



**PGECons**  
PostgreSQL Enterprise Consortium

# PostgreSQL適用するための 検討ポイントについて

## ～異種DBMSからの移行を考える～

## 2012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 (設計運用WG)

# Agenda

- **WG2(設計運用WG)の活動概要**
- **成果紹介**
- **PostgreSQL移行成否を分けるポイントとは？**
- **WG2の活動を振り返って**

# WG2(設計運用WG)

## 活動の概要

# WG2(設計運用WG)の概要

## ■ 活動方針

- PostgreSQL利用促進を目的とし、PostgreSQL経験の少ないDB技術者に向けた情報発信を行う。
- 他DBMSと比べた際のPostgreSQLの優位点や注意事項を明確化する。
- 実システムを更改する場合の検討項目・手順を作成する。

(例)

- SQL文やストアドプロシージャなどのアプリケーション側の注意点
- データ定義変更の有無を判定する基準や方法
- PostgreSQL活用時にかかるコスト要因の分析

## WG2 活動計画策定の経緯

- **エンタープライズ領域へのPostgreSQLの普及を推進するために**
  - PostgreSQLを普及していく上で課題となるポイントをPGECons設立準備期間に活動テーマ候補としてリストアップ
  - リストアップされた課題の中から、設計運用分野の課題かつ要望および重要度を考慮して、活動テーマを選定

# (参考) 活動テーマ候補抜粋

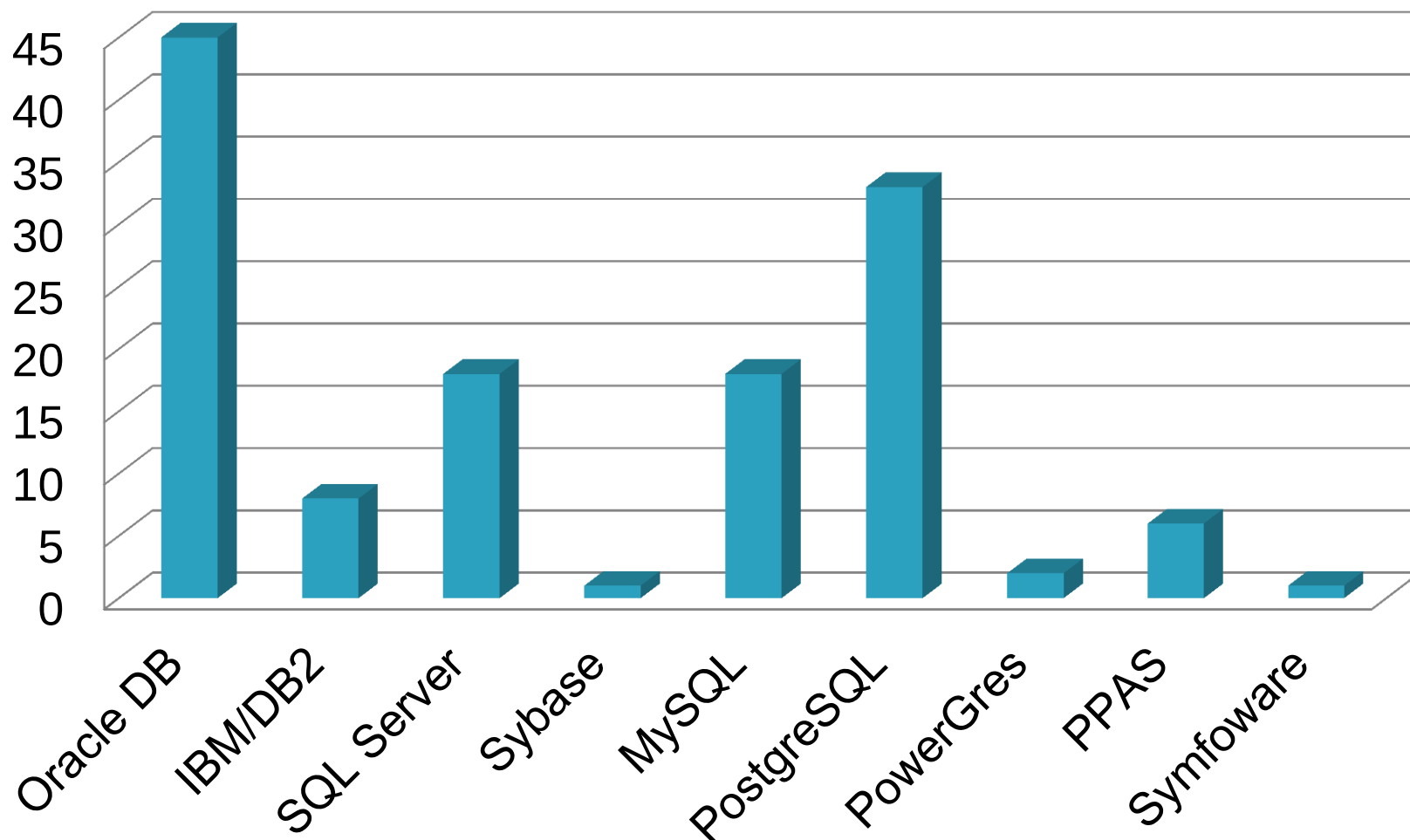
| 大項目    | 小項目               | 概要                           |
|--------|-------------------|------------------------------|
| 保守性    | 保守サポート            | 保守サポート事例紹介                   |
|        | トレーサビリティ          | 障害解析情報の取得方法の整備               |
| 運用性    | 監視運用              | 目的別の監視手法の整備                  |
|        | バックアップ運用          | 目的別のバックアップ手法の整備              |
| セキュリティ | 監査                | 目的別の監査への対応方法(アクセスログの取得など)の整備 |
| 移行性    | データベース            | データ・スキーマの連携・移行手順の整備 ※        |
|        | アプリケーション<br>プログラム | SQLなどの互換性検証 ※                |
|        |                   | ストアドプロシージャに対する移行ノウハウの整備 ※    |
|        | 移行コスト             | 移行コストの見積の考え方の整備 ※            |
| 接続性    | 他ソフトウェア<br>との連携   | 他ソフトウェアとの接続性検証               |
|        |                   | 他ソフトウェアとの連携手法の整備             |
|        |                   | 組合せ(スタック)モデルの整備              |

※ 本年度着手のテーマ

# エンタープライズ領域で使われている異種DBMSシステムへの対応を図る

(参考) PGECons発足セミナー(2012年7月6日) アンケートより

Q: 現在お客様はどんなリレーショナルデータベースをお使いですか？(複数回答可)



# WG2の2012年度活動テーマ

## ■ 「異種DBMSからPostgreSQLへの移行」と設定

### ■ 課題認識

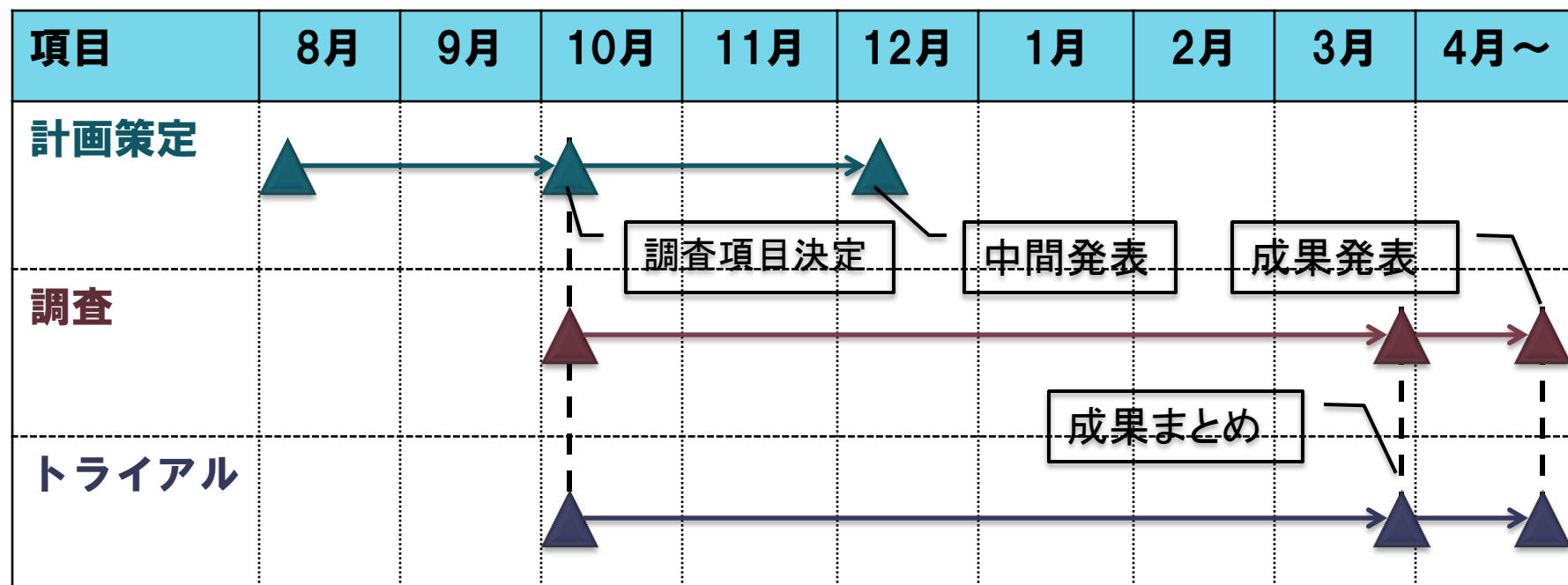
- 異種DBMSシステムをPostgreSQLへ移行するプロセスが確立していないことが、普及を妨げる大きな障壁と認識
  - 移行作業をどのように進めればよいか分からない。
  - 初期段階で移行に必要なトータルコストを算出できない。
  - 過去の経験則や点在するノウハウに依存しているのが現状

### ■ 活動目標

- 異種DBMSからPostgreSQLへの移行を検討する際に留意すべき事項や、難易度判断、調査方法といったガイドラインを本WGから提唱する。

# 2012年度の活動スケジュール

- 中間発表(12月)まで
  - 全体計画策定
  - 調査項目洗い出し、決定
- 年度末(3月)まで
  - 各項目の調査の実施
  - 移行トライアルの実施



# 実施体制

## ■ 参加企業（企業名順）

- ☐ 日本電気株式会社
- ☐ NECソフト株式会社
- ☐ 日本電信電話株式会社
- ☐ NTTソフトウェア株式会社
- ☐ SRA OSS, Inc. 日本支社
- ☐ TIS株式会社
- ☐ 株式会社アシスト
- ☐ クオリカ株式会社
- ☐ サイオステクノロジー株式会社
- ☐ 株式会社日立製作所
- ☐ 株式会社日立ソリューションズ
- ☐ 株式会社富士通ソーシアルサイエンスラボラトリ

# 検討会の様子

## ■ 隔週でWG2参加各社による検討会を開催



# 今年度の成果物

- 「2012年度WG2活動成果報告書」として、PGECconsサイトに公開しました。
- 移行作業の調査ドキュメントとともに、検証時に実際に移行を行ったアプリケーションのソースコードも公開しています。



PostgreSQL エンタープライズ・コンソーシアム 技術部会 WG#2

2012年度WG2活動成果報告書

© 2012 PostgreSQL Enterprise Consortium

# 作成文書と担当会社

| 作成文書            | 担当会社                          |
|-----------------|-------------------------------|
| DB移行フレームワーク編    | NECソフト(株)、日本電信電話(株)           |
| システム構成調査編       | 日本電気(株)                       |
| 異種DB間連携調査編      | サイオステクノロジー(株)                 |
| スキーマ移行調査編       | TIS(株)                        |
| SQL移行調査編        | SRA OSS, Inc. 日本支社            |
| ストアドプロシージャ移行調査編 | クオリカ(株)                       |
| 組み込み関数移行調査編     | TIS(株)                        |
| データ移行調査および実践編   | (株)富士通ソーシャルサイエンスラボラトリ、(株)アシスト |
| アプリケーション移行調査編   | TIS(株)                        |
| アプリケーション移行実践編   | NTTソフトウェア(株)                  |
| 運用設計            | (今期実施せず)                      |
| 試験項目            | (今期実施せず)                      |
| チューニング          | (今期実施せず)                      |

# 成果紹介

# 作業文書の構成

- 移行作業の全体像を解説
  - DB移行フレームワーク編
- 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
  - 異種DB間連携調査編
  - スキーマ移行調査編
  - SQL移行調査編
  - ストアドプロシージャ移行調査編
  - 組み込み関数移行調査編
  - アプリケーション移行調査編
- 移行作業を試行する検証
  - データ移行調査および実践編
  - アプリケーション移行実践編

# 作業文書の構成

## ■ 移行作業の全体像を解説

- DB移行フレームワーク編

## ■ 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
- 異種DB間連携調査編
- スキーマ移行調査編
- SQL移行調査編
- ストアドプロシージャ移行調査編
- 組み込み関数移行調査編
- アプリケーション移行調査編

## ■ 移行作業を試行する検証

- データ移行調査および実践編
- アプリケーション移行実践編

# DB移行フレームワーク編（1/2）

- PostgreSQLに移行するにはどんな作業が必要か？
  - 「移行プロセスに対する認識を共有する」のが目的
  - DB移行作業の主な流れ（一部抜粋）



# DB移行フレームワーク編（2/2）

## ■ アセスメントの作業内容について検討

### □ 簡易アセスメントリスト（一部抜粋）

| 確認事項             | 確認内容                                                                                                                             | 判定    |
|------------------|----------------------------------------------------------------------------------------------------------------------------------|-------|
| 可用性要件            | 障害による一時的なサービス停止や操作の再実行が許容できる。<br>・障害発生時にノード切り替えによる分単位のサービス停止が許容できる<br>・障害による影響範囲を制限するシステムの閉塞が可能                                  | ○ / × |
| スキーマ移行上の制約       | 移行元DBMSやオプション製品による固有な属性を使用しない。<br>・固有な属性（ROWIDやBFILEなど）を使用しない<br>・オプション製品が追加する非標準属性を使用しない                                        | ○ / × |
| パーティション          | パーティションの利用を前提とした大容量データ処理および運用を行わない。<br>・100以上に分割するパーティションは存在しない<br>・複雑なパーティション分割を行わない<br>・構成変更によるリバランス等は必要ない                     | ○ / × |
| ストアドプロシージャの移行困難性 | 移行作業自体が難しいプロシージャや存在しない<br>・PACKAGE定義やグローバル変数を使用しない<br>・移行元DBMSに固有なユーティリティパッケージを使用していない<br>・ソースの更新権限を持たないプロシージャや難読化したプロシージャは存在しない | ○ / × |
| 組み込み関数           | PostgreSQL, Oracleの何れも対応していない関数を利用していない。                                                                                         | ○ / × |

⋮

# 作業文書の構成

- 移行作業の全体像を解説
  - DB移行フレームワーク編
- 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
  - 異種DB間連携調査編
  - スキーマ移行調査編
  - SQL移行調査編
  - ストアドプロシージャ移行調査編
  - 組み込み関数移行調査編
  - アプリケーション移行調査編
- 移行作業を試行する検証
  - データ移行調査および実践編
  - アプリケーション移行実践編

# システム構成調査編（1/2）

## ■ DBMSの主なシステム構成とその特徴について調査し、PostgreSQLに移行時の構成検討の参考となる資料を作成

### □ 一般的なシステム構成の特性を紹介

- シングルサーバ
- HAクラスタ
- データベースレプリケーション
- マルチマスタ負荷分散クラスタ

### □ 主に検討のポイントは以下

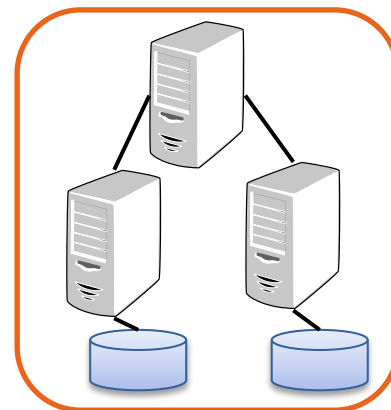
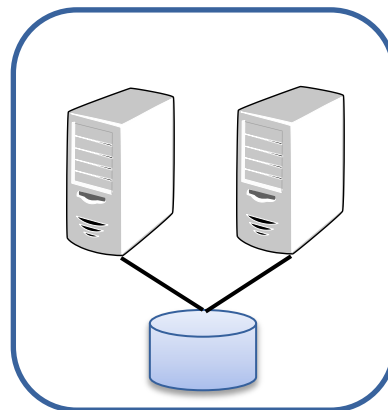
- 可用性
- 性能（参照・更新）
- 拡張性
- 設計・運用面
- 初期導入コスト

今は“XXXX”DBMSで  
Active-Standby構成  
だけど、PostgreSQLで  
同じ構成でよいのか？

性能？

可用性？

コスト？



# システム構成調査編（2/2）

## ■ HAクラスタ構成の比較内容（一部抜粋）

|         |           | 共有ディスク                                                            | ディスクミラーリング                                               | ストリーミングレプリケーション                                                             |
|---------|-----------|-------------------------------------------------------------------|----------------------------------------------------------|-----------------------------------------------------------------------------|
| 可用性     | サービス継続性   | 取れる                                                               | 取れる                                                      | 取れる                                                                         |
|         | ディザスタリカバリ | ストレージレイヤーでレプリケーションをすれば可能                                          | 取れる                                                      | 取れる                                                                         |
| 性能      | 参照系       | オーバヘッドはなし                                                         | オーバヘッドはなし                                                | スケールアウトが可能                                                                  |
|         | 更新系       | オーバヘッドはなし<br>HA構成の中では一元化されているメリットが出る                              | オーバヘッドあり<br>同期・非同期の設定もあるが、ストレージレベルの非同期ではDBの論理破壊の可能性がある   | オーバヘッドあり<br>同期・非同期の設定で性能とトランザクション反映の信頼性のトレードオフになる                           |
| 拡張性     |           | スケールアップで対応<br>データ容量追加は行いやすい                                       | スケールアップで対応                                               | カスケードレプリケーションによって参照系のスケールアウトが可能                                             |
| 運用・設計面  |           | ストレージ全体を一括管理できる<br>メリットあり<br><br>特徴的な検討項目<br>・クラスタ設計<br>・ストレージの設計 | 特徴的な検討項目<br>・クラスタ設計<br>・ストレージ層の同期設計<br>・フェイルバック時のデータ同期設計 | 特徴的な検討項目<br>・クラスタ設計<br>・レプリケーションの同期やPostgreSQLの切替動作の設計<br>・フェイルバック時のデータ同期設計 |
| 初期導入コスト |           | 共有ディスクが必要となり比較的高価                                                 | 汎用のサーバのみで構成を組み事が可能                                       | 汎用のサーバのみで構成を組み事が可能                                                          |

# 異種DB間連携調査編（1/2）

## ■ 異種DBMSとPostgreSQLをどう連携させるか？

### □ 異種DBMSとPostgreSQLを連携させる目的

- データベースの段階移行時における並行稼動
- マスターDBの負荷分散
- 災害対策
- DWH 構築

### □ 異種DBMS間レプリケーションを実現するソフトウェアを調査

- InfoFrame DataCoordinator（NEC）
- DBMoto（クライム）
- DataSpider Servista（アプレッソ）
- xDB Replication Server（EnterpriseDB）

# 異種DB間連携調査編（2/2）

## ■ 異種DBMS間連携トライアル

- xDB Replication を使用した異種DBMS間連携を実施し、連携時の制限事項や注意点を調査



## ■ 実施結果

- 異種DBMSとPostgreSQLのデータ連携を容易に実現
- 柔軟なレプリケーション設定が可能であることを確認

# スキーマ移行調査編 (1/2)

## ■ 異種DBMSとPostgreSQLのスキーマの違いは？

- Oracle, PostgreSQLのDDL互換性を調査
- 各種DDLの相違点をもとにした移行方針を解説
- 組み込みデータ型対応表（一部抜粋）

| 属性 | Oracle                  |      |                               | PostgreSQL                       |        |          |                                              |
|----|-------------------------|------|-------------------------------|----------------------------------|--------|----------|----------------------------------------------|
|    | データ型                    | 単位   | 備考                            | データ型                             | サイズ    | 単位       | 備考                                           |
| 文字 | VARCHAR2                | byte | 4000 バイト                      | varchar(n)                       |        |          | 上限付き可変長                                      |
|    |                         | char | 4000 バイト                      | varchar(n)                       |        |          | 上限付き可変長                                      |
|    | NVARCHAR2               | char | 4000 バイト                      | varchar(n)                       |        |          | 上限付き可変長                                      |
|    | CHAR                    | byte | 2000 バイト                      | char(n)                          |        |          | 空白で埋められた固定長                                  |
|    |                         | char | 2000 バイト                      | char(n)                          |        |          | 空白で埋められた固定長                                  |
|    | NCHAR                   | char | 2000 バイト                      | char(n)                          |        |          | 空白で埋められた固定長                                  |
|    | LONG (下位互換)             |      | 2G-1 バイト                      | text                             |        |          | 可変長(最大1GB)                                   |
| 真数 | CLOB                    |      | 4G-1 バイト                      | text                             |        |          | 可変長(最大1GB)                                   |
|    | NCLOB                   |      | 4G-1 バイト                      | text                             |        |          | 可変長(最大1GB)                                   |
|    | NUMBER                  |      | 精度(10進数38桁)、位取り(10進数-84~127桁) | decimal                          | 可変長    | 固定小数点    | 小数点前までは131072桁、小数点以降は16383桁                  |
|    |                         |      |                               | numeric                          | 可変長    | 固定小数点    | 小数点前までは131072桁、小数点以降は16383桁                  |
|    |                         |      |                               | smallint                         | 2バイト   | 整数データ    | 整数(-32768~+32767)                            |
|    |                         |      |                               | integer                          | 4バイト   | 整数データ    | 整数(-2147483648~+2147483647)                  |
|    |                         |      |                               | bigint                           | 8バイト   | 整数データ    | 整数(-9223372036854775808~9223372036854775807) |
|    |                         |      |                               | real                             | 4バイト   | 単精度浮動小数点 | 6桁精度                                         |
|    |                         |      |                               | double precision                 | 8バイト   | 倍精度浮動小数点 | 15桁精度                                        |
|    | FLOAT                   |      | 精度(2進数126桁)                   | float                            | 4、8バイト | 浮動小数点    | 精度(2進数53桁)                                   |
| 概数 | BINARY_FLOAT            |      | 単精度浮動小数点数                     | real                             | 4バイト   | 単精度浮動小数点 | 6桁精度                                         |
|    | BINARY_DOUBLE           |      | 倍精度浮動小数点数                     | double precision                 | 8バイト   | 倍精度浮動小数点 | 15桁精度                                        |
| 日時 | DATE                    | 日~秒  | -4712/01/01 ~ 9999/12/31      | timestamp                        | 8 バイト  | 1μ秒単位    | 日付と時刻両方(時間帯なし) 4713 BC~294276 AD (1μ秒、14桁)   |
|    |                         |      |                               | date                             | 4 バイト  | 日単位      | 日付のみの場合はPostgreSQLのdate型で表現可能                |
|    | TIMESTAMP               |      | DATE 型に加えてミリ秒、最小でナノ秒単位        | timestamp                        | 8 バイト  |          | 日付と時刻両方(時間帯なし) 4713 BC~294276 AD (1μ秒、14桁)   |
|    | TIMESTAMP WITH TIMEZONE |      | タイムスタンプ型に加えてタイムゾーン情報          | timestamp [ (p) ] with time zone | 8バイト   |          | 日付と時刻両方、時間帯付き 4713 BC~294276 AD (1μ秒、14桁)    |

# スキーマ移行調査編 (2/2)

## ■ Ora2Pgを使った移行で手作業が必要な部分を解説

```
CREATE TABLE DATA_TYPE_TEST1 (  
  ID                VARCHAR2(5)  NOT NULL,  
  CHAR_VARCHAR2_BYTE VARCHAR2(20),  
  CHAR_VARCHAR2_CHAR VARCHAR2(20 CHAR),  
  CHAR_NVARCHAR2    NVARCHAR2(20),  
  CHAR_CHAR_BYTE    CHAR(10),  
  CHAR_CHAR_CHAR    CHAR(10 CHAR),  
  CHAR_NCHAR         NCHAR(10),  
  CHAR_LONG          LONG,  
  CHAR_CLOB          CLOB,  
  CHAR_NCLOB         NCLOB,  
  NUM_NUMBER1        NUMBER(5),  
  NUM_NUMBER2        NUMBER(10),  
  NUM_NUMBER3        NUMBER(11),  
  NUM_NUMBER4        NUMBER(10,2),  
  NUM_NUMBER5        NUMBER(15,5),  
  DATE_DATE          DATE,  
  DATE_TIMESTAMP     TIMESTAMP(6),  
  DATE_TIMESTAMP_TIMEZONE TIMESTAMP(6) WITH TIME ZONE,  
  DATE_TIMESTAMP_LOCAL  TIMESTAMP(6) WITH LOCAL TIME ZONE  
)
```

# スキーマ移行調査編 (2/2)

## ■ Ora2Pgを使った移行で手作業が必要な部分を解説

```
CREATE TABLE DATA_TYPE_TEST1 (
```

```
|| CREATE TABLE "data_type_test1" (  
||   "id"                varchar(5) NOT NULL,  
||   "char_varchar2_byte" varchar(20),  
||   "char_varchar2_char" varchar(20),  
||   "char_nvarchar2"    varchar(20),  
||   "char_char_byte"    char(10),  
||   "char_char_char"    char(10),  
||   "char_nchar"        char, → 手作業で char(10) に書き換える  
||   "char_long"         text,  
||   "char_clob"         text,  
||   "char_nclob"        text,  
||   "num_number1"       numeric(5), → 手作業で int に書き換える  
||   "num_number2"       numeric(10), → 手作業で bigint に書き換える  
||   "num_number3"       numeric(11), → 手作業で bigint に書き換える  
||   "num_number4"       decimal(10,2),  
||   "num_number5"       decimal(15,5),  
||   "date_date"         timestamp,  
||   "date_timestamp"    timestamp, → 手作業で timestamp(6) に書き換える  
||   "date_timestamp_timezone" timestamp, → 手作業で timestamp(6) with time zone に書き換える  
||   "date_timestamp_local" timestamp → 手作業で timestamp(6) に書き換える  
|| );
```

# SQL移行調査編 (1/2)

## ■ 異種DBMSのSQLはPostgreSQLでも動くのか？

- Oracle, SQL Server, PostgreSQLのSQL互換性を調査
- 互換性がないSQLの書き換え方針を調査
- SQL差異表（一部抜粋）

| SQL機能                     | 標準SQL | 説明                                         | PostgreSQL 9.2.1 |                             | Oracle 11g R2 |                             | SQL Server 2008 R2 |               |
|---------------------------|-------|--------------------------------------------|------------------|-----------------------------|---------------|-----------------------------|--------------------|---------------|
|                           |       |                                            | 対応               | 備考                          | 対応            | 備考                          | 対応                 | 備考            |
| <b>SELECT</b>             |       |                                            |                  |                             |               |                             |                    |               |
| WITH句                     | ○     | WITH 問い合わせ(共通テーブル式)                        | ○                | WITH 句の中で更新系コマンドが使用可能       | ○             |                             | ○                  | 列の別名に独自の構文あり  |
| DISTINCT                  | ○     | 重複している行を取り除く                               | ○                | DISTINCT ON は PostgreSQL 特有 | ○             |                             | ○                  |               |
| UNIQUE                    | ×     | 重複している行を取り除く                               | ×                |                             | ○             | Oracle 特有                   | ×                  |               |
| TOP句                      | ×     | 取得する行数の指定                                  | ×                |                             | ×             |                             | ○                  | SQL Server 特有 |
| FROM句                     | ○     | テーブルの指定                                    | ○                | FROM 句を省略可能                 | ○             | FROM 句は省略不可<br>サブクエリの別名を省略可 | ○                  | FROM 句を省略可能   |
| CONNECT BY / START WITH 句 | ×     | 階層問い合わせ                                    | ×                |                             | ○             | Oracle 特有                   | ×                  |               |
| JOIN 句                    | ○     | テーブルの結合                                    | ○                |                             | ○             | 独自の結合演算子(+)あり               | ○                  |               |
| WINDOW句                   | ○     | OVER 句で参照するウィンドウ                           | ○                |                             | ×             |                             | ×                  |               |
| LIMIT 句                   | ×     | 取得する行数などの指定                                | ○                |                             | ×             |                             | ×                  |               |
| FETCH 句                   | ○     | 取得する行数などの指定                                | ○                |                             | ×             |                             | ×                  |               |
| SELECT FOR UPDATE         | ○     | 更新ロックを取得する<br>(標準 SQL ではカーソルのオプションとしてのみ有効) | ○                | 任意の SELECT で使用可能            | ○             | サブクエリでは使用不可                 | ○                  | カーソルでのみ使用可能   |
| SELECT FOR SHARE          | ×     | 共有ロックを取得する                                 | ○                | PostgreSQL 特有               | ×             |                             | ×                  |               |
| <b>更新系</b>                |       |                                            |                  |                             |               |                             |                    |               |
| INSERT                    |       | 行の挿入                                       |                  | RETURNING句を使用可能             |               |                             |                    | TOP句を使用可能     |

# SQL移行調査編 (2/2)

## ■ 代表的な書き換え例

- SELECT結果の行数を制限するSQLはDBMS毎に異なる。

| Oracle                                                                              | SQL Server                                        | PostgreSQL                                                   |
|-------------------------------------------------------------------------------------|---------------------------------------------------|--------------------------------------------------------------|
| SELECT * FROM<br>(SELECT * FROM tbl<br>ORDER BY id)<br>WHERE <b>ROWNUM &lt;= 10</b> | SELECT <b>TOP 10</b> *<br>FROM tbl<br>ORDER BY id | SELECT * FROM tbl<br>ORDER BY id<br><b>LIMIT 10 OFFSET 0</b> |

※PostgreSQL (8.4以降) は標準SQLの構文に対応

SELECT \* FROM tbl **OFFSET 0 ROW FETCH FIRST 10 ROWS ONLY**

## □ MERGE

- PostgreSQLでは未実装。WITH句を使った書き換え方法を紹介。

## □ CONNECT BY

- Oracle独自の階層問い合わせ
- 再帰SQL(標準SQL準拠)を使った書き換え方法を紹介

# ストアドプロシージャ移行調査編

- 異種DBMSのストアドプロシージャをどのようにPostgreSQLに移行すればよいか？
  - PL/SQLのストアドプロシージャをPostgreSQLへ移行する際に注意すべき点について調査
  - 手続き言語の文法的な違いを明確にすることを目的とした。
- 調査結果
  - SQLや手続き言語の差は機械的に変換可能と思われる。
  - DBMSのアーキテクチャにかかわる部分(トランザクション制御など)が存在すると移行の難易度が高くなる。
  - 無理にプロシージャに移行せず、アプリケーションとして書き換えた方が良いパターンもある。

# 組み込み関数移行調査編 (1/2)

## ■ 異種DBMSの関数はPostgreSQLでも動くのか？

- Oracle, PostgreSQLの組み込み関数互換性を調査
- 組み込み関数対応表（一部抜粋）

| Oracle関数                            | 説明                                                         | PostgreSQL関数                                                         | 対応可否 |
|-------------------------------------|------------------------------------------------------------|----------------------------------------------------------------------|------|
| 日時ファンクション                           |                                                            |                                                                      |      |
| ADD_MONTHS (date, integer)          | 日付dateに月数integerを加えて戻す。                                    | + 演算子を使って書き換え可能<br>例: select date '2013-03-22' + interval '1 months' | ○    |
| CURRENT_DATE                        | セッション・タイムゾーンの現在の日付を戻す。                                     | current_date, current_timestamp                                      | ○    |
| CURRENT_TIMESTAMP                   | セッション・タイムゾーンの現在の日付および時刻をTIMESTAMP WITH TIME ZONEデータ型の値で戻す。 | current_timestamp                                                    | ○    |
| DBTIMEZONE                          | データベースのタイムゾーンの値を戻す。                                        |                                                                      | ×    |
| EXTRACT (element FROM date)         | 日時式または期間式から指定された日時フィールドの値を抽出して戻す。                          | extract(field from timestamp)                                        | ○    |
| FROM_TZ(timestamp, time_zone_value) | タイムスタンプ値およびタイムゾーンをTIMESTAMP WITH TIME ZONE値に変換する。          |                                                                      | ×    |
| LAST_DAY(d)                         | d を含む月の最後の日付を戻す。                                           | last_date(date)                                                      | ○    |
| LOCALTIMESTAMP                      | セッション・タイムゾーンの現在の日付および時刻をTIMESTAMPデータ型の値で戻す。                | localtimestamp                                                       | ○    |
| MONTHS_BETWEEN(d1, d2)              | d1 と d2 の間の月数を戻す。                                          | months_between(d1, d2)                                               | ○    |
| NEW_TIME(d, z1, z2)                 | 時間帯z1の日時がdの時点の時間帯z2の日時を戻す。                                 |                                                                      | ×    |
| NEXT_DAY(d, char)                   | charで指定した曜日で日付d以降の最初の日付を戻す。                                | next_day(date, text)                                                 | ○    |
| NUMTODSINTERVAL(n, 'char_expr')     | n をINTERVAL DAY TO SECOND リテラルに変換する。                       |                                                                      | ×    |
| NUMTOYMINTERVAL(n, 'char_expr')     | n をINTERVAL YEAR TO MONTH リテラルに変換する。                       |                                                                      | ×    |
| ROUND(d, fmt)                       | d を書式 fmt で指定した単位に丸めた結果を戻す。                                | round(date, text)                                                    | ○    |

# 組み込み関数移行調査編 (2/2)

## ■ 代表的な書き換え例

### □ ADD\_MONTHS : 月を演算する関数

```
SQL> SELECT ADD_MONTHS('2013/3/22',1) FROM DUAL;  
ADD_MONT  
-----  
13-04-22
```

### □ PostgreSQL に ADD\_MONTHS 関数は存在しない。

### □ 算術演算子を使った書き換え

```
postgres=# SELECT date '2013-03-22' + interval '1 months';  
?column?  
-----  
2013-04-22 00:00:00  
(1 行)
```

### □ Oracle互換関数ライブラリ(Orafce)を利用する方法も

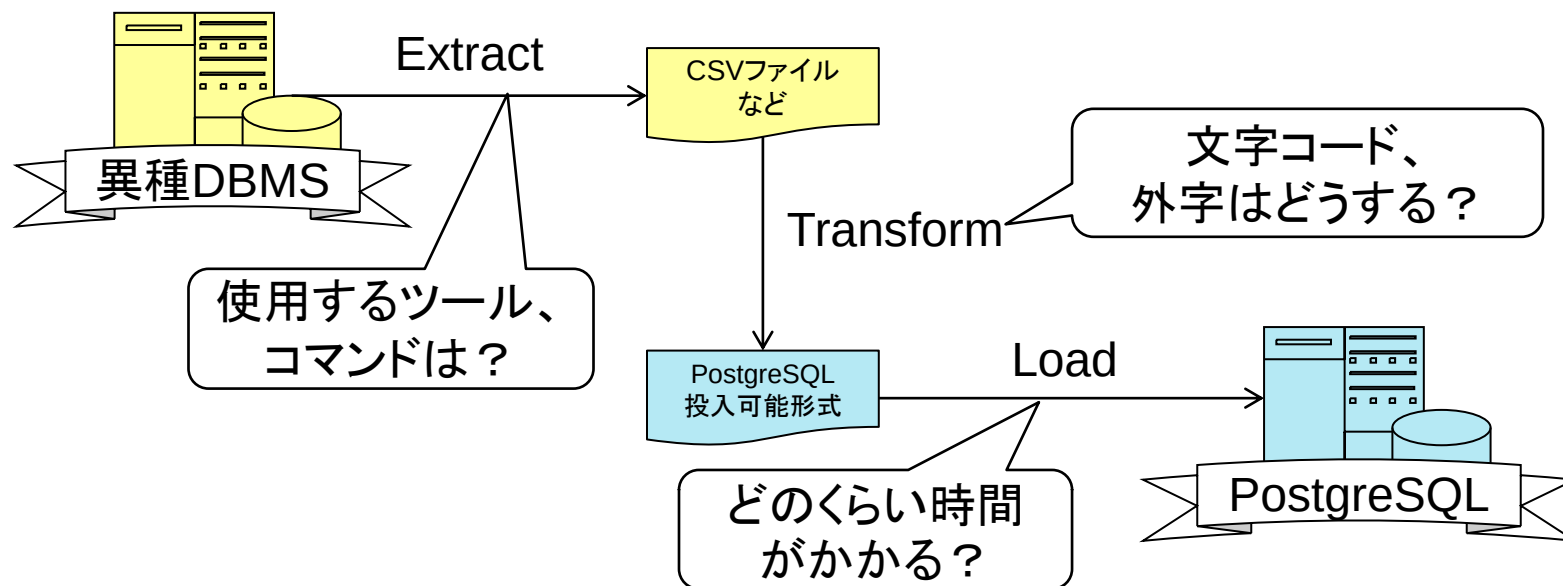
# 作業文書の構成

- 移行作業の全体像を解説
  - DB移行フレームワーク編
- 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
  - 異種DB間連携調査編
  - スキーマ移行調査編
  - SQL移行調査編
  - ストアドプロシージャ移行調査編
  - 組み込み関数移行調査編
  - アプリケーション移行調査編
- 移行作業を試行する検証
  - データ移行調査および実践編
  - アプリケーション移行実践編

# データ移行調査および実践編（1/3）

## ■ 異種DBMSのデータをPostgreSQLに持ってくるにはどうすればいい？

- 異種DBMSからPostgreSQLへデータ移行を行う場合に発生する作業や注意点について調査
- 実際にデータ移行を試行し、調査結果の妥当性を検証



# データ移行調査および実践編（2/3）

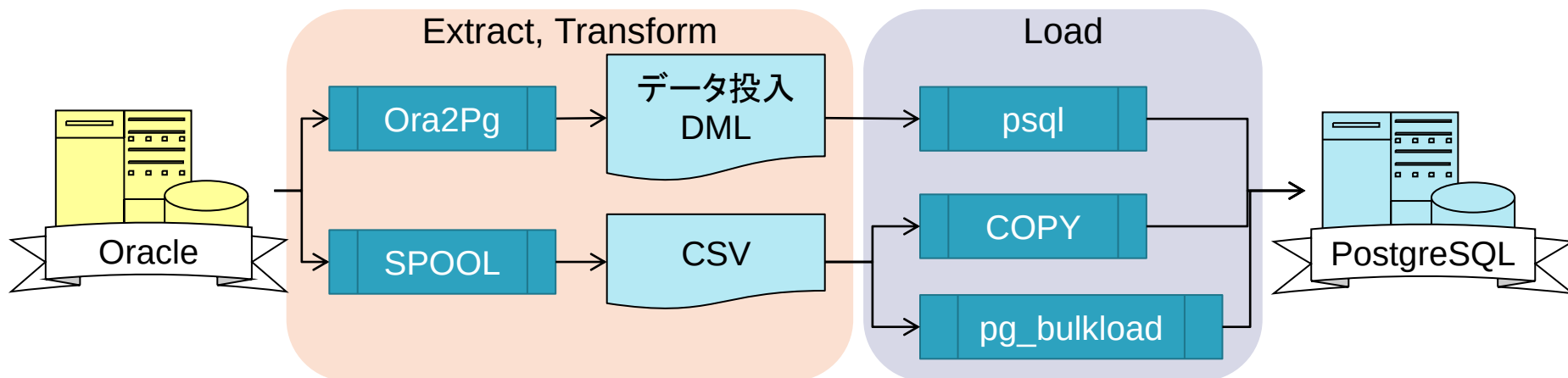
## ■ 移行調査編

- Oracle, SQLServerからPostgreSQLへのデータ移行
- データ移行作業の手順と注意点を解説
- データの抽出 (Extract)
  - CSVへ出力する際の表現形式
  - 中間データへ出力するツール (SP00L, Ora2Pg, bcpユーティリティ)
- データの変換 (Transform)
  - 中間データの文字エンコーディングの変換
- データの投入 (Load)
  - データをロードするツール (COPY, psql, pg\_bulkload)
- データ移行後の作業
  - データロード後に確認すべき点 (エラー有無、投入行数)
  - VACUUM, ANALYZE の実施

# データ移行調査および実践編（3/3）

## ■ 移行実践編

- OracleからPostgreSQLへのデータ移行を実践
- 移行元DBはJdbcRunner同梱の「TinyTPC-C」で作成



## ■ 実施結果

- 各種ツールで正常に移行でき、ツールの利便性を確認
- 用途によっては使いづらいケースもあった。
- ツール利用時は操作方法及びフィジビリティの確認が重要



# アプリケーション移行実践編（2/5）

- SQL変更箇所の抽出に db\_syntax\_diff を使用
- db\_syntax\_diff
  - Oracle DatabaseからPostgreSQLへアプリケーションを移行する際に修正が必要となる箇所を報告するツール
  - NTT OSSセンタで開発されたOSS
  - ライセンスは The PostgreSQL License
  - <https://github.com/db-syntax-diff> より取得可能
  - アプリケーションのソースコードを指定して実行すると、修正が必要なSQLのパターンとマッチングして報告。
  - 修正が必要なSQLのパターンはXML形式で定義するため、必要に応じてチェックパターンのカスタマイズが可能

# アプリケーション移行実践編（3/5）

## ■ db\_syntax\_diff の出力例（一部抜粋）

|                                                              |      |    |      |           |                                           |
|--------------------------------------------------------------|------|----|------|-----------|-------------------------------------------|
| /home/postgres/b4/src/com/commander4j/db/JDBTable.java       | 83   | 7  | SQL  | WARNING   | user_idのデータ・ディクショナリ・ビューまたは動的パフォーマンス・ビュー   |
| /home/postgres/b4/src/com/commander4j/db/JDBTable.java       | 83   | 15 | SQL  | WARNING   | user_commentのデータ・ディクショナリ・ビューまたは動的パフォーマンス  |
| /home/postgres/b4/src/com/commander4j/db/JDBDespatch.java    | 428  | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |
| /home/postgres/b4/src/com/commander4j/db/JDBQuery.java       | 67   | 13 | SQL  | ERROR LV1 | 全列指定は未サポートです。SELECT句にすべてのカラム名を指定してく       |
| /home/postgres/b4/src/com/commander4j/db/JDBQuery.java       | 77   | 23 | SQL  | LOW1      | ROWNUM擬似列は未サポートです。                        |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameLoca | 239  | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameLoca | 248  | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameLoca | 297  | 0  | FUNC | WARNING   | このまま移行する場合、引数のデータ型の確認が必要です。なぜなら、C         |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameQMIn | 449  | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameQMIn | 466  | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameQMIn | 485  | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameProc | 238  | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameMHN  | 252  | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameMHN  | 261  | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameMHN  | 314  | 0  | FUNC | WARNING   | このまま移行する場合、引数のデータ型の確認が必要です。なぜなら、C         |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameMHN  | 1115 | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameQMF  | 78   | 82 | SQL  | WARNING   | User_Data_1のデータ・ディクショナリ・ビューまたは動的パフォーマンス・ビ |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameQMF  | 78   | 95 | SQL  | WARNING   | User_Data_2のデータ・ディクショナリ・ビューまたは動的パフォーマンス・ビ |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameQMF  | 166  | 80 | SQL  | WARNING   | USER_DATA_1のデータ・ディクショナリ・ビューまたは動的パフォーマンス   |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameQMF  | 166  | 92 | SQL  | WARNING   | USER_DATA_2のデータ・ディクショナリ・ビューまたは動的パフォーマンス   |
| /home/postgres/b4/src/com/commander4j/app/JInternalFrameMate | 230  | 7  | SQL  | LOW2      | FROMの省略は未サポートです。                          |

# アプリケーション移行実践編（4/5）

## ■ 移行実施の手順

- db\_syntax\_diffをJavaソースコードと環境構築用のSQLファイル(スキーマ定義DDL、データ投入DML)に対して実行
- OracleのJDBCドライバクラスを呼び出す部分を修正
- db\_syntax\_diffの実行結果をもとに、SQL文を修正
- 修正後アプリケーションの動作確認
  - Commander4j の基本機能が正常動作するか？

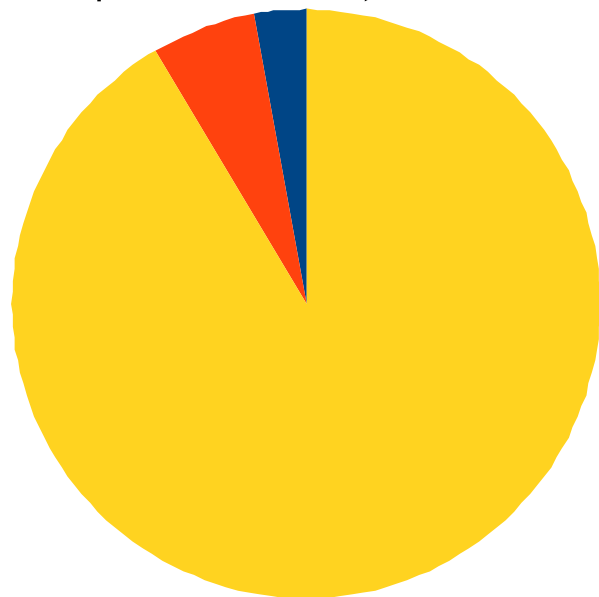
# アプリケーション移行実践編（5/5）

## ■ 移行実施結果

### □ テスト工程が全工程の90%以上を占めた

- ツールでSQL修正箇所を自動検出し、修正作業が短く済んだ。
- 修正有無に関わらず、関連するSQLはすべてテストを実施した。
- db\_syntax\_diff で検出できなかった箇所はテスト工程で対応した。

各フェーズの所要時間比率



| 項番 | 作業項目                       | 所要時間比率 |
|----|----------------------------|--------|
| #1 | AP修正(PostgreSQLへの接続)       | 2.8%   |
| #2 | AP修正(db_syntax_diffに基づく修正) | 5.8%   |
| #3 | テスト・およびエラー修正               | 91.4%  |

- 移行作業項番#1
- 移行作業項番#2
- 移行作業項番#3

# PostgreSQL移行の成否を 分けるポイントとは？

# PostgreSQL移行の成否とは？

- そもそもどういう形で終われば移行に成功した、と言えるのでしょうか？
- 莫大な費用をかけて移行できても、それは成功と言えるのでしょうか？
- 移行の成功とは、

# PostgreSQL移行の成否とは？

- そもそもどういう形で終われば移行に成功した、と言えるのでしょうか？
- 莫大な費用をかけて移行できても、それは成功と言えるのでしょうか？
- 移行の成功とは、

**当初見積りと実作業量の差が小さい**

# PostgreSQL移行の成否を分けるポイント

## ■ アセスメントの精度を高める

- 当初見積りを正確にする
- 移行元、移行先システムの要件を正しく把握する
- 移行元DBMSとPostgreSQLの機能差異を理解しておく
- 移行ツールをうまく活用する

## ■ 動作検証、テストの時間を十分に確保する

- 結局実作業量は増えることを見越しておく
- アプリケーション移行実践編では、  
テスト工程が全体の9割以上という結果に
- 移行ツールの利用で修正の実作業量は縮小
- 移行ツールで吸収できないケースへの対応は必ず発生

# 参加各社からの声 「WG2の活動を振り返って」

# WG2の活動で苦勞したこと

## 1. 作業時間の捻出

- ミーティング、調査時間を確保するのが大変だった。  
(NECソフト、クオリカ)
- WG1とWG2を兼務しており作業も多く大変だった。  
社内リソースも含め作業分担をしっかりと、  
より参加しやすくなると思う。(NEC)
- 通常業務をこなしながら、2週に1度の検討会に  
向けた活動を実施していくのは大変だった。  
(NTT SOFT)

# WG2の活動で得られたこと

## 1. 技術者交流の機会

- PostgreSQLをはじめとするDBMSに造詣の深い、他社の技術者と議論する場が得られた。  
(NECソフト、SRA OSS、富士通SSL、サイオス、NTT SOFT)

## 2. 実用性の高い技術情報

- これまで知らなかったツールの知識やノウハウを得た。  
(NECソフト、NTT SOFT)
- db\_syntax\_diff 等のツールについて知ることが出来、実際のシステム移行案件にも役立てることができた。(クオリカ)
- 他社が持つ成果物を参照する機会があり、参考になった。  
(NTT SOFT)

# WG2の活動で得られたこと

## 3. PostgreSQLに対する理解度の向上

- 成果物としてまとめることで、PostgreSQLに対する理解が深まった。(NEC)
- 自身の理解が不足している点を、他社のDB技術者と交流することで気づくことができた。(NEC)
- 参加企業で作業を分担することで、1社では調査できない範囲のノウハウを得られた。(アシスト)

## 4. ドキュメンテーション技術の向上

- 成果物作成を通じ技術文書のまとめ方が参考になった。(アシスト)
- 書籍を出されている方も少なくなかった為、ドキュメンテーションのテクニックや、レビューのコメントなどが大変勉強になった。(サイオス)



**PGECons**  
PostgreSQL Enterprise Consortium

**PGEConsと一緒に  
活動しませんか？**

**Java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。**

**Microsoft、Windows Server、SQL Server、米国 Microsoft Corporationの米国及びその他の国における登録商標または商標です。**

**MySQL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。**

**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。**

**TPC, TPC Benchmark, TPC-C, TPC-E, tpmC, TPC-H, QphHは米国Transaction Processing Performance Councilの商標です**

**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。**