Migration de code

Aide à la migration de code en listant les équivalences

- Postgre SQL

Postgre SQL

TOP

Limiter le nombre de résultat de la requête

```
-- TSQL
SELECT TOP 0
FROM filtres
ORDER BY aaa;

-- Postgre SQL
SELECT *
FROM filtres
ORDER BY aaa
LIMIT 0
```

PRINT

Affiche un message dans la console

```
-- TSQL
PRINT '-- Suppression de la table temporaire _filtres';

-- Postgre SQL
RAISE NOTICE '-- Suppression de la table temporaire _filtres';
```

SELECT INTO

Créée une table à partir du résultat d'une requete

```
-- TSQL
SELECT TOP 0 * INTO _filtres FROM filtres;

-- Postgre SQL
CREATE TABLE _filtres
AS (
    SELECT *
    FROM filtres
    LIMIT 0
);
```

IF ... ELSE

Documentation disponible ici: https://www.postgresql.org/docs/7.2/plpgsql-control-structures.html

```
-- TSQL
IF (ma condition) BEGIN

END

-- Postgre SQL
IF ma condition THEN
ELSIF ma condition2 THEN
ELSE
END
```

ISNULL()

```
-- TSQL
SELECT t.carticle,
ISNULL(t.CDEPOT, 0) AS cdepot

-- Postgre SQL
SELECT t.carticle,
  COALESCE(t.CDEPOT, 0) AS cdepot
```

SCRIPT PL/PGSQL

Pour exécuter un script, il est important d'englober dans les balises suivantes:

```
DO $$
  -- Déclarer ici toutes les variables utilisées dans le script
BEGIN
  -- Contenu du script
END $$

-- Exemple:
DO $$

  DECLARE nbLignes int = 0;

BEGIN
  -- Supprimer les lignes autogénérées
  RAISE NOTICE '-- Supprimer les lignes autogénérées';
  DELETE FROM multifam
  WHERE multifam.refresh=1;

  GET DIAGNOSTICS nbLignes = ROW_COUNT;
  RAISE NOTICE '-- % lignes supprimées', nbLignes;
END $$
```

Variables

Assigner une valeur à une variable

```
-- TSQL
DECLARE @myvar nvarchar(50) = '';

SET @myvar = 'test';
```

```
-- Postgre SQL
DO $$
  DECLARE myvar varchar(50) = '';
BEGIN
  myvar := 'test';
END $$
```

Concaténer du string

```
-- TSQL
DECLARE @myvar NVARCHAR(MAX) = '';

SET @myvar += 'str1' + 'str2' ;

SET @myvar += CONCAT('str1', 'str2');

PRINT @myvar;

-- Postgre SQL
DO $$
  DECLARE myvar text = '';
BEGIN
  myvar := myvar || 'str1' || 'str2';

  myvar := CONCAT(myvar, 'str1', 'str2');

  RAISE NOTICE '%', myvar;
END $$;
```

@@ROWCOUNT

Permet d'obtenir le nombre de lignes modifiées par la requête précédente

```
-- TSQL
DELETE FROM multifam
WHERE refresh=1;
```

```
PRINT('-- ' + CAST(@@ROWCOUNT AS VARCHAR) + ' lignes supprimées');

-- Postgre SQL
DELETE FROM multifam
WHERE refresh=1;

GET DIAGNOSTICS nbLignes = ROW_COUNT;
PRINT '-- % lignes supprimées', nbLignes;
```

GetDate()

```
-- TSQL
SELECT GetDate();

-- Postgre SQL
SELECT NOW();
```

TRUNCATE

Utiliser RESTART IDENTITY pour réinitialiser les compteurs.

```
-- TSQL
TRUNCATE TABLE cli_modele;

-- Postgre SQL
TRUNCATE TABLE cli_modele RESTART IDENTITY;
```

ALTER TABLE

```
-- TSQL
ALTER TABLE LK_PCDE_EVT ALTER COLUMN numero INT;
```

```
-- Postgre SQL
ALTER TABLE LK_PCDE_EVT ALTER COLUMN numero TYPE int USING numero::integer
```