

Bases de données *SQL*

LDD

Fonctions et procédures

BD110
v121c

2022-04-01

Département d'informatique
Faculté des sciences



Christina.Khnaisser@USherbrooke.ca
<http://info.USherbrooke.ca/ckhnaisser>
Luc.Lavoie@USherbrooke.ca
<http://info.USherbrooke.ca/llavoie>

PLAN

- Motivation
- Un exemple
- Les fonctions
- Les procédures
- Exemples
- Références



FONCTIONS ET PROCÉDURES

MOTIVATION

- Qu'ont en commun une fonction, une procédure et un automatisme ?
 - Ce sont des abstractions paramétrables d'un traitement.
- Qui fait quoi ?
 - Une *fonction* **calcule une valeur** suite à un appel (sur la base de ses paramètres ou de l'état de la BD).
 - Une *procédure* **change l'état de la BD** suite à un appel (sur la base de ses paramètres **et** de l'état de la BD).
 - Un *automatisme* change l'état de la BD **suite à un évènement** (sur la base de ses paramètres et de l'état de la BD).

PORTÉE

- Le présent module traite des fonctions et des procédures.
- Le module BD111 traitera des automatismes.

EXEMPLE 0

DOMAINES SQL

Définition de domaines utilisés dans le cadre des prochains exemples.

Le numéro de téléphone ne comprend que les chiffres qui le composent (aucune marque d'édition telle que tirets, parenthèses, points, etc.).

Les numéros des pays membres de l'ITU comprennent entre 8 et 13 chiffres.

Les chiffres des numéros canadiens et états-uniens se répartissent ainsi :

- indicatif national : 1;
- indicatif régional : 3;
- équipement : 7.

```
CREATE
  DOMAIN Telephone
  VARCHAR(13)
  CONSTRAINT Telephone_ITU
  CHECK(VALUE SIMILAR TO '[0-9]{8,13}')
```

```
CREATE
  DOMAIN Telephone_CA
  VARCHAR(13)
  CHECK(VALUE SIMILAR TO '[1][0-9]{10}');
```

```
CREATE
  DOMAIN Montant_CAD
  NUMERIC(12,2)
  CHECK(0 <= VALUE);
```

EXEMPLE 0

FONCTION SQL

Détermination de l'indicatif régional d'un numéro de téléphone canadien ou états-unien.

```
CREATE OR REPLACE FUNCTION IndicatifRegional
(t Telephone_CA)
RETURNS VARCHAR(3)
LANGUAGE SQL AS
'SELECT SUBSTR(t, 2, 3)'
```

EXEMPLE 0

FONCTION SQL

Calcul du prix d'un produit p à une date d en fonction des promotions enregistrées. Le prix de base est utilisé à moins qu'il n'y ait au moins une promotion applicable, auquel cas la plus basse des promotions est utilisée. Il n'en résulte donc pas nécessairement une baisse de prix : si toutes les promotions applicables comportent un prix plus élevé que le prix de base, alors le "rabais" lui sera supérieur !

```
CREATE OR REPLACE FUNCTION Rabais
  (p NoProduit, d DATE)
  RETURNS Montant_CAD
LANGUAGE SQL AS
$$
  SELECT
    CAST
      ( COALESCE
        ( SELECT MIN(R.prix)
          FROM Produit
            JOIN Promotion AS R USING(noProduit)
          WHERE (noProduit = p)
              AND d BETWEEN R.début AND R.fin
        ,
          SELECT prix
            FROM Produit
          WHERE noProduit = p
        )
      AS Montant_CAD
  )
$$
```

EXEMPLE 0

PROCÉDURE SQL

La procédure d'entretien mensuel consiste à

- appliquer une variation uniforme du prix des produits.
- supprimer les produits périmés à partir de la date spécifique.

```
CREATE OR REPLACE PROCEDURE Entretienir
    (taux NUMERIC(6,4) , d DATE)
LANGUAGE SQL AS
    $$
        UPDATE Produit
            SET prix = prix * taux;
        DELETE FROM Produit
            WHERE peremption < d ;
    $$
```


FONCTION – SYNTAXE SIMPLIFIÉE (POSTGRESQL)

function_def ::=

```
CREATE [ OR REPLACE ] FUNCTION name ( [ argument [, ...] ] )  
RETURNS ret_type  
{ other_fun_qual }  
LANGUAGE lang_name  
AS body
```

argument ::=

```
[ arg_mode ] [ arg_name ] arg_type [ DEFAULT default_expr ]
```

arg_mode ::=

```
IN | VARIADIC
```

ret_type ::=

```
type_denotation
```

```
| TABLE ( [ column_name column_type [, ...] ] )
```

PROCÉDURE – SYNTAXE SIMPLIFIÉE (POSTGRESQL)

procedure_def ::=

```
CREATE [ OR REPLACE ] PROCEDURE name ( [ argument [, ...] ] )  
  { other_proc_qual }  
LANGUAGE lang_name  
AS body
```

argument ::=

```
[ arg_mode ] [ arg_name ] arg_type [ DEFAULT default_expr ]
```

arg_mode ::=

```
IN | VARIADIC | OUT
```

CORPS DE ROUTINE SYNTAXE (POSTGRESQL)

- La routine, aussi appelée *permanently stored module* (PSM), est un objet stocké dans la base de données, comme les domaines, les types, les tables et les vues.
- Le corps (**body**) d'une routine est la description textuelle du traitement dans un langage (**lang_name**) spécifié.
- Plusieurs dénnotations sont disponibles pour la description textuelle.
 - Voir au TABLEAU
- Les langages spécifiés
 - varient d'une dialecte à un autre ;
 - comprennent minimalement SQL ;
 - comprennent souvent Pascal, C, C++, Java et Python ;
 - comprennent souvent un « PSM language ».
- Chaque dialecte SQL semble avoir défini son propre PSM language plutôt que d'adhérer à celui prescrit par la norme ISO :
 - **plpgsql** est celui de PostgreSQL ;
 - **PL/SQL** est celui d'Oracle.

PLPGSQL

- Le langage plpgsql est décrit dans le manuel de PostgreSQL
 - <https://www.postgresql.org/docs/current/plpgsql.html>

APPEL DES ROUTINES SYNTAXE (POSTGRESQL)

○ Fonction

- comme un terme au sein d'une expression
- exemples
 - 5.00 + Rabais('p001', '2020-10-31')
 - SELECT Rabais('p008', '2020-10-31')
 - SELECT idProduit, Rabais(idProduit, '2019-10-31') AS rab FROM Produit

○ Procédure

- grâce à l'instruction CALL
- exemple
 - CALL Entretien(1.10, '2020-10-24')

EXEMPLE DE DONNÉES

Étudiant

<u>matricule</u>	<u>nom</u>	<u>adresse</u>
15113150	Paul	>Δ ^ϕ σ ^ϕ b
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Activité

<u>sigle</u>	<u>titre</u>
IFT159	Analyse et programmation
IFT187	Éléments de bases de données
IMN117	Acquisition des médias numériques
IGE401	Gestion de projets
GMQ103	Géopositionnement

TypeÉvaluation

<u>code</u>	<u>description</u>
IN	Examen intra
FI	Examen final
TP	Travail pratique
PR	Projet

Résultat

<u>matricule</u>	<u>TE</u>	<u>activité</u>	<u>trimestre</u>	<u>note</u>
15113150	TP	IFT187	20133	80
15112354	FI	IFT187	20123	78
15113150	TP	IFT159	20133	75
15112354	FI	GMQ103	20123	85
15110132	IN	IMN117	20123	90
15110132	IN	IFT187	20133	45
15112354	FI	IFT159	20123	52

EXEMPLES - FONCTION

Créer une fonction pour identifier les activités informatique.

Définition

```
CREATE DOMAIN SigleCours
  VARCHAR(6)
  CHECK
    (VALUE SIMILAR TO '[A-Z]{3}[0-9]{3}');

CREATE OR REPLACE FUNCTION
  estActiviteInfo(sigle SigleCours)
  RETURNS BOOLEAN
  LANGUAGE SQL AS
  $$
  SELECT SUBSTR(sigle,1,3) IN ('IFT', 'IGE', 'IGL', 'IMN')
  $$;
```

Utilisation

```
SELECT sigle, estActiviteInfo(sigle)
FROM activite
```

EXEMPLES – FONCTION (v1)

Créer calculer la cote pour une note.

Clarifications :

Définition

```
CREATE OR REPLACE FUNCTION
  calculerCote(_note NUMERIC(3))
  RETURNS CHAR(1)
  LANGUAGE SQL AS
  $$
  SELECT
    CASE WHEN _note >= 90 THEN 'A'
         WHEN _note BETWEEN 80 and 89 THEN 'B'
         WHEN _note BETWEEN 60 and 79 THEN 'C'
         WHEN _note BETWEEN 40 and 59 THEN 'D'
         ELSE 'E'
    END
  $$;
```

Utilisation

```
SELECT matricule,
       activite,
       calculerCote(note) as Cote
FROM Resultat
WHERE trimestre = '20133'
GROUP BY matricule, activite
```


EXEMPLES – FONCTION (v1)

Créer calculer la cote pour une note.

Clarifications :

Définition

```
CREATE OR REPLACE FUNCTION
  calculerCote(_note NUMERIC(3))
  RETURNS CHAR(1)
  LANGUAGE plpgsql AS
$$
DECLARE
  cote CHAR(1)
BEGIN
  IF _note >= 90 THEN cote := 'A'
  ELSIF _note BETWEEN 80 and 89 THEN cote := 'B'
  ELSIF _note BETWEEN 60 and 79 THEN cote := 'C'
  ELSIF _note BETWEEN 40 and 59 THEN cote := 'D'
  ELSE cote := 'E'
  END IF;
  RETURN cote;
END;
$$;
```

Utilisation

```
SELECT matricule,
       activite,
       calculerCote(note) as Cote
FROM Resultat
WHERE trimestre = '20133'
GROUP BY matricule, activite
```

EXEMPLES – PROCEDURE (v1)

Inscrire un nouvel étudiant en informatique.

Clarifications :

- Créer l'étudiant.
- l'inscrire à tous les cours informatique.

Supposons que nous avons une table inscription(matricule, sigle, trimestre).;

Définition

```
CREATE OR REPLACE PROCEDURE
    IncrireInfo(_matricule Matricule,
               _nom VARCHAR(30),
               _adresse VARCHAR(30),
               _trimestre Trimestre)
LANGUAGE SQL AS
$$
INSERT INTO
    Etudiant(matricule, nom, adresse)
VALUES(_matricule, _nom, _adresse);

INSERT INTO
    Inscription(matricule, sigle, trimestre)
SELECT _matricule, sigle, _trimestre
FROM Activite
WHERE estActiviteInfo(sigle);
$$;
```

Utilisation

```
CALL
    IncrireInfo('20122334', 'Jeanne', '20201');
```

EXEMPLES – PROCEDURE (v2)

Inscrire un nouvel étudiant en informatique.

- Créer l'étudiant.
- L'inscrire à tous les cours informatique.

Supposons que nous avons une table inscription(matricule, sigle, trimestre);

Vérifier si l'étudiant est déjà défini puis insérer les cours.

Exercices :

Inscrire l'étudiant au seulement les cours manquant.

Définition

```
CREATE OR REPLACE PROCEDURE
  IncrireInfo(_matricule Matricule,
             _nom VARCHAR(30),
             _adresse VARCHAR(30))
  LANGUAGE plpgsql AS
$$
BEGIN
  SELECT matricule FROM etudiant
  IF NOT FOUND THEN
    INSERT INTO
      Etudiant(matricule, nom, adresse)
      VALUES(_matricule, _nom, _adresse);
  END IF;

  INSERT INTO
    Inscription(matricule, sigle, trimestre)
    SELECT _matricule, sigle, _trimestre
    FROM Activite
    WHERE estActiviteInfo(sigle);
END;
$$;
```

Utilisation

```
CALL
  IncrireInfo('20122334', 'Jeanne', '20201');
```

RÉTRO-COMPATIBILITÉ

PROCÉDURE SQL – « THE OLDE STYLE » #1

La procédure d'entretien mensuel consiste à

- appliquer une variation uniforme du prix des produits.
- supprimer les produits périmés à partir de la date spécifique.

```
CREATE OR REPLACE FUNCTION Entretienir
    (taux NUMERIC(6,4), d DATE)
    RETURNS VOID
    LANGUAGE SQL AS
    $$
        UPDATE Produit
            SET prix = prix * taux ;
        DELETE FROM Produit
            WHERE peremption < d ;
    $$

-- appel
SELECT Entretienir (1.10, '2020-10-24')
```

RÉTRO-COMPATIBILITÉ

PROCÉDURE SQL – « THE OLDE STYLE » #2

La procédure d'entretien mensuel consiste à

- appliquer une variation uniforme du prix des produits.
- supprimer les produits périmés à partir de la date spécifique.

```
CREATE OR REPLACE FUNCTION Entretienir
(taux NUMERIC(6,4), d DATE)
RETURNS BOOLEAN
LANGUAGE plpgsql AS
$$
BEGIN
UPDATE Produit
    SET prix = prix * taux ;
DELETE FROM Produit
    WHERE peremption < d ;
RETURN TRUE ;
END
$$
```

RÉFÉRENCES

- [Loney2008]
Loney, Kevin ;
Oracle Database 11g: The Complete Reference.
Oracle Press/McGraw-Hill/Osborne, 2008.
ISBN 978-0071598750.
- [Date2012]
Date, Chris J. ;
SQL and Relational Theory: How to Write Accurate SQL Code.
2nd edition, O'Reilly, 2012.
ISBN 978-1-449-31640-2.
- Le site d'Oracle (en anglais)
 - http://docs.oracle.com/cd/E11882_01/index.htm
- Le site de PostgreSQL (en français)
 - <http://docs.postgresqlfr.org>

