# 【B3】SQL Server から PostgreSQLへの移行

PostgreSQL Conference Japan 2020

株式会社インプリム
CEO 内田 太志

# 自己紹介

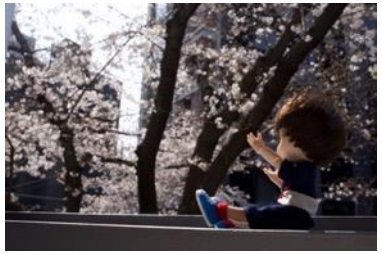
株式会社インプリム

CEO 内田 太志

中野が大好き♥

## 1998年　株式会社富士通エフサス（約19年）

- 主にインフラの運用、保守、プロマネ
- 効率化大好き！　プログラミング好き
- ASP.NET / SQL Server / VB



## 2017年　株式会社インプリム設立　～　現在４期目

Pleasanter

Business Application Platform

オープンソースのWebデータベース

プリザンター開発中！

# インプリムとは

## 業務内容： マネジメント快適化

- OSS プリザンターの開発
- プリザンターのサポート
- 商用ライセンスの提供

## 業務絶賛拡大中

- 創立3年で3つの名誉ある受賞
- C#エンジニア・プリセールス・営業募集中！



# プリザンターとは

**ノンプログラミング**で素早く簡単に**業務アプリケーション**が作成できる**オープンソースソフトウェア**です。 Excel や E-mail で行っていた業務を**高機能なWebシステム**に置き換えることが可能です。

## Web画面を素早く簡単に作成


自由に項目が設定できる編集画面

見やすく使い勝手のよい一覧画面

## 複数のデータを自在に紐づけ


マウス操作で簡単に紐づけ

いつでも柔軟に拡張可能

紐づけの階層に制限なし

関連データをCSV出力可能

**オープンソース（AGPL）で、無料で利用可能！**

# アプリケーションの移行

| Pleasanter<br>.NET Framework版 | Pleasanter<br>.NET Core版 |
| --- | --- |
| IIS | Nginx |
| ASP.NET<br>.NET Framework<br>C# | ASP.NET<br>.NET Core<br>C# |
| SQL Server | PostgreSQL |
| Windows | Cent OS |

# 様々な環境で動作可能

## マルチプラットフォーム対応、Azure上でのサーバレス構成も可能

### ■.NET Framework版(.NET Framework 4.5以上)**

**-- Windows --**

| | 名称 |
|---|---|
| OS | Windows Server 2012 R2 / 2016 / 2019 |
| Webサーバ | Internet Information Services 8.0 / 8.5 / 10.0 |
| DBサーバ | SQL Server 2016 / 2017 / 2019 |

**-- Microsoft Azure サーバレス構成 --**

| | 名称 |
|---|---|
| Webサーバ | Azure App Service(ASP.NET) |
| DBサーバ | Azure SQL Database |

### ■.NET Core版(.NET Core 3.1以上)**

**-- Windows --**

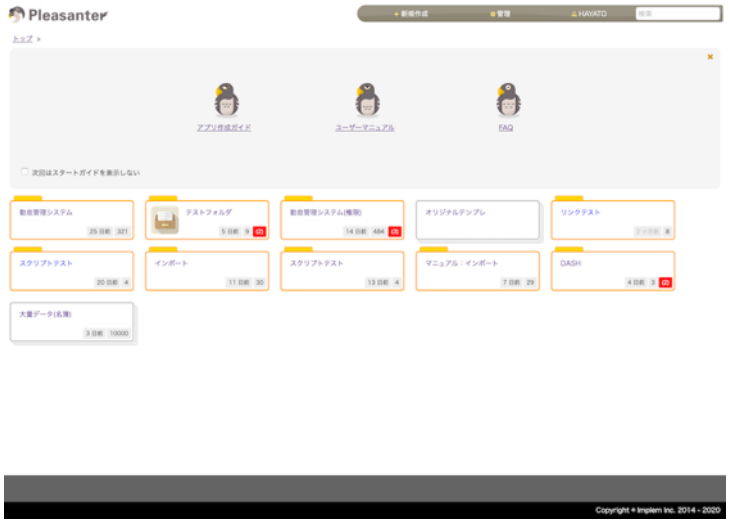
| | 名称 |
|---|---|
| OS | Windows Server 2012 R2 / 2016 / 2019 |
| Webサーバ | Internet Information Services 8.0 / 8.5 / 10.0 |
| DBサーバ | PostgreSQL 11 / 12 |
| | FUJITSU Software Enterprise Postgres 11 |
| | SQL Server 2016 / 2017 / 2019 |

**-- Linux --**

| | 名称 |
|---|---|
| OS | Red Hat Enterprise Linux 8 |
| | CentOS 7 / 8 |
| Webサーバ | Nginx 1.16 |
| | Apache 2.4 |
| DBサーバ | PostgreSQL 11 / 12 |
| | FUJITSU Software Enterprise Postgres 11 |
| | SQL Server 2019 |

# Pleasanter

# DEMO



# 移行要件

- ✓ データベースのスキーマはなるべく変えずに、アプリケーションの変更で対応する。
- ✓ SQL ServerとPostgreSQLの両方に1つのプログラムで対応できるようにする。
- ✓ オンライン処理のDBアクセスだけでなく、インストール時のDDLにも対応する。
- ✓ 今後、他のDBMSへの対応も考慮した作りにする。
- ✓ SQL ServerからPostgreSQLへデータを移行できるようにする。

# プリザンターの移行でわかった
# .NETアプリケーションの
# SQL ServerからPostgreSQLへ
# 移行時の注意点


# テーブル名、カラム名に大文字

Pleasanterではテーブル名やカラム名の命名規則にアッパーキャメルケースを用いており、PostgreSQLではSQLが動かない。

SQL ServerのSQLではテーブル名やカラム名の大・小文字を識別しない

```
select * from Items
```

プリザンターは[]で囲うスタイルを採用 → PostgreSQLでは動かない

```
select * from [Items]
```

ダブルクォートに変更する事でSQL Server/PostgreSQLの両方で動作

```
select * from "Items"
```

# ユーザ名等の大文字・小文字

Pleasanterではユーザ名の大文字・小文字を区別しない仕様だが、
PostgreSQLでそのまま動かすと区別されるため認証できない。

SQL ServerのSQLで大・小文字を識別しない照合順序を使用

```
select * from Users where LoginId = @LoginId
```

プリザンターでは大文字・小文字を変換せずに照合

```
select * from [Users] where [Users].[LoginId] = @LoginId
```

lower関数を使用することでSQL Server/PostgreSQLの両方で動作

```
select * from "Users"
where lower("Users"."LoginId") = lower(@LoginId)
```

# bool値の型誤り

Pleasanterではbool型の項目の照合の一部に 1 / 0 を使用していたが、PostgreSQLでは動かない。

SQL ServerのSQLでBool型の項目は 1 / 0 で抽出できる

```
select * from Issues where CheckA = 0
```

プリザンターではクエリのパラメータにInt型の 0 を指定（C#）

```
select * from [Issues] where [Issues].[CheckA] = @CheckA
```

パラメータにBool型で渡すことでSQL Server/PostgreSQLの両方で動作

```
select * from "Issues" where "Issues". "CheckA" = @CheckA
```

# シーケンスの番号取得

Pleasanterではシーケンスが設定されたテーブルでinsertした後に番号を@@identityで取得しているがPostgreSQLで動かない。

SQL ServerのSQLで@@identityが使用できる

```
select @@identity
```

プリザンターではinsert後にこれを使用してidを取得している

```
select @@identity
```

PostgreSQLの場合だけRETURNINGに切り替える処理を実装

```
returning "ReferenceId"
```

# top句

Pleasanterではselect結果の先頭のレコードを取得するためにtop句を用いており、PostgreSQLでは動作しない。

SQL ServerのSQLでtop句が使用できる

```
select top 20 * from Issues
```

プリザンターでもtop句を使用してページング等を行っている

```
select top 20 * from [Issues]
```

PostgreSQLの場合だけtop句からlimit句に切り替える処理を実装

```
select * from "Issues" limit 20
```

# 主キーの制約

Pleasanterでは主キーに降順インデックスを指定しており、DDLでdescの指定をしているがPostgreSQLでは動かない。

SQL Serverの主キーに降順インデックスを指定できる

```
alter table Issues add constraint PK primary key clustered
(SiteId, UpdatedTime desc, IssueId)
```

プリザンターでも主キーに降順インデックスを使用している

```
alter table [Issues] add constraint [PK] primary key clustered
([SiteId], [UpdatedTime] desc, [IssueId])
```

PostgreSQLの場合は主キーに降順インデックスを使用しない

```
alter table "Issues" add constraint "PK" primary key
("SiteId", "UpdatedTime", "IssueId")
```

# フルテキスト検索

PleasanterではSQL Serverのフルテキスト検索を使用しており、
PostgreSQLでは動作しない

SQL Serverはオプションでフルテキスト検索が使用可能

```
select * from Items where contains(*,@SearchText)
```

プリザンターではcontains句でフルテキスト検索を実装

```
select * from [Items] where contains([Items].[FullText],@SearchText)
```

PostgreSQLの場合だけpg_trgmをインストールして切り替え処理を実装

```
select * from "Items" where ("Items"."FullText" %> @SearchText)
```

# 組み込み関数の違い

SQL Serverの組み込み関数の一部はPostgreSQLで動作しない。

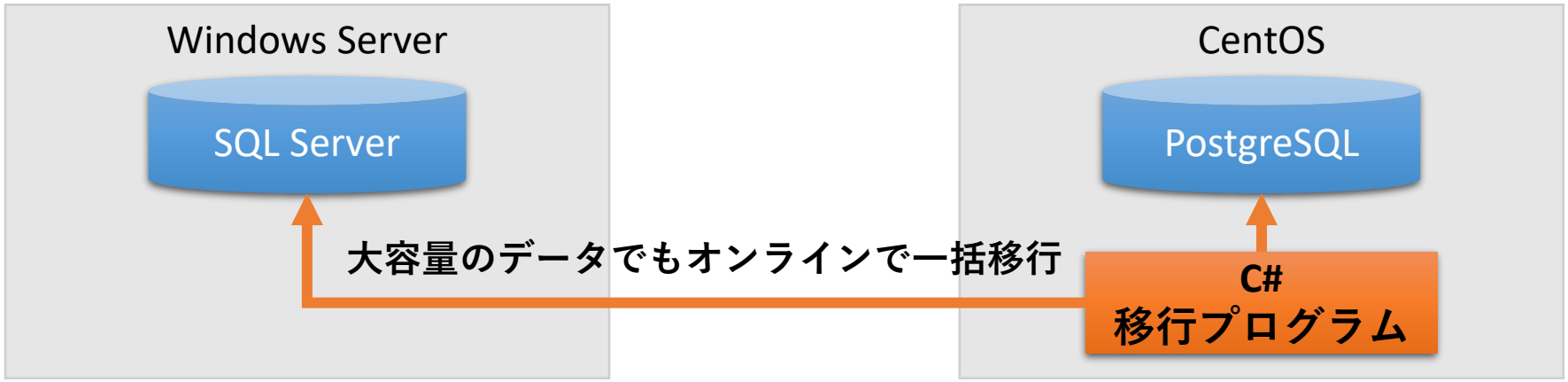
| | SQL Server | PostgreSQL |
|---|---|---|
| 1 | getdate | CURRENT_TIMESTAMP |
| 2 | dateadd | + interval '3 day' |
| 3 | try_cast | case |

# インストール時のDDLの違い

| No | 項目 | SQL Server | PostgreSQL |
|---|---|---|---|
| 1 | スキーマ情報の取得 | sys.columns / sys.tables / sys.types | information_schema.columns |
| 2 | テーブル名 | sys.tables.name | table_name |
| 3 | カラム名 | sys.columns.name | column_name |
| 4 | タイプ名称 | sys.types.name | udt_name |
| 5 | データ長 | sys.columns.max_length | character_octet_length |
| 6 | 有効桁数 / 小数点以下桁数 | sys.columns.precision / sys.columns.scale | numeric_precision / numeric_scale |
| 7 | NULL許可 | sys.columns.is_nullable | is_nullable = 'YES' |
| 8 | IDENTITY | sys.columns.is_identity | is_identity = 'YES' |
| 9 | PK名、インデックス名 | 最大128Byte | 最大63Byte |
| 10 | ユーザ作成 | create login | create user |

# データ移行

C#のプログラムを使用し、SQL Server to PostgreSQLの移行を可能にした。
PostgreSQLのシーケンスのsetvalがハマリポイント。




| No | 項目 | SQL Server | PostgreSQL |
|---|---|---|---|
| 1 | C# 接続クラス | SqlConnection | NpgsqlConnection |
| 2 | C# コマンドクラス | SqlCommand | NpgsqlCommand |
| 3 | シーケンス | SET IDENTITY_INSERT TableName ON/OFF | INSERT後にsetval |

https://github.com/Implem/Implem.Pleasanter.NetCore/blob/master/Implem.CodeDefiner/Functions/AspNetMvc/CSharp/Migrator.cs

# Thank you for your attention

プリザンター　検索