Transpilation

D’un dialecte SQL à un autre

Florent Jardin, Étienne Bersac

mercredi 7 février 2024

Qui sommes-nous ?

- @bersace Marmotte 🥔. Prêt à livrer ! code ⛳🗑️
- @fljdin Database inspired, powered by passion and curiosity

Sommaire

- 1) Concepts
- 2) Outils open-source
- 3) Cas concrets

1) Concepts

Dialectes SQL

- Standard ISO/IEC 9075-1:2023 (juin 2023)
- PostgreSQL tend à couvrir la totalité de la norme
- … les autres systèmes aussi

```sql
SELECT *
  FROM t1, t2
 WHERE t1.col1 = t2.col3 (+);
```

```sql
SELECT lastname, job, IF(active, 'yes', 'no')
  FROM employees;
```

```
USE AdventureWorks2022;
GO
CREATE PROCEDURE HumanResources.uspGetEmployeesTest2
    @LastName nvarchar(50),
    @FirstName nvarchar(50)
AS
    SET NOCOUNT ON;
    SELECT FirstName, LastName, Department
      FROM HumanResources.vEmployeeDepartmentHistory
     WHERE FirstName = @FirstName AND LastName = @LastName
       AND EndDate IS NULL;
GO
```

Transpiler

Deux stratégies reconnues

- Expressions rationnelles
- Analyse syntaxique

Expressions rationnelles

- Décrire des ensembles réguliers d'une chaîne de texte
- Détecter la présence d'un motif (*pattern*)
- Valider la forme d'une chaîne
- Remplacer un sous-esemble par un autre

```
(\b25[0-5]|\b2[0-4][0-9]|\b[01]?[0-9][0-9]?)(\.(25[0-5]|2[0-4][0-9]|
[01]?[0-9][0-9]?)){3}
```

```
^[\w\-\.]+@([\w-]+\.)+[\w-]{2,}$
```

Arbre syntaxique

- AST : *Abstract syntax tree*
- analyse lexicographique et sémantique
- représentation structurée du code
- compilateur, linteur, transpileur

Analyse lexicale : lexer

lexicale : identifier la nature des mots et des symboles (*tokens*)

```
>>> lexer("SELECT 1, TRUE, NULL, 'chaine', col FROM t1;")
[SELECT, 1, TRUE, NULL, 'chaine', "col", 'FROM', "t1", ;]
```

Analyse syntaxique : parser

syntaxique : grouper les mots sémantiquement

```
>>> parse([SELECT, 1, TRUE, NULL, 'chaine', "col", 'FROM', "t1", ;])
- SELECT:
  - 1
  - TRUE
  - NULL
  - 'chaine'
  - "col"
  FROM:
  - "t1"
  ";"
```

Édition de l’AST

Transpiler: ajouter, modifier ou supprimer un nœud de l’AST.

```
>>> rewrite([SELECT [SYSDATE] FROM [DUAL]])
SELECT
- CURRENT_TIMESTAMP
```

2) Les outils open-source

- Ora2Pg de Gilles DAROLD
- sqlglot de Toby MAO

Ora2pg

- Lit le code uniquement depuis l’instance
- Expressions rationnelles
- Préserve commentaires et les indentations
- Le plus avancé dans les règles de conversion

Ora2Pg : TRUNC

```
$str =~ s/\bTRUNC\s*\($field\)/date_trunc('day', $1)/is;
if ($str =~ s/\bTRUNC\s*\($field,$field\)/date_trunc($2, $1)/is ||
    $str =~ s/\bTRUNC\((\%\%REPLACEFCT\d+\%\%)\s*,\s*(\?TEXTVALUE\d+\?)\)/date_trunc($2, $1
)
{
    if ($str =~ /date_trunc\(\?TEXTVALUE(\d+)\?/)
    {
        my $k = $1;
        $class->{text_values}{$k} =~ s/'(SYYYY|SYEAR|YEAR|[Y]+)'/'year'/is;
        $class->{text_values}{$k} =~ s/'Q'/'quarter'/is;
        $class->{text_values}{$k} =~ s/'(MONTH|MON|MM|RM)'/'month'/is;
        $class->{text_values}{$k} =~ s/'(IW|DAY|DY|D)'/'week'/is;
        $class->{text_values}{$k} =~ s/'(DDD|DD|J)'/'day'/is;
        $class->{text_values}{$k} =~ s/'(HH|HH12|HH24)'/'hour'/is;
        $class->{text_values}{$k} =~ s/'MI'/'minute'/is;
    }
}
```

sqlglot

- 20 dialectes
- Préserve commentaires et les indentations
- Traduction simple
- Optimiseur, interprêteur

Notre contribution

Besoins

- Transpiler du code SQL arbitraire
- Réécrire lourdement l’AST
- Préserver indentation, casse et commentaires
- Simplicité de l’implémentation

Hors du besoin

- Performances
- Interprétation et validation
- Optimisation de requête

Nouveau projet : transqlate

- gitlab.com/dalibo/transqlate
- CLI & API Go
- Parser TDOP : *Top-down Operator Precedence*
- Édition de l'AST
- **Alpha**

CLI

```
transqlate <file>
```

3) Cas concrets

Fonction NVL

```
SELECT NVL(description, short_description) FROM articles;
```

réécrit en:

```
SELECT COALESCE(description, short_description) FROM articles;
```

NVL : transqlate

```
rules = append(rules, RenameFunction{From: "nvl", To: "COALESCE"})
```

NVL : transqlate

```
func (r RenameFunction) Match(n ast.Node) bool {
    c, _ := n.(ast.Call)
    f, _ := c.Function.(ast.Identifier)
    return f.In(r.From) // e.g. NVL
}
```

NVL : transqlate

```
func (r RenameFunction) Rewrite(n ast.Node) (ast.Node, error) {
    c := n.(ast.Call)
    f := c.Function.(ast.Identifier)
    f.Token.Set(r.To) // e.g. COALESCE
    c.Function = f
    return c, nil
}
```

Fonction TRUNC

```
SELECT trunc(hired_date, 'Y') FROM employees;
```

réécrit en:

```
SELECT date_trunc('year', hired_date) FROM employees;
```

TRUNC : transqlate

```
func (_ replaceTrunc) Match(n ast.Node) bool {
    c, _ := n.(ast.Call)
    f, _ := c.Function.(ast.Indentifier)
    return f.In("trunc")
}
```

```go
func (_ replaceTrunc) Rewrite(n ast.Node) (ast.Node, error) {
    c := n.(ast.Call)

    // Call DATE_TRUNC.
    f := c.Function.(ast.Indentifier)
    f.Token.Set("DATE_TRUNC")
    c.Function = f

    // Swap arguments.
    c.Args.Items[0].Expression, c.Args.Items[1].Expression =
      c.Args.Items[1].Expression, c.Args.Items[0].Expression

    // Get optionnal format.
    f, _ = c.Args.Items[0].Expression.(ast.String)
    datefmt := translateDateFormat(f.Token.Str)
    f.Token.Set(lexer.QuoteString(datefmt))
    c.Args.Items[0].Expression = f

    return c, nil
}
```

Feuille de route

- `CONNECT BY`, `MINUS`, `ADD_MONTHS`, etc.
- Jointures externes, jointures récursives
- Casse des identifiants
- Compléter le lexer
- Compléter le parser
- Plus de SGBD: MySQL, SQL Server, Sybase
- Cf. gitlab.com/dalibo/transqlate/-/issues

Conclusion

- Simple et puissant
- Une contribution à l’écosystème de la migration

https://gitlab.com/dalibo/transqlate

Questions ?