



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

— document de travail, ne pas citer —

Plan

Introduction	3
1. DVCT	4
2. Exemples	7
3. Mise en oeuvre	9
Conclusion	18
Références	19
Définitions	20

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 ciblés. 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

- une déterminer la conformité d'une représentation ;
- l'autre pour déterminer la valeur associée à une représentation conforme.

Pour un type $\langle T \rangle$, leurs entêtes 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 $\langle T \rangle$, l'entête de la fonction est typiquement celle-ci :

```
function  $\langle T \rangle$ _CONV (argument Varchar) returns  $\langle T \rangle$ 
```

La fonction $\langle T \rangle$ _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
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 ;
```


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 Note_CONF (v VARCHAR) returns Boolean
language plpgsql as
$$
begin
    return case when CAST(v as Note) is null then false else true
end;
exception when others then
    return false;
end
$$;
```

Exemple_VAL

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

Exemple_CONV

```
create or replace function Note_CONV (v VARCHAR) returns Note
language plpgsql as
$$
begin
    return CAST(v as Note);
exception when others then
    return null;
end
$$;
```

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

La définition d'une IMM de type DVCT offre plusieurs avantages, dont ceux-ci :

- *À développer...*

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 (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).

Produit le 2025-05-21 06:10:15 UTC



Collectif francophone pour l'enseignement libre de l'informatique