SGBD : PROGRAMMATION ET ADMINISTRATION DES BASES DE DONNÉES [M2106]

TD(TP) $N^0 7$ - DROITS

Objectifs

- Organisation logique d'une base de données
- Gestion des droits d'accès

Enoncés

Exercice I : Droits , API

On considère deux tables `m_envoyes` et `m_recus` identiques, sauf que `m_envoyes` stockent les messages envoyés par chaque utilisateur et `m_recus` stockent les messages recus par chaque utilisateur (chacune est une copie de l'autre).

```
1 create table m_envoyes(
2   numero serial,
3   message text,
4    expediteur name,
5    destinataire   name,
6   date_envoie timestamp
7     DEFAULT
8     current_timestamp
9 );
```

```
1 create table m_recus(
2   numero serial,
3   message text,
4    expediteur name,
5    destinataire   name,
6   date_envoie timestamp
7     DEFAULT
8     current_timestamp
9 )
```

Question 1.1. *Décrire (en complétant le code ci-dessous) une fonction* `post` *qui permet à un utlisateur d'envoyer un message à un autre utilisteur. Cette fonction retourne vrai si le message est bien posté (faux si le destinataire n'est pas un utilisateur connu). Chaque message est inséré à la fois dans* `m_envoyes` *et* `m_recus`*. Cette redondance permet à chaque utilisateur de gérer ses propres messages.*

Date: 27 février 2014.

Hocine ABIR - IUT Villetaneuse .

```
create function post(
        destinataire  varchar,
       message text)
returns  boolean as
$$

   -- corps de fonction

$$ language  plpgsql
   SECURITY  DEFINER;
```

Question 1.2. *Décrire (en complétant le code ci-dessous) une fonction* `get` *qui permet à un utlisateur de consulter l'ensemble de son courrier (dans l'ordre des envois)*

```
create  function  get(
   out num int ,
   out expe name,
   out quand timestamp,
   out mess  text
 )
returns  setof  record  as
$$

   -- corps de fonction

$$ language  plpgsql
   SECURITY  DEFINER;
```

Comme dans l'exemple suivant d'echanges entre paul et jean :

```
jean=>
 num | expe |           quand            |                    mess
-----+------+----------------------------+---------------------------------
  10 | paul | 2014-02-27 10:15:02.633956 | ->Bonjour paul
  11 | paul | 2014-02-27 10:15:23.306027 | ->Tu me reponds
  11 | paul | 2014-02-27 10:15:54.44197  | <-qu'est ce que tu veux !!
  13 | paul | 2014-02-27 10:18:33.290095 | ->tu as compris l'histoire de  +
     |      |                            | SECURIY DEFINER
  13 | paul | 2014-02-27 10:20:35.05014  | <-Oueh... Utilise SESSION_USER +
     |      |                            |  pour determiner l'utilisateur +
     |      |                            |  qui opère
  15 | paul | 2014-02-27 10:20:53.178059 | ->Merci
  15 | paul | 2014-02-27 10:21:08.826106 | <-Envoir
```

```
(7 rows)

paul=>
 num | expe |           quand            |                    mess
-----+------+----------------------------+--------------------------------
   9 | jean | 2014-02-27 10:15:02.633956 | <-Bonjour paul
  10 | jean | 2014-02-27 10:15:23.306027 | <-Tu me reponds
  12 | jean | 2014-02-27 10:15:54.44197  | ->qu'est ce que tu veux !!
  12 | jean | 2014-02-27 10:18:33.290095 | <-tu as compris l'histoire de  +
     |      |                            | SECURIY DEFINER
  14 | jean | 2014-02-27 10:20:35.05014  | ->Oueh... Utilise SESSION_USER +
     |      |                            |  pour determiner l'utilisateur +
     |      |                            |  qui opère
  14 | jean | 2014-02-27 10:20:53.178059 | <-Merci
  16 | jean | 2014-02-27 10:21:08.826106 | ->Envoir
(7 rows)
```

Exercice II : table systèmes

La table système `pg_database` contiennent des informations sur les bases de données existantes, en particulier les colonnnes suivantes donnent :

Table **pg_table**

Colonne	Type	Description
datname	name	le nom de la base de données
datdba	oid	le propriétaire de la base de données pg_catalog.pg_get_userbyid(datdba) retourne le nom
datacl	aclitem[]	privilèges d'accès

Question 2.1. *Décrire la fonction* `l` *suivante :*

```
1  CREATE or replace FUNCTION l
2    ( out "Base" varchar ,
3      out "Owner" varchar,
4      out "Privileges" text
5    )
6    RETURNS setof record as
7  $$
8     -- corps fonction
9
10 $$ LANGUAGE SQL;
```

de sorte que son résultat soit de la forme :

```
=> select * from l()
    where "Base"~'^template.';
   Base    |  Owner   |      Privileges
-----------+----------+----------------------
 template1 | postgres | =c/postgres          +
           |          | postgres=CTc/postgres
 template0 | postgres | =c/postgres          +
           |          | postgres=CTc/postgres
(2 rows)
```

La table système `pg_namespace` contiennent des informations sur les schémas d'une base de données existante avec les colonnnes suivantes :

Table **pg_namespace**

Colonne	Type	Description
nspname	name	le nom du schéma
nspowner	oid	le propriétaire du schéma pg_catalog.pg_get_userbyid(spowner) retourne le nom
nspacl	aclitem[]	privilèges d'accès

Question 2.2. *Décrire la fonction* `ls` *suivante :*

```
CREATE or replace FUNCTION ls
  ( out "Schema" varchar ,
    out "Owner" varchar,
    out "Privileges" text
  )
  RETURNS setof record as
$$
   -- corps fonction

$$ LANGUAGE SQL;
```

de sorte que son résultat soit de la forme :

```
=> select * from ls()
    where "Schema"~'^publi*';
  Schema  |  Owner   |      Privileges
----------+----------+----------------------
 public   | postgres | postgres=UC/postgres+
          |          | =U/postgres         +
          |          | abir=UC/postgres
 publique | abir     |
```

```
(2 rows)
```

La table système `pg_class` contiennent des informations sur différents objets d'une base de données existante , en particulier les colonnnes suivantes donnent :

Table **pg_namespace**

Colonne	Type	Description
relname	name	le nom de l'objet
relnamespace	oid	identifiant du schéma contenant l'objet
relowner	oid	le propriétaire de l'objet pg_catalog.pg_get_userbyid(relowner) retourne le nom
relkind	"char"	Type de l'objet (r=table, S=séquence, v=vue)
relacl	aclitem[]	privilèges d'accès

Question 2.3. *Décrire la fonction* `z` *suivante :*

```
1  CREATE or replace FUNCTION z
2    ( out "Schema" varchar ,
3      out "Nom" varchar,
4      out "Owner" varchar,
5      out "?" "char",
6      out "Privileges" text,
7      out "ColonneS" text
8    )
9    RETURNS setof record as
10 $$
11    -- corps fonction
12
13 $$ LANGUAGE SQL;
```

de sorte que son résultat soit de la forme :

```
abir=> select * from z() where "Nom"='employe';
 Schema |   Nom    | Owner | ? |    Privileges     | ColonneS
--------+----------+-------+---+-------------------+-----------
 public | employe | abir  | r | abir=arwdDxt/abir | empnom:  +
        |          |       |   |                   |   =r/abir+
        |          |       |   |                   | empgrade:+
        |          |       |   |                   |   =r/abir
(1 row)
```

Corrigés

Exercice I :

Question 1.1. `prop` *est le login de chaque etudiant.*

```
1  create function post( destinataire varchar,message text)
2  returns boolean as
3  $$
4   DECLARE
5    dest name;
6   BEGIN
7     select rolname into dest from pg_roles
8          where rolcanlogin
9            and rolname= destinataire ::name;
10    IF NOT FOUND THEN
11         raise notice 'destinataire "%" introuvable',destinataire;
12         return false;
13    END IF;
14     insert into prop.m_envoyes(message,expediteur,destinataire)
15          values(message,session_user,destinataire::name);
16     insert into prop.m_recus(message,expediteur,destinataire)
17          values(message,session_user,destinataire::name);
18    return true;
19  END;
20 $$ language plpgsql
21  SECURITY DEFINER;
```

Question 1.2. `prop` *est le login de chaque etudiant.*

```
1  create function get(
2     out num int ,
3     out expe name,
4     out quand timestamp,
5     out mess text
6   )
7  returns setof record as
8  $$
9   DECLARE
10     mail CURSOR FOR
11     SELECT numero, expediteur, date_envoie,'<-'||message
12       FROM prop.m_recus
13       WHERE destinataire = SESSION_USER
14     union
15     SELECT numero, destinataire, date_envoie,'->'||message
16       FROM prop.m_envoyes
17       WHERE expediteur = SESSION_USER
```

```
18      order by date_envoie;
19  BEGIN
20    OPEN mail;
21    LOOP
22       FETCH mail into num,expe,quand,mess;
23       EXIT WHEN NOT FOUND;
24       RETURn next;
25    END LOOP;
26     return ;
27  END;
28 $$ language plpgsql
29   SECURITY DEFINER;
```

Exercice II :

Question 2.1.

```
1  CREATE or replace FUNCTION l
2    ( out "Base" varchar ,
3      out "Owner" varchar,
4      out "Privileges" text
5    )
6    RETURNS setof record as
7  $$
8  SELECT d.datname::varchar ,
9         pg_catalog.pg_get_userbyid(d.datdba)::varchar as "Owner",
10        pg_catalog.array_to_string(d.datacl, E'\n') AS "Privileges"
11 FROM pg_catalog.pg_database d
12 $$ language SQL;
```

Question 2.2.

```
1  CREATE or replace FUNCTION ls
2    ( out "Schema" varchar ,
3      out "Owner" varchar,
4      out "Privileges" text
5    )
6    RETURNS setof record as
7  $$
8  select s.nspname::varchar,
9    pg_catalog.pg_get_userbyid(nspowner)::varchar,
10  pg_catalog.array_to_string(s.nspacl,E'\n')
11  from pg_catalog.pg_namespace s ;
12 $$ language sql;
```

Question 2.3.

```
CREATE or replace FUNCTION z
  ( out "Schema" varchar ,
    out "Nom" varchar,
    out "Owner" varchar,
    out "?" "char",
    out "Privileges" text,
    out "ColonneS" text
  )
  RETURNS setof record as
$$
SELECT n.nspname::varchar ,
      c.relname::varchar  ,
      pg_catalog.pg_get_userbyid(c.relowner)::varchar ,
      c. relkind  ,
      pg_catalog.array_to_string(c. relacl , E'\n') ,
      pg_catalog.array_to_string(ARRAY(
       SELECT attname || E':\n ' ||
            pg_catalog.array_to_string( attacl , E'\n ')
          FROM pg_catalog.pg_attribute a
          WHERE attrelid = c.oid
          AND NOT attisdropped
          AND attacl IS NOT NULL
     ), E'\n')
 FROM pg_catalog.pg_class c
 LEFT JOIN pg_catalog.pg_namespace n
        ON n.oid = c.relnamespace
  WHERE c.relkind IN ('r', 'v', 'S')
  AND n.nspname !~ '^pg_'
  AND pg_catalog.pg_table_is_visible(c.oid);
$$ language sql;
```