列指向型データ格納RDBMSを利用した分散OLAP問合せの処理最適化

塩井　隆円†　　波多野賢治††

† 同志社大学大学院文化情報学研究科　〒610–0394 京都府京田辺市多々羅都谷 1-3
†† 同志社大学文化情報学部　〒610–0394 京都府京田辺市多々羅都谷 1-3
E-mail: †shioi@ilab.doshish.ac.jp, ††khatano@mail.doshisha.ac.jp

あらまし　現在データ管理に利用されているデータベースシステムは，主に行指向型データ格納方式もしくは列指向型データ格納方式を採用している．しかし，高頻度に大量・複雑な非構造化データを短時間で格納・集計するビッグデータ解析の要求を満たすためには，これら二つのデータ格納方式を併用し，かつ負荷集中の問題を解決するために複数の計算機を利用した分散処理環境の元でデータベースシステムを運用する必要がある．本稿では，複数の計算機による分散処理環境として行指向/列指向双方のデータ格納方式をサポートする RDBMS の同期レプリケーションを利用した問合せ処理環境を構築し，効率的な問合せ処理のためのストレージ選択方式を提案する．評価実験の結果，提案手法は既存のストレージ選択条件と比較し，OLAP 問合せ処理速度を 5~25%高速化することができた．

キーワード　OLAP, RDB, 列指向型ストレージ, 負荷分散

1. はじめに

近年，リアルタイムに蓄積する大量のセンサデータを分析し，企業の意思決定支援に利用するビッグデータ分析が注目されている．特に，ビッグデータ解析のための情報システムでは Database Management System (以下，DBMS) として NoSQL 系 DBMS と Relational Database Management System (以下，RDBMS) を採用することで，大量のセンサデータ格納と効果的なデータ解析を実現している．

近年の DBMS では，高速なデータ分析のためにデータの参照効率が高く，Online Analytical Processing (以下，OLAP) 問合せを高速に実行できるとされている列指向型ストレージを採用することが一般的となっている．しかしながら，列指向型ストレージを採用した DBMS では Online Transactional Processing (以下，OLTP) 問合せの実行に適さず，センサデータに含まれるエラー修正のための更新処理がボトルネックとなり DBMS の作業負荷が高くなってしまうという問題点がある．

こうした問題点を解決するために C-store [4,8] や Fractured Mirrors [6] のような DBMS では，列指向型ストレージに加えて RDB で採用されている行指向型ストレージを一つの DBMS で扱う事で，OLAP と OLTP 両方の問合せを効率的に処理している [1,2,7].

しかし，ビッグデータを格納する場合には，分散処理環境を構成することに加え，データテーブルのサイズがリアルタイムに増加しつづけることや，列の値の幅が増加することが考えられるため，既存のオプティマイザにおける OLAP 問合せにおいて，コストベースのクエリプランの生成に時間が掛かり，上手く生成されたクエリプランも実行する時点には見積り通りの問合せ処理が可能であるとは言い難いのが現状である [3]．また，計算機の故障に備えて複数計算機を用いたレプリケーションによってシステムの可用性（availability）を高め，レプリケーションされた計算機に対してもホットスタンバイによる負荷分散を行うことでシステムの信頼性（reliability）を保つことが一般的となっているため，コストベースのクエリプランの生成がこれまで以上に困難となる．そのため，二種類のストレージを運用する DBMS を用いる際に，OLAP 問合せに対しどちらのストレージを用いて処理を行うかをルールベースで適切に処理する必要がある．

そこで本稿では，二種類のストレージを運用する DBMS の OLAP 問合せを効率的に実行するためにルールベースに基づくストレージ選択法を提案することで問合せ処理時間の高速化を行う．具体的には，DBMS を複数ノードに同期レプリケーションし，ホットスタンバイによる負荷分散を行う分散処理環境の下で，問合せの種類に対して使用するストレージ選択基準を作成することで問合せ処理高速化につなげていく．

2. 予備実験

OLAP 問合せにおいては，テーブル内の特定の列データを集計する処理が多く，行指向型ストレージでは処理に使用しない列まで読み出してしまうため，列ごとにデータを格納する列指向型ストレージを利用することで効率良く問合せ処理が可能であるとされている．この点を考慮して，C-Store [4,8] や Fractured Mirrors [6] では行指向型／列指向型ストレージを併用し，問合せの種類に応じて使用するストレージを選択する条件を設定した上で問合せ処理を行っている．本節では，既存の行指向型／列指向型ストレージの選択基準が問合せ処理にどれほど影響を与えているかを評価する予備実験の結果を述べる．

2.1 実験環境

予備実験に使用したデータは，米国トランザクション処理性能協議会 (Transaction Processing Performance Council: TPC) によって策定されたテストコレクションである TPC-H(注1) であり，それを行指向型／列指向型ストレージに格納した．本稿

(注1)：http://www.tpc.org/tpch/spec/tpch2.8.0.pdf

の実験では PostgreSQL 9.4.4 と PostgreSQL に列指向型ストレージ機能を追加するための cstore_fdw 1.3 を利用することで行指向型ストレージと列指向型ストレージの併用を実現している．列指向型ストレージに格納するデータは図 1 のようにテーブルごとに行指向型ストレージのデータから複製し，列ごとに圧縮して格納している．また，TPC-H によって定義されている OLAP 問合せ 22 種類をそれぞれ 5 回ずつ実行した平均実行時間をそれぞれの問合せの処理時間とした．

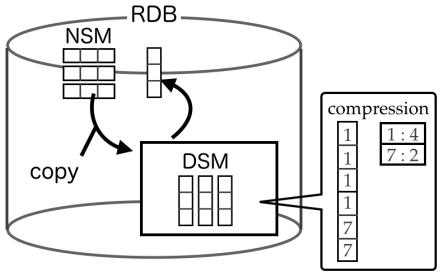

図 1　行指向型／列指向型ストレージを用いた DBMS

このとき分散処理環境を実現するために，図 2 のように 2 種類のストレージを利用した複数の計算機を同期レプリケーションした上で，OSS である Pgpool-Ⅱ 3.4 を用いて各計算機に問合せを分散する負荷分散環境を構築した．この実験環境を利用して，TPC-H によって定義されている 22 種類の OLAP 問合せの中からそれぞれランダムに 8 種類同時に実行して実験を行った．

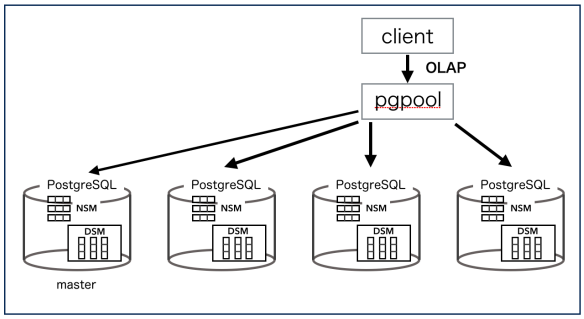

図 2　レプリケーションを用いた評価実験環境

2.2　既存のストレージ選択基準の比較

TPC-H を用いて作成されるデータのサイズはスケールファクターによって定義されており，本実験ではおよそ 10GB のデータ量となるスケールファクター 10 を使用した．また，行指向型／列指向型ストレージ選択法の比較には行指向型ストレージのみを用いた場合と，C-Store のように列指向型ストレージのみ（テーブルの結合が必要な場合は Hash Join，Merge Join のみを利用）を利用した場合，そして Fractured Mirrors のように行指向型／列指向型ストレージを併用する場合の三種類の比較を行ったところ，問合せ実行時間は図 3 の結果であった．ただし，C-Store [4, 8] や Fractured Mirrors [6] では以下のようなストレージの選択基準を設定した．

C-Store:　OLAP 問合せを処理する場合はディスク上の列指向型ストレージを使用（列指向型）

Fractured Mirrors:　多数列／少数行を利用する OLAP 問合せを処理する場合には行指向型ストレージを使用，少数列／多数行を利用する OLAP 問合せを処理する場合には列指向型ストレージを使用（併用型）

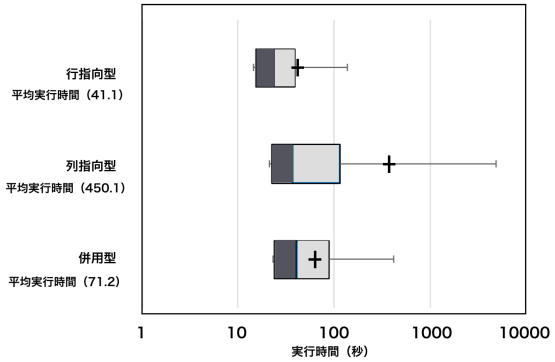

図 3　問合せ処理実行時間

図 3 が示す結果から，行指向型ストレージのみを使用した場合が最も平均実行時間が速く，OLAP 問合せを効率良く実行できると言われている列指向型ストレージを用いる場合は問合せを効率良く実行できなかった．特に，列指向型ストレージを利用する場合は，問合せ処理を利用する条件の場合は，平均実行時間が遅く，かつその分散が大きいため極端に実行時間が遅い問合せ存在することがわかった．その一方で，行指向型ストレージを用いた場合に比べ，図 4 に示すように問合せ#9，#21 は，行指向型／列指向型の併用型ストレージを用いる場合が最も高速に問合せ処理を実行できるケースがあることが判明した．

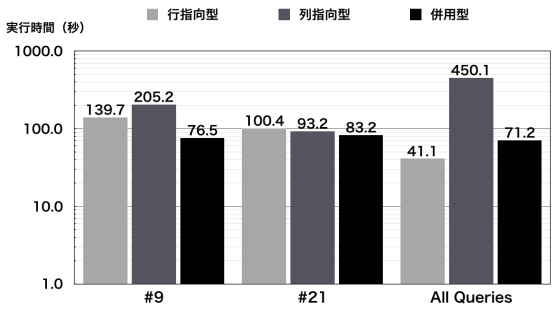

図 4　問合せ#9，#21 の平均実行時間

このことから，行指向型／列指向型ストレージの併用は，それぞれのストレージのみを利用するよりも高速に問合せを実行できる可能性がある．しかしながら，特に問合せ#2，#5，#8 においては，図 5 が示すようにストレージ併用型に比べて，行指向型ストレージで問合せ処理する場合の方が大幅に処理時間

が短い．つまりこれらの問合せに併用型ストレージを用いる際のディメリットが含まれると考えられる．

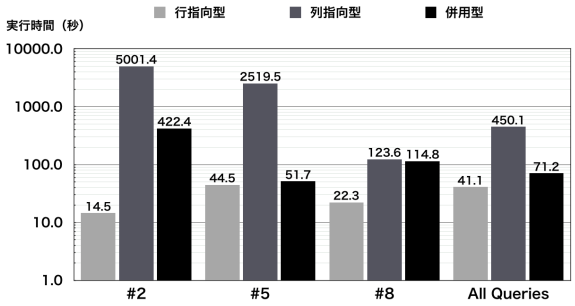

図 5　問合せ#2，#5，#8 の平均実行時間

以上のことから，問合せ#2，#5，#8 には行指向型ストレージを選択すべき条件があり，また，問合せ#9，#21 のように併用型ストレージを使用すべき条件が存在することがわかるため，これらを考慮しながらルールベースのストレージ選択条件を提案を行い，平均問合せ処理時間の短縮を図ることとする．

3. 提案手法

2 節で述べたように，列指向型ストレージの利用においては問合せ#2，#5，#8 の平均実行時間が遅いため，列指向型ストレージを利用するには不適切な問合せ処理が行われていると考えられる．よって本節では，問合せ#2，#5，#8 を高速化するために，行指向型ストレージを効果的に利用するための条件の考案と，問合せ#9，#21 を高速に実行できたストレージ併用型条件との適切な組合せ方法について述べる．また，併用型ストレージには，図 4 に示したように問合せ処理を高速に実行できた問合せも含まれているため，併用型ストレージにおけるストレージ選択条件をも考慮に入れた上で，問合せ内の FROM 句で指定される各テーブルごとにストレージ選択条件を提案する．

3.1 ストレージ選択条件の考案

まず，問合せ#2 は，相関副問合せが含まれている．この場合，一般的に副問合せで扱うデータサイズが大きくなると問合せ処理に時間が掛かる傾向にある．また，内部結合が比較的多く実行されるため，Nested Loop Join の処理の効率化にインデックスを利用するべきだと考えられる．このことは，問合せ#2 が行指向型ストレージのみを使用した場合に最も高速に問合せを処理することができたことからも明らかである．よって，相関副問合せを含む問合せが実行される場合には，以下のような条件 1 が設定できる．

> **条件 1:** 相関副問合せの結合条件にインデックスが創られている場合，行指向ストレージを使用する

一方，問合せ#5，#8 は結合処理を行う問合せである．このとき，サイズの大きなテーブルを扱っているが，テーブルの結合をする際にデータを絞り込むための条件が存在せず，全てのデータを走査するという理由から処理速度が遅い．もし，サイズの大きいテーブルの走査条件にインデックスが作成されている列が指定されていれば，インデックスを用いた Nested Loop Join が行われた上でデータの走査が可能なので，行指向型ストレージを用いたほうが問合せ#5，#8 を高速に実行できたと考えられる．

また，インデックスが利用できないパタンでは，WHERE 句で指定される条件は基本的に列指向型ストレージを用いた場合に余分な列を読み出す必要がないため扱うデータ量が少なく，効率的に問合せ処理を実行できると考えられる．さらに，コストベースでは無くルールベースによる条件ではテーブルサイズを考慮することは難しい．

そこで，テーブルサイズは考慮せず WHERE 句内の条件のインデックスを考慮したストレージ選択条件を以下のように設定した．

> **条件 2:** WHERE 句の条件のうちインデックスが作られていない数を n，WHERE 句の条件のうちインデックスが作られている数 m とし，$n < m$ を満たす場合は行指向ストレージを利用する．

これらの条件 1，2 によって問合せ#2，#5，#8 は高速に処理を実行できると予想されるが，行指向型ストレージのみを用いた場合に問合せ処理の実行が遅かった問合せ#9，#21 に関しては，予備実験で使用したストレージ併用型の選択条件が効果的に作用することが分かったために，以下のような条件を設定した．

> **条件 3:** テーブルの全列数の半分よりも問合せ内で扱われる列数のほうが多い場合，行指向ストレージを利用する

条件 3 は予備実験で使用したストレージ併用型の選択条件の一部であり，問合せ#9，#21 を行指向型ストレージのみを用いた場合よりも処理を高速化できると考えられる．

3.2 ストレージ選択条件の組合せ

3.1 節においてストレージの選択条件を提案したが，この条件をどのように組み合わせていくのかは未だ決まっていないため，それぞれの条件の優先順位を設定する必要がある．

そこで本稿では考案した条件を，問合せ#2 を唯一高速に実行できた条件である条件 1 を筆頭に，平均問合せ処理時間を最も高速に実行できた条件 2，そして残りの条件 3 のように優先順位を決めて条件の適用順を設定した．条件 4 についてはテーブルのストレージ選択を行った後にのみ判断できる条件であるため，ストレージを選択した後に適用するようにしている．これらの優先順位を適用した条件の概略を以下の図 6 ように示す．

4. 評価実験

3 節で述べた条件を図 6 のように適用し，問合せ内の各テーブルごとにストレージの選択を行った問合せ処理の速度，行指向型ストレージのみを用いた問合せ処理速度，列指向型ストレージのみを用いる C-store 型条件の問合せ処理速度の比較を

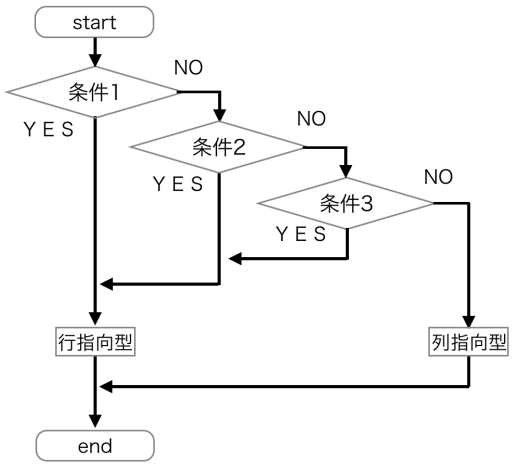

図 6　条件の優先順位

行った．使用した実験環境は，2.1 節と同じものを使用した．その実験結果を図 7 に示す．

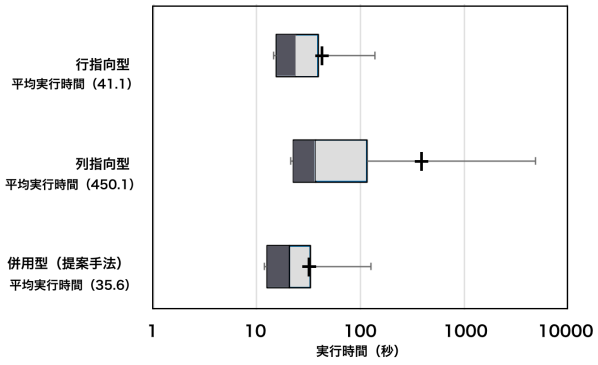

図 7　問合せ処理実行時間（Scale Factor 10）

図 7 に示す結果から，平均問合せ処理時間は 3 節で提案したストレージ選択条件を適用した場合が最も早く，効率良くストレージの選択を行うことで問合せ処理速度を高速化できたことがわかる．

また，およそ 100GB のデータ量となるスケールファクター 100 を使用した同様の実験を行ったところ，図 8 のように扱うデータ量が増えた場合においても，本提案によるストレージの選択条件が最も平均問合せ処理時間を高速化できたことが示されたため，行指向型／列指向型ストレージを単独で用いるよりも二種類のストレージを併用運用したほうが有用であることが分かる．

一方，負荷分散がストレージに与える影響を調べるために，TPC-H によって定義されている 22 種類の OLAP 問合せの中からそれぞれランダムに 24 個の問合せを同時に実行して実験を行ったところ，図 9 のような結果となった．

しかし，この結果から，負荷分散によって同時に処理する問合せ処理数が 3 倍に増えた場合においては，列指向型ストレージのみを利用した場合が最も平均問合せ処理時間の増加が少なく，行指向型ストレージに比べて問合せ処理に利用しない列を読み込む必要がなく，扱うデータ量が少なくなるため負荷分散の際に問合せ処理を効率的に行えることがわかった．具体的には，行指向型ストレージと併用型ストレージでは，スケールファクター 10 の時は約 3.5 倍，スケールファクター 100 の時は約 1.7 倍となったのに対し，列指向型ストレージではそれぞれ約 2.3 倍，1.3 倍の増分となっている．このような結果となったのは，図 10 に示す TPC-H の問合せ#7 のように，同時に処理する問合せ処理数が増えた場合においても，列指向型ストレージでは問合せ処理速度の増加が少ない問合せがあったからである．

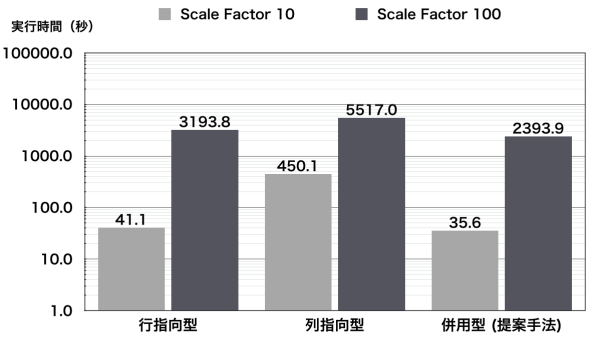

図 8　平均問合せ処理実行時間（Scale Factor 10，100）

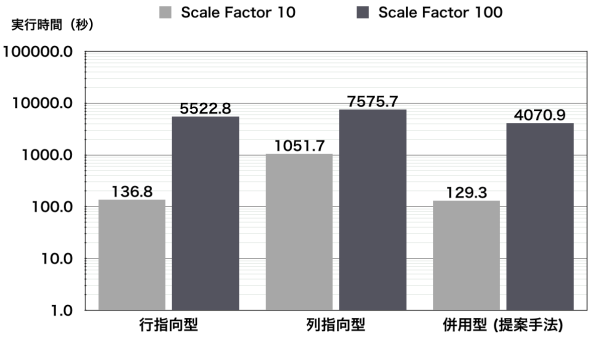

図 9　同時問合せ処理数 24 の場合の平均問合せ処理実行時間（Scale Factor 10，100）

問合せ#7 は中間結果に対して $sum()$ や一つの列の全データに対して乗算する処理が利用されているため，一度ディスクから読み込まれたデータを出来る限り計算機上のメモリにキャッシュすることで，ディスクにアクセスする回数を減らすことができると考えられる．そこで，PostgreSQL の共有バッファサイズを 2.1 節で述べた各計算機の総メモリサイズに対して 1/2 のサイズとなる 8GB に設定して実験を行った．実験に使用したデータは，およそ 10GB となるスケールファクター 10 のデータではほとんどのデータがキャッシュされてしまい，行指向型ストレージと列指向型ストレージを利用した際の扱うデータ量の差が及ぼす問合せ処理時間の性能差が表れないと判断したため，およそ 100GB のデータサイズとなるスケールファクター 100 の場合でのみ実験を行った．その実験結果を以下の図 11

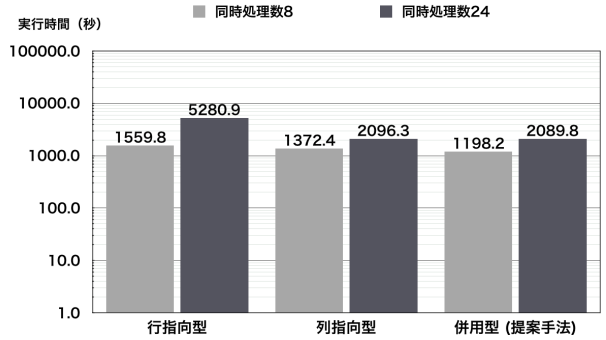

図 10　問合せ#7 の同時問合せ処理数 8，24 の場合の平均問合せ処理実行時間（Scale Factor 100）

に示す．

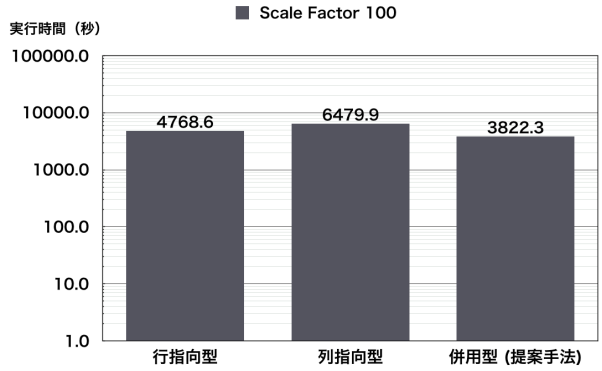

図 11　共有バッファサイズ 8GB での同時問合せ処理数 24 の場合の平均問合せ処理実行時間（Scale Factor 100）

図 9，11 の結果から，提案手法を用いてストレージを併用運用した場合に最も平均問合せ処理時間が早いことが分かる．また，列指向型ストレージが最も平均問合せ処理時間が遅く，TPC-H を用いた OLAP 処理性能が良いと判断できないため，既存の列指向型ストレージのみを利用した問合せではなく行指向型ストレージとの併用運用が必要であると考えられる．しかしながら，共有バッファサイズを増やした場合には列指向型ストレージのみを利用した場合が最も平均問合せ処理時間の改善が見られ，提案手法を用いたストレージの併用運用が最も平均問合せ処理時間を改善できていない．具体的には，列指向型ストレージでは約 15%改善できたのに対し，提案手法では共有バッファを利用したディスクアクセス回数を考慮したストレージ選択を行っていないため，併用型ストレージでは約 5%のみの改善となっている．

5. 関連研究

列指向型ストレージはテーブルのデータを列ごとに格納するデータ格納方式であり，近年多くの DBMS で採用されるようになった．列ごとに格納されたデータは，OLAP 問合せのような各列ごとに値を集計する処理に適しており，列単位でデータ型が一致するためデータの圧縮効果が高いというメリットがある．

しかし，行指向型ストレージに比べ OLTP 問合せのような行を特定し，取得することでその行に含まれる列データを更新する処理には適していないというデメリットを持ち合わせている．そのため，近年の DBMS は行指向型ストレージと列指向型ストレージを併用することで，OLTP ／ OLAP 双方の問合せを効率的に処理することを目指したものが多い．

この場合，2 節で述べたように，二種類のストレージを併用した問合せに応じてストレージの選択を適切に設定しなければ問合せ処理に時間が掛かることとなる．本節では，1 節で述べた行指向型と列指向型ストレージを併用する DBMS の例として C-store や Fractured Mirrors のアーキテクチャとストレージの選択基準を紹介する．

5.1 C-Store

C-Store は，近年の DBMS で用いられるようになった一般的な行指向型／列指向型併用ストレージを採用するアーキテクチャとなっている．ストレージの選択基準は OLTP 問合せの場合は行指向型ストレージを利用し，OLAP 問合せの場合は列指向型ストレージを利用するというものであり近年用いられる一般的なものとなっている．

このようなアーキテクチャでは，行指向型ストレージは基本的にメモリ上に構築し，列指向型ストレージはディスク上に構築するようになっている．つまり，OLTP により格納されるデータは逐次，列指向型ストレージに格納され，さらに格納されるデータは圧縮されることによって総データ量を減らすことができる．一方，OLAP においては，基本的に列指向型ストレージを用いて問合せ処理を行うが，行指向型に一時的に保存されているデータを参照する場合には，行指向型ストレージからデータを読み出すことによって，更新途中のデータ参照においても一貫性が保たれるようになっている．しかし，本稿で述べたように列指向型ストレージのみを用いた OLAP は，問合せによっては行指向型ストレージを利用した場合の方が処理を高速に実行できるため，C-store のストレージ選択基準が正しいとは言えない．C-store では列指向型ストレージのみを用いて OLAP 問合せを処理するが，本稿で述べた提案手法では 2 種類のストレージを併用利用するという点でストレージ選択基準の違いがある．C-store が採用する OLTP ／ OLAP における処理の概略を図 12 に示す．図に示す矢印は，OLTP によるデータ格納から，OLAP によるデータの読み出しまでの流れを示している．

5.2 Fractured Mirrors

Fractured Mirrors は，C-Store のように行指向型／列指向型ストレージを併用する DBMS だが，C-Store とは異なるアーキテクチャを構成している．

図 13 に示すように行指向型ストレージと列指向型ストレージはそれぞれディスク上に構築され，RAID1 のようなミラーリングを利用することで二種類のストレージを扱う DBMS を二台運用し，それぞれ OLTP 用と OLAP 用の DBMS として利用する．また，片方の DBMS に処理が集中した場合には，もう片方の DBMS を利用することで負荷分散を行うことも可能

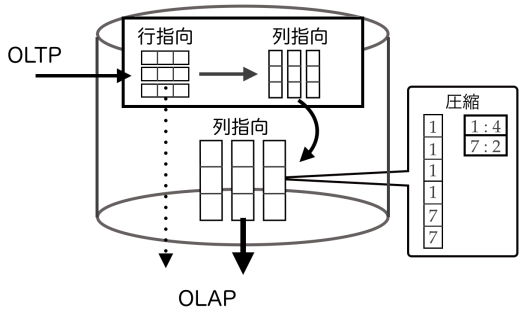

図 12　C-Store

となっている．さらに，OLAP の問合せ処理の性能を向上させるために，データを列ごとにハードディスクのシリンダごとに格納して利用することで処理性能を向上させている．

Fractured Mirrors で考慮されているストレージ選択基準は，基本的に OLAP 問合せは列指向型ストレージを使用するが，多数列／少数行が選択される問合せ処理の場合には行指向型ストレージを利用するという条件を設定している．2 節で行った実験では，Fractured Mirrors のストレージ選択基準を利用した OLAP 問合せの平均処理時間が行指向型ストレージのみを用いた場合に比べて遅くなってしまっていたため，本稿ではこの Fractured Mirrors のストレージ選択基準も利用しつつ新たにストレージ選択基準を提案することで，行指向型ストレージのみを用いた場合よりも問合せ処理時間を短縮できた．

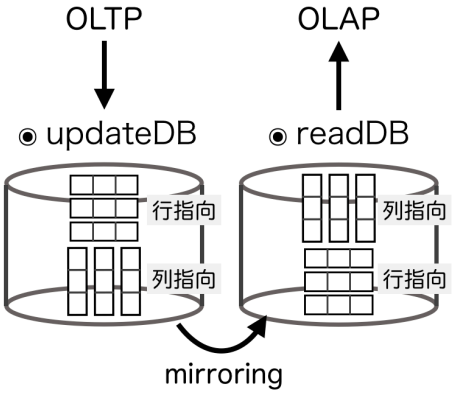

図 13　Fractured Mirrors

6. おわりに

本稿では，負荷分散環境の下で行指向型ストレージと列指向型ストレージを併用した DBMS を運用する際に，それぞれのストレージを選択するための条件を定めることで OLAP 問合せの処理最適化を図った．その結果，行指向型ストレージを利用する RDBMS に比べて，提案手法を用いて行指向型／列指向型ストレージを併用運用した際に，TPC-H の問合せ処理の平均実行時間を 5~25%短縮できた．

しかし，TPC-H の問合せ#2，#9 はおよそ 100GB のサイズとなるスケールファクター 100 のデータを使用した場合に，提案手法のストレージ選択条件が行指向型ストレージのみを使用した場合よりも，それぞれ約 2.2 倍，約 1.3 倍問合せ処理速度が遅くなってしまったことから，ストレージの選択が効率良く行うことができていない可能性が残されているため，この点も考慮したストレージの併用運用方法も考える必要がある．また，問合せ内の SELECT 句で利用されている `sum()` や `average()` などは列指向型ストレージを利用したほうが扱うデータ量も少ないため，負荷分散環境においても効率よく問合せ処理を行えると考えられる．特に，今回の図 11 の結果から，負荷分散環境での同時に実行される問合せ処理の数が多い場合に，行指向型／列指向型ストレージではそれぞれ約 13%，15%改善できたことに対して，提案手法を用いた併用型ストレージでは約 5%のみしか改善できていない．そのため，今後のストレージ併用運用には共有バッファを利用した際のディスクアクセス回数を考慮して，問合せ処理で利用されるテーブルの列数とディスクページ数によって行指向型／列指向型ストレージを選択しつつ，Sequential Scan 回数によって扱うデータ量の少ない列指向型ストレージを選択する等の，分散処理環境でも利用可能なコストベースクエリプラン生成方法をルールベースクエリプラン生成方法と組み合わせてストレージ選択方法を提案する必要がある．

さらに，本稿で評価に利用した TPC-H だけでなく，Star Schema Benchmark [5] や OLTP 問合せを含む TPC-C を用いたベンチマークテストを行い，提案手法のパフォーマンス測定を行う必要がある．

謝　辞

本研究の一部は JSPS 科研費 26280115，および文部科学省私立大学戦略的研究基盤形成支援事業 S1411030 の助成を受けたものである．

文　献

[1] M. Colgan, J. Kamp, and S. Lee. *Oracle Database In-Memory*. Oracle Corporation, October 2014. An Oracle White Paper.

[2] F. Färber, S.-K. Cha, J. Primsch, C. Bornhövd, S. Sigg, and W. Lehner. SAP HANA Database: Data Management for Modern Business Applications. *ACM SIGMOD Record*, 40(4):45–51, December 2011.

[3] H. Garcia-Moluna, J. D. Ullman, and J. Widom. *Database Systems: Pearson New International Edition: The Complete Book*. Pearson, August 2013.

[4] A. Lamb, M. Fuller, R. Varadarajan, N. Tran, B. Vandiver, L. Doshi, and C. Bear. The vertica analytic database: C-store 7 years later. In *Proceedings of the 1985 ACM SIGMOD international conference on Management of data*, pages 1790–1801. VLDB Endowment, August 2012.

[5] P. O'Neil, B. O'Neil, X. Chen, and S. Revilak. The Star Schema Benchmark and Augmented Fact Table. In *Performance Evaluation and Benchmarking*, pages 237–252. Au-

gust 2009.

[6] R. Ramamurthy, D. J. DeWitt, and Q. Su. A Case for Fractured Mirrors. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 430–441. VLDB Endowment, August 2002.

[7] R. Seiffert. Online Transactional and Analytics Processing using SPSS, DB2 z/OS, DB2 Analytics Accelerator. `http://www-304.ibm.com/connections/blogs/systemz/entry/online_transactional_and_analytics_processing`, March 2013. Retrieved April 20, 2015.

[8] M. Stonebraker, D. J. Asadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, and S. Zdonik. C-Store: A Column-Oriented DBMS. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 553–564. VLDB Endowment, August 2005.