Nom & Prénom :

Année académique 2012/2013

TP 4 - « Les procédures stockées »

1. Initialisation :

A faire : Lancer pgAdmin III

2. Concepts de procédures stockées :

Définition : « Une procédure stockée (*Stored Processus*) est une fonction, écrite dans un langage de programmation ou langages de procédures et stockée dans la base de données. Elle est exécutée sur demande par le serveur de bases de données.»

PostgreSQL ne gère pas lui-même les langages de procédures mais délègue cette tâche à un gestionnaire dédié (sauf dans le cas de fonctions SQL).
PostgreSQL peut supporter de nombreux langages de programmation pour écrire des procédures stockées. Cependant, seul un interpréteur de langage PL/pgSQL est livré avec PostgreSQL, pour d'autres langages (PL/Perl, PL/Java, PL/PHP…) il faut installer des interpréteurs externes.

Intérêt : Les requêtes SQL envoyées à l'exécuteur de requête font l'objet d'une analyse syntaxique puis d'une interprétation avant d'être exécutées.

Dans le cas des procédures stockées une requête n'est envoyée à l'exécuteur qu'une seule fois pour être analysée, interprétée et stockée sur le serveur sous forme exécutable (précompilée). Pour qu'elle soit exécutée, le client n'a qu'à envoyer au serveur une requête comportant le nom de la procédure stockée avec ses paramètres.

3. Fonctions SQL :

Il est possible d'écrire une fonction avec le langage SQL.

Syntaxe : (Se référer à l'aide de PostgreSQL pour les options)

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = }
default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
  { LANGUAGE lang_name
    | AS 'definition'
...  }
```

Nom & Prénom :

Exemple :

```
CREATE OR REPLACE FUNCTION GetSalleCapaciteSuperieurA(integer)
RETURNS SETOF Salle
AS '
SELECT * FROM Salle WHERE Capacite > $1;
'
LANGUAGE SQL;
```

A faire : Que fait la fonction `GetSalleCapaciteSuperieurA` ? Exécuter la requête précédente. Que se passe-t-il ?

Appel de la fonction : L'appel de la fonction se fait par son nom et en donnant ses arguments de la manière suivante :

```
SELECT * FROM GetSalleCapaciteSuperieurA(50);
```

A faire : Analyser la requête précédente.

Utiliser une fonction dans une requête : Une fonction peut être utilisée dans une requête comme suit :

```
SELECT DISTINCT batiment
FROM salle
WHERE batiment IN (SELECT batiment FROM
GetSalleCapaciteSuperieurA(50));
```

Supprimer une fonction : Pour supprimer une fonction utiliser la commande
`DROP FUNCTION NomFontion (Arguments)`

A faire : Supprimer la fonction `GetSalleCapaciteSuperieurA`

Fonction ne retournant pas d'arguments : Si une fonction ne retourne pas d'arguments écrire RETURNS VOID

Modifier une fonction : La commande `ALTER FUNCTION NomFonction {Options}` permet de changer la définition d'une fonction.

Fonction avec plusieurs arguments :

```
CREATE OR REPLACE FUNCTION Puissance(numeric,numeric)
RETURNS numeric
AS '
SELECT power($1,$2);
'
LANGUAGE SQL;
```

Exemples de fonctions mathématiques : abs(x) : Valeur absolue / mod(y,x) : Reste de la division des arguments / power (x,y) : x à la puissance y / sqrt (x) : Racine carrée de x

Nom & Prénom :

A faire : Créer la fonction Puissance et tester la.

A faire : Ecrire une fonction qui prend en argument le nom d'un département et retourne son identifiant. Utiliser cette fonction dans une requête permettant de retrouver, à partir de la table ENSEIGNANT, le nom et le prénom des enseignants du département MIG.

`table_name.column_name%TYPE`

Récupérer le type d'une colonne (TYPE) :

```
CREATE OR REPLACE FUNCTION BatimentAReserver(integer)
RETURNS SETOF character varying(1)
AS '
SELECT batiment FROM Salle WHERE Capacite > $1;
'
LANGUAGE SQL;
```

On peut remplacer le type de la colonne BATIMENT character varying(25) par la syntaxe suivante : NomTable.NomColonne%TYPE

A faire : Tester la méthode précédente.

Fonction retournant une colonne :

```
CREATE OR REPLACE FUNCTION BatimentAReserverBis(integer)
RETURNS TABLE (ID_Batiment character varying(1))
AS '
SELECT batiment FROM Salle WHERE Capacite > $1;
'
LANGUAGE SQL;
```

A faire : Expliquer la syntaxe de la fonction `BatimentAReserverBis.`

A faire : En se basant sur la fonction `BatimentAReserverBis`, écrire une nouvelle fonction `BatimentSalleAReserver` retournant deux colonnes IDBat et IDSalle représentant l'identifiant du bâtiment et de la salle ayant une capacité supérieure à une valeur.

Exercice 1 :

Ecrire une fonction qui vérifie que le créneau horaire choisi pour une réservation n'est pas contenu dans le(s) créneau(x) horaire(s) de réservations existantes ou ne chevauche pas le(s) créneau(x) horaire(s) de réservations existantes.
Cette fonction prendra en paramètres un numéro de bâtiment et un numéro de salle (sous forme de chaînes de caractères), une date de réservation et une heure de début et de fin de réservation.
La fonction retournera les identificateurs des réservations qui rendent la réservation demandée impossible (ou ne retourne rien sinon).