消費電力を考慮したOLAP問合せ処理

OLAP Query Optimization for Energy Consumption

福澤 優 † 宮崎 純 † 藤澤 誠 ‡ 天野 敏之 †† 加藤 博一 †
Yu FUKUZAWA Jun MIYAZAKI Makoto FUJISAWA Toshiyuki AMANO Hirokazu KATO

1 はじめに

近年の計算機の発展により，その処理速度は大きく向上してきた．しかし，地球温暖化などの環境問題や企業におけるコスト低減の問題などから処理速度のみならず消費電力の抑制の需要が高まり，より高い処理速度を維持しながら，消費電力を削減するといった電力管理も考慮したシステムが求められている．

一方，データベース管理システム (DBMS) は現在の情報化社会において様々なシステムの根幹を担っており，銀行のオンラインシステムやネットショッピングなど多くのシステムに必要不可欠なソフトウェアである．近年では，情報化の進展により計算機上で扱うデータ量の急速な増加が見られている．それに伴い，大量のデータセットから有意義なデータを抽出することを目的としたデータマイニングやビジネスインテリジェンスなどのシステムがデータベースの分野で需要が高まっている．

またクラウドコンピューティング技術の発展により，機器導入のコストや消費電力の削減を目的として多くの企業においてクラウドコンピューティング技術の導入が進行している．しかし、一方ではクラウド化によってデータセンタに多くのデータが集約されることとなり，より一層ビッグデータに対する取り組みが注目されている．

本稿ではデータベースの処理の中でも，これらのビッグデータに対して行われる OLAP 処理に着目し，従来の DBMS において最も重要とされてきた処理速度の向上だけではなく，消費電力量削減に対する問題を考慮した OLAP 問合せの最適化の構想を述べる．

2 関連研究

従来から多くのシステムで用いられる PostgreSQL や MySQL などの RDBMS では，テーブルデータの格納方法に行指向というテーブルを行単位でページに格納していくという手法がとられてきた．しかしデータ量の爆発的な増加が伴う中で OLAP 処理に有効な DBMS の設計が考えられてきている．なかでも行指向に対して C-Store[1] に代表されるテーブルを列単位，つまり同じ属性データだけををまとめてページに格納していく列指向の DBMS が効果的とされている．他にも Sybase IQ[8]，Vertica Analytic Database[9] などの商用 DBMS においても列指向 DBMS の概念が取り入れられており，多くのシステムで利用されている．

OLAP 処理において列指向 DBMS である C-Store が従来の行指向 DBMS と比べ多くの利点が挙げられている [2] が，その評価は簡単な問合せにおいての比較にとどまっている．このことから，より多くの特性をもつ問合せに対しての検証を行う必要性があると考える．

また行指向と列指向のハイブリッドな DBMS を構成することでクエリやデータの特性による行指向 DBMS と列指向 DBMS のそれぞれの利点を利用する Fractured Mirrors[3] が提案された．Fractured Mirrors では行指向と列指向のデータをミラーリングして格納する．このハイブリッドな DBMS によって行指向または列指向のみでのクエリ実行よりも柔軟かつ有効なクエリプランの作成が可能になっている．

これらの手法は主にクエリ実行時間の高速化を目指して研究が行われてきたが，データセンタなどにおける大規模システムを中心とした背景に省電力化の需要が高まってくると，これまでのクエリ実行における処理速度の向上だけでなくクエリ実行時に消費される電力量のコストも抑えたクエリ最適化の考えが出てきた．文献 [4] においては，PostgreSQL で行われている様々なクエリプランの中から実行時間のコストパラメータによる最適化の手法に，新たに演算子ごとの消費電力のパラメータを付加させることにより消費電力と実行時間両方のコストパラメータを用いたコスト関数を用いてクエリ最適化が可能となるシステムを構築している．このシステムでは OLTP 処理中心の TPC-C ベンチマークにおいては実行時間を 2.5% 改善した上で，消費電力量も 19.0% 低減しているのに対して，OLAP 処理中心の TPC-H ベンチマークにおいては実行時間が 19.7% の低下するものの，3.3% の消費電力量削減しか

† 奈良先端科学技術大学院大学，Nara Institute of Science and Technology
‡ 筑波大学，University of Tsukuba
†† 山形大学，Yamagata University

行えていない．また消費電力量に関して，ディスク 1 台の消費電力は CPU の消費電力に比べ極端に小さいとし，ディスクの消費電力量は考慮せず CPU の消費電力量の計測のみに留まっている．しかし，現在の OLAP などの処理を行うシステムでは複数のディスクを論理的に 1 つにまとめて利用するディスクアレイを構築するのが一般的であるため，このような構成においてはディスクの消費電力量も考慮すべきである．

本稿では，この結果に対して OLAP 処理において，より効果的にクエリ実行時間を維持しながら低消費電力化を実現するために，これまで OLAP 処理で行われてきた実行時間高速化手法を用いながら消費電力量の低減を可能とするクエリ最適化の実現を目指す．

3 列指向 DBMS

この節では OLAP 処理で有効とされている列指向 DBMS の特徴を，行指向 DBMS と比較しながら見ていく．従来の DBMS では電子商取引など OLTP 処理のようなタプルに対する更新が頻繁におこる処理を中心に考えられ設計されてきた．そのためタプル全体のデータをページ内に格納する行指向の格納手法がとられている．そのため，タプル中の全データを別々のページに格納する列指向の格納手法よりも更新操作の実行が効率かつ高速に行われる．一方，経営の意思決定支援システムに見られるような蓄積されたビッグデータに対する解析的なクエリ実行を行う OLAP 処理では，ページに対して読み込みアクセスが中心であり，タプル中の特定の属性に対してのみクエリ内で利用されるといった特徴をもつ．

OLAP 処理を対象として考えた場合，テーブル中の特定の属性データのみの参照を行うようなクエリにおいて行指向 DBMS ではタプル単位でページにデータが格納されるため，処理に必要なデータ以外のデータを読み込むこととなり，ディスク I/O の回数が多くなってしまう．それに比べ列指向の DBMS においては各属性データが別々のページに格納されているため，処理に必要な属性データが存在するページにのみアクセスすれば行指向 DBMS よりディスク I/O の回数が低減されることとなる．このディスク I/O 回数の低減は，処理速度の高速化にディスク I/O が主なボトルネックになっている現在の計算機に対しては有効な手法である．

このように列指向 DBMS では読み込み中心の OLAP 処理において，いくつかの有効な手法が存在すると考えられている．

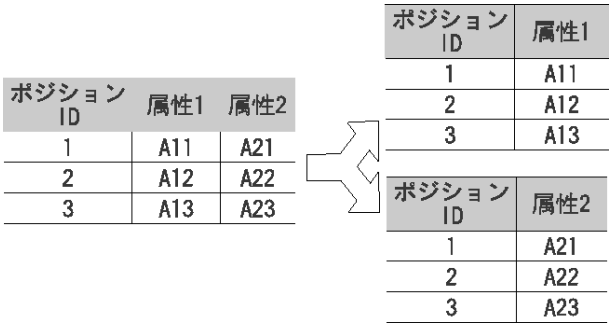

図 1: 列指向 DBMS の格納データ抽出

3.1 データの圧縮

列指向 DBMS では，あるページに格納されているデータは同じ属性のデータである為，行指向 DBMS に比べ効率的にデータを圧縮しページ内に格納することが可能となる．可逆圧縮手法としてはハフマン符号化や Lempel-Ziv 符号化などの圧縮手法が提案されている．しかし，圧縮されたデータはクエリ実行の際に解凍する必要性がある．そのため復号化におけるコストが比較的低いことと，圧縮したままでの演算が可能であるような圧縮方法が列指向 DBMS では有効だとされている．ここでは，列指向 DBMS で発生するポジション ID に対する圧縮，ランレングス圧縮とビットマップにおける圧縮手法を述べる．

3.1.1 ポジション ID に対する圧縮

列指向 DBMS はタプル中の特定データを効率良く読み込めるように提案された [5]．ここであるクエリ対してテーブルの一部を再構築する必要性が出てくると，各属性データに元テーブル上での位置情報 (ポジション ID) を記録しておかなければならない (図 1)．このポジション ID より元テーブルを再構築することが可能となる．しかし，すべてのデータがポジション ID を保持することになると，格納すべきデータ量が増加してしまうばかりか，その分のディスク I/O が発生してしまう．そこでテーブル内の属性データをポジション ID に対してシーケンシャルに格納する際は，ページ内にヘッダ情報としてページ中の最初に格納するデータのポジション ID を格納しておけば，ヘッダ情報から各データのポジション ID の情報が得ることができる (図 2)．ページに対し先頭データのポジション ID のみを格納すればよいので，これによりデータ量の増加やディスク I/O の増加が最小限に抑えることが可能となる．

3.1.2 ランレングス圧縮

ランレングス圧縮は連続したデータを，データとそのデータの連続出現回数で表現する圧縮法である (図

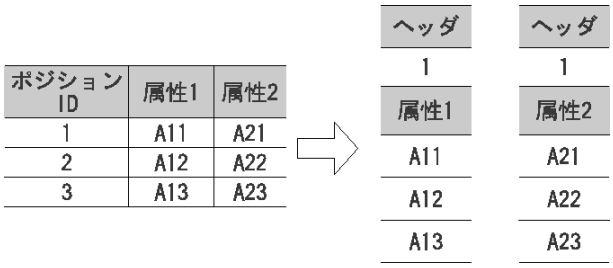

図 2: ポジション ID に対する圧縮

図 3: ランレングス圧縮

3)．ランレングス圧縮はその圧縮方法から同じデータが連続している場合に効率的に圧縮が可能だが，逆に同じデータが連続して出現しない場合のデータに対してランレングス圧縮を施すとデータ量が多くなってしまうという欠点が生じてしまうのが特徴である．

列指向 DBMS に対してランレングス圧縮を考えたとき，外部キーに対応する属性などは同じデータが複数回出現する可能性があるためランレングス圧縮に対して有効となることがある，しかし重複するデータが連続して出現するとは限らないため，重複データが多く存在しても効果的に圧縮されるとは限らない．この場合はデータを整列することにより解決することが出来る．ここで先ほどポジション ID に対する圧縮からの整列を考えると，データが元テーブルに対するデータ順と異なって位置してしまうため，ヘッダ情報に格納されている先頭データのポジション ID からでは元のデータ位置が認識出来ず，テーブルの再構築が不可能となってしまう．

そこで，新たに元テーブルでのポジションと整列後のポジションの変換を行う Join Index を作成することで問題を解決する (図 4)．格納データは整列後のデータ順を元に新たにポジション ID を設定する．それと同時に整列前と整列後の対応を Join Index に格納していく．新たにポジション ID を設定したことによってデータの順番が対応するポ本研究の一部は，科研費補助金基盤研究 (A)(課題番号:22240005) ならびに基盤研究 (C)(課題番号:23500121) の支援による．ここに記して謝意を表す．ジション ID に対して昇順に格納されることになり，ポジション ID に対する圧縮が可能となる．クエリ実行時には，まずページ内のヘッダ情報のポジション ID を元に整列後のデータ位置を取得する．その後 Join Index を参照することによって整列前のデータ位置情報を取得することができテーブルの再構築が可能となる．

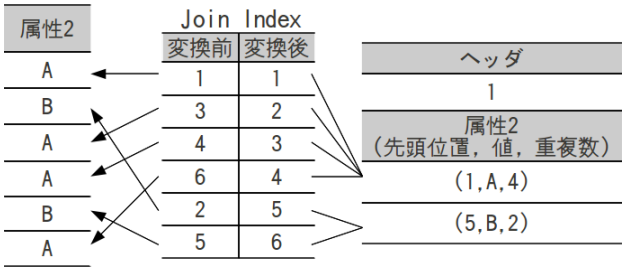

図 4: 列指向におけるランレングス圧縮

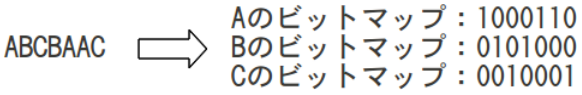

図 5: ビットマップ圧縮

3.1.3 ビットマップ圧縮

ビットマップ圧縮は各データ値を元にビットマップを生成することによってデータを圧縮する手法である (図 5)．データ集合に対してデータの種類が少ない場合には，各単一データをビットに対応させるためのビット列の生成が少ないことから効率的に圧縮が可能となるが，データの種類が多くなればなるほど生成しなければならないビット列が増えるために，逆にデータ量が増加してしまうことがある．

従来の行指向 DBMS においては Bitmap Index などに用いられるが，列指向 DBMS に対しては 3.2 に示すように，演算中の中間データとしてビットマップを用いてデータを保持することで演算の効率化を行える．これは行指向 DBMS に比べ列指向 DBMS では，ある属性から特定データのビットマップ生成とビットマップから対応データの抽出が効果的に行える為，有効な手法である．

3.2 遅延実体化

OLAP 処理のような複雑なクエリ処理においては，結合演算の前に属性データについての選択や射影演算が複数回行われることが少なくない．これは，一般的に問合せ最適化ではクエリから実行プランを生成する際に，問合せの処理のなるべく早い段階でデータの絞り込みを行い，その後の処理の対象となる中間結果のデータをできる限り削減することが処理の最適化に重要となるからである．複雑なクエリになればなるほど，そのクエリから DBMS 内部で生成させる処理木が複雑になり，処理の実行中に生じる中間結果のデータをメモリ上ないしディスク上に格納する場合が多々存在する．特にビッグデータに対するクエリ処理では，そのデータ量の多さから選択や射影演算におけるデータの絞り

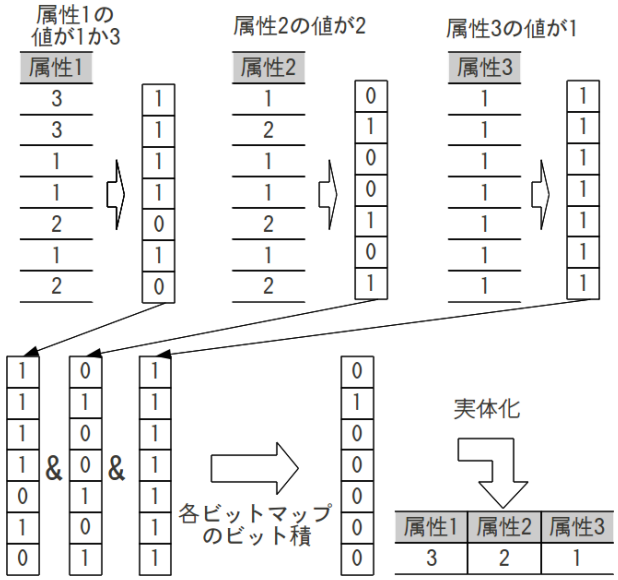

図 6: 遅延実体化

込みを行ったとしても，ディスク上に中間結果のデータを格納することがほとんどとなる．このような場合，圧縮により中間結果のデータサイズを小さくすることによって中間結果のデータをディスクに格納しなければならない状態でも，ディスク I/O の回数が少なくなるため，処理時間の高速化が見込める．

列指向 DBMS ではデータの格納方法から，クエリ実行時における列データ同士の結合演算が複数回行われることとなる．また，列データからのビットマップの生成とビットマップからのデータの復元が行指向 DBMS より有効に行える為に，クエリ実行の早い段階でのデータの絞り込みを行う演算において生成される中間結果のデータをビットマップで格納し，その後の結合演算などの演算における処理をビットマップ対して行うことで，ディスク I/O の回数を抑えながら行うことが可能となり，最後にビットマップから実体データを読み出すことによって結果を出力する．このようにクエリ実行中は圧縮されたビットマップデータに対して演算を行ない，出力前に実体化を行う遅延実体化によって，中間結果のデータに対して行われる処理で発生するディスク I/O の回数を削減することによって処理の高速化が見込める．

図 6 では，列指向 DBMS に格納されたテーブルから属性 1 は 1 または 3 の値を，属性 2 は 2 の値を，属性 3 は 1 の値をとるレコードの抽出をビットマップを利用した遅延実体化手法を用いて行っている．列指向データより各属性データにアクセスし，各選択演算の条件を満たすデータの抽出を行い，その抽出した中間結果をビットマップを利用して格納している．これによって各選択演算後の結合演算においては実体データを持たず条件を満たすレコードの抽出を行っている．まず各選択演算の条件を全て満たすようなレコードの絞り込みを，中間結果として生成されたビットマップデータ同士のビット積をとり，抽出すべきレコードの情報をビットマップデータとして得る．最後にビット積から得られたビットマップの情報を基に，抽出すべき各属性データが格納されているページにアクセスし，データの実体化を行うことで，結合された抽出データを得ることが可能となる．

このような列指向 DBMS における遅延実体化手法は Invisible Join[6] などで用いられており，結合演算が多い列指向 DBMS に対する処理の高速化を実現している．

4 提案手法

ここではこれまで述べた OLAP の高速化手法を用いながら，消費電力を考慮した OLAP 処理の問合せ手法について記述する．

4.1 列指向と行指向のハイブリッドシステム

OLAP 処理では一般的に列指向 DBMS が有効とされているが，クエリの特性によっては行指向 DBMS の方が有効なケースも存在すると考えられる．具体的には，あるテーブル上の特定の属性データに対して選択演算を行い，そのデータに対する同一タプル上にある異なる属性データの抽出を考える．このとき列指向 DBMS では，選択演算の対象とする列を参照する．その後，そのデータのポジション ID より対応するデータを抽出するために抽出対象属性データのページを参照しなければならない．一方，行指向 DBMS の場合は，選択演算の対象であるデータと同じページ上に，同じタプルの異なる属性データが格納されているので，選択演算の条件を満たすタプルの任意の属性データを，他のページを参照することなく抽出が可能となる．列指向 DBMS では列同士の結合演算をおこなうための処理が発生してしまうため，行指向 DBMS 上での処理の方が処理速度において有効になると考えられる．

文献 [7] においては列指向 DBMS である C-Store と行指向 DBMS である MySQL のそれぞれで処理速度と消費電力量の両面において特性の検証を行っている．これより，列指向 DBMS と行指向 DBMS のそれぞれにおいて有効となるクエリが存在するため，問合せ最適化において処理速度と消費電力量の二つのコストにおける最適化を考えたとき Fractured Mirrors のように行指向と列指向の両方のデータを格納することで，クエリ実行時に各インデックスや先に述べた遅延実体化など各格納方法に有効な演算処理を利用することができ

る．よって，一方だけや両方の格納データを参照しながら処理を行うといった柔軟なクエリプランが可能となるため，より有効な最適化が行えると考える．

さらに，行指向と列指向でデータをミラーリングすることによってクエリに対する柔軟性だけでなく，データに対する信頼性を向上させることができる．しかし一方でミラーリングを行うことにより，データの整合性が損なわれることが考えられる．これはデータに対する更新処理が行われたとき，行指向と列指向の両方のデータに対して更新処理を行う必要性が生じるためである．このことから，行指向のデータと列指向のデータが常に一致するとは限らず，クエリが同時実行される場合には，両指向のデータに対する整合性を保証した同時実行制御を行う必要がある．

本稿では処理の対象とするのは OLAP 処理であるため，データに対する更新処理はまとまったデータの挿入処理が定期的に発生するのみで，処理の中心はデータ読み込みだけの処理であると考えられる．このことから，同時実行制御におけるデータ整合性の保証は厳密には行わなくてもよく，データの挿入処理のみ読み込み中心の処理から独立して行えばよいと考える．

4.2 消費電力に対するコスト処理

消費電力量を考慮したクエリ最適化を考える際には，クエリ最適化の処理において消費電力量のコストの設定を考えなければならない．従来の DBMS では各演算子について，演算子固有の処理時間に準ずるコストパラメータが設定されている．クエリ最適化処理ではそれらのコストパラメータやテーブル情報から動的計画法などのアルゴリズムを用いて最適なクエリプランを抽出する．

本稿では，処理時間と消費電力量の両方のコストに対して考えるので，[4] と同様にこれまでの処理時間に関するコストパラメータに消費電力量の関するコストパラメータを追加し，コスト関数において二つのコストを統合することによって処理時間と消費電力量の 2 つに対して最適化が行えると考える．

コストパラメータの設定では，経験則から値を設定することが妥当であると言える．これは各演算子に対する消費電力量の関係が不透明であることが大きな要因として挙げられる．そのため，複数のクエリパターンやデータパターンより，各演算子に設定するパラメータを導出する必要性があると考える．

また消費電力量に対するパラメータを設定することによって，ユーザに処理速度を重視するようなクエリプランや，消費電力量を重視するようなクエリプランの選択が可能となる．これより深夜におけるバッチ処理など，比較的実行時間が遅くても良いという状況に対しては消費電力量をなるべく抑えたクエリプランで実行させるなど，時間帯や他の状況に合わせてクエリ最適化においてどちらかコストを重要視した最適化が行えるという利便性が生まれると思われる．

4.3 クエリ実行プラン

ここでは，実際に行指向と列指向のハイブリッドな格納データにおいて，いくつかのクエリ特性を考え，それらのクエリに対してどのようなクエリプランで実行すれば処理速度と消費電力量に有効であればよいのかを考えていく．

文献 [7] では処理速度と消費電力量の両面から，どのようなクエリ特性が行指向 DBMS と列指向 DBMS において有効化を考察している．これによりクエリ内において完全一致の問合せを含くみ，かつタプル内の複数の属性データを利用するものは，行指向 DBMS が有利であることが検証されている．これは完全一致におけるページ問合せの際に条件を満たすタプル内の全ての属性データを取得出来るためである．一方，列指向 DBMS では行指向 DBMS よりも結合の演算のコストが低いために，クエリ内において結合演算が複数含まれている場合に対しては，列指向 DBMS を利用した方が有効であることが検証されている．結合演算においては演算前に中間結果データを格納する処理が発生することが多いので，ディスク I/O が発生しやすい演算であると言える．このため演算に必要データだけにアクセスすることが可能であり，さらにビットマップなどの圧縮データなどによるディスク I/O の低減によって，結合演算では列指向 DBMS が有効であると考えられる．

消費電力量の観点では CPU と HDD の消費電力量を計測しており，単位時間の消費電力量の電力では平均的に列指向 DBMS の方が低く，行指向 DBMS の 77.8W に対し 73.7W という結果になっている．これよりクエリやクエリ中の演算の実行時間が比較的近い値をとるようであるならば，消費電力量を低減させるためには列指向のデータにおいてクエリや演算を実行する方が最適であると考えられる．

さらに行指向と列指向の両方の格納データを持っているため，[3] のように片方だけのデータを用いてクエリ処理を行うのではなく，両方の格納データを利用したクエリプランも考えることが可能である．実際に処理時間に関しては片方だけの格納データを利用するクエリプランよりも，両方の格納データを利用したクエリプランの方が効果的に処理が行えるクエリの例も示されている．このことからも，消費電力量の観点も考

慮しながら行指向と列指向の格納データを組み合わせたクエリの実行がどれくらい有効か，検証が必要となると考えられる．

5 まとめ

本稿では，データの爆発的な増加と消費電力量削減の需要を背景とし，ビックデータに対する OLAP 処理に対して，これまでの DBMS で重要視されてきた処理速度の最適化に消費電力量の最適化の概念を追加し，低消費電力化を目指した OLAP 処理に特化した DBMS の具体的な構想を述べた．これまで，OLAP 処理における高速化を目指して提案された行指向と列指向のミラーリングを行う Fractured Mirrors の構成を利用し，消費電力量のコストパラメータを追加することによって，処理速度だけではなく消費電力量を考慮したクエリ最適化を行う．また，ミラーリングを行うことにより行指向と列指向それぞれの DBMS の利点を利用しながら，一方だけの構成だけでは行えない有効なクエリプランの生成ができ，より最適なクエリプランの抽出が可能であると考えられる．

今後の課題としては，本稿での提案したシステムの開発を行い，行指向や列指向の一方だけの DBMS と比べ，Fractured Mirrors のような行指向と列指向のハイブリッドな DBMS における実行時間と消費電力量の両方のコストにおいて，どれくらいの優位性があるのかの検証を行う予定である．また，SSD や GPU などのハードウェア構成によって更なる処理実行時間を抑えながらの低消費電力化アプローチの可能であると考えているため，それらのアプローチについても考えていきたいと思っている．

謝辞

本研究の一部は，科研費補助金基盤研究 (A)(課題番号:22240005) ならびに基盤研究 (C)(課題番号:23500121) の支援による．ここに記して謝意を表す．

参考文献

[1] Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xue-dong Chen, Mitch Cherniack, Miguel Ferreira, EdmondLau, Amerson Lin, Samuel Madden, Elizabeth J. O ’Neil, Patrick E. O ’Neil, Alex Rasin, Nga Tran, and Stanley B.Zdonik. C-Store: A Column-oriented DBMS.. In *Proceedings of VLDB 2005*, pp. 553–564, 2005.

[2] Daniel J. Abadi, Samuel R. Madden and Nabil Hachem. Column-Stores vs. Row-Stores: How Different Are They Really?. In *Proceedings of ACM SIGMOD 2008*, pp. 967–980, 2008.

[3] Ravishankar Ramamurthy, David J. DeWitt, and Qi Su. A Case for Fractured Mirrors. *Proceedings of VLDB 2002*, pp. 430–441, 2002.

[4] Zichen Xu, Yu-Cheng Tu, and Xiaorui Wang. Exploring power-performance tradeoffs in database systems. In *Proceedings of ICDE 2010*, pp. 485–496, 2010.

[5] George P. Copeland and Setrag N. Khoshafian. A DECOMPOSITION STORAGE MODEL. In *Proceedings of ACM SIGMOD 1985*, pp. 268–279, 1985.

[6] D. J. Abadi. Query execution in column-oriented database systems. *MIT PhD Dissertation*, 2008. PhD Thesis.

[7] 福澤優, 宮崎純, 藤澤誠, 天野敏之, 加藤博一. カラムストアデータベースの処理性能と電力の関係について. In *Proceedings of DEIM 2011*.

[8] Sybase IQ White papers. ⟨http://www.sybase.com⟩, (参照 2010-07-29).

[9] Vertica White papers. ⟨http://www.vertica.com⟩, (参照 2010-07-29).