Procédures stockés en Python pour PostgreSQL

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 3 avril 2007

https://www.bortzmeyer.org/postgresql-python.html

Une des grandes forces du SGBD PostgreSQL est la possibilité d'écrire des **procédures stockées**, des bouts de code exécutés par le serveur via une requête SQL. Ces procédures stockées sont typiquement écrites en PL/pgSQL mais on peut aussi les écrire en Python.

PL/pgSQL est un langage très pratique, puisque assez simple et présent sur toutes les installations de PostgreSQL. Mais il est limité et les calculs complexes sont mieux sous-traités à un langage plus général comme Perl ou Python. Supposons qu'on veuille stocker dans la base de données des noms de domaine IDN. La conversion de ces noms Unicode en un encodage compatible avec ASCII (ACE pour "*ASCII Compatible Encoding*") se fait selon un algorithme décrit dans le RFC 3492 [1]. Cet algorithme, qui met en jeu de grosses tables Unicode, est bien trop complexe pour PL/pgSQL mais est trivial en Python qui dispose d'une bibliothèque pour cela <http://docs.python.org/lib/module-encodings.idna.html>.

Voici un exemple d'un programme Python qui convertit un IDN en ACE :

```
# Converts the Unicode *labels* (*not* domain names) given on the
# command line to ACE (xn--something).

import encodings.idna
import sys

# Change it to your locale. TODO: deduce it from the environment
locale = "Latin-1"

for name in sys.argv[1:]:
    # TODO: check with UseSTD3ASCIIRules true (we should not accept
    # names with ; or &)
    print "%s -> %s" % (name,
                        encodings.idna.ToASCII(unicode(name, locale)))
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc3492.txt

Maintenant, si on veut le faire depuis PostgreSQL, par exemple pour remplir automatiquement le champ `label_ascii` dans cette table ?

```
CREATE TABLE Domains (
 id SERIAL UNIQUE,
 label TEXT NOT NULL, -- label is the left most label.  It is in Unicode.
 label_ascii TEXT NOT NULL, -- If the domain
    -- is an IDN, it is the ACE. Otherwise, same as label.
 parent INTEGER REFERENCES Domains (id));
```

PostgreSQL permet d'écrire un "*trigger*" qui va être déclenché à chaque écriture dans la table. D'abord, on installe le langage PL/pgSQL, qui sert à une fonction intermédiaire (on peut aussi le faire en SQL avec `CREATE LANGUAGE`) :

```
% sudo -u postgres  createlang plpgsql registry
```

Ensuite, on crée le "*trigger*" :

```
CREATE OR REPLACE FUNCTION add_ace() RETURNS TRIGGER
   AS 'BEGIN
          NEW.label_ascii = to_ace(NEW.label);
          RETURN NEW;
       END;'
     LANGUAGE PLPGSQL;

CREATE TRIGGER add_ace
   BEFORE INSERT ON Domains
   FOR EACH ROW
   EXECUTE PROCEDURE add_ace();
```

Et la fonction `to_ace` va être en Python. Mais, d'abord, installons PostgreSQL avec le support Python. Sur une Debian, il suffit d'installer le paquetage `postgresql-plpython`.

Sur Gentoo, il faut compiler PostgreSQL avec le support Python. Le fichier `ebuild` de PostgreSQL contient :

```
IUSE="doc kerberos nls pam perl pg-intdatetime python readline selinux ssl tcl test xml zlib"
```

Il faut donc s'assurer que l'option `python` est dans la variable USE. Par exemple, dans `/etc/make.conf` :

```
USE="... python ..."
```

On peut aussi la mettre dans `/etc/portage/package.use` si on veut ne l'appliquer qu'à PostgreSQL :

```
dev-db/postgresql python
```

Une fois cette option présente (on peut vérifier avec `emerge --pretend postgresql`), on compile PostgreSQL :

```
% sudo emerge postgresql
```

Sur une machine NetBSD, utilisant le système pkgsrc, on doit installer le paquetage plpython, par exemple :

```
% cd /usr/pkgsrc/databases/postgresql82-plpython
% make install
```

Une fois PostgreSQL installé avec le support de Python, il faut installer le langage :

```
% sudo -u postgres  createlang plpythonu registry
```

On peut enfin ajouter la fonction en Python :

```
CREATE OR REPLACE FUNCTION to_ace(TEXT) RETURNS TEXT
    AS '
        import encodings.idna
        return encodings.idna.ToASCII(unicode(args[0], "UTF-8"))
    '
    LANGUAGE 'plpythonu';
```

Python étant un langage "*untrusted*" (le 'u' final) dans PostgreSQL, cette opération doit être effectuée par un super-utilisateur PostgreSQL.

Il ne nous reste plus qu'à tester :

```
registry=> INSERT INTO Domains (label) VALUES ('foobar');
INSERT 0 1

registry=> SELECT * from Domains;
 id | label  | label_ascii | parent
----+--------+-------------+--------
  1 | foobar | foobar      |
(1 row)

registry=> INSERT INTO Domains (label) VALUES ('café');
INSERT 0 1

registry=> SELECT * FROM Domains;
 id | label  | label_ascii | parent
----+--------+-------------+--------
  1 | foobar | foobar      |
  3 | café   | xn--caf-dma |
(2 rows)
```

Et voilà, les labels en Unicode ont bien été convertis, grâce au programme Python.