PGECons
PostgreSQL Enterprise Consortium

2019年度CR部会活動報告

PostgreSQL エンタープライズ・コンソーシアム
CR部会

アジェンダ

- CR部会のご紹介
  - CR部会の目的、参加メンバ
- 2019年度の取り組みのご紹介
  - コミュニティへのフィードバックに向けた取り組み
  - PostgreSQL機能改善への協力
- おわりに
  - 2020年度にむけて

CR部会のご紹介

CR(Community Relations)部会の目的

- PostgreSQL開発コミュニティへのフィードバック
  - エンタープライズ領域へのPostgreSQLの適用に向けて、開発コミュニティに技術的課題をフィードバック

抱えている課題が解決される！

ユーザの声が届く！

CR部会 2019年度参加メンバ

- SRA OSS, Inc.日本支社　【主査】
- NECソリューションイノベータ株式会社
- NTTテクノクロス株式会社
- 日本電信電話株式会社
- 富士通株式会社

（企業名50音順・敬略称）

2019年度の取り組み

2019年度の取り組み内容

1. 開発コミュニティへのフィードバックを目指し、エンタープライズ領域における技術的課題を詳細化

2. PostgreSQLの機能改善活動への協力
   - Incremental View Maintenance (IVM)

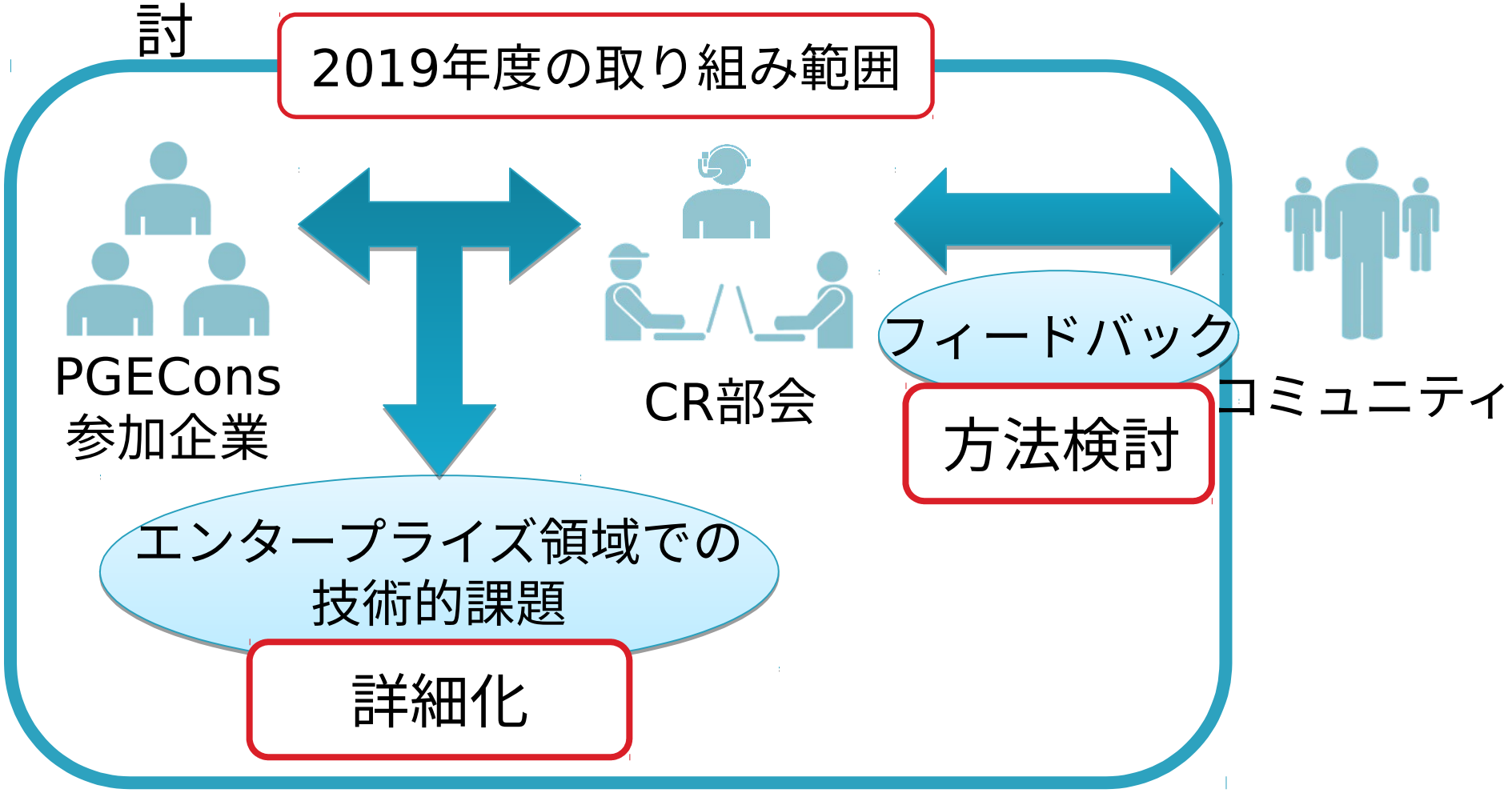
2019年度の取り組み内容

1. 開発コミュニティへのフィードバックを目指し、エンタープライズ領域における技術的課題を詳細化

2. PostgreSQLの機能改善活動への協力
   - Incremental View Maintenance (IVM)

フィードバックに向けた2019年度の活動概要

- 技術的課題の詳細化とフィードバック方法の検討



課題の詳細化への取り組み

- [課題]データのロードが遅い
  - 課題詳細
    - COPY文でデータを取り込むのが遅い
    - 外部ツール(pg_bulkload)もあるが、課題もある
      - 初期ロードにしか使えない
      - パーティション機能への適用は制限がある
      - 追加モジュールなので、クラウドでのDaaS環境では使えない
  - 本課題に対する取り組み
    - COPYの改善に注力する
    - 開発コミュニティで議論する方向で、そのための事前調査を行った

COPYに関する開発コミュニティでの議論分析

- 103本の議論スレッドから、比較的最近の議論(2017/11以降)を中心に、47本を解析
- 大きく分けて機能改善と性能改善の提案がある
  - 機能改善では、エラーが起きてもアボートせずにCOPY TOを継続する提案が過去何度かあった
  - 性能改善では、WALの出力を抑止する、COPY TOのパラレル化、COPY FREEZEでVisible flagをセットしてI/Oを改善するなどの提案があった
  - 以下は現在も進行中(2020/8/6現在)
    - COPY TOのパラレル化
    - COPY FREEZEでVisible flagをセットしてI/Oを改善

実行計画に起因する性能課題(事例)の収集

- [課題]実行計画が制御できない
  - 課題詳細
    - 不適切な実行計画が採用されクエリの実行性能が出ない
    - 統計情報の変化に起因して実行計画が変化することで性能が不安定に
      ⇒ pg_hint_planなどのExtensionをインストールして実行計画を制御
  - 本課題に対する取り組み
    - 抱えている課題をコミュニティに投稿（フィードバック）し、最良の解決策を探りたい
    - 投稿に向けてコミュニティの課題感と摺合せられるようにユーザのニーズを調査したい
    - [2019年度の取り組み] 現場で見られる実行計画の問題事例を収集するためのアンケート項目を検討中

2019年度の取り組み内容

1. 開発コミュニティへのフィードバックを目指し、エンタープライズ領域における技術的課題を詳細化

2. PostgreSQLの機能改善活動への協力
   - Incremental View Maintenance (IVM)

Incremental View Maintenance (IVM)とは

- Materialized viewの更新をリアルタイムかつ高速に行う機能
  - Materialized viewはSELECT結果を保存し、次回以降の検索を高速化する機能
  - 元のテーブルを更新してもMaterialized viewは更新されない
  - 手動でmaterialized viewを更新する必要あるが、SELECTをやり直すので遅い
  - もし自動でMaterialized viewを更新したければ、トリガを使ってMaterialized view相当の機能を実装しなければならない

Incremental View Maintenance (IVM)開発への協力

- PostgreSQL開発者メーリングリストでの議論に参加
  - 20通以上のレビュー・意見を投稿
- GitHubでのレビュー、テストで協力
  - 10個のIssueの登録、修正確認を実施
- IVM開発は現在も継続中
  - PostgreSQL 14に入れるのが目標

Incremental View Maintenance (IVM)の特徴

- IVMのメリット
  - 元のテーブルを更新するとMaterialized viewも自動的に更新される
  - 必要な部分だけmaterialized viewを更新するので高速
  - 更新はリアルタイムで行われる
  - トリガは元のテーブルに自動設定される。自前のトリガは必要ない。
    - トリガのコーディングに起因するバグを心配しなくて良い

Incremental View Maintenance (IVM)の特徴

- IVMの課題
  - すべてのSELECTに対応しているわけではない。2020/8/4時点で以下を含むSELECTには対応していない
    - WITH句、WINDOW関数、UNION/INTERSECT/EXCEPT、FOR UPDATE/SHARE、GROUPING SETS、複雑な副問合せ、OUTER JOINの一部、ユーザ定義集約
    - TPC-DSの99本のクエリのうち、20本をサポート
      - TPC-DSとは、ベンチマーク標準化団体TPCが定義した意思決定支援システムをモデル化したベンチマーク
  - 元になったテーブル(base table)を更新する際にトリガのオーバヘッドがかかる
    - このテーブルを参照しているIVMの数が多いほどオーバヘッドも大きくなる

Incremental View Maintenance (IVM)の利用

- 使い方
  - 例：pgbench(PostgreSQLの付属ベンチマークツール)が生成する1000万件のテーブルの行数をカウントするクエリのマテリアライズドビューを作りたい
    - PostgreSQLでは行数カウントは順スキャンになり、大きなテーブルでは遅くなりがち
    - CREATE INCREMENTAL MATERIALIZED VIEW mv1 AS SELECT count(*) FROM pgbench_accounts;
  - ビューの参照

```
SELECT * FROM mv1;
  count
----------
 10000000
(1 row)
```

Incremental View Maintenance (IVM)の利用

- 使い方
  - データ更新
    - データ更新は元になったテーブル(base table、ここではpgbench_accounts)を普通に更新する。更新結果は直ちにビューに反映される
    - DELETE FROM pgbench_accounts WHERE aid =1;
  - 結果の参照はビューをSELECTするだけ。ビューの作り直しや、REFRESHは必要ない

```
SELECT * FROM mv1;
  count
----------
 9999999
(1 row)
```

Incremental View Maintenance (IVM)の利用

- サポートしていない問い合わせでビューを作成しようとしたら？
  - CREATE INCREMENTAL MATERIALIZED VIEWを発行した段階でエラーとなる
    - ERROR: CTE is not supported with IVM

Incremental View Maintenance (IVM)の利用

- インデックスは必要？
  - なくても動くが、インデックスを作った方が格段に速くなることがある。
    - ベーステーブルにインデックスを作成
    - ビューの行数が大きい場合は、ビューにもインデックスを作成

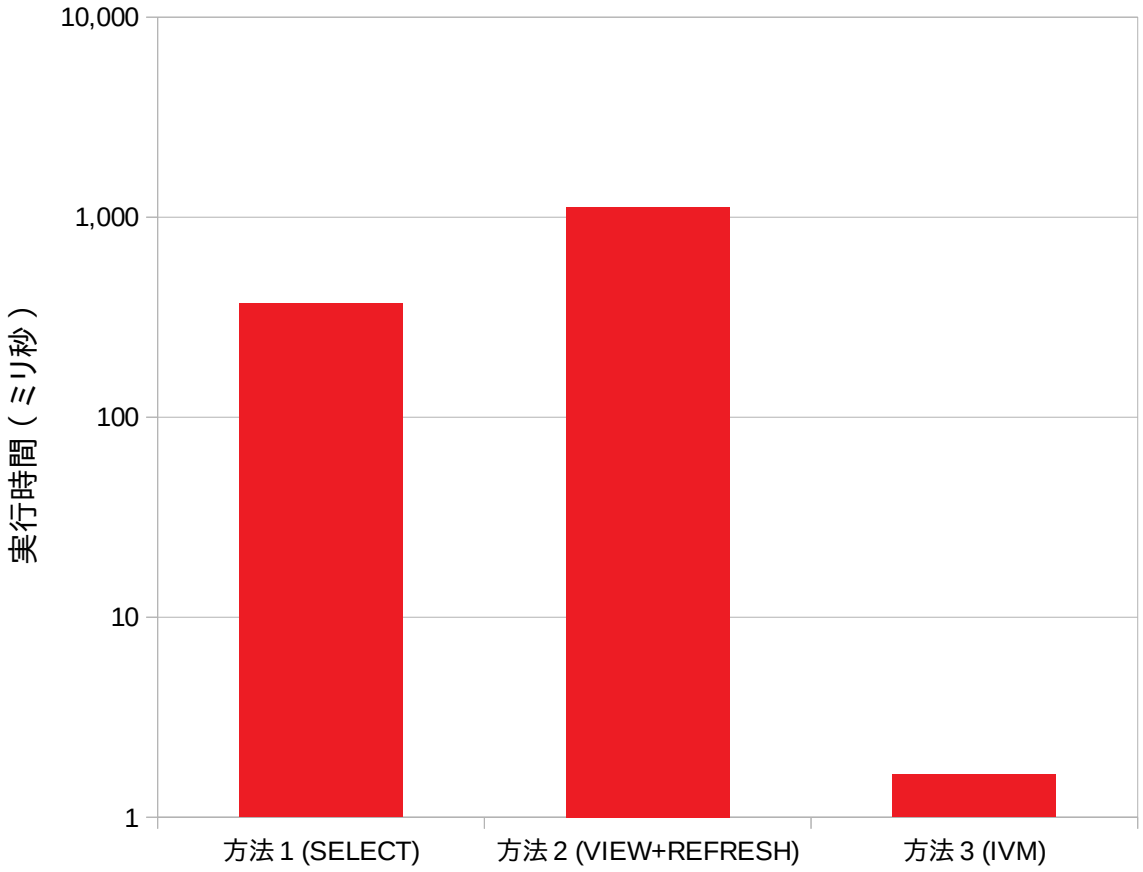
Incremental View Maintenance (IVM)の性能

- 想定シナリオ
  - 大きなテーブル(pgbench_accounts)の行数をリアルタイムで知りたい
    - 方法1: 普通にSELECT count(*)する
    - 方法2: Materialized viewを作り、pgbench_accountsが更新されたらビューをREFRESHしてからビューをSELECTする
    - 方法3: Incremental Materialized viewを作り、pgbench_accountsが更新したらビビューをSELECTする

Incremental View Maintenance (IVM)の性能

- DELETE FROM pgbench_accounts WHERE aid = 1;
  - 方法1: SELECTをそのまま実行
    - 370.214ms
  - 方法2: base tableのDELETE+ビューのREFRESH, count数取得
    - 1,122.455ms
  - 方法3: base tableのDELETE+IVMからcount数を取得
    - 1.636ms

Incrementa View Maintenance (IVM)の性能



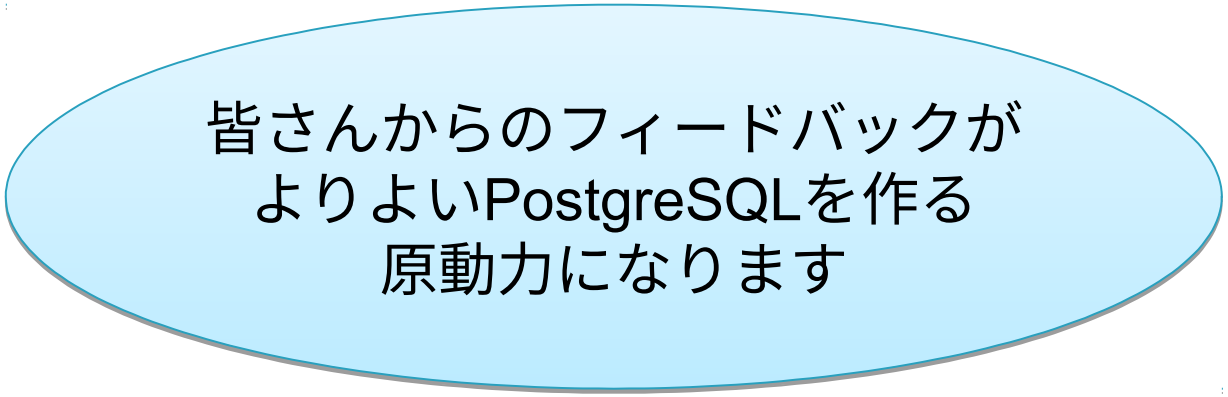
おわりに

2020年度に向けて

- 現在の取り組みを継続
  - エンタープライズ領域での技術的課題詳細化を継続
  - 実行計画を制御できないために発生する課題の収集
  - Incremental View Maintenaceの開発支援
- コミュニティへのフィードバック活動を活性化
  - 実行計画についての課題を投稿して解決へと導く
  - COPY関係の改善議論へのフィードバック
  - Incremental View Maintenaceの議論へのフィードバック

皆様へのお願い

- 今年度（2020年度）は、ヒント句やIncremental View Maintenaceなどの PostgreSQLの改善活動の支援に取り組む予定です
  - 是非皆様のご意見をお寄せください



- 一緒にフィードバックに活動しませんか？

PGECons

PostgreSQL Enterprise Consortium