納順序不正率の高いクラスタ鍵分布範囲を同定し，その後，クラスタ鍵分布統計により，当該クラスタ鍵分布範囲に対する物理アドレス空間を同定する．このとき，同定された物理アドレス空間におけるクラスタ鍵範囲が広い空間をオーバーフローページ，それ以外をオーバーフローレコードの発生源ページ (オーバーフロー元ページ) とする．以上の手順により，クラスタ化表に対して局所的な挿入が発生したことにより生じたオーバーフローレコードに対する構造劣化空間，即ち，オーバーフローページとオーバーフロー元ページを同定することができる．

4.2.2　構造劣化空間の部分再編成

同定された構造劣化空間，即ち，オーバーフローページとオーバーフロー元ページに対して，オーバーフローレコードの解消を行う部分再編成は，図 4 に示すように，以下の 2 ステップの手順で実施する．

（1）　オーバーフロー元ページをアンロードし，レコードをクラスタ鍵順に整列し，新たな空間にロードする．オーバーフロー元ページでは，レコードは殆んど整列化されている，若しくは本来格納されるべきページの近傍に格納されていることが

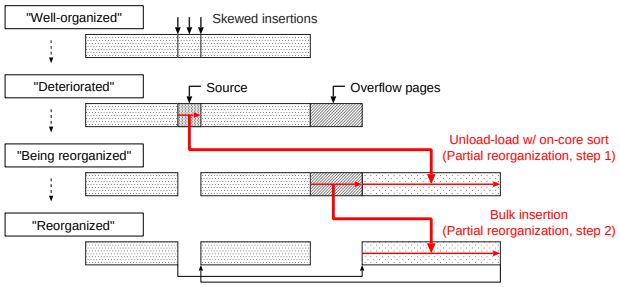


図 4　Reorganization of structurally deteriorated space (case study #2.)

期待されるため，一定のウィンドウ内で隣接するページ群内で整列操作を行うことにより，レコードを確実に整列することができる．即ち，多くの場合において，整列は外部整列ではなくオンコアで行うことが可能である．この際，ロードにおいては，次のステップでの挿入に備えて，一定の空き率を以って，即ち，予め充填率を低く設定してレコードを格納する必要がある．

（2）　次にオーバーフローページをアンロードし，レコードをクラスタ鍵順に整列し，先にロードされた空間にバルク挿入を実施する．

以上の手順により，クラスタ化表全体の外部整列を避け，必要な整列のみを実施することにより，オーバーフローレコードを解消することが可能となる．即ち，再編成時間の大幅な高速化が期待できる．

5.　自己再編成ストレージ試作機を用いた部分再編成の有効性検証

本章では，自己再編成ストレージ試作機を用いた部分再編成の実装，並びにそれを用いた有効性の検証に関して述べる．

5.1　自己再編成ストレージ試作機における部分再編成の設計と実装

5.1.1　自己再編成ストレージにおける部分再編成と実験環境

自己再編成ストレージは，オンラインデータベース再編成機能を有する高機能ディスクアレイである．従来，データベース再編成は，サーバ上の管理ソフトウェアによって実施されていたが，当該処理をストレージ内で実行することにより，ストレージ装置の有する高い IO 内部帯域を有効に活用して再編成を 1 桁以上高速にすることができるとともに，トランザクション処理への副作用を最小限に抑えることを可能とするなど，著

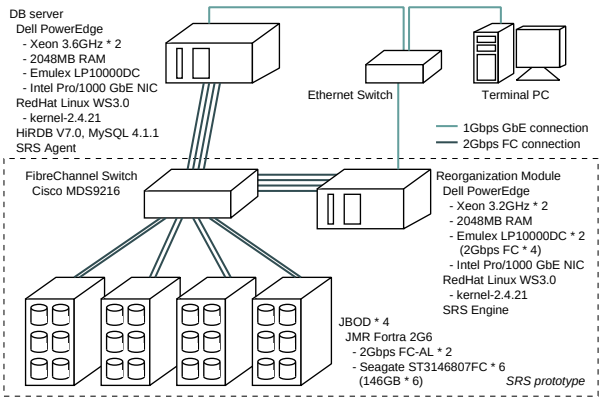


図 5　SRS prototype and its experimental system

しい利点が見込まれている [37].

著者らは前章で示したケーススタディの部分再編成を，自己再編成ストレージの試作機において実装した．この際，データベースシステムとして HiRDB [39] を対象とした．自己再編成ストレージの試作機並びに実験環境については，図 5 に概要を示す．本論文では部分再編成の設計，並びに実装に関して焦点を当てて述べることとし，自己再編成ストレージ試作機の設計，並びに実装の詳細に関しては [37] に稿を譲る．

部分再編成の実装にあたっては，自己再編成ストレージにおいて，ストレージ内の再編成モジュールにおいて実行されるソフトウェアとして，以下を開発した．

- 構造劣化空間同定機構
- 構造劣化空間再編成機構

前者がデータベース空間において構造が劣化した空間を同定し，後者が当該空間の再編成を実施する．

5.1.2　自己再編成ストレージにおける構造劣化空間の同定に関する検討

構造劣化空間同定機構は，常にデータベース空間内の構造劣化度を把握しており，部分再編成を開始するにあたっては，ケーススタディで検討した方式により，構造劣化空間を直ちに同定することが可能である．この際，当該機構は，ストレージ装置内のキャッシュメモリ上を盗み見ることにより，構造劣化度をインクリメンタルに，かつ低オーバヘッドで更新することが可能である．

ケーススタディ 1 に関しては，構造劣化度としてページ毎の充填率を監視する．サーバ上のデータベースシステムが，ページをストレージ側に書き込むと，当該ページはストレージ上のキャッシュメモリに一時的に存在する．構造劣化空間同定機構は，上記のイベントに応じて，キャッシュメモリ上の書き込まれたページを参照することにより，当該ページの充填率を容易に計測することができる．この際，構造劣化空間同定機構が必要とするメモリ長はデータベース長と比較するとさほど大きくはない．

ケーススタディ 2 に関しては，構造劣化空間の同定のために，クラスタ鍵範囲毎のレコード格納不正順序率，並びにページ毎のクラスタ鍵分布率を監視する．前者に関しては，データベースシステムが，クラスタ索引の葉ページをストレージに書き込んだ際に，構造劣化空間同定機構が当該ページをキャッシュメモリ上で参照することにより，当該ページに相当するクラスタ鍵範囲毎のレコード格納不正順序率を容易に計測することが可能である．後者に関しては，同様に，構造劣化空間同定機構がクラスタ化表のページをキャッシュメモリ上で参照することにより，容易に計測することが可能である．同様に，この際，構造劣化空間同定機構が必要とするメモリ長はデータベース長と比較するとさほど大きくはない．

なお，以下の実験においては，構造劣化の局所性に着目した部分再編成の有効性を検証することに焦点を当て，構造劣化空間同定機構はデータベースサーバ上で模擬することとした．ストレージプロセッサ上でキャッシュメモリを参照することの有効性に関しては，今後，実装による評価を行いたい．

5.2　ケーススタディ 1 の検証

代表的なデータベースベンチマークである TPC-H [29] のデータセットを利用して，ケーススタディ 1 における構造劣化空間の同定，並びに構造劣化空間の再編成の検証を行った．

データベースシステムである HiRDB の設定はページ長を 16KB，エクステント長を 32 ページ，共有バッファ長を 512MB とし，ストライピングユニット長を 64KB として 3 台のディスクドライブからストライピング構成される論理ユニット (LU) を 2 つ作成し，それぞれ表，並びに索引用の表空間として利用した．また，自己再編成ストレージの設定はページバッファ長を 512MB とし，IO 制御のキュー長をディスクドライブ毎に 64 とした．

局所的な疎空間の生成を模擬するために，表空間に TPC-H におけるスケールファクタ (SF) を 1 とした `ORDERS` 表 (150 万行) を初期データとして非クラスタ化表に格納し，当該表に対して，150 万レコードの挿入及び削除を実施した．この際，挿入及び削除は当該表の主鍵である `O_ID` について，[95%, 5%] の局所性を有することとした．即ち，レコードの挿入及び削除のうち，95%は `O_ID` のうち最大値から 5%の範囲内の鍵を対象に行われることを意味する．その後，ケーススタディ 1 において検証した部分再編成を実行した．

図 6 には，初期状態，並びに 150 万行が挿入及び削除された後の，ページ毎の空き率を示す．初期状態では，全ての有効なページの空き率がほぼ 0%であるのに対し，更新後には，データベース空間の後方にページ割り当てが広がり，周辺ページの空き率が低下していることが分かる．空き率の計測から，構造劣化空間同定機構は，ページ範囲 [13921, 28800] を構造劣化空間として同定した．この際の空き率閾値は 20%である．

また，図 7 には，上記において同定された構造劣化空間の再編成を実施した場合の実行時間，並びにその構造劣化の解消効果を，従来的にデータベース全体を再編成する場合と比較して，まとめる．図 7(a) に表空間全体の再編成と部分再編成の実行時間を比較する．図 7(b) には，初期状態，再編成前状態，並びに従来的な再編成、部分再編成それぞれに関して、再編成後状態の全表検索の実行時間を示す．データベースの更新により疎空間が発生し，全表検索の実行時間が 2 割超悪化したが，部分再編成によって十分に実行時間を回復することができたことが

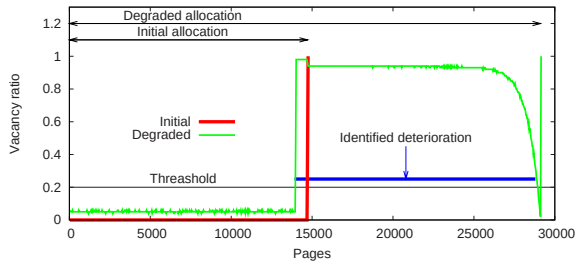
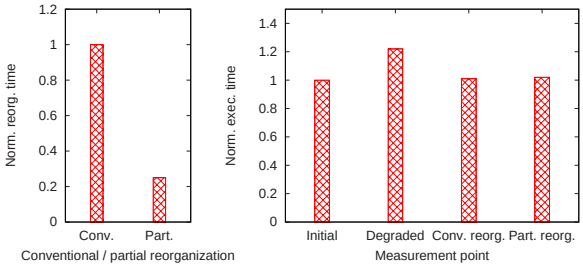


図 6　Identification results (case study #1.)



(a) Reorg. exec. time. (b) Query exec. time.

図 7　Partial reorganization results (case study #1.)

分かる．また，部分再編成により，従来的な再編成と比較して 77%の再編成実行時間の削減が可能となったことが分かる．

5.3　ケーススタディ 2 の検証

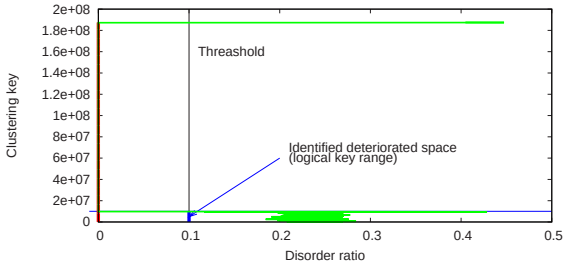
前節と同様に，ケーススタディ 2 における構造劣化空間の同定，並びに構造劣化空間の再編成の検証を行った．

局所的なクラスタ鍵空間への挿入を模擬するために，表空間に TPC-H におけるスケールファクタ (SF) を 1 とした ORDERS 表 (150 万行) を初期データとして充填率 70%でクラスタ化表に格納し，さらに，当該表に対して 150 万行の挿入を実施した．挿入は当該表の主鍵である O_ID について，[95%, 5%] の局所性を以って行った．即ち，挿入のうち，95%は O_ID のうち最大値から 5%の鍵を対象に行われる．その後，ケーススタディ 2 において検証した部分再編成を実行した．

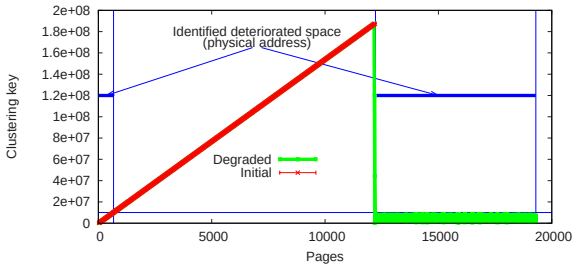
図 8 には，初期状態，並びに 150 万行が挿入及び削除された後の，クラスタ鍵に対するレコード不正順序格納率の変動，並びにページ毎のクラスタ鍵分布を示す．初期状態では，レコード不正格納が全く存在しないのに対し，更新後には，クラスタ鍵範囲 $[0, 1.0 \times 10^7]$ のレコードが不正順序で格納されていることが分かる(注6)．このため，当該鍵空間を，ページ毎のクラスタ鍵分布に照らし会わせることにより，レコードの不正順序格納はページ範囲 [1, 672] と [12225, 19296] において発生していることが分かる．更に，[1, 672] におけるクラスタ鍵の分布範囲は約 $\pm 2.5 \times 10^5$ であるのに対し，[12225, 19296] における分布範囲は約 $\pm 4.7 \times 10^6$ であることから，[1, 672] をオーバーフロー元ページ，[12224, 19296] をオーバーフローページと同定することができる．

また，図 9 には，上記において同定された構造劣化空間の再

(注6)：クラスタ鍵 1.8×10^8 近傍の範囲はオーバーフローページとの境界による変則性によっており，実際にその範囲は極めて小さいことから，ヒューリスティックなルールを同定機構に与えることにより，当該範囲は対象から除去した．

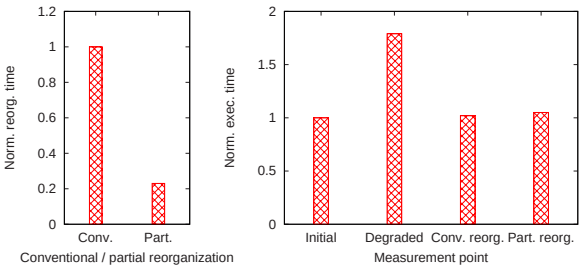


(a) Disorder ratio



(b) Key distribution

図 8　Identification results (case study #2.)



(a) Reorg. exec. time. (b) Query exec. time.

図 9　Partial reorganization results (case study #2.)

編成を実施した場合の実行時間，並びにその構造劣化の解消効果を，従来的にデータベース全体を再編成する場合と比較して，まとめる．図 9(a) に表空間全体の再編成と部分再編成の実行時間を比較する．図 9(b) には，初期状態，再編成前状態，並びに従来的な再編成、部分再編成それぞれに関して、再編成後状態の範囲検索の実行時間を示す．この際，実行時間はデータ量により正規化されている．データベースの更新によりオーバーフローレコードが発生し，範囲検索の実行時間が約 8 割悪化したが，部分再編成によって十分に実行時間を回復することができたことが分かる．また，部分再編成により，従来的な再編成と比較して 75%の再編成実行時間の削減が可能となったことが分かる．

以上，2 つのケーススタディの検証により，部分再編成により、従来的な再編成と比較して大幅な再編成実行時間の削減が可能となる一方、ほぼ同等の構造劣化の解消効果が達成可能であることが分かり，その有効性が確認された．

6.　ま　と　め

本論文では，データベースの更新が局所性を有することから，その構造劣化も局所的に発生することに着目し，データベース空間の一部のみを再編成することにより，再編成実行時間を大

きく削減し，空間全体を再編成する従来的な手法とほぼ同程度の構造回復効果を目指す部分再編成を提案した．部分再編成を実施するための，構造劣化空間の同定並びに再編成の方式に関して，典型的なデータベースの更新例を２つ示し，ケーススタディとして検討した．また，自己再編成ストレージ試作機において，当該２例の部分再編成を実装し，その有効性を検証した．ケーススタディにおいては，部分再編成により再編成に係る時間を 75-77%削減可能であり，かつ，同等の再編成効果を得られることを示した．

本論文は部分再編成の研究に関する第一ステップであると考えている．今後は，構造劣化空間の同定に関してストレージにおける構造劣化度の計測コストの検証を行う他，時間軸，空間軸双方における再編成管理の高度化に関して研究を進めたい．

謝　　辞

本研究の一部は，文部科学省リーディングプロジェクト e-Society 基盤ソフトウェアの総合開発「先進的なストレージ技術」の助成により行われた．協力企業である株式会社日立製作所より多くの有益なコメントを頂戴した．感謝する次第である．

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] BMC Software, Inc. Easing the Pain of an IMS Reorganization. White Paper, 2002.

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

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

[5] Carsten A. Gerlhof, Alfons Kemper, and Guido Moerkotte. On the Cost of Monitoring and Reorganization of Object Bases for Clustering. *SIGMOD Record*, Vol. 25, No. 3, pp. 22–27, 1996.

[6] 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.

[7] Hitachi Ltd. Nonstop operation: HiRDB. `http://www.hitachi.co.jp/Prod/comp/soft1/global/prod/hirdb/nonstop.html`.

[8] IBM. DB2 Product Family. `http://www.ibm.com/software/data/db2/`.

[9] IBM Corp. *Information Management System, Utilities Reference: Database and Transaction Manager, Version 8.*, 2004. SC27-1308-02.

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

[11] 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.

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

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

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

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

[16] E. Omiecinski. Incremental File Reorganization Schemes. In *Proc. Int'l. Conf. on Very Large Data Base*, pp. 346–357, 1985.

[17] 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.

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

[19] Oracle Corp. Oracle Database. `http://www.oracle.com/database/`.

[20] Oracle Corp. Oracle Database 10g Online Data Reorganization & Redefinition. White Paper, 2004.

[21] 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.

[22] PostgreSQL Global Development Group. The world's most advanced open source database. `http://www.postgreSQL.com/`.

[23] 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.

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

[25] 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.

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

[27] BMC Software. *REORG PLUS for DB2 General Information*, 1997. ARUPMG093097.

[28] 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.

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

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

[31] Vlad I. Wietrzyk and Mehmet A. Orgun. Dynamic Reorganization of Object Databases. In *Proc. Int'l. Symp. on Database Eng. and Applications*, pp. 110–118, 1999.

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

[33] 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.

[34] 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.

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

[36] Chendong Zou and Betty Salzberg. Safely and Efficiently Updating References During On-line Reorganization. In *Proc. Int'l. Conf. on Very Large Data Base*, pp. 512–522, 1998.

[37] 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.

[38] 星野 喬, 合田 和生, 喜連川 優. データベース更新差分を用いた範囲検索の IO コスト推定. 日本データベース学会 Letters, Vol. 4, No. 2, pp. 37–40, 2005.

[39] 日立製作所. Hitachi HiRDB Version 7. `http://www.hitachi.co.jp/Prod/comp/soft1/hirdb/`.