PGECons
PostgreSQL Enterprise Consortium

PostgreSQLの強化・変更に対する移行検討

2019年度活動成果報告

PostgreSQLエンタープライズ・コンソーシアム
WG2（移行WG）

アジェンダ

- PGECons WG2 ～ これまでの活動内容 ～
- 2019年度活動報告
  - ストアドプロシージャの移行
  - パラレル処理の適用
  - メジャーバージョンアップ
  - 「移行ガイドブック」の改訂
- おわりに

PGECons WG2

～ これまでの活動内容 ～

ワーキンググループ活動について

- 現在3つのワーキンググループにて活動中

新技術検証WG（WG1）

- 新バージョンの性能や新技術の検証を通じて有用性を明確化
- スケールアップ検証、新機能における性能特性調査等

移行WG（WG2）

- 異種DBMSからの移行をテーマに活動
- 商用DBMSからの移行プロセスに伴う技術調査や検証を実施

課題検討WG（WG3）

- データベース管理者やアプリケーション開発者が抱える現場の課題や困り事に対するテーマを設定
- 可用性・運用性・保守性・セキュリティ・接続性が主な課題領域

移行WG(WG2)活動内容

活動テーマ:異種DBMSからPostgreSQLへの移行

課題認識

- 異種DBMSシステムをPostgreSQLへ移行するプロセスが確立していないことが、普及を妨げる大きな障壁と認識
  - 移行作業をどのように進めればよいかがわからない。
  - 初期段階で移行に必要なトータルコストを算出できない。
  - 過去の経験則や点在するノウハウに依存しているのが現状

活動目標

- 異種DBMSからPostgreSQLへの移行を検討する際のガイドラインを提示する。
  (難易度判断、留意すべき事項、移行手順)

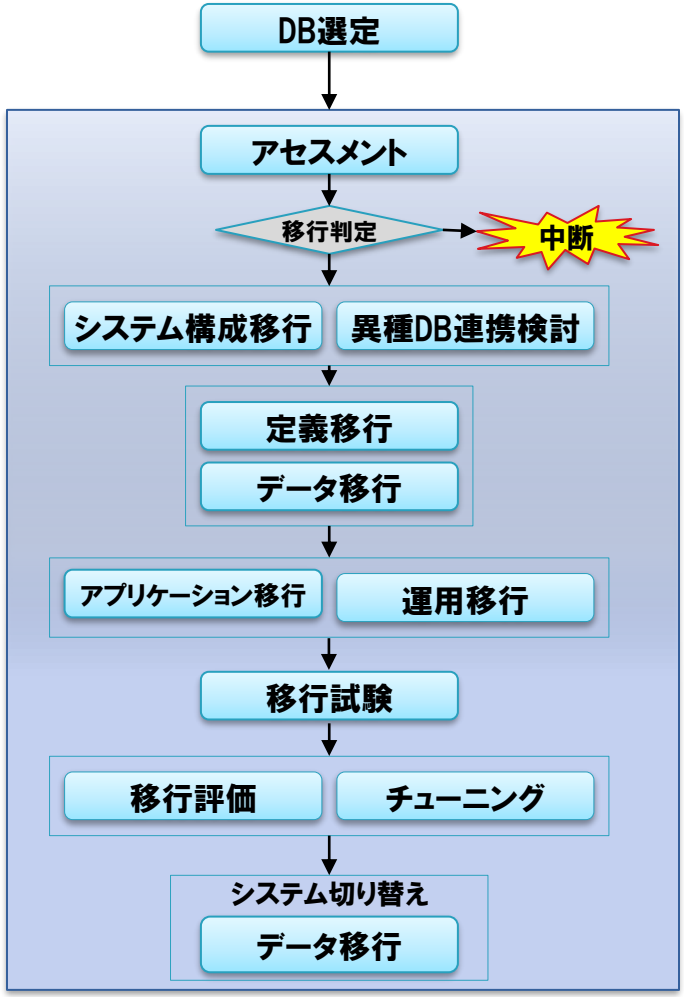
成果物

- 「異種DBMSからPostgreSQLへの移行ガイド」を作成

「異種DBMSからPostgreSQLへの移行ガイド」の構成

- 移行作業の全体像を解説
  - DB移行フレームワーク編 （21ページ）
  - DB移行開発見積り編（26ページ）
- 移行作業に含まれる作業内容、手順の調査
  - システム構成調査編（29ページ）
  - 異種DB間連携調査編（18ページ）
  - スキーマ移行調査編（25ページ+別表）
  - データ移行・文字コード変換編（49ページ）
  - ストアドプロシージャ移行調査編（23ページ）
  - アプリケーション移行調査編（10ページ）
  - SQL移行調査編（20ページ+別表）
  - 組み込み関数移行調査編（30ページ+別表）
  - チューニング編（30ページ+別表）
  - バージョンアップ編（39ページ+別表）
  - 試験編（71ページ+別表）
- 移行作業を試行する検証
  - データ移行調査および実践編（60ページ+別表）
  - アプリケーション移行実践編（25ページ+別表）
- DBMSに求められる要件整理
  - DB選定基準編（43ページ+別表）

本編のみでも500ページ越え



移行プロセス全体像

2018年度の活動

活動テーマ：異種DBMSからPostgreSQLへの移行

課題認識

- 過去の成果物は500ページもあり、初めて移行を検討する技術者にとってどれを見ていいのかわからない
- 過去の成果物からは移行作業の全体像がわかりづらい

活動目標

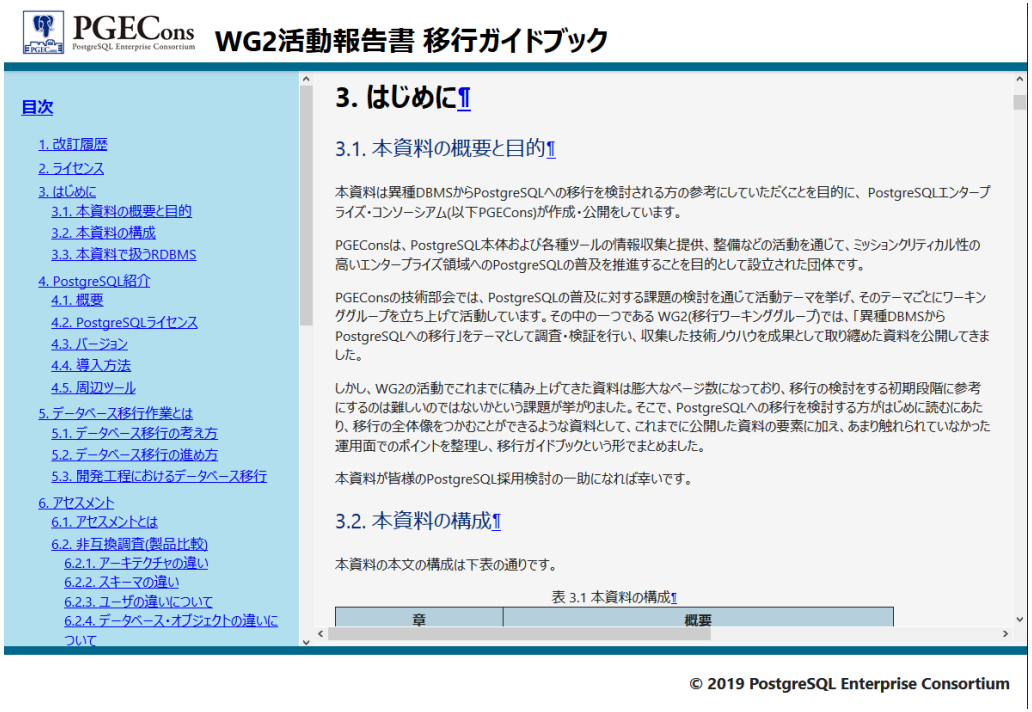
- PostgreSQLへの移行を検討している技術者が、簡単に移行の概要をつかめる入門書を用意しよう

取り組み

- 技術者視点で、DB移行時に知っておくべきことは何か検討
- **入門者向けの「移行ガイドブック」作成**

移行ガイドブック全体像

- 目次
  3. はじめに
  4. PostgreSQL紹介
  5. データベース移行作業とは
  6. アセスメント
  7. 移行作業
  8. 運用



2019年度活動報告

活動メンバー

- 2019年度は下記6社にて活動しました
  - NECソリューションイノベータ株式会社（主査）
  - 株式会社中電シーティーアイ NEW
  - 日本電子計算株式会社
  - 富士通株式会社
  - 富士通エフ・アイ・ピー株式会社
  - 三菱電機株式会社

2019年度 活動テーマ案

PostgreSQLの強化への対応

- ストアドプロシージャ
  - PostgreSQL 11であらためて実装された
    ストアドプロシージャに対応
- パラレルクエリ
  - パラレルクエリを中心に、PostgreSQLで動作する
    パラレル処理にフォーカス
- 宣言的パーティショニング
  - 2018年度 WG3成果物が存在する中で、
    どういう内容にするかまとまらず保留

2019年度 活動テーマ案

PostgreSQLの変更への対応

- メジャーバージョンアップ
  - PostgreSQL 10で変わっているところも多く、
    あらためてメジャーバージョンアップ時の
    非互換などの影響に着目

既存成果物の更新

- 移行ガイドブックの改訂
  - より実用性のある文書を目指して継続的に改善を進める

2019年度活動報告

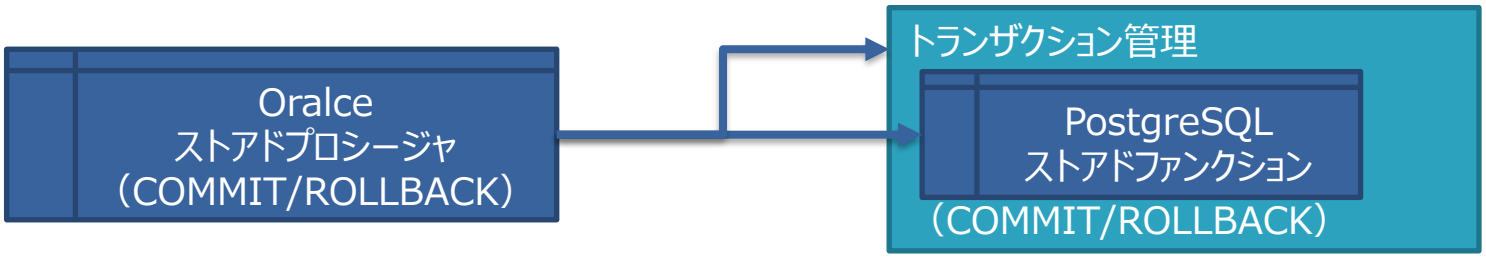
- ストアドプロシージャの移行
- パラレル処理の適用
- メジャーバージョンアップ
- 「移行ガイドブック」の改訂

ストアドプロシージャの移行

- PostgreSQL 11からストアドプロシージャ追加
- ストアドプロシージャの移行制限
- 移行制限対応策

PostgreSQL 11からストアドプロシージャ追加

- 「ストアドプロシージャ移行調査編」改訂
  - PostgreSQL 10以前
    - ファンクションとして移行。トランザクション管理は不可
    - 移行調査編は手続き言語としての移行方法を説明



  - 改訂版：PostgreSQL 11版
    - プロシージャオブジェクトとして移行。制限も記載

ストアドプロシージャの移行制限

■ 処理パターン別の制限

制限＝全てのパターンが“単純に”移行できるわけではない

No	処理パターン	単純移行	補足
1	単純パターン （SQL文とCOMMIT,ROLLBACK)	○	
2	例外ハンドラのあるブロック内での COMMIT,ROLLBACK	×	**Oracle プロシージャ移行での頻出パターン** サブブロック化、ストアドのネスト可が必要
3	ストアドプロシージャのネスト	○	
4	ファンクションからのストアドプロシージャ （COMMITあり）の起動	×	ファンクション起動をストアドプロシージャ起動に変更すればよい
5	明示的なトランザクション内でのストアドプロシージャ（COMMITあり）の呼び出し	×	明示的なトランザクションをやめる、またはストアドプロシージャ内ではCOMMIT/ROLLBACKしない
6	自律型トランザクションが使われている場合	×	PostgreSQLに機能なし。DBLINKなど代替え手段で対応

表のNo 2,6について対応策を説明

移行制限対応策

- 例外ハンドラのあるブロック内COMMIT,ROLLBACK
  - Oracleでよく見るパターン

    BEGIN～EXCEPTION間でCOMMITは記述できない

    ＝そのままでは移行できない

Oracle

```
CREATE PROCEDURE プロシージャ名　（引数）
IS
　～
BEGIN
　業務処理
　～
　COMMIT;
EXCEPTION
WHEN　XXXXXX THEN
　ROLLBACK;
　エラー処理
　COMMIT;
END プロシージャ名;
```

PostgreSQL

```
CREATE PROCEDURE プロシージャ名　（引数）
LANGUAGE plpgsql AS $$
DECLARE
　～
BEGIN
　業務処理
　～
　COMMIT;
EXCEPTION
WHEN　XXXXXX THEN
　ROLLBACK;
　エラー処理
　COMMIT ;
END;
$$;
```

NG
BEGIN～EXCEPTION間に
COMMITは記述できないので
EXCEPTIONに飛ぶ

移行制限対応策

- 例外ハンドラのあるブロック内COMMIT,ROLLBACK
  - 対応例1

    BEGIN～EXCEPTION～ENDをサブブロックとし、COMMITはサブブロックの外に記述

PostgreSQL

```
CREATE PROCEDURE プロシージャ名（引数）
LANGUAGE plpgsql AS $$
DECLARE
    ～
BEGIN
    BEGIN
        業務処理
        ～
    EXCEPTION
        WHEN  XXXXXX THEN
        エラー処理
        COMMIT;
    END;
    COMMIT;
END;
$$;
```

サブブロック化

移行制限対応策

■ 例外ハンドラのあるブロック内COMMIT,ROLLBACK

□ 対応例2

BEGIN～EXCEPTION～ENDを子プロシージャとしCALLする

子プロシージャで業務処理を部品化することで保守性も上がる

PostgreSQL

```
CREATE PROCEDURE メインプロシージャ名（引数）
LANGUAGE plpgsql AS $$
DECLARE
   ～
BEGIN
   CALL 子プロシージャ１(引数)
    IF c_RCD = 0 THEN
       CALL 子プロシージャ２(引数)
    END;
    ～
   IF c_RCD = 0 THEN
      COMMIT;
    ELSE
       ROLLBACK;
       CALL エラー出力（ IN c_RCD INT,IN c_MSG TEXT ）
END;
$$;
```

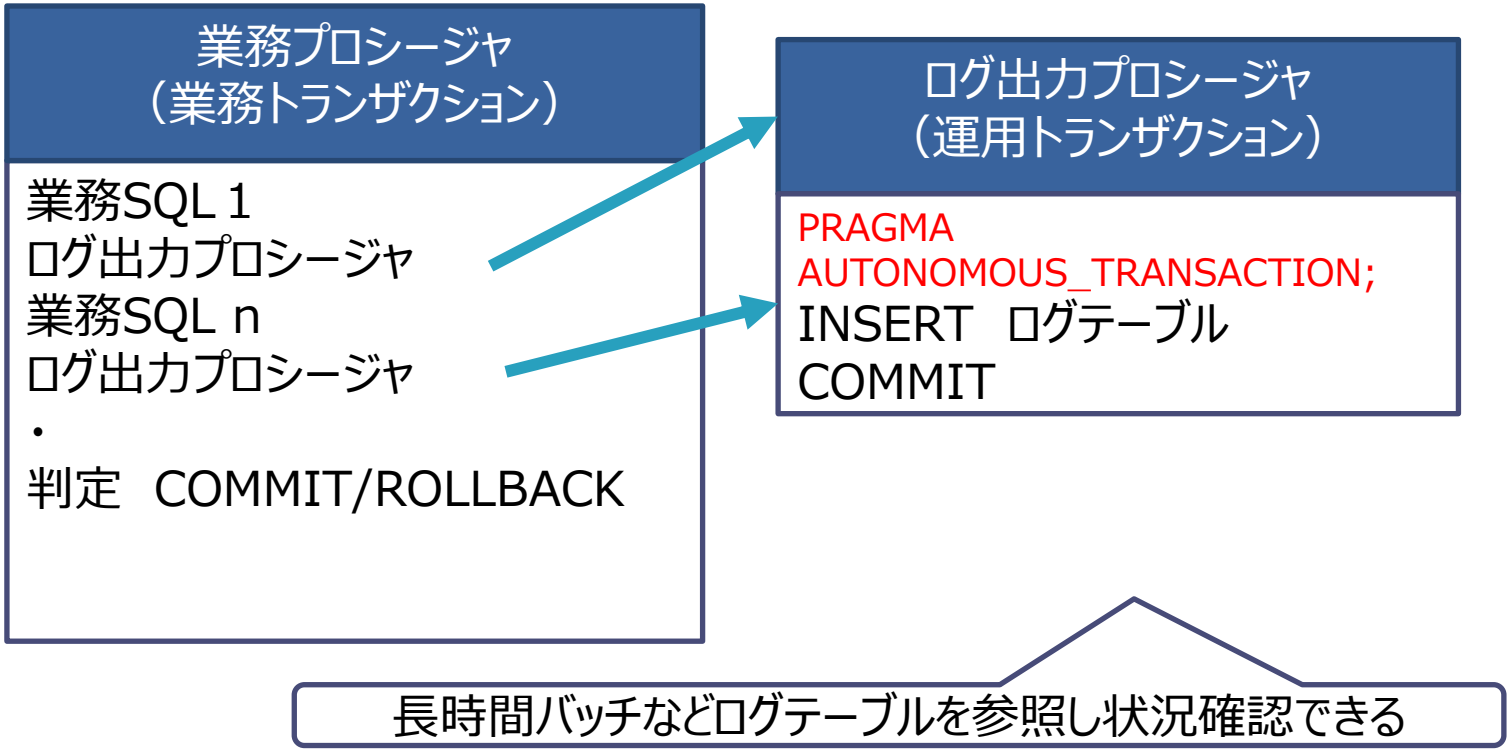
```
CREATE PROCEDURE 子プロシージャ１
 （～,INOUT c_RCD INT,INOUT c_MSG TEXT）
```

```
CREATE PROCEDURE 子プロシージャ２
 （ ～,INOUT c_RCD INT,INOUT c_MSG TEXT）
LANGUAGE plpgsql AS $$
DECLARE
   ～
BEGIN
   業務処理２
   ～
   c_RCD  = 0;
EXCEPTION
   WHEN  XXXXXX THEN
   エラー処理
   c_RCD  = 9999;
   c_MSG = 'XXXXXXXXX';
END;
$$;
```

メインプロシージャはコードが短く、かつ何をしている処理か理解しやすくなる

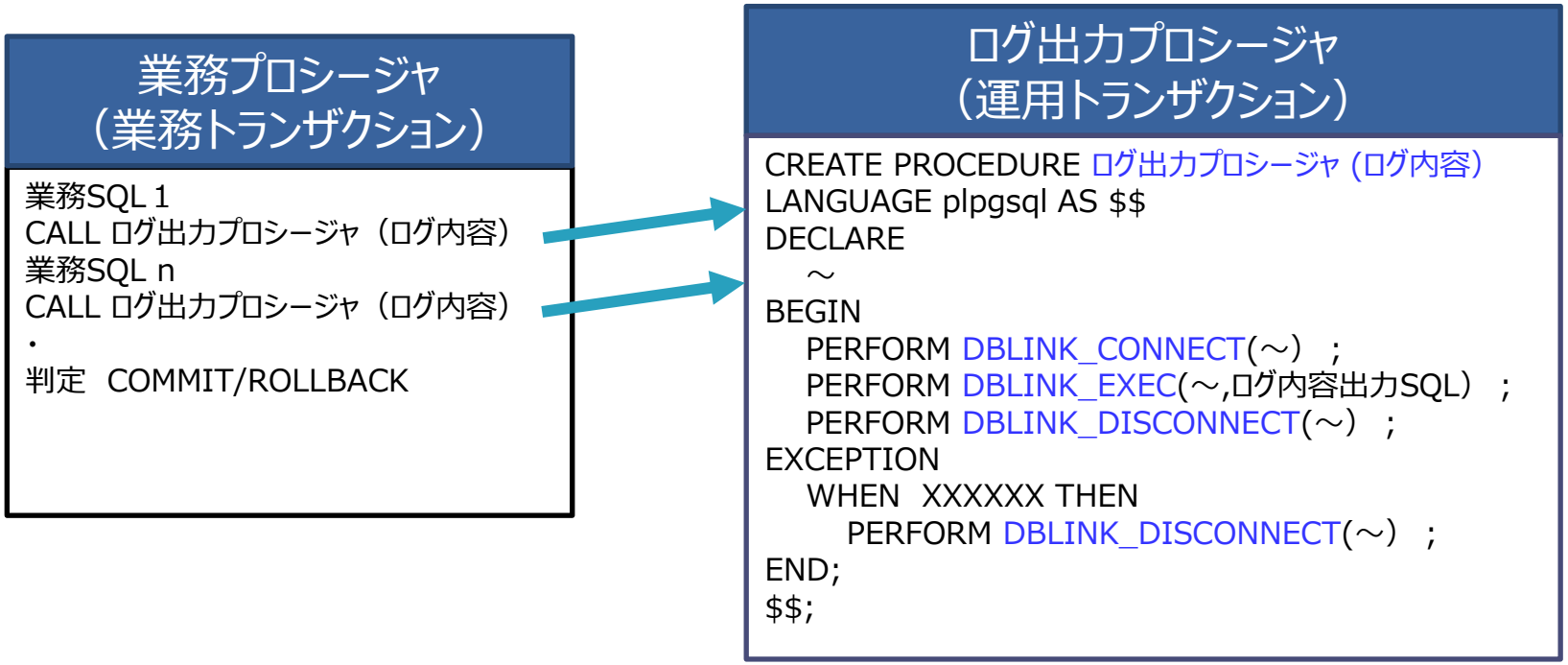
移行制限対応策

- 自律型トランザクションが使われいる場合
  - 自律型トランザクションのユースケース例
    - 業務としてのトランザクションとログを出力する運用トランザクションを分ける。運用トランザクションは業務トランザクションに依存しない



移行制限対応策

- 自律型トランザクションが使われいる場合
  - 代替案
    - DBLINKを使うことで業務トランザクションと運用トランザクションを分離できる。データベースを分ける必要はない。



まとめ

- ストアドプロシージャ移行（PotgreSQL 11以降）
  - トランザクション管理ができるようになった
    - 移行がシンプルになる

      Oracleプロシージャ → PostgresSQL プロシージャ

  - 制限有のパターンもある
    - ロジック変更を伴う対応が必要

2019年度活動報告

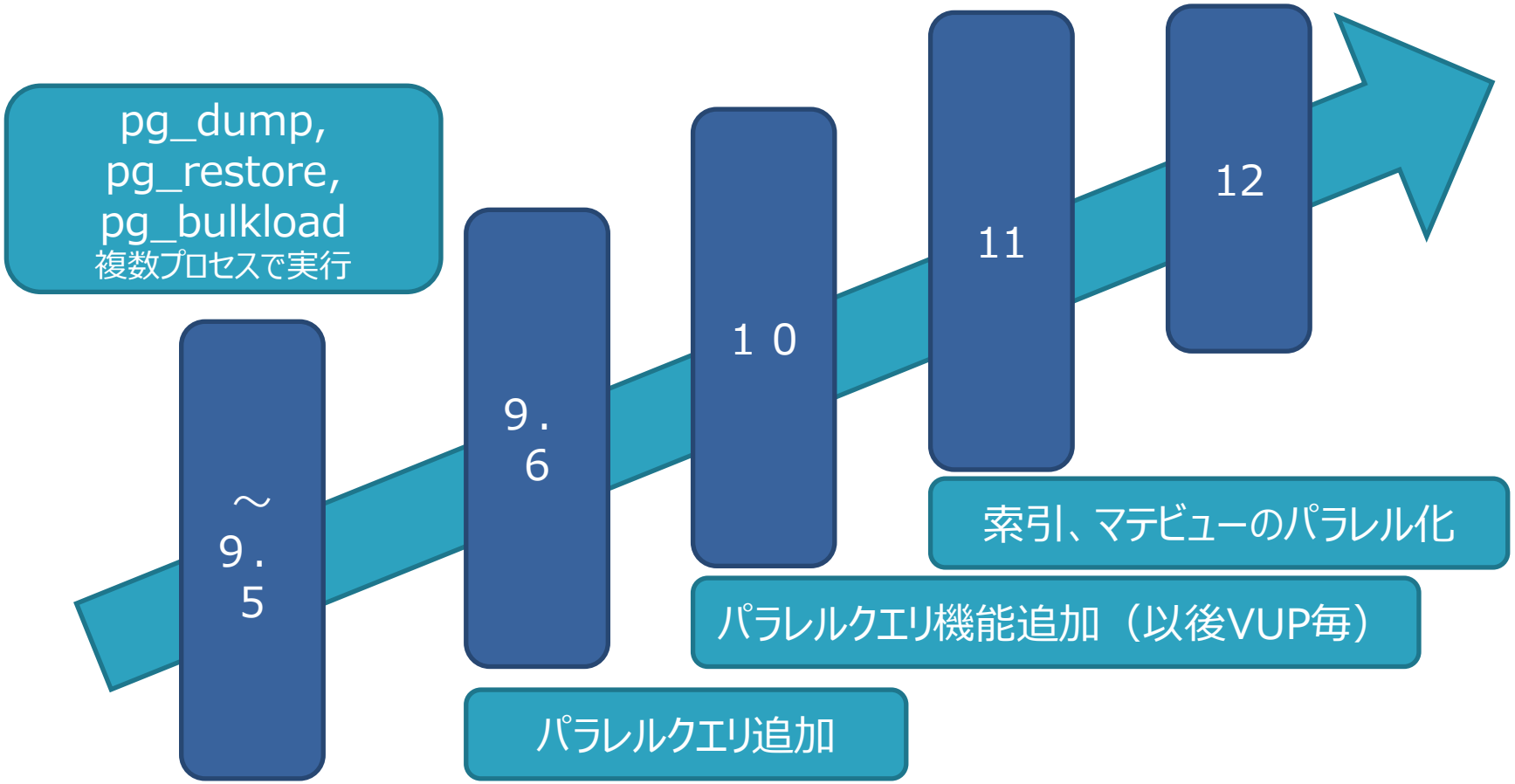
- ストアドプロシージャの移行
- パラレル処理の適用
- メジャーバージョンアップ
- 「移行ガイドブック」の改訂

パラレル処理の適用

- PostgreSQLのパラレル処理
- パラレル処理の指定
- パラレル処理活用時に注意する点
- まとめ

PostgreSQLのパラレル処理

- PostgreSQLのパラレル処理の歴史
  - パラレル処理：1つの処理を複数のプロセスで分割実行



PostgreSQLのパラレル処理

■ バージョン毎の詳細①

機能分類	機能	PostgreSQLのバージョン				
		9.5以前	9.6	10	11	12
クエリ	Seq Scan	–	○	○	○	○
	Hash Join	–	○	○	○	○
	NL Join	–	○	○	○	○
	集約（Aggregation）	–	○	○	○	○
	ヒント句（pg_hint_plan）	–	○	○	○	○
	Index Scan(B-tree)	–	–	○	○	○
	Merge Join	–	–	○	○	○
	Bitmap Heap Scan	–	–	○	○	○
	サブクエリ	–	–	○	○	○
	ソート	–	–	○	○	○
	Parallel Hash Join	–	–	–	○	○
	パラレルアペンド	–	–	–	○	○
	SELECT ～ INTO ～	–	–	–	○	○

PostgreSQLのパラレル処理

■ バージョン毎の詳細②

機能分類	機能	PostgreSQLのバージョン				
		9.5以前	9.6	10	11	12
クエリ	トラン分離レベルSERIALIZABLE対応	－	－	－	－	○
	外部表（FILE_FDW）	－	－	－	－	－
DML	INSERT,UPDATE,DELETE	－	－	－	－	－
DDL	CREATE TABLE ～ AS SELECT	－	－	－	○	○
	CREATE MATERIALIZED VIEW	－	－	－	○	○
	CREATE INDEX (B-tree)	－	－	－	○	○
	REINDEX	－	－	－	○	○
その他	統計取得（ANALYZE）	－	－	－	－	－
	バックアップ（pg_basebackup, pg_rman）	－	－	－	－	－
	論理バックアップ・リストア	○	○	○	○	○
	データロード（pg_bulkload）	○	○	○	○	○

パラレル処理の指定

- どのように指定すればよいか？

以下の順で指定することで、パラレル化の管理コストを下げる

1. 自動でパラレル化する
   - 各種パラメータ、オブジェクトでの動作指定
     → postgresql.conf,テーブル定義など
   - ユーザ毎の指定
     → ロール定義

2. 明示的にパラレル化する
   - コマンド、ツール等の個別のリファレンスに従い指定
   - クエリ中のヒント句での指定

パラレル処理の指定

- 自動でパラレル化する
  - 指定個所

指定個所	説明
postgresql.conf（パラメータ）	• パラレル度の制限 • 実行計画の調整要素を指定
表定義	set句でparallel_workersを指定
ロール定義	set句でpostgresql.confのパラメータを上書き※
セッション	set文でpostgresql.confのパラメータを上書き※

※：max_worker_processesなど上書きできないものもある

パラレル処理の指定

- 明示的にパラレル化する

  明示的に指定するケースはツールや処理毎にパラレル化を指定する場合になる

  以下は運用で使用頻度が高いと想定されるもの

  - 論理バックアップ・リストア

    例　test01データベースを4パラレルでバックアップする

    ```
    pg_dump -Fd -j 4 -f /home/postgres/backup/test01_d test01
    ```

  - ヒント句（pg_hint_plan）

    例　pgbench_accountsに対して4パラレルでクエリーを実行

    ```
    /*+ Parallel (pgbench_accounts 4) */
    select * from pgbench_accounts where XXXXXXXX;
    ```

パラレル処理活用時に注意する点

- **実行優先度がある**

パラレル指定は複数の個所に指定できる。複数指定した場合、基本の実行優先度は以下のようになる（PostgreSQL11.4での検証結果）

ヒント句 > セッション指定 > ロール指定 > postgresql.conf

表のパラレル度

上記2つの“より制限される”パラレル度が適用される

パラレル処理活用時に注意する点

- 期待通りにパラレル化されない場合もある

  以下のような場合は指定通りにパラレル化されません

  - 複数処理の同時実行によりパラメータの制限を超えた
  - 実行計画で非パラレルが早いと推定したとき
    （セッション指定、ロール指定、postgresql.conf）
  - 表のパラレル度未指定のとき表のサイズが小さい場合
    （表サイズに依存したパラレル度（仕様）がある）

パラレル処理活用時に注意する点

- パラレル処理に向く処理、向かない処理
  - パラレル処理に向く処理
    - 長時間のバッチ処理
    - 運用保守の処理

      論理バックアップ、索引の再構築など
  - パラレル処理に向かない処理
    - オンライン処理

      同時に多数のユーザが接続されるような処理はパラレル化に必要なバックグランドプロセスが確保できず期待通りにパラレル化されない

まとめ

■ パラレル処理の適用方針案

DB規模	前提条件	指針	考え方
小	• 性能要件の規定なし • CPUコアは4以下 • DBサイズは50GB以下	指針なし （パラメータはデフォルトのまま）	管理コストを掛けない （管理しない）
中	• 性能要件の規定あり • CPUコアは32以下 • DBサイズは50GB～1TB	パラレル化方式を規定	優先度の高い処理の性能確保
大	• 性能要件の規定あり • CPUコアは32以上 • DBサイズは1TB以上	パラレル化方式を規定＋パーティショニングなど含め性能確保を検討	• 優先度の高い処理の性能確保 • パラレル化は一つのチューニング要素として考える

考え方の例であり、中規模でも数百GBのテーブルがある場合など
パーティショニング含め検討するなど柔軟に適用する

2019年度活動報告

- ストアドプロシージャの移行
- パラレル処理の適用
- メジャーバージョンアップ
- 「移行ガイドブック」の改訂

メジャーバージョンアップ

■ 活動概要

PostgreSQL 9.4が2020年2月に最終リリースを迎えました。これを機に、PostgreSQL 9.4からのメジャーバージョンアップを検討、実施するケースも多くなると考えられます。

そこで、今回はPostgreSQL 12へのメジャーバージョンアップをモデルケースとし、PostgreSQLのメジャーバージョンアップ手順、考慮点などについて調査し、簡易的な実機検証をおこないました。

メジャーバージョン	最終リリース
12	2024年11月14日
11	2023年11月9日
10	2022年11月10日
9.6	2021年11月11日
9.5	2021年2月11日
9.4	2020年2月13日

■ 既存成果物との関係

PGECons WG2の成果物にはすでに、『バージョンアップ編』というものがあります。これは、バージョンアップ時に利用するツールの紹介や、その利用方法に主眼をおき、それらツールの実機検証をおこない、まとめたものです。

これに対し、今回作成した『メジャー・バージョンアップ編』は、メジャー・バージョンアップ作業そのものに主眼をおいてまとめました。

メジャーバージョンアップ

■ コンテンツ

メジャーバージョンアップ編は、下表の構成としています。

章	概要
メジャーバージョンアップ手順	PostgreSQLのメジャーバージョンアップ手順について • メジャーバージョンアップをおこなう際の、工程概要を記述
メジャーバージョン間の非互換	メジャーバージョン間の主な非互換について • メジャーバージョン間の非互換について、主要なものを抜粋して記述
メジャーバージョンアップ検証	メジャーバージョンアップを実機検証した結果について • 実際にメジャーバージョンアップを実施した結果を、ターミナルログを主体に記述

メジャーバージョンアップ

■ メジャーバージョンアップ手順

この章では、メジャーバージョンアップの基本的な流れについて記述しました。

PostgreSQLのメジャーバージョンアップは、バイナリを更新するだけで完了するマイナーバージョンアップとは異なり、データベースクラスタの移行が必要です。そのため、その移行をどのように行うか、なにを検証しなければならないのか、を考えなければなりません。

この章では、

- ✓ 要件確認
- ✓ 非互換確認
- ✓ 移行手順

として、各フェーズで考えるポイントを概略レベルで記載しました。

移行手順では、

- pg_upgradeを利用する場合
- pg_dumpallを利用する場合
- pg_dump/pg_restoreを利用する場合

と、3パターンについて、概略レベルでまとめています。

メジャーバージョンアップ

■ メジャーバージョン間の非互換

この章では、PostgreSQL 9.4からPostgreSQL 12へのメジャーバージョンアップ時に把握しておくべき差異や非互換について、記述しました。

非互換といっても、ちょっと調べればわかるものから、リリースノートを読み込まなければわからないものまで、様々です。それらすべてを列挙することは紙面の都合上無理ですので、ここでは代表的な非互換を挙げるにとどめています。

より細かい、とくにアプリケーションレイヤーでの非互換については、リリースノートを熟読するようにしてください。

またこの章では、PostgreSQLサーバーの非互換だけではなく、代表的な機能拡張についても、触れています。それぞれの機能拡張のバージョンが、どのメジャーバージョンに対応しているのか、といった一覧表を用意しました。メジャーバージョンアップによって、機能拡張のバージョンをどうすればよいのか？　といった場合にお役に立てると思います。

メジャーバージョンアップ

■ メジャーバージョン間の非互換

ここでまとめた非互換は、下記のとおりです。

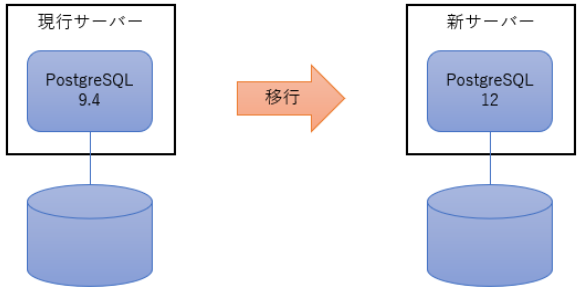
- サーバープロセス
- データベースクラスタを構成するディレクトリやファイル
- サーバーパラメータ
- システムカタログ
- システムビュー
- バックアップ・リカバリや監視などの非機能要件
- 主な拡張機能
  - pg_dbms_stats
  - pgBadger
  - pg_hint_plan
  - pg_monz
  - pg_repack
  - pg_rman
  - pg_statsinfo
  - pg_bulkload

メジャーバージョンアップ

■ メジャーバージョンアップ検証

この章では、実機でメジャーバージョンアップを実際に行った結果について記述しました。

モデルケースは、PostgreSQL 9.4が稼働する現行サーバーから、PostgreSQL 12をインストールした新サーバーへ移行する、という別サーバーへ移行するものです。



移行の手法として、今回は2パターンで実施しました。

1. pg_dump/pg_restore を利用するパターン
2. pg_upgradeを利用するパターン

検証結果は、これから初めてメジャーバージョンアップを行う方でも、イメージしやすいよう、実際のターミナルログを主体にまとめています。

■ 留意事項

小規模な検証データでの移行ですので、実行時間はあくまで目安となります。また、実際にはおこなうはずの、アプリケーション目線での移行後検証などは省略しています。

2019年度活動報告

- ストアドプロシージャの移行
- パラレル処理の適用
- メジャーバージョンアップ
- 「移行ガイドブック」の改訂

「移行ガイドブック」の改訂

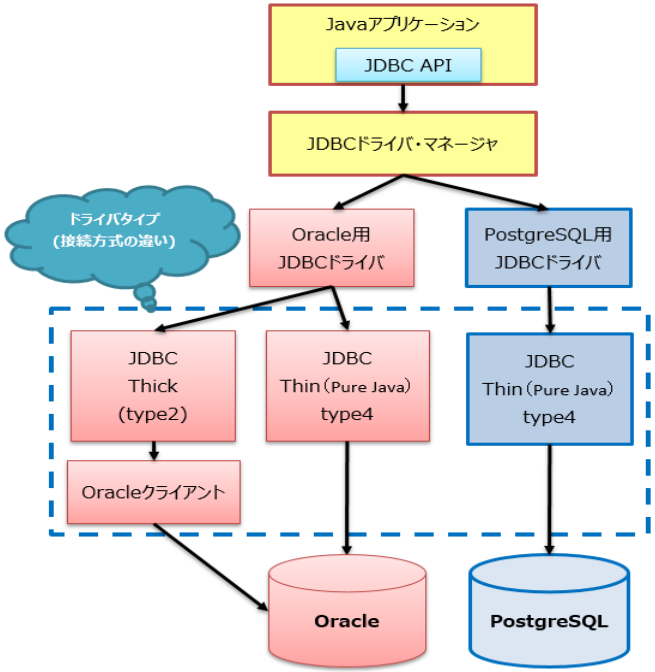
- 移行ガイドブックとは　<2018年度成果物>
  - PostgreSQLへ移行するための全体感がわかるコンパクトなガイドブックを作り、移行検討のバイブルのような存在になることを目的に作成した。
  - 移行に必要な考え方、移行の検討や手順、運用に至るまでを網羅的に取り上げている。
  - 移行に関わる人たち(DB/APエンジニア、マネジャー)にも有益な情報やノウハウをまとめている。
  - これまでの活動成果を集約し、技術情報の最新化を図っているので、このガイドブックを足掛かりに関連する過去の成果物を活用してほしい。
- 今後の改訂方針
  - 有効性から実用性へ
    - ガイドブックは、移行時に役立つ(有効性)はあるものの、実際に使える(実用性)とまでには至っていない部分があるため、内容を掘り下げ肉付けしていく。
  - 活動成果との連動
    - 今後の活動成果は、ガイドブックに反映していく。

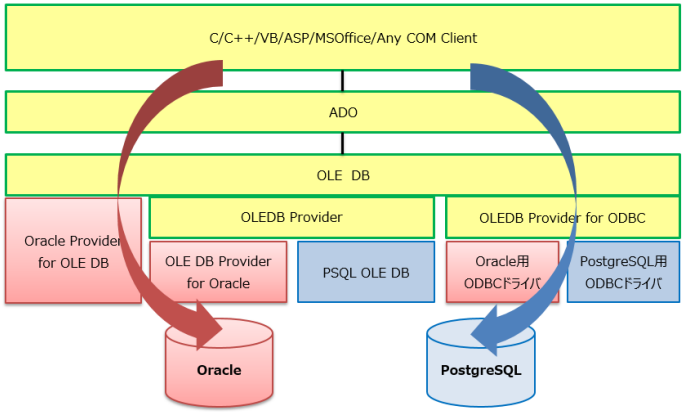
「移行ガイドブック」の改訂

- アプリケーション移行
  - 現状評価
    - アプリケーションは様々な形態があり、総論的な内容となっている
    - DBライブラリなどの具体的な影響点がないので、実用性に欠けている
  - 改訂方針
    - 標準規格インタフェースのJDBC、ODBCを取り上げる
    - それぞれで影響する箇所や対処方法の具体例を記述する

「移行ガイドブック」の改訂

- 標準規格インタフェースの対応
  - JDBC系
    - 変更範囲
    - 変更例
      例）java.sql.DriverManager
      1. JDBCドライバのロード
      2. 接続文字列
      3. エラー判定
      4. 例外ハンドリング
  - ODBC系 (Windows)
    - 変更範囲
    - 変更例
      例）ODBC Provider
      OLEDB Data Provider





おわりに

成果物の公開

- 成果物は無償でダウンロード可能
  - https://pgecons-sec-tech.github.io/tech-report/

移行WG（WG2）

「異種DBMSからPostgreSQLへの移行」をテーマとして調査・検証を行い、収集した技術ノウハウを成果として取り纏めた資料を公開しています。

成果物一覧

文書名	概要	リンク(HTML・PDF版)	リンク(圧縮版)	最終更新日
移行ガイドブック	これからPostgreSQLへの移行を検討される方の一助となる、DBMS移行の概要をつかめるガイドブックです。	• 本文(HTML) • 本文(PDF)		2019/05/20
異種DBMSからPostgreSQLへの移行ガイド	各成果物の活用場面をイメージして頂くために、一般的なシステム移行手順を提示した上で、各タスクとWG2成果物の関係を表現しています。	• 本文(HTML) • 本文(PDF)		2017/06/22
DB移行フレームワーク編	異種DBMSからの移行とは具体的に何を行うのかを紹介します。 DBMSの移行作業において一般的に発生すると考えられる作業工程を定義し、各工程における検討結果をベースとして移行可否判断の手がかりとなる情報を提供します。	• 本文(PDF)	• 2013年度WG2成果物(zip) • 2013年度WG2成果物(tar.bz2)	2014/04/16
システム構成調査編	DBMSの代表的なシステム構成とその特徴を挙げ、PostgreSQL移行時に採用可能な構成を紹介します。	• 本文(PDF)	• 2013年度WG2成果物(zip) • 2013年度WG2成果物(tar.bz2)	2014/04/16
異種DB間連携調査編	異種 DBMSで稼動する既存システムとの連携を想定し、異種DBMSとPostgreSQL の連携について、実現方法や移行前	• 本文(PDF)	• 2013年度WG2成果物(zip)	2014/04/16

2019年度活動を振り返って

更新中

- 成果

- 反省

今後の活動について

更新中

- 過去成果物の最新バージョン対応
  - メンテナンスが必要なドキュメントは継続して更新、公開したい
- PostgreSQLの新機能の移行観点での評価
  - 宣言的パーティショニングは今回保留としたが、メジャーバージョンアップにより継続的に改善が入っている

移行で検討してほしいテーマがあれば
アンケートに記入をお願いします。

PGECons

PostgreSQL Enterprise Consortium