PGECons
PostgreSQL Enterprise Consortium

PostgreSQLのパーティショニング運用

2022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 ～
- 2022年度活動報告
  - 既存資料の更新に向けた調査状況
  - パーティショニングしたテーブルの移行と運用
- 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

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
PostgreSQL自習書	Oracleデータベース経験者がPostgreSQLの概要を理解することを目的とした技術文書です。	• 本文(PDF)		2021/05/25 NEW!
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)		2020/09/02
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

2022年度活動報告

活動メンバー

- 2022年度は下記6社にて活動しました
  （五十音順）
  - 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
  - 日本電子計算株式会社
  - 富士通株式会社
  - 富士通Japan株式会社
  - 三菱電機株式会社

2022年度 活動テーマ案

継続テーマ

- 既存資料の更新
  - 移行に関わる機能強化の反映
- 移行ツールの調査
  - Ora2Pgを中心とした移行ツールの調査・検証

新規テーマ

- パーティションの移行調査
  - 宣言的パーティショニングによる移行・運用

2022年度活動報告

- 既存資料の更新に向けた調査状況

PostgreSQLの進化

■ 近年の移行に関連する機能強化

PostgreSQL バージョン	機能強化内容
11	• ストアドプロシージャのサポート (トランザクション制御が可能に) • ECPGにOracle互換モード追加
12	• 生成列のサポート
14	• ストアドプロシージャのOUTパラメータのサポート • SEARCH/CYCLE句のサポート
15	• MERGE構文のサポート • 正規表現関数の追加 (regexp_count、regexp_instr、regexp_substr)

MERGE構文（PostgreSQL 15）

■ 構文の主な差異

- OracleはUPDATE/DELETEを1つのMATCHEDに記述し、UPDATEの結果に対してDELETEを実行するが、PostgreSQLは異なる分岐処理として実行される

- 条件指定をOracleはWHERE句に記述するのに対し、PostgreSQLは[NOT] MATCHEDの後ろにANDで記述する

- PostgreSQLは更新列の修飾不可 (エラーになる)

Oracle

```
MERGE INTO t1
USING t2 ON (t1.id = t2.id)
WHEN MATCHED THEN UPDATE SET t1.value = t1.value + t2.value
                 DELETE WHERE t1.value > 1000
WHEN NOT MATCHED THEN INSERT (id, value) VALUES (t2.id, t2.value);
```

PostgreSQL

```
MERGE INTO t1
USING t2 ON (t1.id = t2.id)
WHEN MATCHED AND t1.value > 1000 THEN DELETE
WHEN MATCHED THEN UPDATE SET value = t1.value + t2.value
WHEN NOT MATCHED THEN INSERT (id, value) VALUES (t2.id, t2.value);
```

MERGE構文（PostgreSQL 15）

- PostgreSQL固有機能
  - [NOT] MATCHEDの記述は複数可能、
    上位から条件判定されマッチしたものが選択される
  - WITH句が使用可能
  - DO NOTHING句：なにもしない
  - ON句に指定した条件に含まれる列も更新可能

- INSERT ON CONFLICTとの差異
  - DELETE処理が可能
  - 並行するトランザクション処理と競合し、
    重複キーエラーが発生する可能性あり

2022年度活動報告

- パーティショニングしたテーブルの移行と運用

パーティショニングしたテーブルの移行と運用

- テーブルのパーティショニングのメリット
- Oracle パーティション表の移行
- パーティショニングしたテーブルの運用
- 2022年度 WG2成果物

テーブルのパーティショニングのメリット

- 大規模表の課題
  - データ量が増えると検索が遅くなる
  - VACUUM、索引再構築など保守作業に時間がかかる
- 大規模表への対応

テーブルのパーティショニング

パーティション単位の検索による性能向上

パーティション単位の保守が可能（保守性向上）

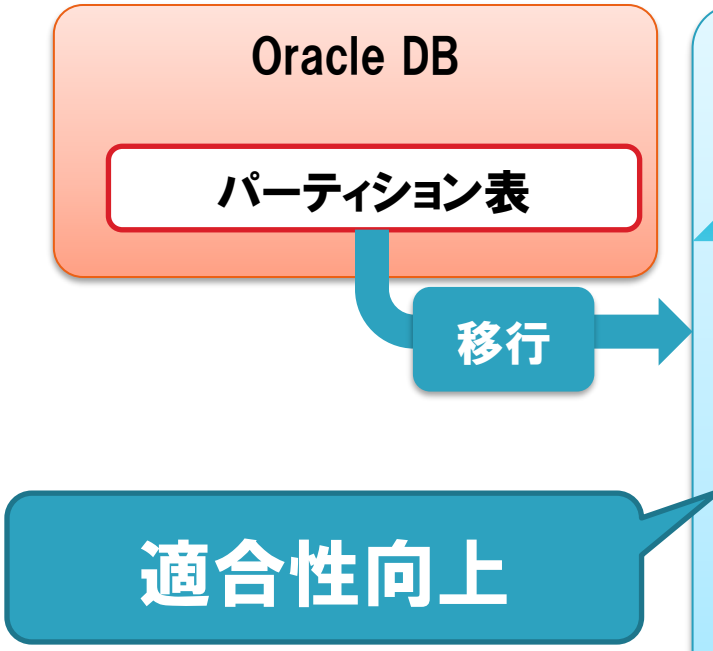
テーブルのパーティショニングのメリット

- メリット
  - パーティションキーに従う、SQLの性能向上の効果が特に高い。
  - パーティションキーに従う時系列テーブルでの古いデータの整理、削除はパーティションの切り離し(DETACH)、DROPとなるので性能向上しリソース負荷を掛けずに処理ができる。
  - 追加、更新、削除の集中する特定のパーティションのみ保守することができる。(VACUUM、ANALYZE、索引再構築)

- デメリット
  - パーティションキーが有効とならない場合はパーティション分割がオーバーヘッドとなり、非パーティションテーブルより性能が劣化するケースがある。

Oracle パーティション表の移行

- 移行のポイント
  - 性能の維持
  - 運用保守性の維持



PostgreSQL

宣言的パーティショニング	
15	性能改善、CLUSTERコマンド対応他
14	更新・削除時のパーティションプルーニング運用性向上他
13	パーティション結合の改善他
12	多数パーティションでの性能改善他
11	パーティションキーを跨ぐ更新 ハッシュパーティション 索引自動生成
10	宣言的パーティショニング追加

Oracle パーティション表の移行

■ パーティショニングの機能差

□ 機能比較

機能		Oracle	PostgreSQL	備考
パーティション	レンジパーティション	○	○	範囲パーティションと同意
	リストパーティション	○	○	Oracle：複数カラム PostgreSQL：1カラム
	ハッシュパーティション	○	○	
	コンポジットパーティション	○	○	サブパーティションと同意
索引	グローバル索引	○	ー	表パーティションに依存しない索引
	ローカル索引	○	○	表パーティション単位の索引
他	参照パーティション	○	ー	外部参照するパーティションと同じ単位でパーティション化
	読取り専用パーティション	○	ー	パーティション単位に読取り専用指定

上記の機能はOracle12cR2以降、PostgreSQL 11以降

Oracle パーティション表の移行

- パーティショニングの運用保守の差
  - 保守面

機能	Oracle	PostgreSQL	備考
パーティション追加	○	○	新規パーティションの追加
パーティション削除	○	○	パーティション単位の削除
パーティション切捨て	○	○	パーティション単位のTRUNCATE
パーティション交換	○	○	パーティション表から、非パーティション表への変更またはその逆
インターバルパーティション	○	△ pg_partman	時間隔のレンジパーティションの自動追加
自動リストパーティション	○	ー	リストパーティションの自動追加
パーティションマージ	○	ー	複数のパーティションのマージ。レンジパーティションは隣接するパーティションが対象
パーティション分割	○	ー	1つのパーティションを複数に分割

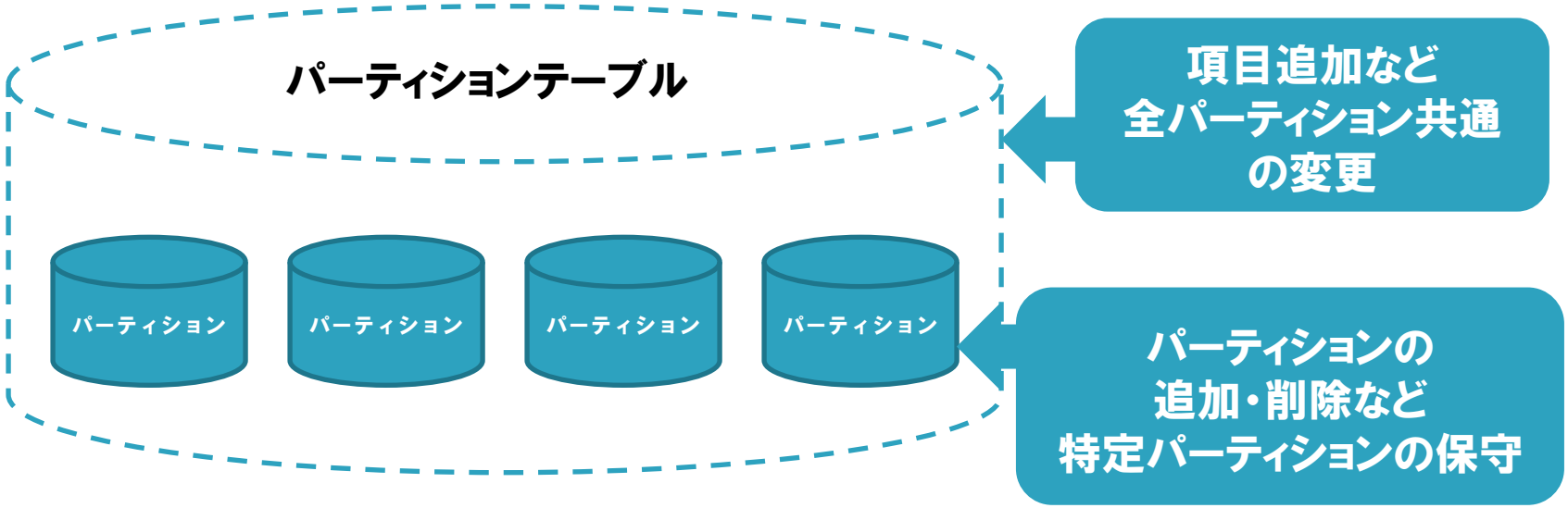
パーティショニングしたテーブルの運用

- **保守の対象**

  PostgreSQLのパーティショニングしたテーブル

  - パーティションテーブル：パーティションをまとめた仮想的な表
  - パーティション：1つ1つが物理的な表

  保守は上記のいずれか、または両方に対して実施



パーティショニングしたテーブルの運用

■ 保守一覧(1/3)

保守内容	保守対象		説明
	パーティションテーブル	パーティション	
パーティショニングしたテーブルの作成	○	○	①パーティションテーブル作成 ②①に対しパーティションを追加
パーティショニングしたテーブルの削除	○	ー	パーティションテーブル、パーティションの両方が削除される。
項目の追加、修正	○	ー	全てのパーティションに対して実施される。パーティション個別には実行できない。
パーティションの追加　方法1(CREATE TABLE)	ー	○	パーティションテーブルに対しパーティションを追加作成
パーティションの追加　方法2(ATTACH)	○	○	①パーティションの元のテーブルを作成 ②パーティションテーブルに①をパーティションとして組み込みを行う。

パーティショニングしたテーブルの運用

■ 保守一覧(1/3)

保守内容	保守対象		説明
	パーティションテーブル	パーティション	
パーティショニングしたテーブルの作成	○	○	①パーティションテーブル作成 ②①に対しパーティションを追加
パーティショニングしたテーブルの削除	○	ー	パーティションテーブル、パーティションの両方が削除される。
項目の追加、修正	○	ー	全てのパーティションに対して実施される。パーティション個別には実行できない。
パーティションの追加　方法1(CREATE TABLE)	ー	○	パーティションテーブルに対しパーティションを追加作成
パーティションの追加　方法2(ATTACH)	○	○	①パーティションの元のテーブルを作成 ②パーティションテーブルに①をパーティションとして組み込みを行う。

複数の方法のあるものは運用により、どちらかの方法を選択する

パーティショニングしたテーブルの運用

- パーティション追加の方法の選択
  - 方法１　保守時間を確保できる場合
    - メリット　　１つのSQL文でパーティション追加できる。
    - デメリット　　ロックによる業務影響が発生する。（下の例）

トランザクション１	トランザクション２	トランザクション３

test01=# begin;
BEGIN
test01=*# select count(*) from PART_T
where TEST_YMD = '2023-01-10';
count
------- 	
38
(1 row)		

トランザクション１が完了しないと業務が止まってしまう！	test01=# CREATE TABLE PART_T_P2023_02 PARTITION OF PART_T test01-# FOR VALUES FROM ('2023-02-01') to ('2023-03-01') with (FILLFACTOR = 80); **トランザクション１の完了待ち**	応答しない！
		test01=# select count(*) from PART_T where TEST_YMD = '2023-01-12 '; **トランザクション2の完了待ち**

パーティショニングしたテーブルの運用

- パーティション追加の方法の選択
  - 方法2　保守時間を確保できない場合
    - メリット　　ロックによる業務影響は発生しない。
    - デメリット　　2つのSQL文を発行する必要あり。

作成手順例

①パーティションの元のテーブルを作成

```
create table PART_T_P2023_03 (like PART_T including
defaults including constraints) with ( FILLFACTOR = 80);
```

②パーティションテーブルに①をパーティションとして組み込みを行う。

```
alter table PART_T attach partition PART_T_P2023_03
for values from ('2023-03-01') to ('2023-04-01' );
```

パーティショニングしたテーブルの運用

■ 保守一覧(2/3)

保守内容	保守対象		説明
	パーティションテーブル	パーティション	
パーティションの削除 (DROP)	ー	○	特定のパーティションを削除する。 (パーティション追加　方法1と同様に業務影響あり)
パーティションの切り離し (DETACH)	○	○	特定のパーティションをパーティションテーブルから切り離す。 (PG14以降:CONCURRENTRY句を利用すれば業務影響を最小限にできる)
パーティションの切り捨て (TRUNCATE)	○	○	パーティションのデータを切り捨てる。パーティションテーブルに実施した場合は全てのパーティションが対象。
テーブルのオプション等の変更 (FILLFACTOR,TABLESPACE)	△	○	FILLFACTORはパーティション単位に変更可能。表領域はパーティションテーブル、パーティションどちらも可能。パーティションテーブルに対して実施の場合は以後作成されるパーティションのデフォルトが変更される。

パーティショニングしたテーブルの運用

■ 保守一覧(2/3)

保守内容	保守対象		説明
	パーティションテーブル	パーティション	
パーティションの削除 (DROP)	ー	○	特定のパーティションを削除する。 (パーティション追加　方法1と同様に業務影響あり)
パーティションの切り離し (DETACH)	○	○	特定のパーティションをパーティションテーブルから切り離す。 (PG14以降:CONCURRENTLY句を利用すれば業務影響を最小限にできる)
パーティションの切り捨て (TRUNCATE)	○	○	パーティションのデータを切り捨てる。パーティションテーブルに実施した場合は全てのパーティションが対象。
テーブルのオプション等の変更 (FILLFACTOR,TABLESPACE)	△	○	FILLFACTORはパーティション単位に変更可能。表領域はパーティションテーブル、パーティションどちらも可能。パーティションテーブルに対して実施の場合は以後作成されるパーティションのデフォルトが変更される。

パーティショニングしたテーブルの運用

- パーティションの切り離し
  - CONCURRENTLY句を使用しない（使用できない）場合
    - DETACHに関してはロックにより業務影響を与えてしまう

トランザクション1	トランザクション2	トランザクション3

testO1=# begin;
BEGIN
testO1=*# select count(*) from PART_T
where TEST_YMD = '2023-02-10';
count
------- 	
0
(1 row)
	トランザクション1，3とは異なるパーティションの切り離し	

トランザクション1が完了しないと業務が止まってしまう！	testO1=# ALTER TABLE PART_T DETACH PARTITION PART_T_P2023_01 ; トランザクション1の完了待ち	応答しない！
		testO1=# select count(*) from PART_T where TEST_YMD = '2023-02-11 '; **トランザクション2の完了待ち**

パーティショニングしたテーブルの運用

■ 保守一覧(3/3)

保守内容	保守対象		説明
	パーティションテーブル	パーティション	
索引の追加　方法1 (業務影響あり)	○	ー	パーティションテーブルに対し索引追加するとすべてのパーティションに索引が追加。ただし追加処理中はパーティションテーブルに対し書き込みロックがかかる。CONCURRENTLY句は付与できない。
索引の追加　方法2 (業務影響小)	○	○	①パーティションテーブルに対し索引をONLY句を付加して追加 ②各パーティションでCONCURRENTLY句を付与して索引追加 ③②の索引を①の索引にATTACHする。
VACUUM FULL	○	○	パーティションテーブルに対し実行すれば管理される全てのパーティションが対象。 パーティションに対して実行すれば指定したパーティションのみが対象。
VACUUM			
ANALYZE			

2022年度WG2の成果物
『パーティショニングした表の移行と運用』

- テーブルのパーティショニングの概要
- テーブルのパーティショニングのメリットとデメリット
  - パーティショニングの効果のあるパターンとないパターン
- Oracle　パーティション表の移行
  - 移行において性能を維持できないケースもある
- パーティショニングしたテーブルの運用
  - 保守作業の業務影響を実例を含め詳細に説明

おわりに

今後の活動について

- 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
  - メジャーバージョンアップなど
- 利用者のニーズに応じた活動テーマとしたい

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。

ライセンス

文書の内容、表記に関する誤り、ご要望、感想等につきましては、PGEConsのサイトを通じてお寄せいただきますようお願いいたします。

- Linux は、Linus Torvalds 氏の日本およびその他の国における登録商標または商標です。
- Oracleは、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。
- PostgreSQLは、PostgreSQL Community Association of Canadaのカナダにおける登録商標およびその他の国における商標です。
- その他、本資料に記載されている社名及び商品名はそれぞれ各社が商標または登録商標として使用している場合があります。

PGECons

PostgreSQL Enterprise Consortium