2C-02

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　田原 司睦　中村 実　宇治橋 善史　河場 基行　原田 リリアン

（株）富士通研究所

1. はじめに

近年、データ解析への関心の高まりから、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）上で、OLAP 処理も同時に扱えるOLXP[1][2]システムが注目されている。OLXP システムは、OLTP 業務の傍らで、最新状態のデータで OLAP 処理を行うことができることを特徴としている。

我々は、OSS の RDBMS である PostgreSQL ベースに OLXP システムを実現した。OLTP 性能と OLAP 性能の双方を引き出すため、データを OLTP に適した行形式と、OLAP に適した列形式の双方で 2 重に保持するアプローチを採用した。PostgreSQL に列形式のインデックス（カラムストアインデックス、CSI）を追加することで、一つの表を二つの方式で格納することを実現した[3]。

我々の OLXP システムの実現に際した課題の一つにトランザクションの実現がある。性能面ではトランザクション処理に伴うデータ更新速度の低下の抑制、機能面ではインデックスにアクセスする際の MVCC(Multi-Version Concurrent Control)サポートが課題となる。本稿ではこれらの問題への対応案について述べる。

2. カラムストアインデックスの構成

本研究で開発する CSI の概要を図 1 に示す。行形式データと列形式データを 2 重に保持している。CSI では、カラム毎に別々のファイルに書き込むので、1 レコードの挿入・削除ごとにカラムに格納していると OLTP 性能を悪化させる。このため一定量のレコードを行形式で保存する Write-Optimized-Store（WOS）と呼ばれるライトバッファ機構[4]を採用した。WOS へのデータの挿入は、PostgreSQL のカスタムインデックスの枠組みを利用しており、オリジナルのテーブルへの行挿入の後即座に行われる。WOS への挿入コストを最小限に抑えるために、WOS にはオリジナルテーブル上の行要素のユニークな ID(Tuple-ID, TID)のみを保存する。

Implementing a transaction mechanism on PostgreSQL with columnar tables

Takushi Hashida, Minoru Nakamura, Tsuguchika Tabaru, Yoshifumi Ujibashi, Motoyuki Kawaba, Lilian Harada

Fujitsu Laboratories LTD.

実際に列形式 1 でデータを保持している機構は Read-Optimized-Store（ROS）と呼ばれ、列形式のデータが必要な場合はこれを用いて処理を行う。WOS から ROS への変換は、定期的かつユーザー側の実施するクエリとは非同期的に、変換デーモンによって行われる。

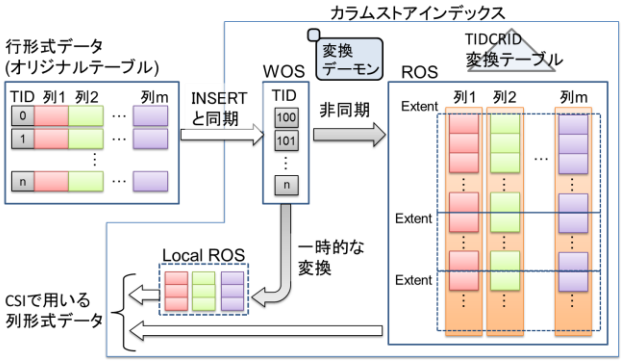

図 1　全体構成図

変換デーモンは、WOS に記録された TID を元にオリジナルテーブルから実際のデータを参照し、列ごとに分解して ROS へと格納する。

ROS 上のデータ管理の効率化を目的として、エクステントを導入した。変換作業やファイルへの書き出し、CSI を用いたクエリの際のクエリ実行エンジンへのデータの受け渡しはこのエクステント単位で行われる。エクステントは原則的には上書きによる更新は行わず、データの追加や削除された行の回収には、別のエクステントを生成し新たにデータを格納しなおす。これにより、後述する、ROS の MVCC や、異常時のリカバリーを可能にした。ROS 上のデータは、格納されたエクステントの番号、およびエクステント内での論理的な位置から一意に定まる CRID を持っており、オリジナルテーブル上で同じ行のデータは必ず同じ CRID の位置に格納されることで同じ行のデータを特定することが可能である。TID と CRID の対応は、TID-CRID 変換テーブルに記録されており、ROS 上からの行の削除時に参照される。行の削除も変換デーモンによって非同期的に行われるため、データの挿入時と同様、オリジナルテーブル上からのデータの削除時にはホワイトアウト情報として専用の WOS へと格納される。

変換デーモンの動作は非同期的であるため、

ある瞬間にすべてのデータが ROS 上に存在するわけでは無く、WOS に残っているデータが存在する。CSI を用いた検索処理の実施時にはこれらを一時的にメモリ上で列形式へと変換する。これは、クエリ単位のスコープを持った ROS と見なすことが可能で、これを local ROS と呼ぶ。

local ROS の作成と ROS への変換は排他的に実施されるが、それらを用いたクエリの実行は、同時に行われる。

3. カラムストアインデックスの MVCC 動作

CSI を用いた場合でも、トランザクションの一貫性は保たれる必要があり、次の 2 点に留意する必要がある。①クエリの実行中にも ROS 変換は行われるため、local ROS と ROS 上のエクステントから重複してデータを読むなどの不整合が起こらないようにする必要がある。②検索クエリを含むトランザクションの実行中に挿入されたデータの扱いで、これについては、PostgreSQL 本来の可視性チェックに従わなければならない。

PostgreSQL は、内部的に MVCC を用いて、トランザクションごとのデータの一貫性を管理している。各行要素には、自身を生成、および削除した Transaction-ID(XID) として、Xmin・Xmax が記録されており、検索クエリは自身の XID とこの Xmin・Xmax を比較することで、その行を読んで良いかの可視性判定を行っている。ROS 上に移したデータの全てにこれらを備え、同じ方法で可視性をチェックするのはオーバーヘッドが大きいため、エクステント単位で可視性をチェックできる手法をとった。

WOS 上のデータは、WOS への挿入や削除がオリジナルテーブルへの挿入・削除後同じトランザクション内で即座に行われるため、オリジナルテーブルと全く同じ Xmin・Xmax を持つ。この Xmin・Xmax を用いて、現存するすべてのクエリから「可視である」と判定できる行に限り WOS から ROS に移してよいと判断している。

ROS への変換は、エクステント単位で行われるが、このエクステントに ROS 変換を行った際の XID を記録する。概念的には Xmin と同等のものであるが、ここでは区別のために Xgen と名付けた。Xmax に相当するものは、Xdel と名付けた。エクステントは上書きを行わず、コピーに対して追記や削除行の回収を行うが、その際にコピー元となるエクステントに操作を実施した XID が記録される。

カラムストアインデックスを用いた検索クエリは、クエリの開始時に最後に成功した ROS 変換の XID を取得する。この XID と、エクステントに記録された Xgen、Xdel を比較し「 Xgen ≦ 取得した XID ＜ Xdel」が成立するエクステントのみを読むことができる。これにより①の点を解決した（図 2）。

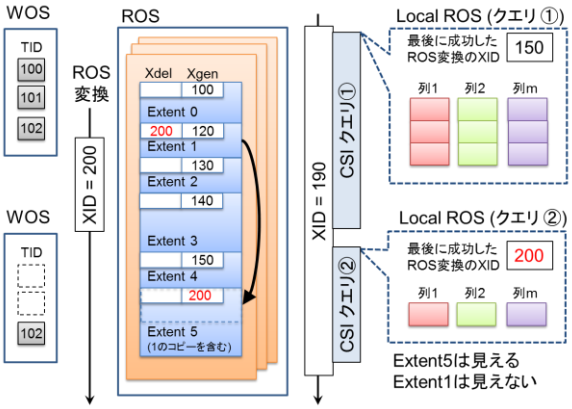

図 2　ROS の MVCC

②については、local ROS があることで解決できる。クエリの開始時には、local ROS も作成する必要があるが、local ROS に含まれるデータは、WOS 上のアイテムの Xmin・Xmax とトランザクションと同じ可視性チェックを用いて判定が行われる。これはすなわちオリジナルテーブル上の行を判定しているのと同じである。

エクステントの可視チェックと、local ROS 作成時の可視チェックを組み合わせて考えると、CSI を用いた場合でもオリジナルと全く同じトランザクションの一貫性が保たれることがわかる。

4. まとめ

OLTP 向けの OSS である PostgreSQL を用い、OLXP を実現するためのカラムストアインデックスのストレージ構造及びトランザクション動作に対する制御の実装について述べた。実装したシステムでは、PostgreSQL の OLTP 性能を損なうことなく、またインデックス作成以外の特別な操作を必要とせずに、モードレスに OLAP を実施することが可能な OLXP システムを実現した。

参考文献

[1] Hasso Plattner, The Impact of Columnar In-Memory Databases on Enterprise Systems, VLDB, pages 1722-1729, 2014.
[2] Oracle Database In-Memory, Oracle, http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html
[3] 田原 司睦 他,カラムナデータをサポートするための PostgreSQL 拡張,第 77 回日本情報処理学会全国大会
[4] M. Stonebraker, et.al., C-Store: A Column-oriented DBMS. VLDB, pages 553-564, 2005.