

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)

CoFELI/Scriptorum/MLD_11-DVCT, version 0.1.0.a, en date du 2024-05-19

Ń document de travail, ne pas citer Ń

Plan

Introduction	3
1. DVCT	4
2. Exemples	7
3. Mise en oeuvre	10
Conclusion	19
Références	21
Définitions	22

Introduction

Ce document présente les éléments nécessaires pour définir une interface applicative (API), aussi appelée interface machine-machine (IMM), pour vérifier et convertir les valeurs des types.

1. DVCT

Une IMM de type DVCT est conçue pour vérifier et convertir les valeurs des types. Pour cette raison, il est fortement suggéré de définir les fonctions à la suite des types cibles. D'où l'origine de l'acronyme DVCT (définition, vérification et conversion de types).

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 $\langle T \rangle$, leurs entités sont typiquement les suivantes:

```
function  $\langle T \rangle$ _CONF (argument Varchar) returns Boolean  
function  $\langle T \rangle$ _VAL (argument Varchar) returns  $\langle T \rangle$ 
```

La fonction $\langle T \rangle$ _CONF détermine si l'argument est conforme.

La fonction $\langle T \rangle$ _VAL, ayant pour antécédent que l'argument est conforme, retourne la valeur correspondant à l'argument.

1.2. Fonction combinée

Chaque type est assorti d'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 utilisant ces fonctions.

Alimentation A

```
insert into ObsTemperature
Ê select
Ê   Date_VAL(date) as date,
Ê   Temperature_VAL(temp_min) as temp_min,
Ê   Temperature_VAL(temp_max) as temp_max,
Ê   coalesce (note, '') as note
Ê from CarnetMeteo
Ê where (NOT Date_CONF(date))
Ê      or (NOT Temperature_CONF(temp_min))
Ê      or (NOT Temperature_CONF(temp_max)) ;
```

Alimentation B

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

Voici un exemple de dŽtections des tuples non conformes

DŽtection A

```
select  
Ê date,  
Ê temp_mi n,  
Ê temp_max,  
Ê note  
from CarnetMeteo  
where Date_CONF(date)  
Ê and Temperature_CONF(temp_mi n)  
Ê and Temperature_CONF(temp_max) ;
```

3. Mise en oeuvre

3.1. Rgles de dŽnomination

```
<foncti on> ::= <type> <catŽgori e> <spŽci al i sati on>  
<type> ::= IDENT  
<catŽgori e> ::= '_CONF' | '_VAL' | '_CONV'  
<spŽci al i sati on> ::= ('_' IDENT)*
```

- ¥ L'identifiant <type> dŽsigne le type de rŽfŽrence de la fonction (donc le type de sa valeur de retour).
- ¥ La <catŽgorie> dŽsigne la catŽgorie 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
$$
begin
  Ê return case when CAST(v as Trimestre) is null then false else
true end;
exception when others then
  Ê return false;
end
$$;
```

Exemple_VAL

```
create or replace function Trimestre_VAL (v Varchar) returns  
Trimestre  
Ê -- Attention !  
Ê -- 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  
$$  
begin  
    Ê return CAST(v as Trimestre);  
exception when others then  
    Ê return null;  
end  
$$;
```

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"), ^ 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. Synthes

Developer

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 réalisée 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).

RŽfŽrences

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 de persistance (ou *durability* en anglais) relativement au traitement transactionnel.

ASTBD

Application ou service tributaire d'une base de données.

fMIR

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).

fMIRA

Variante de l'fMIR 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 2025-05-21 10:56:08 UTC

Collectif francophone pour l'enseignement libre de l'informatique