PostgreSQL インサイド

プラガブルストレージのテーブルへの展開！

テーブルアクセスメソッドインターフェイスへの取り組み

－富士通の技術者に聞く！PostgreSQL の技術－

PostgreSQL は、他の多くの RDBMS と異なり、テーブル内のデータを格納する方法（テーブルストレージ機構）として 1 つの形式しかサポートしていません。この制約に対し、以前から PostgreSQL 開発コミュニティーでは他のテーブルストレージへの対応について議論されてきました。PostgreSQL 12 ではその基盤となる、自由にテーブルストレージ機構を追加できる仕組み（テーブルアクセスメソッド）として、「テーブルアクセスメソッドインターフェイス」機能がサポートされました。これにより、独自のテーブルアクセスメソッドを定義できるようになり、PostgreSQL の適用範囲を大きく広げていくことが可能になりました。
この機能は富士通グループの社員が PGCon 2017 へ議題を提案し、コミュニティーでの議論に参加し、PostgreSQL 12 でのコミット（注 1）を実現しました。
本特集では、テーブルアクセスメソッドインターフェイス開発の取り組みについて、FUJITSU AUSTRALIA SOFTWARE TECHNOLOGY（以降、FAST）の「Pankaj Kapoor」が語ります。

注 1　ここでのコミットとは、PostgreSQL の正式機能として採用される、の意味です。

Pankaj Kapoor

専門分野：データベース
アプリケーションから通信まで、さまざまな分野での 15 年以上の経験、FUJITSU Software Enterprise Postgres の主要開発者で、約 3 年前から PostgreSQL エコシステム業務に従事。

テーブルアクセスメソッドインターフェイスとは

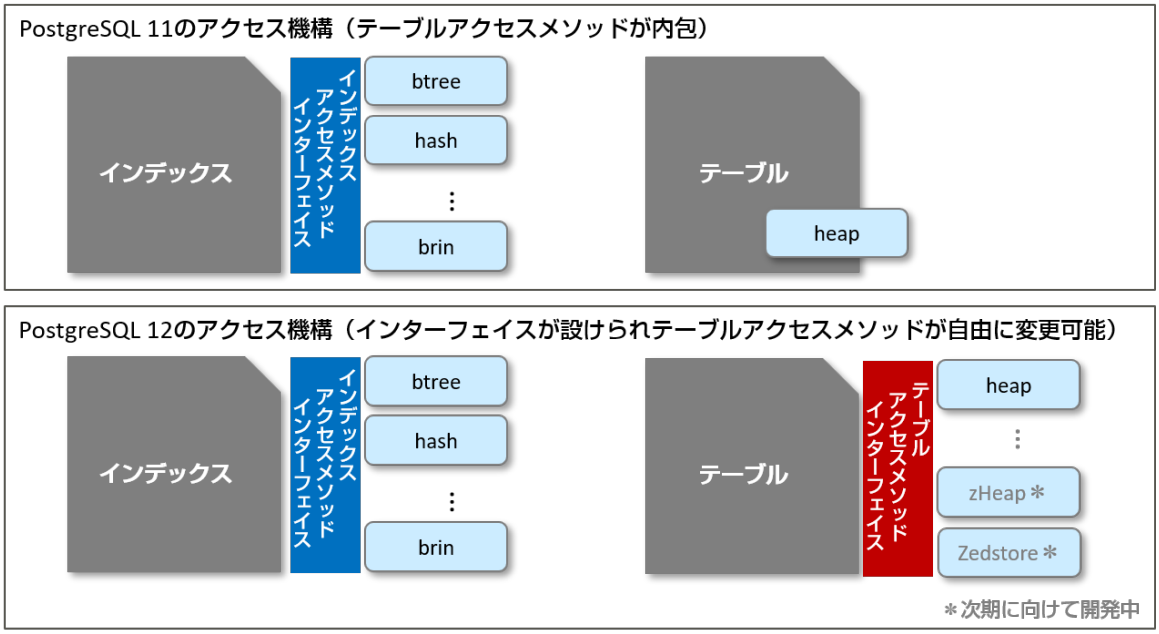
テーブルアクセスメソッドインターフェイスとは、どのようなことができる機能でしょうか。

Pankaj

これは、テーブル内のデータの格納を現状の heap 以外の方法で実装することを可能とした機能です。
PostgreSQL 11 までは、インデックスに対しては btree、hash といった異なる格納方式を選択できるようにアクセスメソッドが用意されていますが、テーブルに対して同様の機構はありませんでした。
PostgreSQL 12 では、インデックスと同様にテーブル用のアクセスメソッドを実装し、異なるテーブルストレージ機構の選択が可能になりました。これが、プラガブルテーブルアクセスメソッドです（注 2）。

プラガブルテーブルアクセスメソッドでは、ユーザーが独自に作成できるテーブルアクセスメソッドを実装するためのインターフェイスを公開しています。これにより、PostgreSQL の開発者や開発チームがテーブル用に独自のテーブルアクセスメソッドを作成することができます。PostgreSQL 12 では、従来から利用してきた heap 形式をテーブルアクセスメソッド上に移行し、デフォルトで利用できます。

注 2　プラガブルとは、自由に取り付け可能な、拡張可能な、の意味です。



PostgreSQL 開発者向けにインターフェイスが設けられたのですね。では、ユーザーがテーブルアクセスメソッドを利用する方法を教えてください。

Pankaj

テーブルアクセスメソッドの定義は、CREATE ACCESS METHOD 文の TYPE 句にアクセスメソッドの型として「TABLE」を指定します。アクセスメソッドを変更したいテーブル単位に、CREATE TABLE 文に USING 句を付けてテーブルアクセスメソッド名を指定します。CREATE TABLE AS SELECT 文や、CREATE MATERIALIZED VIEW 文でも指定できます。
また、postgresql.conf ファイルのパラメーター"default_table_access_method"でデフォルトのテーブルアクセスメソッドを指定する方法もあります。

例）テーブル：tbl1 にテーブルアクセスメソッド：heap1 を指定する場合

```
CREATE ACCESS METHOD heap1 TYPE TABLE HANDLER heap_tableam_handler;
CREATE TABLE tbl1(id int, name text) USING heap1...;
```

テーブルアクセスメソッドの有用性

どうしてこの機能が必要と思われたのか、開発の背景と経緯について教えてください。

Pankaj

開発コミュニティーでは、他の多くの RDBMS と同じように UNDO ログを利用できるようにするために、専用のテーブルアクセスメソッドを追加選択できるようにする必要があり、インデックスと同様にテーブルアクセスメソッドを自由に取り付け可能にすることが重要である、と議論されてきました。heap しか対応していない PostgreSQL のテーブルストレージは、コミュニティー内の他のオープンソースの活動（例えば、zHeap や Zedstore など）にも影響を与えていましたし、そのニーズを満たすことが困難でした。
開発の発端は、2016 年に現コミッターの Alvaro Herrera さんによって、カラムナストレージの機構に対応するパッチが作成されたことでした。また、富士通は、自社が持つカラム型ストレージ技術「Vertical Clustered Index（VCI）」をコミュニティーに寄贈したいと考えており、これを推進するために、FAST の Haribabu Kommi さんが開発に参加しました。その後、現コミッターの Robert Haas さんがストレージ層に重点を置いた、汎用的なテーブルアクセスメソッドへの取り組みを提案しました。コミュニティーメンバーは、現コミッターの Andres Freund さんの支援を受けながら、開発、レビュー、修正に多大な労力を費やして、2019 年 3 月に PostgreSQL 12 の機能としてコミットに至りました。

この機能のアピールポイントと、この機能が想定している使われ方について教えてください。

Pankaj

この機能のアピールポイントは 4 点あります。

- 簡潔で自由に取り付け可能なアーキテクチャー、つまり、使いやすくて PostgreSQL 開発者に優しい
- ユーザーがテーブルごとに異なるテーブルアクセスメソッドを指定できる
- オープンソースと商用データベースの両方に道を開く
- 1 つのデータベース内で異なるテーブルアクセスメソッドの同時共存ができる

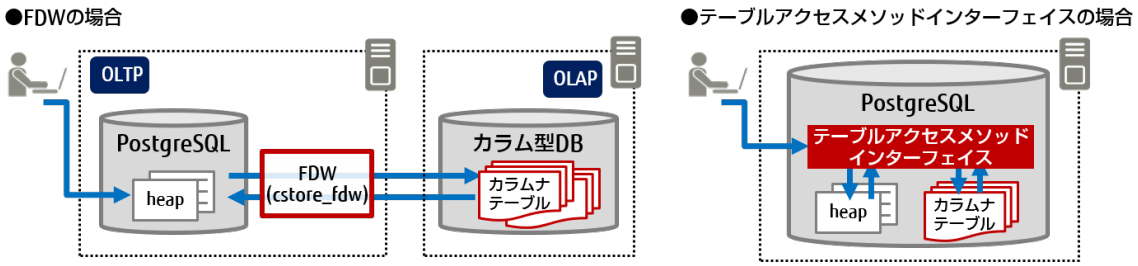
PostgreSQL 12 ではインターフェイスが設けられただけで、対応するテーブルアクセスメソッドは heap のみですが、PostgreSQL の次バージョン以降では、カラムナテーブルやインメモリーテーブルなどの新しいテーブルアクセスメソッドの提供が期待されています。将来的に、ユーザーは OLTP 業務には heap、OLAP 業務にはカラムナテーブル、超高速なデータ検索処理にはインメモリーテーブルと、業務用途に応じて適切なテーブルアクセスメソッドを選択できます。ユーザーが適したテーブルアクセスメソッドを選択できるようにすることで、今後のさまざまな業務処理のニーズに対応できます。

PGCon 2019 の発表の中に「Why not FDW」のご説明がありましたが、PostgreSQL の Foreign Data Wrapper（FDW）とどのような違いがあるのでしょうか。

Pankaj

FDW は外部データにアクセスすることを目的としており、ローカルにデータを格納することを目的としていません。例えば、PostgreSQL からカラム型データベースサーバー内にあるカラムナテーブル型のデータを利用する要件があるとします。FDW の場合は、カラム型 DB に対応した FDW（cstore_fdw）を使ってリモートの別サーバーへアクセスする必要があり、その分処理性能が落ちます。テーブルアクセスメソッドの場合は、ローカルにカラムナテーブル型（注 3）のテーブルを格納することができ、処理が速くなります。
要するに、FDW とテーブルアクセスメソッドは異なるニーズをターゲットにしています。FDW が異なるサーバー上のデータにアクセスすることを目的としているのに対し、テーブルアクセスメソッドはローカルにデータを格納することを目的としています。

注 3　PostgreSQL 12 では、カラムナテーブルのテーブルアクセスメソッドには対応していません。



コミュニティーの協力と富士通の支援があってこそ

新しいインターフェイスの実現は大変だったと思いますが、苦労した点や工夫した点を教えてください。

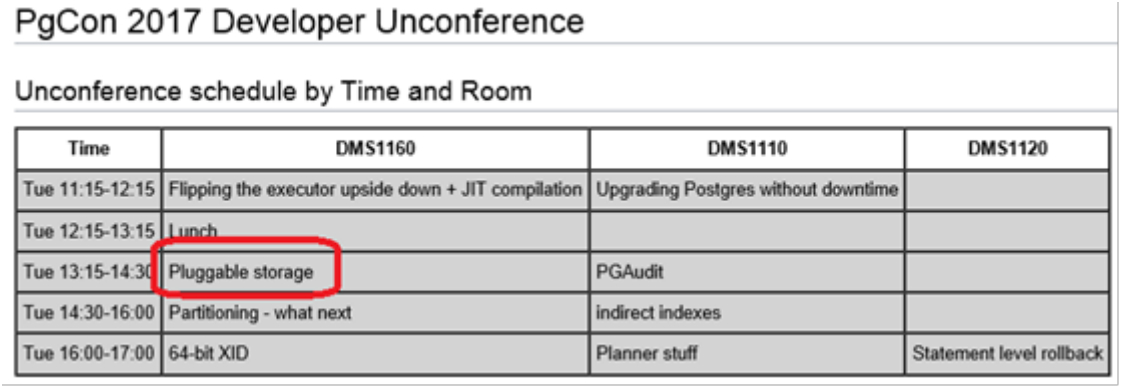
Pankaj

この機能に関わったすべてのメンバーが、大量のインフラストラクチャーのコードの改変に苦労しました。FAST の Haribabu Kommi さんによって提供された最初のパッチセットでさえ、100 以上のファイルに 2500 行以上の変更が発生しました。
PostgreSQL のストレージ周りは他のコンポーネントと密接に連携しているため、調査や調整が大変でした。このようなレベルの機能を実現するのは、サイズだけではなく、複雑さも考慮しなければならず、非常に労力を要する作業でした。
困難な状況において、富士通グループからの継続的な推進とサポートにより、この機能を開発コミュニティーでコミットすることができました。

本機能は PGCon 2017 アンカンファレンスで提案したと聞きましたが、周りの反応はどうでしたか？

Pankaj

アンカンファレンスでは議論する検討テーマを投票で決定するのですが、これはコミュニティーで最も多く投票されたテーマの 1 つでした。個人の関心だけでなく、さまざまな組織でより多くの必要性が認識され、アンカンファレンスで人気のトピックになりました。



PgCon 2017 Developer Unconference

Unconference schedule by Time and Room

Time	DMS1160	DMS1110	DMS1120
Tue 11:15-12:15	Flipping the executor upside down + JIT compilation	Upgrading Postgres without downtime	
Tue 12:15-13:15	Lunch		
Tue 13:15-14:30	Pluggable storage	PGAudit	
Tue 14:30-16:00	Partitioning - what next	indirect indexes	
Tue 16:00-17:00	64-bit XID	Planner stuff	Statement level rollback

PostgreSQL 12 でのコミット後の PGCon 2019 では"Pluggable storage"が 2 つ発表されていましたね。注目のテーマだったのですね。

Pankaj

はい。これはプレゼンテーションルームだけでなく、コーヒーや軽食を摂りながらの議論も活発に行われ、非常に興味深いテーマでした。プレゼンテーションの 1 つはコミッターの発表で、アーキテクチャーと開発者の見解に焦点を当てていました。もう 1 つの私の発表はテーブルアクセスメソッドの使用法に触れ、ユーザー視点で説明しました。
詳しくは、PGCon 2019 のオフィシャルサイトを参照してください。

- PGCon 2019
  https://www.pgcon.org/2019/

PostgreSQL の今後の発展に向けて

PostgreSQL 12 リリース後のカンファレンスで、Zedstore やテーブルアクセスメソッドインターフェイス関連のテーマが議論されているようですが、このような動向についてどうお考えでしょうか？また、注目している議論などがあれば教えてください。

Pankaj

PostgreSQL 14 以降で、テーブルアクセスメソッドの可能な実装をいくつかご紹介します。

- heap の改良版（例えば、zHeap などの行指向の格納、UNDO ログ、VACUUM レスを持つもの）
- カラムナテーブル（例えば、Zedstore などの列指向の格納、OLAP 向き、圧縮機能を持つもの）
- インメモリーテーブル（データをメモリー上に配置して、DB アクセスを高速化するもの）
- インデックス構成テーブル（主キーでソートされた方法でデータが格納されたもの）、など

zHeap と Zedstore はテーブルアクセスメソッドの開発中の実装で、どちらも機能的にとても優れています。私は Zedstore の議論をフォローしており、可能であれば彼らの開発を支援していきたいと考えています。

拡張性の高い機能ですが、さらに追加したい点など、この機能の今後の取り組みについて教えてください。

Pankaj

PostgreSQL 12 では、テーブルアクセスメソッドはタプル（行）ベースだけですが、PostgreSQL の今後の可能性を広げる機能の基盤を築くことができました。将来的には、実行エンジンとテーブルアクセスメソッドインターフェイスを組み合わせた最適な実装をサポートしたいと考えています。

PostgreSQL 12 で導入されたテーブルアクセスメソッドインターフェイスにより、PostgreSQL ユーザーは要件に合わせたテーブルアクセスメソッドのテーブルを作成できるようになったということですね。将来的には、業務の特性に合ったテーブルアクセスメソッドを複数選択することができ、ますます PostgreSQL の活躍の場が広がりそうで楽しみです。今日はありがとうございました。

2020 年 10 月 9 日