PGECons
PostgreSQL Enterprise Consortium

2020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
- 2020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
  - 2021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 2020年度参加メンバ

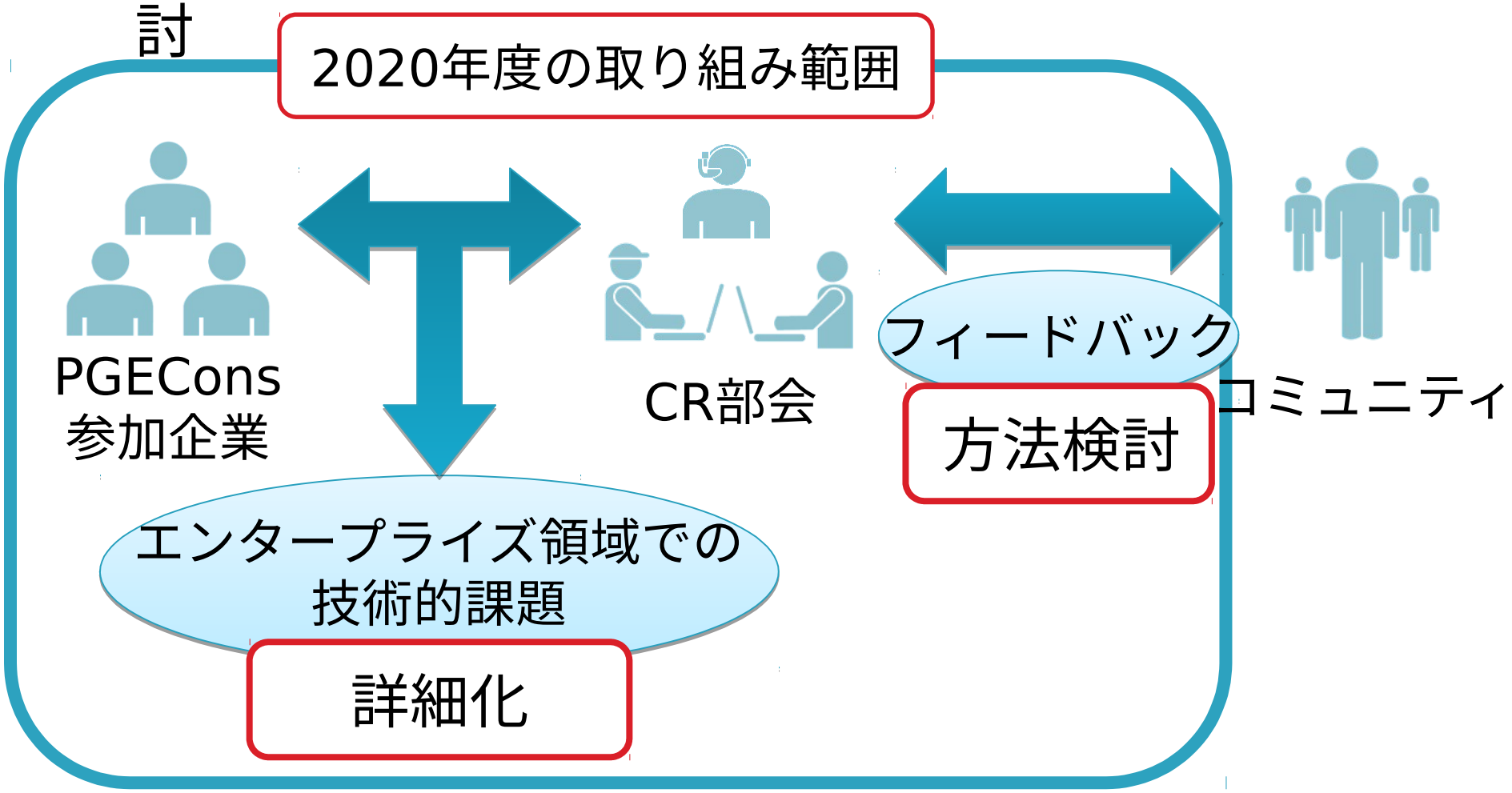
- SRA OSS, Inc.日本支社　【主査】
- NECソリューションイノベータ株式会社
- NTTテクノクロス株式会社
- 日本電信電話株式会社
- 富士通株式会社

（企業名50音順・敬略称）

2020年度の取り組み

フィードバックに向けた2020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



2020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. 実行計画の調整機能(hint)の実現に向けてヒアリングを実施
3. IVM (Incremental View Maintenance、 マテリアライズドビューの増分リフレッシュ)開発の支援活動
4. CoC (Code of Conduct; コミュニティの行動規範)の原文が改訂されたことを受け、日本語訳を更新

2020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. 実行計画の調整機能(hint)の実現に向けてヒアリングを実施
3. IVM (Incremental View Maintenance、 マテリアライズドビューの増分リフレッシュ)開発の支援活動
4. CoC (Code of Conduct; コミュニティの行動規範)の原文が改訂されたことを受け、日本語訳を更新

本作業の経緯

- オリジナルTODO項目「基幹領域への適用におけるPostgreSQLの抱える課題(2012年)」
  - PGECons成立当初に、エンタープライズ用途でPostgreSQLを使えるようにするために、課題を各社持ち寄り検討したのが始まり
  - 技術的・非技術的課題を含めて全46項目
- 2016年度の第2回CR部会会合で、CR部会として以下の取組を行うことを決定
  - 各社で分担して、各項目の検討および詳細化を実施
  - 非技術項目、解決済みの項目は削除

「基幹領域への適用におけるPostgreSQLの抱える課題(2012年)」（一部）

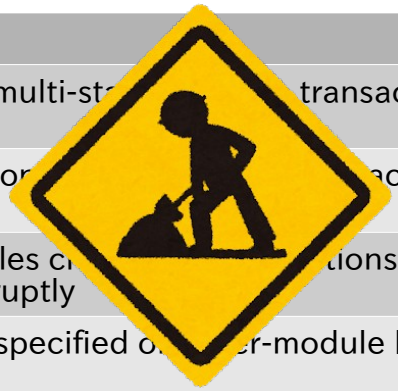
基幹領域への適用におけるPostgreSQLの抱える課題				
項番	カテゴリ		課題	
	大区分	小区分	概要	詳細 ＜課題の詳細とそれを裏付ける具体
1	性能	性能モデル	オンライン業務に対する性能モデルがない	ターゲットシステムへの導入の可能性
2			バッチ業務に対する性能モデルがない	ターゲットシステムへの導入の可能性
3			統計分析処理（BIなど）に対する性能モデルがない	ターゲットシステムへの導入の可能性
4			大容量データ処理に対する性能モデルがない	項番2、3に関連して、大量データ処理 なっていない。（Hadoopなどビッグデー る領域はどこか、PostgreSQLにて対応
5		スケールアップ	CPUマルチコアに対する性能懸念	

コミュニティTODO

- PostgreSQL開発コミュニティのTODO項目を解析・整理
  - PGECons TODOとの突き合わせを実施
  - その結果、コミュニティTODOにないPGEConsの独自のTODOがあることを確認
  - PGECons TODOを深耕する意義が確認できた

整理したコミュニティTODO(一部)

ID	section	subsection	summary
2.0-1	Administration		Allow administrators to cancel multi-st transactions
2.0-2	Administration		Allow administrators to cancel lo actions
2.0-3	Administration		Check for unreferenced table files c tions that were in-progres when the server terminated abruptly
2.0-4	Administration		Allow log_min_messages to be specified o r-module basis
2.0-5	Administration		Simplify creation of partitioned tables
2.0-6	Administration		Allow custom variables to appear in pg_settings()
2.0-7	Administration		Have custom variables be transaction-safe
2.0-8	Administration		Implement the SQL-standard mechanism whereby REVOKE ROLE revokes only privilege granted by the invoking role, and not those granted by other roles
2.0-9	Administration		Prevent query cancel packets from being replayed by an attacker, especially when using SSL
2.0-10	Administration		Provide a way to query the log collector subprocess to determine the name of the currently active log file
2.0-11	Administration		Improve logging of prepared transactions recovered dur startup



TODO項目検討作業の現状

- 初期のTODO 46項目から検討対象を17項目に絞り込みを実施
  - 非技術項目、解決済み項目を削除、類似の項目のまとめ
- 現在取り組んでいる項目の例
  - 性能モデルの欠落
    - 適用事例ごとの性能傾向がわかるように、性能事例を収集する方向で検討
      - https://wiki.postgresql.org/wiki/Performance_Case_Study

TODO項目検討作業の現状

- 現在取り組んでいる項目の例（続き）
  - 実行計画が制御できない
    - 実行計画の調整を可能にする追加モジュールである

      pg_hint_plan の採用を開発コミュニティに働きかける方向で活動中
  - データの初期ロードが遅い
    - COPY本体の性能改善を目指す

課題の詳細化への取り組み

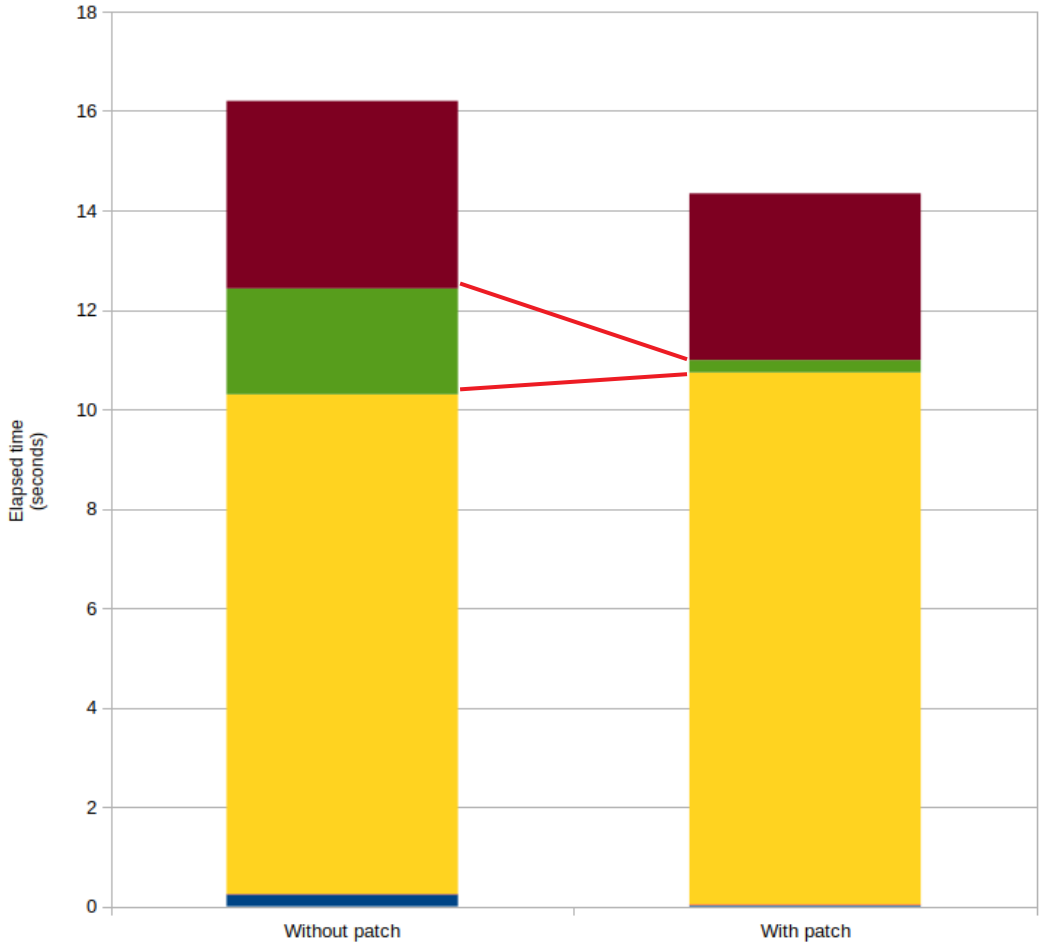
- [課題]データのロードが遅い
  - 課題詳細
    - COPY文でデータを取り込むのが遅い
    - 外部ツール(pg_bulkload)もあるが、課題もある
      - 初期ロードにしか使えない
      - パーティション機能への適用は制限がある
      - 追加モジュールなので、クラウドでのDaaS環境では使えない
  - 本課題に対する取り組み
    - COPYの改善に注力する
    - コミュニティにおけるCOPY関連の47議論スレッドを詳細に解析
    - COPY FREEZE改善パッチのレビューで協力、PostgreSQL 14向けにコミットされた

COPY FREEZE改善について

- COPY FREEZE実装の課題
  - COPY FREEZE後にVACUUMしてもI/Oは発生しないはずが、実際には全ブロックの書き込みが発生していた
    - 原因は、PD_ALL_VISIBILEフラグがセットされていなかったため
    - PD_ALL_VISIBLEフラグは、ページの中の全行が「可視」であることを示すフラグ
  - また、Visibility mapが更新されていなかった
    - これが更新されていないと、index only scan が効かない
- 改善されたCOPY FREEZE
  - PD_ALL_VISIBLEフラグをセット
    - COPY FREEZE後のVACUUMでI/Oが発生しないのでVACUUMが高速
    - COPY FREEZE後にVACUUMをしなくてもindex only scanが効く

COPY FREEZE改善の効果

- pgbench -i -s 100 で検証



緑の部分がVACUUMの時間。
改善により、VACUUM時間が2.13秒から0.25秒に8.5倍高速化された。
トータルでは11%の高速化を達成

https://pgsqlpgpool.blogspot.com/2021/03/speeding-up-pgbench-using-copy-freeze.html より

2020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. 実行計画の調整機能(hint)の実現に向けてヒアリングを実施
3. IVM (Incremental View Maintenance、 マテリアライズドビューの増分リフレッシュ)開発の支援活動
4. CoC (Code of Conduct; コミュニティの行動規範)の原文が改訂されたことを受け、日本語訳を更新

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
      - PGECons内でヒアリングを実施
      - 更にヒアリングを行う予定

2020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. 実行計画の調整機能(hint)の実現に向けてヒアリングを実施
3. IVM (Incremental View Maintenance、 マテリアライズドビューの増分リフレッシュ)開発の支援活動
4. CoC (Code of Conduct; コミュニティの行動規範)の原文が改訂されたことを受け、日本語訳を更新

Incremental View Maintenance（IVM）

- マテリアライズド・ビューの更新を高速に行う機能
  - マテリアライズド・ビューはSELECTクエリの結果を保存することで、応答を高速化する
    - テーブルの更新後、ビューに保存されている結果も更新する必要がある
    - 通常の REFRESH ではクエリの再実行を行うため時間がかかる
  - IVM はテーブル更新によって生じた「増分」のみを計算して、それを適用することでビューを最新化する
    - クエリの再実行よりも効率的にビューを更新できる
    - 現在の PostgreSQL (バージョン13）には未実装の機能

PostgreSQL への IVM 実装の提案（１）

- 機能
  - テーブルを更新すると、その直後にマテリアライズド・ビューが自動的に更新される
- メリット
  - ビュー全体ではなく必要な部分のみを更新するため高速
  - 手動でREFRESHを実行しなくてもビューは最新の状態に保たれる
  - ユーザが自前でトリガ関数を書く必要がない

PostgreSQL への IVM 実装の提案（２）

- 対応しているクエリ
  - SELECT、結合（内部結合、外部結合、自己結合）、DISTINCT
  - 組み込みの集約（sum, count, avg, min, max)、GROUP BY
  - FROM句内の単純なサブクエリ、単純なCTE（WITH 句）
  - 相関サブクエリ（EXISTS）
- 非対応
  - 上記以外の集約、ウィンドウ関数、HAVING
  - 複雑なサブクエリ、CTE

    （内部に集約、DISTINCT、外部結合を含むもの）
  - ORDER BY、LIMIT/OFFSET、
  - 集合演算（UNION, EXCEPT, INTERSECT）

IVM 機能の使用方法（１）

- 自動的に増分更新されるマテリアライズド・ビューの作成
  - CREATE INCREMENTAL MATERIALIZED VIEW 文
    - 例）pgbench(PostgreSQLの付属ベンチマークツール)が生成する1000万件のテーブルに対する集約クエリ

```
CREATE INCREMENTAL MATERIALIZED VIEW mv AS
        SELECT bid, count(abalance), sum(abalance), avg(abalance)
        FROM pgbench_accounts GROUP BY bid;
```

  - 非対応のクエリが指定された場合には、この時点でエラーとなりビューは作成されない

IVM 機能の使用方法（2）

- ビューの参照

```
postgres=# SELECT * FROM mv WHERE bid = 1;
 bid | count  | sum  |          avg
-----+--------+------+------------------------
   1 | 100000 | 5792 | 0.057920000000000000
(1 row)
```

- ビューの更新
  - テーブルを更新すると、ビューが自動で更新される

```
postgres=# UPDATE pgbench_accounts
                SET abalance = abalance + 1000 WHERE aid = 1;
UPDATE 1


postgres=# SELECT * FROM mv WHERE bid = 1;
 bid | count  | sum  |          avg
-----+--------+------+------------------------
   1 | 100000 | 6792 | 0.067920000000000000
(1 row)
```

IVM の性能（１）

- TPC-Hのクエリを用いた評価
  - 使用クエリ
    - Q01: １つの大きなテーブル（lineitem）に対する集約
    - Q09: 6つのテーブルを結合した上で集約
  - スケールファクタは1
    - lineitem が 600万行程度
  - 結果
    - テーブルを１行更新したときの応答時間は、REFRESH MATERIALIZED VIEW コマンドに比べて高速
    - Q01 で 2000 倍以上、Q09 で 200 倍以上

	クエリの実行	ビューへのアクセス	再実行によるビュー更新（REFRESH）	IVMによるビュー更新（テーブルを1行更新）
Q01	10.311 s	1.585 ms	36.811 s	17.104 ms
Q09	3.124 s	2.331 ms	6.153 s	29.301 ms

IVM の性能（２）

- IVM はREFRESHコマンドより高速にマテリアライズド・ビューを更新できる
  - 効率的なビュー更新のためには、適切なインデックスがビューに作成されている必要がある

  → 可能な場合には、ビュー定義時に自動でインデックスが作成される

- ビュー更新のオーバヘッドがテーブル更新性能に影響する
  - テーブルを更新するたびに、ビューを更新する処理が発生
  - ビューの内容に不整合が発生するのを防ぐために排他ロックが使用されているため、同時実行性能が低くなる

  →「テーブル更新頻度は高くないが、更新があった際にはすぐに最新のクエリ結果が欲しい」 という状況にこの機能は有用

前年度からの改良点

- ビュー定義クエリとして CTE（WITH 句）に対応
  - 内部に集約、DISTINCT、外部結合を含まない単純なものに限る
- ビュー定義時に自動で一意インデックスを作成
  - テーブルの主キーのカラムがすべてビューに含まれている場合、これらのカラムにインデックスを作成する
  - GROUP BY がある場合には、そこで指定されたカラムにインデックスを作成する
- ビュー更新時に使用する排他ロック頻度の軽減
  - ビューに含まれるテーブルが１つだけの場合には、排他ロックは不要となるため
- Docker Hub にイメージを公開
  - https://hub.docker.com/r/yugonagata/postgresql-ivm

PostgreSQL ユーザの声

Daniel Westermann
@westermanndanie

This is a great feature for #PostgreSQL, currently in development...no need to manually refresh mviews anymore (commitfest.postgresql.org/31/2138/)

ツイートを翻訳

```
postgres=# \h create materialized view
Command:     CREATE MATERIALIZED VIEW
Description: define a new materialized view
Syntax:
CREATE [ INCREMENTAL ] MATERIALIZED VIEW [ IF NOT EXISTS ] table_name
    [ (column_name [, ...] ) ]
    [ USING method ]
    [ WITH ( storage_parameter [= value] [, ... ] ) ]
    [ TABLESPACE tablespace_name ]
    AS query
    [ WITH [ NO ] DATA ]

URL: https://www.postgresql.org/docs/devel/sql-creatematerializedview.html

postgres=#
```

午前2:25 · 2020年12月7日 · Twitter Web App

61 件のリツイート　22 件の引用ツイート　243 件のいいね

PostgreSQL には IVM 機能が求められている!

Carlo Alberto Ferraris
@CAFxX

Really hoping that #PostgreSQL will eventually gain incremental automatic updates for materialized views: wiki.postgresql.org/wiki/Increment…

ツイートを翻訳

午後2:50 · 2021年4月27日 · Twitter Web App

2 件のリツイート　4 件のいいね

Jon Udell
@judell

"Incremental View Maintenance (IVM) is a technique that ... computes and applies only the incremental changes to the materialized views rather than recomputing the contents. It is not implemented on PostgreSQL yet."

Need it, glad you're working on it.

pgcon.org/2019/schedule/…

ツイートを翻訳

午前9:38 · 2020年2月5日 · Twitter Web App

2 件のいいね

PostgreSQL 開発コミュニティとの関係

- 開発者メーリングリストで議論しながらの開発
  - 「ビューのインデックス自動作成」「排他ロック頻度の軽減」といった改良は、ここでの議論に基づくもの
- PostgreSQL 14 には入らないことが決まった
  - パッチのレビューが十分に行われなかった
  - 機能が大きく複雑であるため、レビューしづらいのが原因？
- 今後は、パッチを細かく分割、機能を削減してサイズを小さくすることで、レビューしてもらいやすくする予定
  - 機能を部分的・段階的に取り入れてもらうことを想定
  - 例えば、まずは集約まではサポートするが、外部結合、サブクエリ、CTEなどはサポートしない、など

2020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. 実行計画の調整機能(hint)の実現に向けてヒアリングを実施
3. IVM (Incremental View Maintenance、 マテリアライズドビューの増分リフレッシュ)開発の支援活動
4. CoC (Code of Conduct; コミュニティの行動規範)の原文が改訂されたことを受け、日本語訳を更新

CoCの改訂

- 英語版が改訂されたことを受け、日本語版の改訂をCR部会内で実施
- 本家のCoC委員会に日本語訳の改訂版を提出
- 翻訳の改訂プロセスが今回定義され、それにしたがってレビューを経て受理された
  - ML(pgsql-general)でフィードバックを募集
  - OKならCoC委員会が掲載
  - pgsql-announceにて、日本語訳が改訂されたことがアナウンスされた
  - これが刺激になったのか、日本語以外にも続々と各国語で翻訳が行われた
    - ヘブライ語、イタリア語、ロシア語

> The PostgreSQL Code of Conduct Committee is pleased to announce that the Japanese translation of the PostgreSQL Community Code of Conduct has been updated, and is now current. The Committee thanks Tatsuo Ishii, Tetsuo Sakata, and Akira Kurosawa for their work in updating the translation.
>
> (pgsql-announceより)

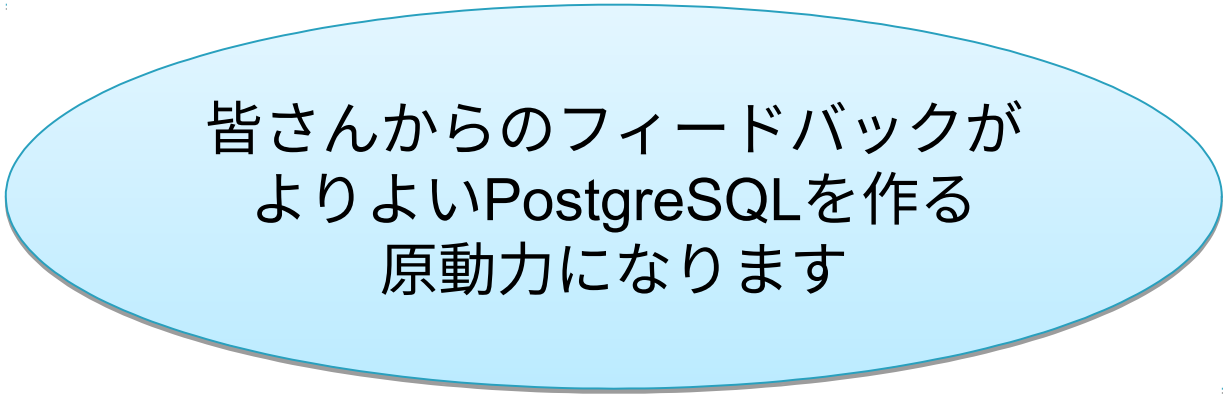
おわりに

2021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の詳細化を継続
  - 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（2021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活動しませんか？

PGECons

PostgreSQL Enterprise Consortium