D-037

列指向型データ格納可能なRDBMSの問合せ内述語に基づく問合せ処理最適化

A Predicate-based Query Optimization on A RDBMS with Columnar-store Function

塩井隆円 † Takamitsu Shioi
波多野賢治 † Kenji Hatano

1. はじめに

近年，大量に蓄積したビジネスデータを多次元的に解析することで企業の意思決定支援に利用する Online Analytical Processing（以下，OLAP）が行われている．しかし，従来の OLAP では業務で運用している Relational Database（以下，RDB）に蓄積したデータを効率的に分析するために，集計処理に特化した多次元データベースへデータを同期する処理が必要となり，リアルタイムなデータ分析や更新ができないという問題がある．それらの問題を解決するために，近年の OLAP は Hybrid OLAP（以下，HOLAP）と呼ばれる処理へと変化している [1].

Fractured Mirrors[2] や C-store[3] では，業務上で運用している RDB を分析に特化した多次元データベースの構築をすることなく低コストでデータ分析を実行するために，RDB と多次元データベースに採用されている表形式で管理するデータを一行ごとに格納する N-ary Storage Model（以下，NSM）と，一列ごとに格納する Decomposition Storage Model（以下，DSM）と呼ばれる二種類のデータ格納方式を一つの DBMS で採用している．しかし，二種類のデータ格納方式を運用する際に問合せ処理の中でストレージを選択する基準が曖昧であるためそれぞれのメリットが OLAP system に活かされていないことが現状となっている．

そこで本研究では，二種類のデータ格納方式をハイブリッドに運用した際に問合せ処理を低コストに実行することができるストレージの選択をする処理分割基準を提案する．

2.NSM/DSM を運用した問合せ処理最適化

NSM/DSM 両方のストレージを用いて HOLAP を高速に実行するためには，双方のデータ格納方式を用いるメリットを考慮する必要がある．今回，結合処理の際に取得するデータの検索において，NSM に基づくストレージは索引の構築に，また DSM に基づくストレージは属性単位のフィルタリングに特徴があると考え，問合せ処理の選択条件を検討することにした．

RDB は主キーを用いてテーブルから一行のデータを取り出していることから，基本，NSM に基づいたストレージを用いていると考えることができる．主キーには必ず索引が作成され，高速検索の実現に一役買っている．一般的に索引を付与する効果が高いのは，

- テーブル内のデータ量が多く少数列を検索する場合
- 値の種類が多い列に索引を付与する場合
- 結合述語として頻繁に使用される列に索引を付与する場合

と言われている．つまりこの条件を満たさない属性には索引を付与しないが，その場合表全体を走査しなければならないという問題が起こる．一方，DSM に基づいたストレージでは各属性単位の検索が可能なため，行ごとに処理を行う NSM に比べ高速に検索が実現できる．

3. 提案手法

2 節に挙げた事柄を考慮し，各問合せごとに NSM/DSM どちらのストレージを用いるかどうかを決定する条件を以下のように設定した．

条件 1: 問合せが相関副問合せであり，かつ WHERE 句に存在する結合述語で指定されている属性に索引が張られている場合

条件 2: テーブルサイズが総テーブルサイズと比較して 0.1%以下のテーブルである場合

条件 3: WHERE 句内の属性のうち索引が作成されていない属性の数 $n_{w/o}$ と，索引が作成されている属性の数 $n_{w/}$ の間に $n_{w/o}$<$n_{w/}$ の関係が成り立つ場合

条件 4: テーブルサイズが総テーブルサイズと比較して 50%以上のテーブルである場合

この条件に対する処理の流れを図示すると図 1 のようになる．

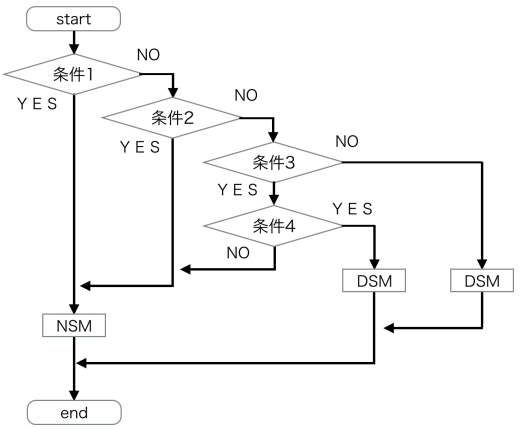

図 1: NSM/DSM 選択のための条件適用

条件 1 は，相関副問合せでは副問合せの結果を逐次主問合せの結果に結合していかねばならないため，効率的な問合せ処理のためには索引を使用したほうが処理の効率化が図れるためこの条件を設定した．

†同志社大学，Doshisha University

条件 2 は，DSM に格納されるデータは圧縮されるため，テーブルサイズが小さい場合，解凍と検索した値の行を特定しテーブルを復元する処理コストが掛かることを考慮し，NSM を選択するためにこの条件を設定した．

条件 3 は，索引を使用できない場合に DSM に基づくストレージを使用するほうが問合せ処理効率がよいため，経験則的にこの条件を設定した．

条件 4 は，テーブルサイズが大きい場合，圧縮されたデータを列単位で処理するメリットを効果的に活用できるためこの条件を設定した．

4. 評価実験

3 節で提案した，各問合せに対するストレージ選択が問合せ処理の効率化にどれほど貢献しているかを評価するために，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 [‡] を使用し評価実験を行った．TPC-H のデータは，データ作成ツール DBGen を用い，そのデータを NSM/DSM 双方のストレージをサポートする PostgreSQL 9.3.4 のデータベースそれぞれに格納している [§]．なお，DSM に基づくストレージに格納する際は，図 2 のようにテーブルごとにデータを圧縮して格納している．

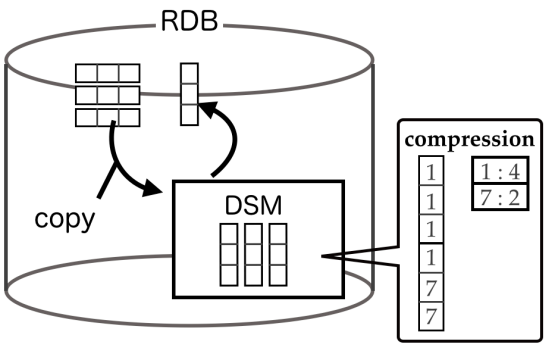

図 2: NSM/DSM を利用する RDB の構成

評価実験の結果を図 3 に示す．図中の NSM は NSM に基づくストレージのみを使用した際の，DSM は DSM に基づくストレージのみを使用した際の，条件 (1) は提案手法で条件 1 のみを適用した際の， 条件 (1,2,3,4) は条件 1, 2, 3, 4 を適用した際の平均問合せ実行時間となっている．

図 3 の結果から分かるように，本提案手法によって NSM/DSM に基づいたストレージを用いる手法が，最も問合せ処理を高速に実行できたことがわかる．また，条件 3 も追加することで列ごとにデータのフィルタリング処理が可能となり，条件 2, 4 によって DSM のデータ圧縮効果を考慮したことで，更なる問合せ処理時間の短縮にもつながった．最も問合せ処理に時間が掛かったのは DSM に基づくストレージだけを使用した場合

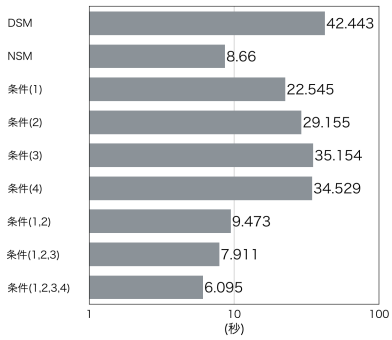

図 3: 各データ格納方式別平均問合せ処理実行時間

であり，相関副問合せに対して処理時間が増大してしまったことが推察される．

結局のところ，22 個の問合せ全体では，NSM/DSM 双方に基づいたストレージを用いる手法が，NSM に基づいたストレージに比べ約 30%，DSM に基づいたストレージに比べ約 85% 処理時間を短縮することができ，提案手法の有効性を示すことができた．

5. おわりに

本研究では，OLAP 型の問合せ処理に対して問合わせ内の結合述語の条件を基に，二種類のデータ格納方式を効率的に選択する手法の提案を行った．

本提案手法を用いて，TPC-H の問合せ処理の実行時間の短縮が可能となり，本提案手法の有用性が確認できた．

謝辞

本研究の一部は JSPS 科研費 26280115，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．

参考文献

[1] Maria Colgan, Jesse Kamp, and Sue Lee. *Oracle Database In-Memory*. Oracle Corporation, October 2014. An Oracle White Paper.

[2] Ravishankar Ramamurthy, David J. DeWitt, and Qi Su. A Case for Fractured Mirrors. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pp. 430–441. VLDB Endowment, August 2002.

[3] Mike Stonebraker, Daniel J. Asadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Sam Madden, Elizabeth O'Neil, Pat O'Neil, Alex Rasin, Nga Tran, and Stan Zdonik. C-Store: A Column-Oriented DBMS. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 553–564. VLDB Endowment, August 2005.

[‡] http://www.tpc.org/tpch/spec/tpch2.8.0.pdf

[§] DBGen により作成されたデータのサイズはスケールファクターで定義されている．本実験では SF = 1 を使用した．