PGECons
PostgreSQL Enterprise Consortium

PostgreSQL移行のその前に

2020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 ～
- 2020年度活動報告
  - アプリケーション移行検証
  - チューニング編アップデート
  - PostgreSQL自習書の作成
- 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)

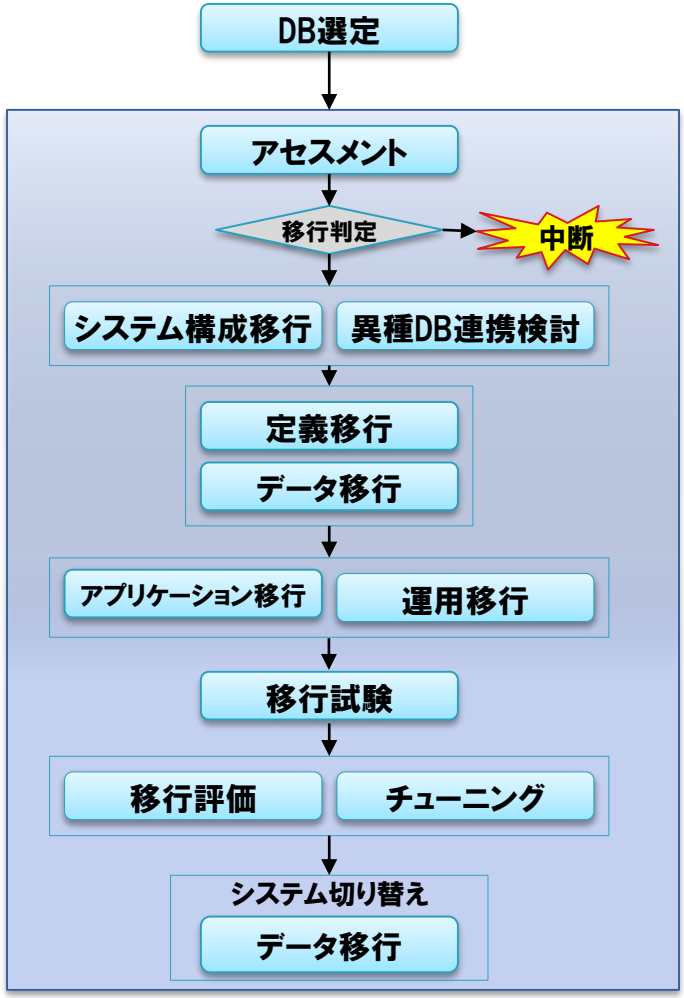
成果物

- 「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

2019年度 活動テーマ

- ストアドプロシージャ移行調査編
  - PostgreSQL 11であらためて実装された
    ストアドプロシージャへの対応
- パラレル処理移行編
  - 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
- メジャーバージョンアップ
  - 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
- 移行ガイドブックの改訂
  - アプリケーション移行観点での内容掘り下げ

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

2020年度活動報告

活動メンバー

- 2020年度は下記7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
  - 日本電気株式会社
  - 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

2020年度 活動テーマ

継続テーマ

- アプリケーション移行検証
  - 主要なアプリケーション・フレームワークやライブラリの移行検証と影響範囲の整理
- チューニング編アップデート
  - 2013年度公開文書をPostgreSQL 13ベースで更新

新規テーマ

- PostgreSQL自習書の作成
  - Oracle経験者向けに、PostgreSQLの概要を理解するための自習書を作成

2020年度活動報告

- アプリケーション移行検証
- チューニング編アップデート
- PostgreSQL自習書の作成

DBライブラリを利用したアプリケーションの移行検証

■目的

基幹システムから移行する際に、アプリケーションの影響範囲を特定するための調査プロセスに活用できる情報を提供すること。

■背景

・データベースの互換性に関する活動が多く、アプリケーション構成の互換性に言及する調査や検討が少なかった

・開発フレームワークやDBライブラリを利用したアプリケーションの影響範囲が知りたいという要望があった

■方針

現在主流なWebアプリケーション開発にフォーカスして移行検証を実施し、その結果からアプリケーションの影響範囲と修正箇所を整理し体系化する。

・現在主流なアプリケーション構成を選出する

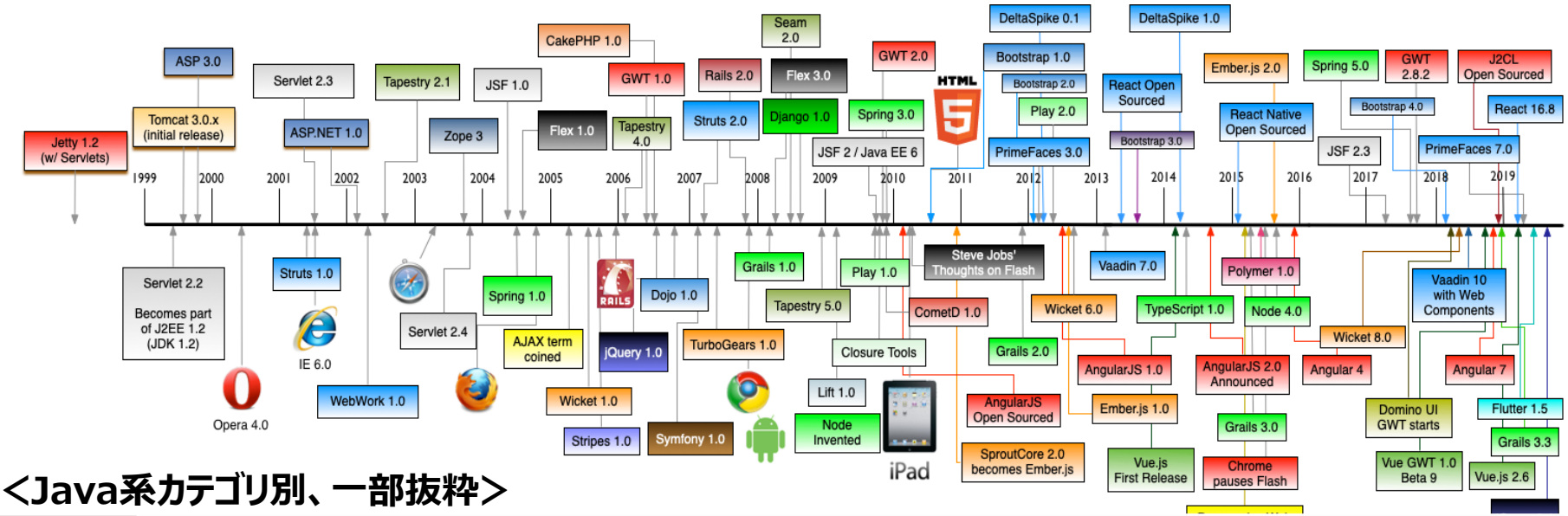
・各種ソフトウェアの組み合わせパターンを定めて移行検証する

・検証結果から共通点や相違点を整理し、影響範囲の特定する情報を示す

■アプリケーション開発構成パターン

①Webアプリケーション関連のソフトウェアの歴史の振り返り

引用：https://github.com/mraible/history-of-web-frameworks-timeline

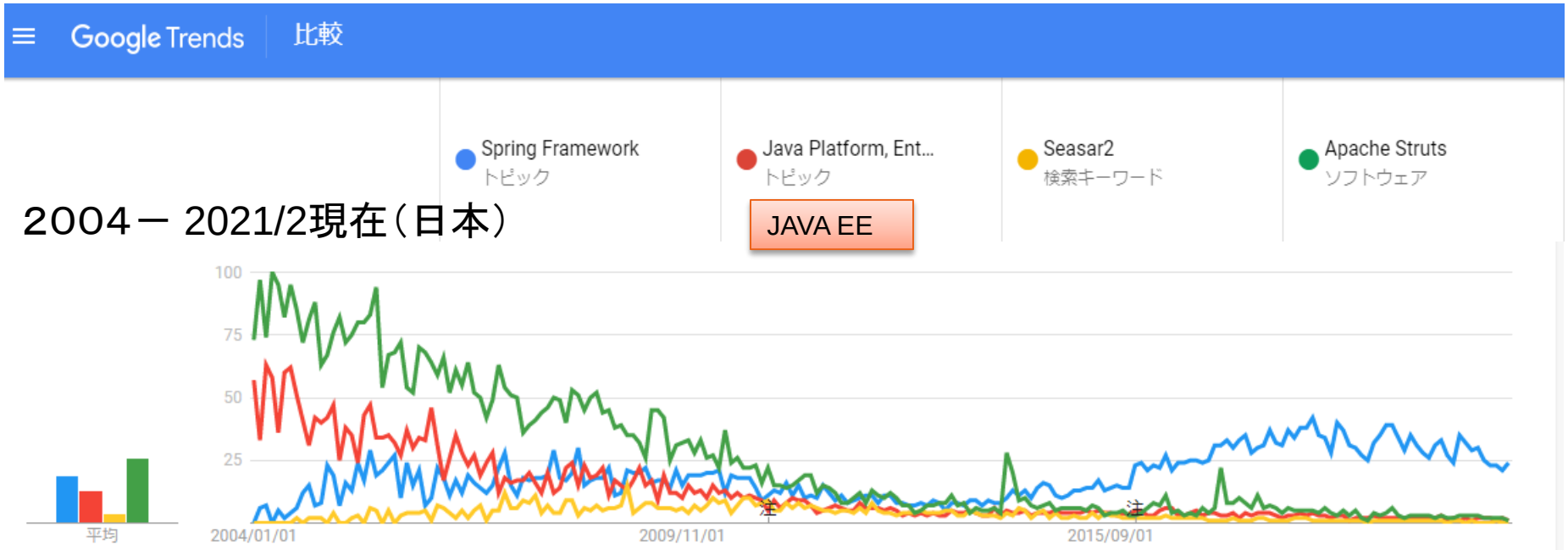


<Java系カテゴリ別、一部抜粋>

MVC	▲Struts 1.0		▲Struts 2.0			
		▲JSF[Sun]	▲Apache MyFaces			
DI x AOP		▲Seasar 2	▲SAStruts			
		▲Spring 1.0	▲Spring 2.0	▲Spring 3.0	▲Spring 4.0	▲Spring 5.0
O/R	▲Hibernate 1.0		▲iBATIS	▲MyBatis		
		▲S2DAO	▲S2JDBC			
JAVA EE	▲ J2EE 1.2	▲ J2EE 1.3	▲ J2EE 1.4	▲ JavaEE 5	▲ JavaEE 6	▲ JavaEE 7	▲ JavaEE 8

②検索ランキング（Google trendsでの比較）

Google Trends：https://trends.google.co.jp/trends/?geo=JP



Web系アプリケーションフレームワークの選定

プラットホーム	開発言語	名称	対象
ALL	Java	Spring Framework	●
		Java EE	
		SAStruts（Seasar2）	●
		Apache Struts	
Windows	C#	Microsoft .NET Framework	●
	Visual Basic (VB.NET/VBS)	Microsoft .NET Framework Active Server Pages(ASP)	●

DBライブラリ/ORMの選定

種類	タイプ	名称	対象
JDBC	JDBC	java.sql.DriverManager	●
ORM	JDBCラッパ	Spring JDBC	
	SQLマッパ	Mybatis（旧iBATIS）	●
		S2JDBC	●
	クエリビルダ	DBFlute	
	ORマッパ（JPA）	Hibernate	
		Spring Data JPA	
ADO	OLE DB Provider	for OLEDB	●
ADO.NET	.NET Framework Data Provider	for OLEDB for Oracle	
		for ODBC	●
	（Oracle専用）	Oracle Data Provider for .NET（ODP.NET）	●
	(PostgreSQL専用)	.NET Access to PostgreSQL（Npgsql）	●

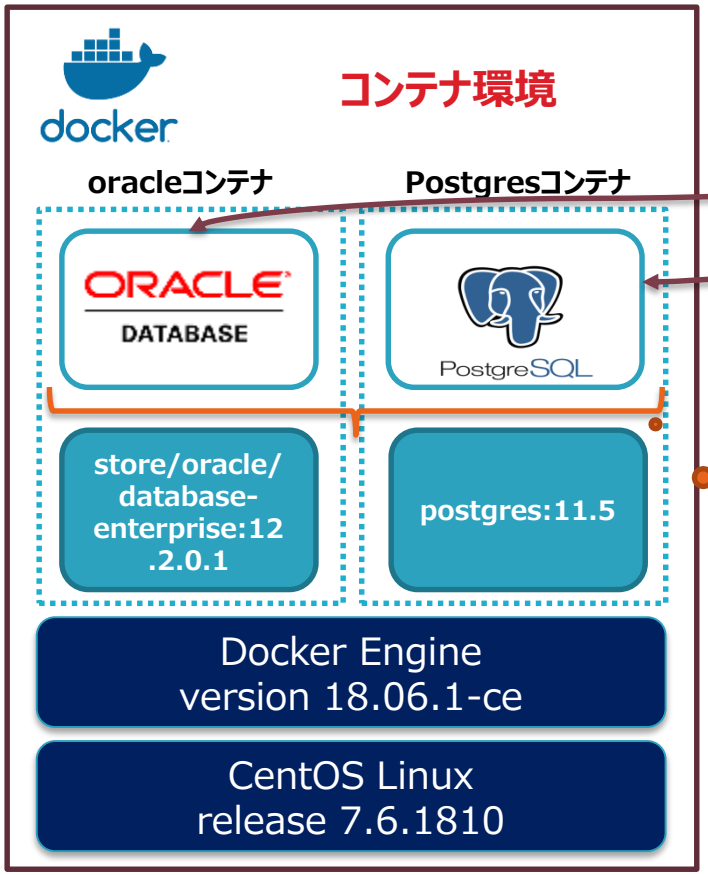
基幹系Webシステムで使われる主要なアプリケーションフレームワークとDBライブラリをメンバーの経験や知識、Google Trendsの検索結果に基づいて選定し、メンバーで検証可能な範囲で検証パターンとして選出しました。

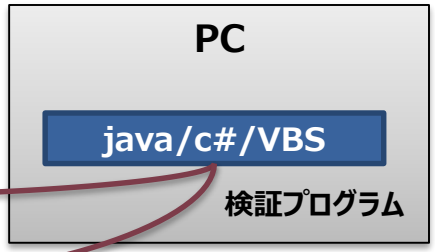
検証パターンの選出

項番	接続形式	言語	DBライブラリ
①	JDBC	Java	Mybatis（Spring Framework）
②			S2JDBC（SAStruts）
③			DriverManager
④	ODBC	VBS	OLE DB Provider
⑤		C#	.NET Framework Data Provider for ODBC
⑥			Oracle Data Provider for .NET

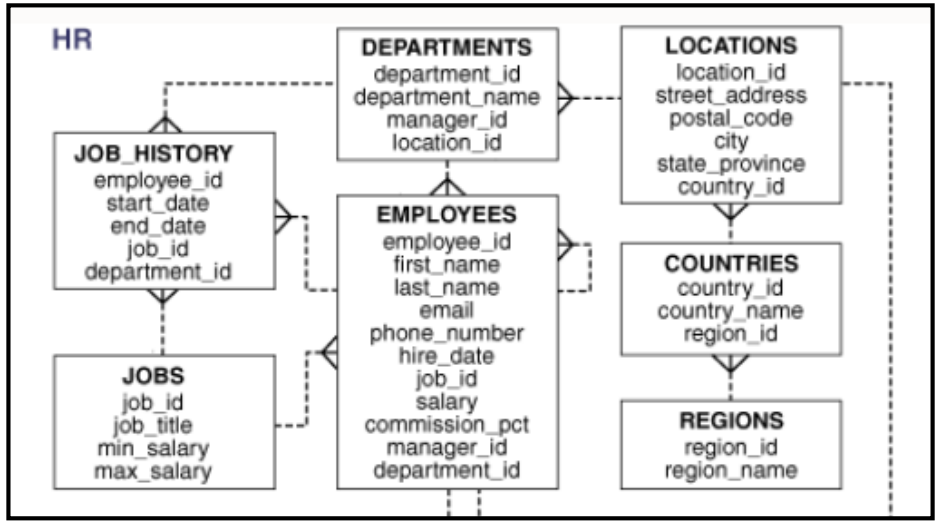
■ 移行検証実施の概要：Dockerによるコンテナ環境構築

PC上のバッチプログラムからコンテナ上のデータベースへ接続する構成を用意した。
厳密なWebシステムの動作環境ではないが、ソースコードの影響範囲は同等なため問題ないと判断した。





検証プログラム：
OracleサンプルスキーマをPostgreSQLにも用意し、双方に接続するプログラムをパターン毎を作成して検証した。



■ 検証結果の詳細(抜粋)

1.Mybatisの移行

■xmlの設定ファイルを変更する

●mybatis-config.xml　定義ファイル

・<property name="driver" value="○○○"/>

→ドライバー情報　上段＝Oracle,下段＝Postgres

oracle.jdbc.driver.OracleDriver

org.postgresql.Driver

・<property name="url" value="○○○"/>

→DB情報　上段＝Oracle,下段＝Postgres

jdbc:oracle:thin:@//133.227.245.82:1521/orclpdb1.localdomain

jdbc:postgresql://133.227.245.82:5432/db01

・<property name=“username” value=“○○○”/> と
<property name="password" value="○○○"/>

→ユーザ／パスワード　必要に応じて変更

2. S2JDBC(Seasar2)の移行

■dicon(設定)ファイルを変更する

●s2jdbc.dicon　利用するDBを定義する

・<include path="○○○.dicon"/>

→データソース情報　上段＝Oracle,下段＝Postgres

jdbc_ora.dicon (インクルードするファイル名は任意)

jdbc_pos.dicon (インクルードするファイル名は任意)

●jdbc_○○○.dicon　データソースを定義する

・<property name="driverClassName">"○○○"</property>

→DB情報　上段＝Oracle,下段＝Postgres

oracle.jdbc.driver.OracleDriver

org.postgresql.Driver

・ <property name="URL">"○○○"</property>

→ DB情報　上段＝Oracle,下段＝Postgres

jdbc:oracle:thin:@//133.227.245.82:1521/orclpdb1.localdomain

jdbc:postgresql://133.227.245.82:5432/db01

3. DriverManagerの移行

■DriverManagerを使用している該当ソースを変更する

●JDBCドライバのロード

・Class.forName("○○○");

→ドライバー情報　上段＝Oracle,下段＝Postgres

oracle.jdbc.driver.OracleDriver

org.postgresql.Driver

● DB情報

・DriverManager.getConnection(url, user, password);

→DB接続先(url)　上段＝Oracle,下段＝Postgres

jdbc:oracle:thin:@//133.227.245.82:1521/orclpdb1.localdomain

jdbc:postgresql://133.227.245.82:5432/db01

→ユーザ(user)、パスワード(password)を必要に応じて変更

● 例外処理

・SQLエラーコードの例外ハンドリング

→エラーコード)　上段＝Oracle,下段＝Postgres

(1)insert文(キーの重複)

if (ex.getSQLState().equals("23000")) {

if (ex.getSQLState().equals("23505")) {

(2)update文(対象行がロック)

if (ex.getSQLState().equals("61000")) {

if (ex.getSQLState().equals("55P03")) {

例外処理の補足

例外ハンドリングは右のように標準のSQL例外クラスで行っているので、
JDBCのエラーコードの値を変更するだけで済む。

※ところが、ベンダー固有の拡張例外クラスでエラーハンドリングしている
場合には、postgres用に変更が必要となる。

例)
[oracle]
java.sql.SQLIntegrityConstraintViolationException:
ORA-00001: 一意制約(HR.EMP_EMAIL_UK)に反しています
[postgres]
org.postgresql.util.PSQLException:
ERROR: 重複キーが一意性制約"employees_pkey"に違反しています

```
try{
    ..........
    //INSERT文を実行する
    int i = ps.executeUpdate();
    conn.commit();
} catch (SQLException ex) {
    //例外発生時の処理
    if (ex.getSQLState().equals("23000")) {
        System.out.println("キーが重複しています.");
    }
    conn.rollback();      //ロールバックする

try{
    ..........
    // 行ロックを取得する
    ResultSet rs = ps.executeQuery();
      ..........
    //UPDATE文を実行する
    int i = ps.executeUpdate();
    conn.commit();
} catch (SQLException ex) {
    //例外発生時の処理
   if (ex.getSQLState().equals("61000")) {
          System.out.println("対象行がロックされています.");
    }
conn.rollback();      //ロールバックする
```

■ 検証結果のまとめ

パターンごと検証結果から、変更箇所の共通点や相違点を整理しました。

※変更箇所は、SQLやデータベースオブジェクトを除いたアプリケーション固有に限定

■共通点

①ドライバの変更

・PostgreSQL用のODBC/JDBCドライバをインストールする（事前準備）

・JDBC系・・・ソースコードや定義ファイルを変更する

・ODBC系・・・ソースコードや定義ファイルを変更する

②接続文字列の変更

・接続文字列[接続先、ポート番号、データベース名]を変更する

・Windowsの“ODBC データソース アドミニストレータ”で、データソース名を登録する

③例外処理の変更

・ベンダー固有のSQLエラーコードの制御を同等のエラーコードに変更する

・ベンダー固有の例外クラスは、同等の例外クラスがあれば変更し、無ければ新規作成する

■相違点

パターンの中で固有の変更があったのは、ODP.NETからのNpgsqlへの移行の場合だけであった。

別のライブラリへの変更になるため、クラス名、メソッド名、データ型名や仕様の一部相違があり、仕様の調査と動作検証しならが対処方法を見極めることが必要となる。また、実装でどこまでライブラリ固有の要素を使用しているかで影響範囲がかなり変わってくる。少なくとも「クラス名やデータ型名の置換」、「一部のメソッドやプロパティの代替」は必要であった。

2020年度活動報告

- アプリケーション移行検証
- チューニング編アップデート
- PostgreSQL自習書の作成

チューニング編アップデート

- 移行ガイドブックのアップデート
  - PostgreSQLの新バージョンにも対応したガイドブックにしたい
  - アクセスが多いガイドブックから順次アップデート予定
- チューニング編とは
  - 異種DBMSからの移行の際のチューニングについて紹介
    - 移行元システムで定義されていた性能要件確認や元システムの性能情報収集について紹介
    - PostgreSQL移行時に注意が必要なチューニングのポイントを紹介
  - 現在公開中のチューニング編はversion 9.3が対象
    - VACUUMのパラメータが現バージョンと異なる
    - レプリケーションのチューニングも改善が施されている
    - 使われなくなったパラメータ等についての記載もある
    - Version 9.3のコミュニティサポート終了している

PostgreSQL v13の新規パラメータ

INSERT時のautovacuumをコントロールする

- autovacuum_vacuum_insert_threshold
  - VACUUMが動く最小のインサート数を指定する
  - デフォルト値は1000 (タプル)
  - -1を指定したときINSERTでVACUUMは発生しない
- autovacuum_vacuum_insert_scale_factor
  - VACUUMが動くテーブルサイズの閾値を指定する
  - デフォルトは0.2
    - テーブルサイズの20%の意味

- 効果
  - Index-onlyスキャンの精度向上
  - 突然大規模なXID凍結処理が走り、性能が劣化することを防ぐ

2020年度活動報告

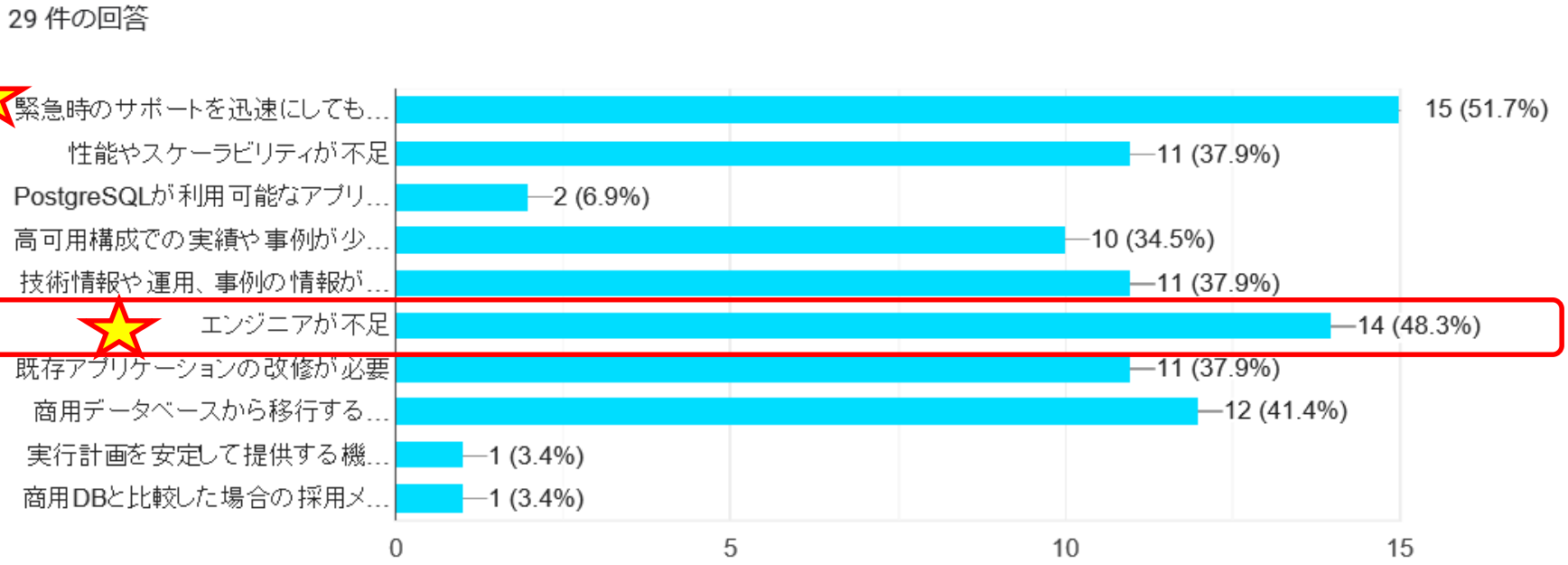
- アプリケーション移行検証
- チューニング編アップデート
- PostgreSQL自習書の作成

PostgreSQL自習書の作成

- 背景
- 自習書の作成方針
- 開発・運用の勘所
- まとめ

背景

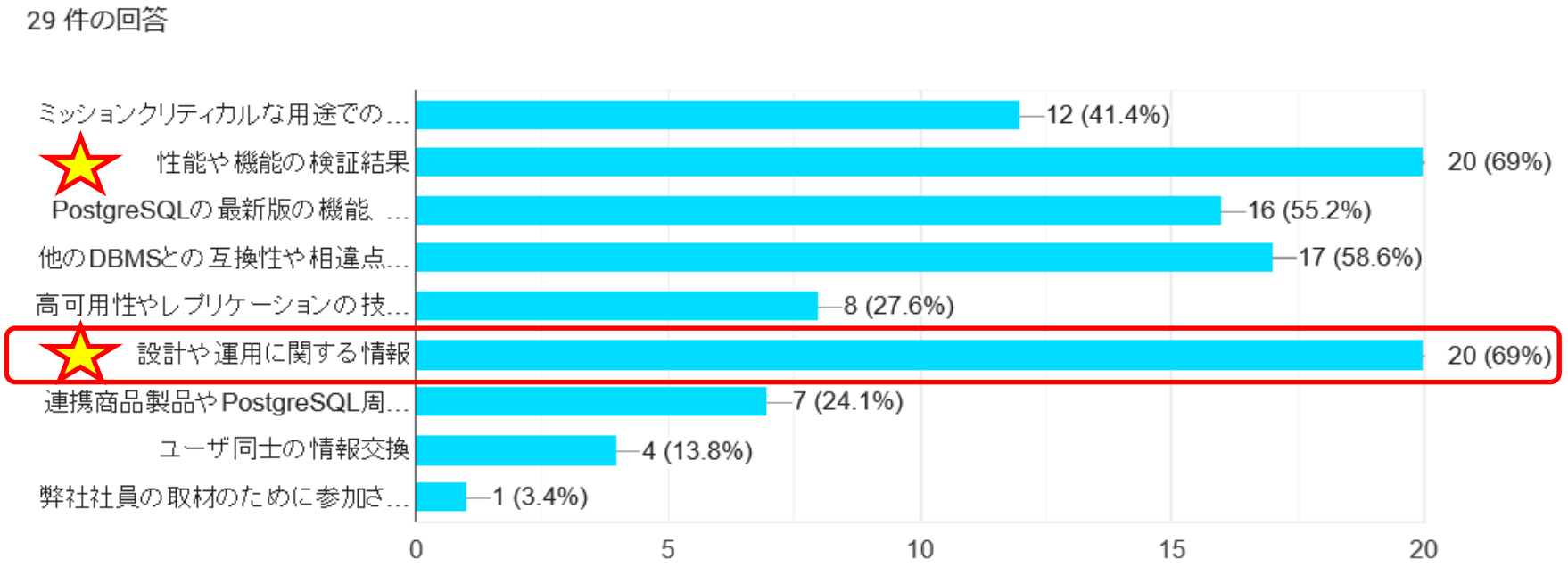
Q7:PostgreSQLをよりミッションクリティカル性の高いエンタープライズ領域で採用するための課題は何だとお考えですか？



2019年度成果発表会アンケートより

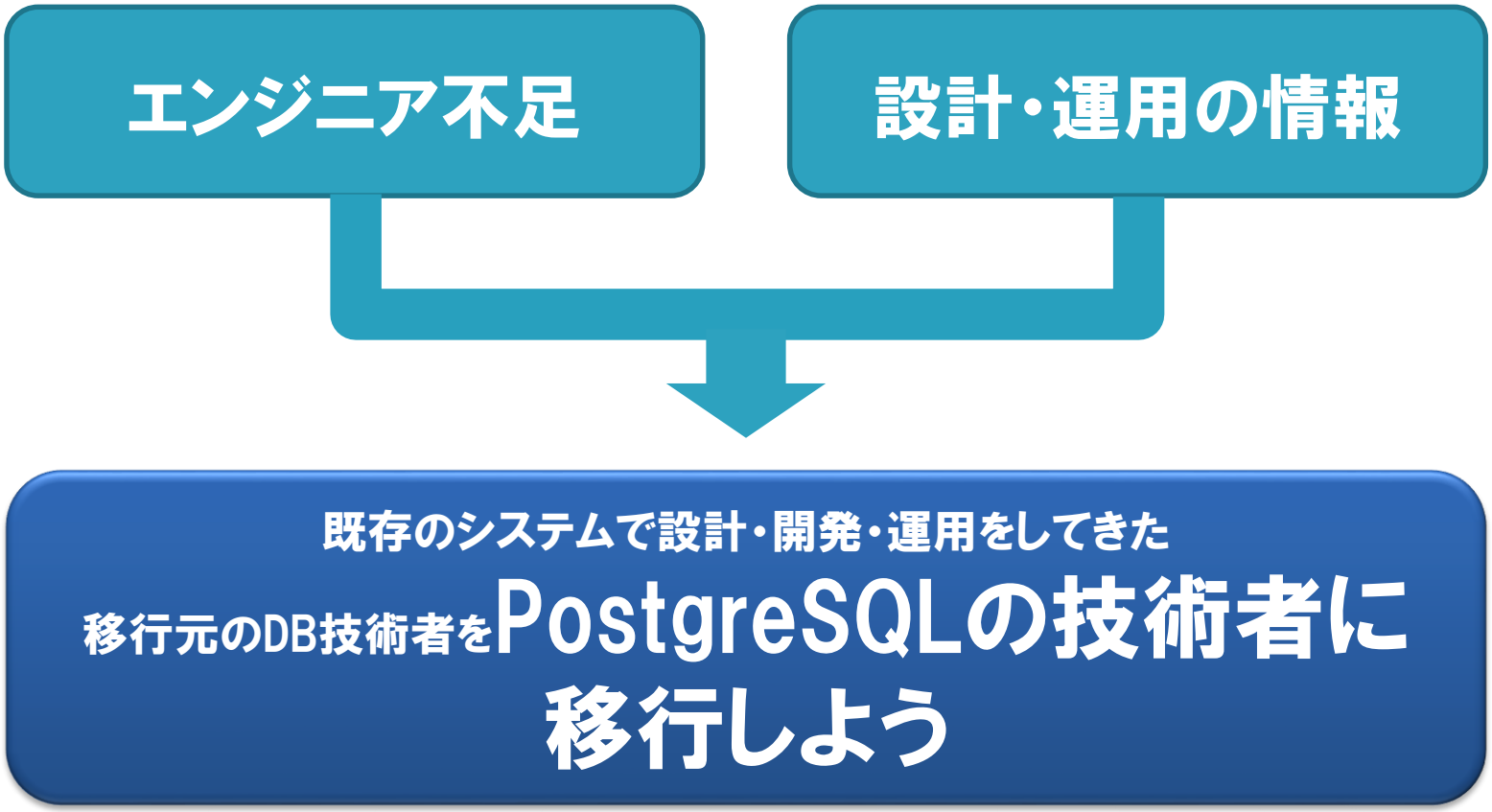
背景

Q21：PGEConsに何を期待していますか？またどんな情報を発信してほしいですか？（複数回答可）



2019年度成果発表会アンケートより

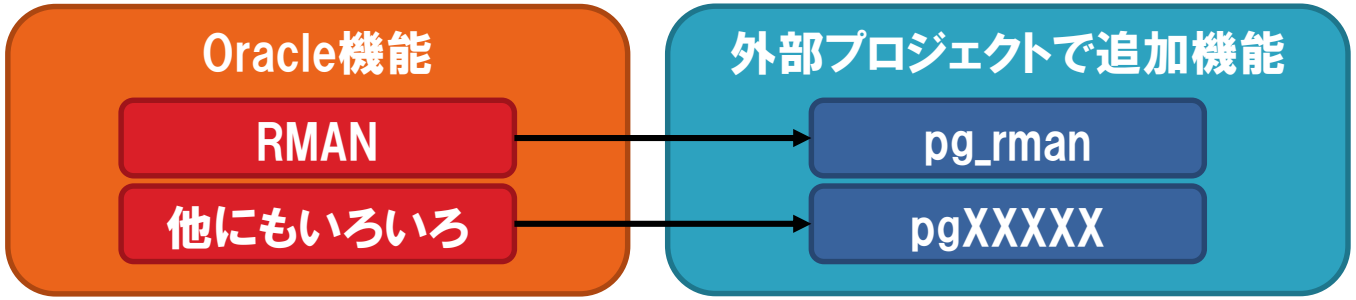
自習書の作成方針



開発・運用の勘所

「PostgreSQL自習書」の目次レベルでのねらい、知って欲しいこと

- PostgreSQLとは
  - オープンソースのリレーショナルデータベース管理システム
  - 特定企業に依存しない開発体制
  - 外部プロジェクトのオープンソースソフトウェアが存在
    - 商用運用では必要とされる機能が多い

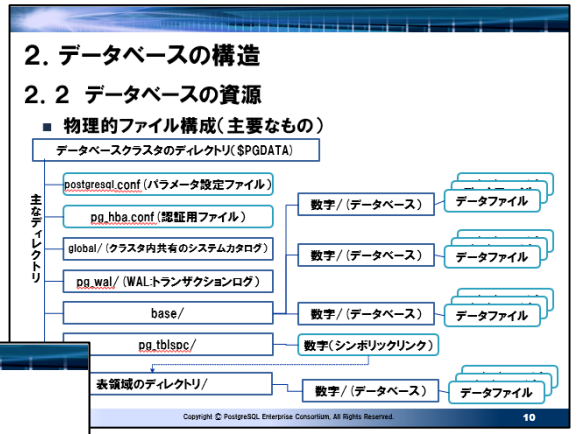


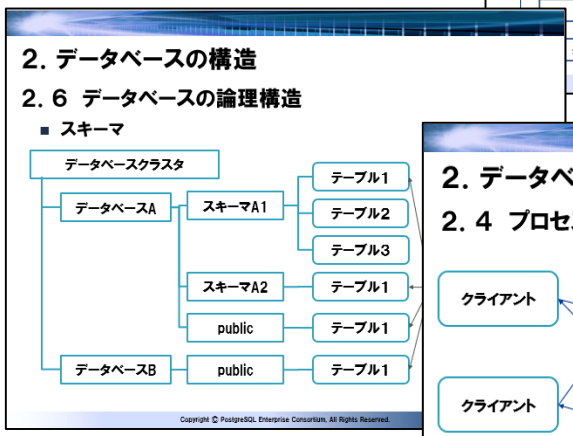
  - サポート
    - いくつかのベンダーから有償でサポートサービスが提供されている

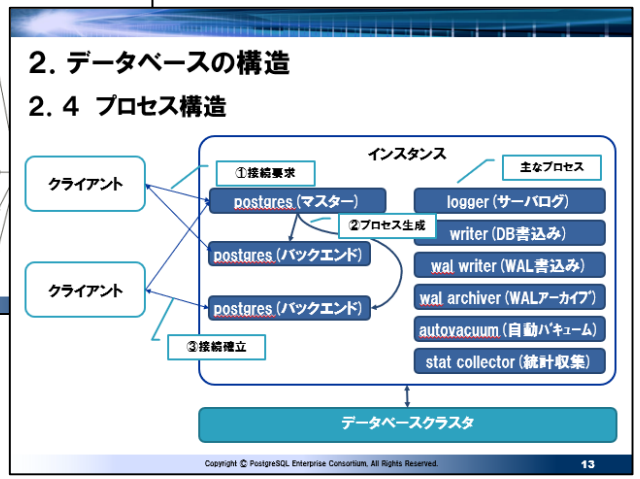
開発・運用の勘所

「PostgreSQL自習書」の目次レベルでのねらい、知って欲しいこと

- **データベースの構造**
  - 物理的に俯瞰
    - データベースクラスタの構成
    - OSのファイルシステム上の物理的な俯瞰
  - 論理的に俯瞰
    - ロール(＝ユーザ)
    - スキーマ
    - オブジェクト
  - 他
    - プロセス構造
    - メモリ構造



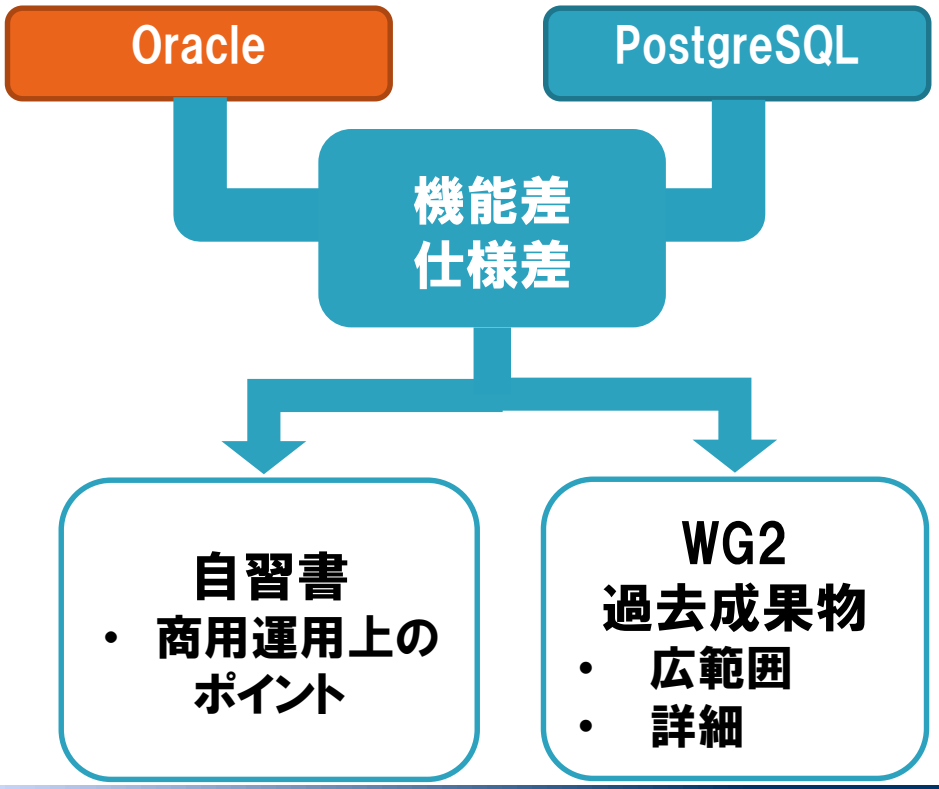




開発・運用の勘所

「PostgreSQL自習書」の目次レベルでのねらい、知って欲しいこと

- データベースの操作
  - RDBMSとしての機能
    - SQL
    - SQL実行処理
    - 関数
    - SQL手続き言語
    - トランザクション制御
    - ロック制御
  - 代表的なツール
    - psql
    - copy



開発・運用の勘所

「PostgreSQL自習書」の目次レベルでのねらい、知って欲しいこと

- **運用管理**
  - 正常運用のための情報取得
    - ﾃﾞｰﾀﾍﾞｰｽで提供される情報
    - サーバログ
    - 性能診断ツール
  - 保守
    - バックアップ・リストア
    - バキューム処理
    - 索引の再構築

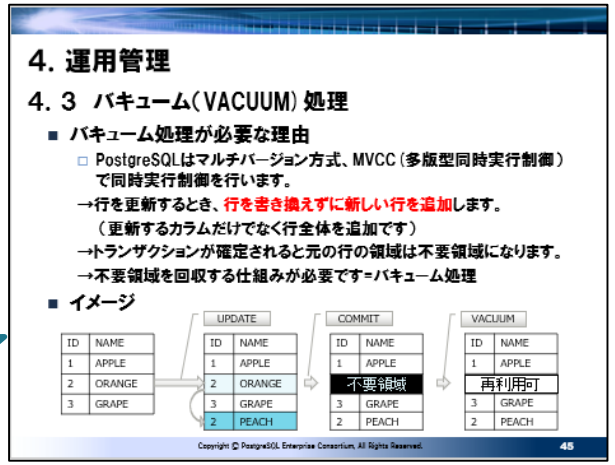
サーバログを出力するための
パラメータが具体的にどう役立つか

- サーバログ出力関連のパラメータ

スライドのパラメータは一部です。パラメータとその指定内容の説明は「PostgreSQL文書」を参照してください。以下にスライド上のパラメータを開発・運用面から説明します。

- log_min_duration_statement：実行時間が長いSQLを簡単に把握するときに指定します。対策が必要な場合はcontribのauto_explainを利用し実行時点の実行計画をログに出力することで正確な対処が実施しやすくなります。オンライン処理のように1回あたりの実行時間は短いが、実行回数が多く負荷の高いSQLは同じくcontribのpg_stat_statementsを使用して分析します。

PostgreSQL特有の保守



まとめ　～「PostgreSQL自習書」は～

- 作成方針はOracle技術者をPostgreSQL技術者へ移行すること
- 全体を短時間で俯瞰できる自習書
- 情報量の多いマニュアルを参照するためのガイドブック

おわりに

2020年度活動を振り返って

- 成果
  - 例年と比較して半年と活動期間が短い中で成果物を出すことができた
  - 全体的にオンラインでの活動となり難しい面はあったものの、活動を継続することができた

- 反省
  - 成果物作成が個人に依存している
  - 成果物に関する議論の活性化に課題あり

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、パーティションなど
- 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。

PGECons

PostgreSQL Enterprise Consortium