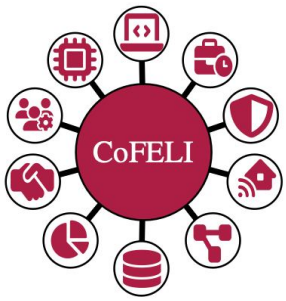

Collectif francophone pour l'enseignement libre de l'informatique

Modèle logique de données

Interface applicative pour vérifier et convertir les valeurs des types

CoFELI:MLD_11-DVCT

Christina KHNAISSER *(christina.khnaisser@usherbrooke.ca)*

Luc LAVOIE *(luc.lavoie@usherbrooke.ca)*

(les auteurs sont cités en ordre alphabétique nominal)

—

CoFELI/Scriptorum/MLD_11-DVCT, version 0.1.0.b, en date du 2025-09-01

— document de travail, ne pas citer —

Sommaire

…

Mise en garde

Le présent document est en cours d'élaboration ; en conséquence, il est incomplet et peut contenir des erreurs.

Historique

diffusion	resp.	description
2025-09-01	LL	Correction de l'inversion entre les exemples d'alimentation et de détection.
2025-05-19	LL	Réorganisation des exemples.
2022-01-23	LL	Récupération de notes diverses.

Table des matières

Introduction

Ce document présente les éléments nécessaires pour définir **DVCT**, un modèle d'interface machine-machine (IMM) pour vérifier et convertir les valeurs des types.

À développer…

Évolution du document

À venir…

Contenu des sections

- La section 1 présente l'organisation et la composition d'un DVCT.
- La section 2 présente deux exemples d'utilisation.
- La section 3 propose une mise en oeuvre.

1. DVCT

Une IMM de type DVCT est conçue pour vérifier et convertir les valeurs des types. D'où l'origine de l'acronyme DVCT (définition, vérification et conversion de types). En pratique, il est donc suggéré de définir les fonctions DVCT à la suite de type ciblé, au sein du module où le type est défini.

1.1. Fonctions ciblées

Chaque type est assorti de deux fonctions complémentaires

- la première pour déterminer la conformité d'une représentation ;
- la seconde pour déterminer la valeur associée à une représentation conforme.

Pour un type <T>, leurs entêtes sont typiquement les suivantes :

```
function <T>_CONF (argument Varchar) returns Boolean
function <T>_VAL (argument Varchar) returns <T>
```

La fonction `<T>_CONF` détermine si l'argument est conforme.

La fonction `<T>_VAL`, ayant pour antécédent que l'argument est conforme, retourne la valeur correspondant à l'argument.

1.2. Fonction combinée

Il est également suggéré de définir une fonction réunissant la vérification et la conversion.

Pour un type <T>, l'entête de la fonction est typiquement celle-ci :

```
function <T>_CONV (argument Varchar) returns <T>
```

La fonction `<T>_CONV` retourne la valeur convertie appropriée si l'argument est conforme et NULL sinon.

2. Exemples

Voici deux exemples d'alimentation simple, en SQL, utilisant ces fonctions.

Alimentation A

```
insert into ObsTemperature
  select
    Date_VAL(date) as date,
    Temperature_VAL(temp_min) as temp_min,
    Temperature_VAL(temp_max) as temp_max,
    coalesce (note, '') as note
  from CarnetMeteo
  where Date_CONF(date)
    and Temperature_CONF(temp_min)
    and Temperature_CONF(temp_max) ;
```

Alimentation B

```
insert into ObsTemperature
  with
    X as (
      select
        Date_eco_CONV(date) as date,
        Temperature_CONV(temp_min) as temp_min,
        Temperature_CONV(temp_max) as temp_max,
        coalesce (note, '') as note
      from CarnetMeteo
    )
  select * from X
  where date is not null
    and temp_min is not null
    and temp_max is not null ;
```

Voici un exemple de détection des tuples non conformes

Détection A

```
select
  date,
  temp_min,
  temp_max,
  note
from CarnetMeteo
where (NOT Date_CONF(date))
   or (NOT Temperature_CONF(temp_min))
   or (NOT Temperature_CONF(temp_max)) ;
```

3. Mise en oeuvre

3.1. Règles de dénomination

```
<fonction> ::= <type> <catégorie> <spécialisation>
<type> ::= IDENT
<catégorie> :== '_CONF' | '_VAL' | '_CONV'
<spécialisation> ::= ('_' IDENT)*
```

- L'identifiant <type> désigne le type de référence de la fonction (donc le type de sa valeur de retour).
- La <catégorie> désigne la catégorie de fonction (confirmation, valeur, conversion).
- Pour la <spécialisation>,
  - lorsqu'elle est omise, le type de l'argument faisant l'objet de la fonction est `Varchar` (ou `Text` dans certains dialectes) ;
  - lorsqu'elle comporte un seul identifiant, il désigne le type de l'argument faisant l'objet de la fonction ;
  - lorsqu'elle comporte plusieurs identifiants, le premier désigne le type de l'argument faisant l'objet de la fonction et les suivants, la caractérisation des fonctions ayant le même type d'argument.

3.2. Programmation

Exemple _CONF

```
create or replace function Trimestre_CONF (v Varchar) returns Boolean
language plpgsql as
$$
begin
  return case when CAST(v as Trimestre) is null then false else true end;
exception when others then
  return false;
end
$$;
```

Exemple _VAL

```
create or replace function Trimestre_VAL (v Varchar) returns Trimestre
  -- Attention !
  -- La fonction retourne une exception si la valeur n'est pas conforme.
return CAST(v as Trimestre)
;
```

Exemple _CONV

```
create or replace function Trimestre_CONV (v Varchar) returns Trimestre
language plpgsql as
$$
begin
  return CAST(v as Trimestre);
exception when others then
  return null;
end
$$;
```

3.3. Association d'un schéma au DVCT

Faut-il mettre le DVCT dans son propre schéma ?

Ceci entre en conflit avec les organisations de code qui privilégient de définir les types au fur et à mesure de leur utilisation par les créations de tables. L'isolation du DVCT dans un schéma simplifie cependant grandement la gestion des droits qui sont en général

- communs aux types et à leurs fonctions de vérification et de conversion ;
- distincts de ceux des tables.

Par contre, cela induit une préfixation systématique des identifiants de type (et de fonctions) par l'identifiant du schéma ("DVCT"), à moins que le dialecte ne comporte une directive non standard semblable au `set search_path` de PostgreSQL.

Les deux organisations sont légitimes :

- la création du schéma facilite la gestion des droits d'accès ;
- l'incorporation dans le schéma principal le code.

Dans le cadre du présent document, l'incorporation a été choisie.

3.4. Synthèse

À développer...

Conclusion

- Les fonctions de l'interface DVCT permettent de vérifier et de convertir les valeurs des types de façon uniforme.
- Elles peuvent être
  - adjointes aux définitions de type dans le schéma principal ou
  - regroupées avec les définitions de type dans un schéma spécifique (utilisé tant par le schéma principal que par les autres interfaces).
- Les fonctions DVCT permettent de définir commodément des fonctions d'alimentation et de détection des tuples non conformes à partir de tables initialisées à partir de fichiers externes, notamment des fichiers au format CSV, TSV et PSV.
- Ces fonctions sont typiquement ajoutées dans une interface EMIR ou regroupées dans un schéma spécifique (ALIM).
- La mise en oeuvre des fonctions DCVT peut être réalisées sans duplications des contraintes grâce à l'utilisation de la fonction CAST et du mécanisme de traitement des exceptions (ce dernier moyen impose toutefois de recourir aux PSM, donc, dans le cas de PostgreSQL au langage plpgsql).

Références

PostgreSQL_17

- https://docs.postgresql.fr/17/sql-createfunction.html
- https://docs.postgresql.fr/17/sql-createprocedure.html

Définitions

ACID

Acronyme désignant conjointement les propriétés d'atomicité, de cohérence, d'isolation et rémanence (pérennité ou *durability* en anglais) relativement au traitement transactionnel.

ASTBD

Application ou service tributaire d'une base de données.

ÉMIR

Une interface machine-machine (IMM) permettant de consulter (évaluer, extraire), mettre à jour (modifier), créer (insérer) et déclasser (retirer) des informations (des données) depuis un système d'information (en particulier depuis une base ou un entrepôt de données).

ÉMIRA

Variante de l'ÉMIR auxquelles s'ajoutent des routines d'archivage.

IMM

Interface machine-machine (interface de programmation ou *application programming interface* - API), par opposition à une interface personne-machine (IPM).

PSM

Permanently stored module, sous-langage de définition de procédures et fonctions défini par le langage SQL.

Produit le 2025-10-03 11:14:05 -0400

Collectif francophone pour l'enseignement libre de l'informatique