2C-01

カラムナデータをサポートするための PostgreSQL 拡張

田原 司睦　中村 実　宇治橋 善史　橋田 拓志　原田 リリアン　河場 基行
株式会社富士通研究所

1 はじめに

リレーショナルデータベースにおいて、オンライントランザクション処理(OLTP)には一件ごとにデータをまとめて格納する行形式が適している。オンライン分析処理(OLAP)には性質毎にデータをまとめて格納する列形式が適していることが多い。

行形式と列形式の格納法は、相反する性質を持っているが、近年、両方の格納形式が利用できるデータベース製品が発表されている[1]。更に、一つの表に対し、行形式と列形式の両方で格納するものもある[2] [3]。両方の形式で記録を行う場合、OLTP 業務の傍らで、最新状態のデータで OLAP を行うことが可能(OLXP)となる。

我々は、オープンソースソフトウェア(OSS)のリレーショナルデータベースマネージメントシステム(RDBMS)である PostgreSQL に列形式ストアを実装した。同様のシステムとしては[4]と[5]があげられるが、前者は OLAP、後者はタイムシリーズの分析に特化しており、OLXP を実現したものではない。我々は、PostgreSQL の行形式の表にインデックス(Column Store Index, CSI)の形で列形式ストアを追加することで、OLTP に大きな遅延を発生させることなく OLXP を可能にした。

また、クエリの並列処理機構、データ共有機構も併せて実装することで、OLAP の高速化を実現している。

2 PostgreSQL

PostgreSQL は、行形式でデータを記録する OLTP 向けオブジェクト RDBMS である。マルチバージョンコンカレンシーコントロール(MVCC)は追記記録方式で実現されており、挿入・削除・更新といった OLTP 処理が効率的に動作するように設計されている。ただし、データの無限増加を防ぐため、走行中のトランザクションと以後の全トランザクションから見ることのできない削除済みデータを実際に削除するバキューム処理が非同期に実行される。

また、問い合わせの高速化のためにインデックスが用意されており、一つの表に対し複数のインデックスを設定可能である。インデックスの利用の有無や、複数の問い合わせ処理の方式(プラン)が考えられる場合、それぞれのプランのコストが評価され、一番低コストのプランが実行される。

3 Column Store Index

我々は、OLTP を行っている環境下でシームレスに列形式ストアが利用できることを目指し、PostgreSQL のプラグインモジュールとして CSI を開発した。インデックスの形を取ることで、列形式のデータを行形式の表に自然な形で付加し、両者の同期を PostgreSQL の枠組み内で遅延なしに実行することを可能にした。

3.1 記録方式

CSI では、列ごとにまとめてデータを格納し、256k 行毎の“エクステント”に区切って管理している。このような列形式では、一件のデータが複数個所に分散して格納されることになるため、挿入・削除・更新処理といった OLTP 処理の性能が低下するという問題がある。これを防止するため、図 1の様に、まず元の表である行形式の格納データの行番号(TID)のみを列形式側の差分記録(WOS)として保存する。WOS がエクステント分たまると、一括して列形式データ(ROS)を作成し、WOS を削除する。データを圧縮して記録する場合は、この段階で圧縮する。以上は、ストレージに保存されるデータである。削除や更新をさらに高速化するため、WOS の TID は共有メモリ上にも赤黒木で保持されている。

データの削除方法は、WOS と ROS で異なる。

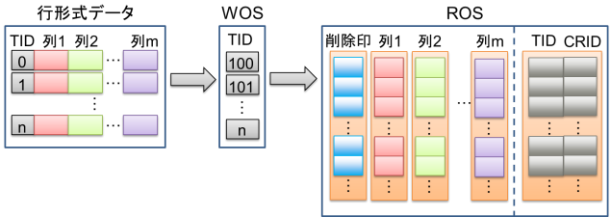

図 1　行形式データから CSI へのデータのコピー。ROS には、PostgreSQL の TID から ROS の行番号(CRID)への変換表も含まれる。

PostgreSQL extension for handling columnar data
Tsuguchika Tabaru, Minoru Nakamura, Yoshifumi Ujibashi, Takushi Hashida, Lilian Harada, Motoyuki Kawaba
Fujitsu Laboratories Ltd.

WOS 上のデータには直接削除印がつけられ、後にバキューム処理される。ROS 上のデータの削除は三段階で行われる。削除直後は、TID で削除情報(ホワイトアウト WOS)を保持する。ホワイトアウト WOS にデータがたまると、一部を ROS の削除印(図 1)に変換し、それまでの削除情報にマージする。エクステント中に削除済みデータが増えると、削除印がついていないデータのみを集めた新たなエクステントを生成し、古いエクステントを破棄する。破棄されたエクステントで使用していた領域は、新たにエクステントを登録する際に利用される。

問い合わせでは WOS のデータも参照するため、クエリ実行開始時に、WOS をクエリ毎の一時データとして列形式に変換し、列形式データ処理エンジンで処理する。ホワイトアウト WOS も同様に、一時削除リストに変換する。これらは、クエリ終了まで共有メモリ上に保持される。

CSI でも MVCC を実現する必要がある。PostgreSQL の行形式ストアと同期した MVCC を実現するための詳細は、[6]を参照されたい。

3.2 問い合わせ処理の実行方式

CSI を利用する処理は、“カスタムノード”[7]を利用して PostgreSQL のプランに埋め込まれる。SQL 文からプランを生成する段階で、PostgreSQL のプランノードの一部をカスタムプランノードに置き換えたプラン(図 2)も生成し、CSI を利用する方が高速と判断された場合のみ利用される。このため、ユーザーは特に意識をせずに CSI を利用できる。

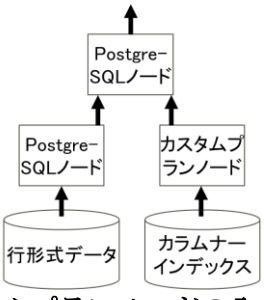

図 2 カスタムプランノードの入ったプランツリーの例。カスタムプランノード内が並列化される。

カスタムプランでは、スキャン、フィルター、並べ替え、集計などを並列処理で実行する。並列処理は、PostgreSQL のダイナミックバックグラウンドワーカー(DBW)で実現している。複数プロセス間でデータ共有を行うための PostgreSQL 向けの新たな仕組みも開発している[8]。

一つの問い合わせ処理は複数のタスクからなるジョブとして処理される。タスクはコンテキストとして扱う。DBW はタスクを一つ選択し、タスクの担当するエクステントの一つを処理する。エクステントの処理が終了すると、再度タスクの選択から行う。この仕組みにより、タスク数と処理する DBW 数を独立させ、複数の問い合わせ間で動的に CPU を割り振っている。

4 まとめ

OLTP 向け OSS RDBMS である PostgreSQL で OLAP を高速化するために、列形式でデータを格納する CSI の実装を進めている。ユーザーは、通常の表に CSI を作成することで、自動的に列形式の処理が利用可能となる。CSI では、問い合わせの並列処理も実装されており、複数の問い合わせ処理に対しても、動的な CPU の割り当てを可能としている。

今後、更なる評価と高速化を進めていく予定である。

参考文献

[1] DB2 with BLU Acceleration: So Much More than Just a Column Store, V. Raman, et al., Proc. VLDB Endow., 2013, Vol. 6, No. 11
[2] Enhancements to SQL server column stores, Per-Ake Larson, et al., Proc. 2013 ACM SIGMOD International Conference on Management of Data, pages 1159-1168
[3] Oracle Database In-Memory, Oracle, http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html
[4] PostgreSQL Columnar Store for Analytic Workloads, citusdata, http://www.citusdata.com/blog/76-postgresql-columnar-store-for-analytics
[5] In-Memory Columnar Store extension for PostgreSQL, Konstantin Knizhnik, http://www.pgcon.org/2014/schedule/events/643.en.html
[6] PostgreSQL ベースのカラムナ機構へのトランザクション実現検討, 橋田 拓志 他, 第 77 回日本情報処理学会全国大会
[7] 海外 浩平氏のカスタムプランのパッチ, http://www.postgresql.org/message-id/9A28C8860F777E439AA12E8AEA7694F8F8F6BA@BPXM15GP.gisp.nec.co.jp
[8] PostgreSQL ベースの並列処理向けの共有メモリ機構の設計, 宇治橘 善史 他, 第 77 回日本情報処理学会全国大会