

SGBD : EXPLOITATION D'UNE BASE DE DONNÉES [R206]

TD(TP) N°7 - DROITS ET PERMISSIONS

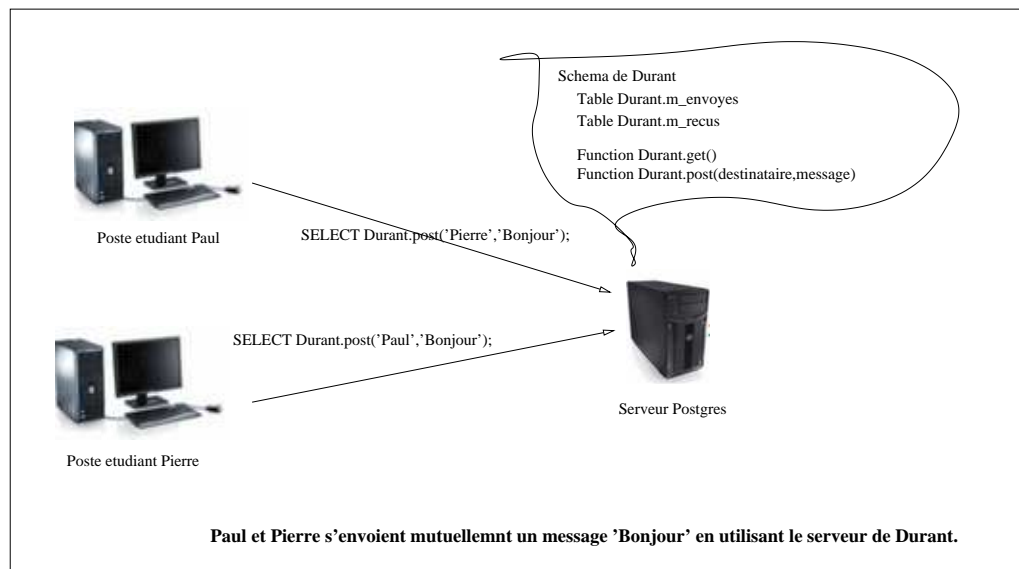
OBJECTIFS

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

ENONCÉS

Exercice I :

On souhaite décrire un schéma de base de données pour un serveur de courrier simplifié comme illustré par le dessin suivant :



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).

Date: March 29, 2022.

Hocine ABIR - IUT Villeteaneuse .

```

1 create table m_envoyes(
2     numero serial,
3     message text,
4     expéditeur 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     expéditeur 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 utilisateur d'envoyer un message à un autre utilisateur. 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.

```

1 create function post(
2     destinataire varchar,
3     message text)
4 returns boolean as
5 $$
6
7     -- corps de fonction
8
9 $$ language plpgsql
10 SECURITY DEFINER;

```

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

```

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
10     -- corps de fonction
11

```

```

12 $$ language plpgsql
13 SECURITY DEFINER;

```

Comme dans l'exemple suivant d'échanges 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	<-Envoyer

(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	->Envoyer

(7 rows)

Exercice II :

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

Table pg_namespace		
Colonne	Type	Description
nspace	name	le nom du schéma
nsowner	oid	le propriétaire du schéma pg_catalog.pg_get_userbyid(spowner) retourne le nom
nsacl	aclitem[]	privilèges d'accès

Question 2.2. Décrire la fonction `ls` suivante :

```

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   $$
8     -- corps fonction
9
10  $$ 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 colonnes suivantes donnent :

Table <code>pg_namespace</code>		
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[]	privileges 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)