

Python dan PostgreSQL

PostgreSQL adalah database *server* yang gratis, andal dan kaya fitur. Untuk aplikasi bisnis, umumnya PostgreSQL dapat diandalkan. Koneksi dari Python pun dapat dilakukan dengan mudah.



Aplikasi bisnis umumnya berhubungan database. Kita dapat memilih salah satu dari sekian sistem database *open source* yang tersebar di belantara *free software*. Salah satu yang sangat populer adalah MySQL. MySQL adalah database server yang cukup mudah digunakan. Sayangnya, MySQL bukanlah sistem database yang lengkap. Apabila Anda butuh sistem yang lebih lengkap dan kaya fitur, namun tetap gratis, gunakanlah PostgreSQL. Yang satu ini memang lebih susah untuk digunakan. Instalasinya juga cukup rumit. Namun, yang jelas, PostgreSQL adalah sistem database *object-relational* yang luar biasa. Untuk kebutuhan bisnis serius, gunakan PostgreSQL. Tidak perlu mengeluarkan banyak uang untuk membeli sistem database saja. PostgreSQL umumnya sudah lebih dari cukup.

Sama dengan pilihan sistem database, pilihan bahasa pemrograman juga cukup banyak. Silakan saja pilih mulai dari yang *low level* seperti C/C++ sampai yang *high level* seperti PHP, Perl, Java, atau Python. Semua memiliki kekurangan dan kelebihan sendiri. Kali ini, kita akan menggunakan Python. Python adalah bahasa pemrograman yang cukup lengkap dan mudah digunakan. Pemula sekalipun cenderung mudah untuk mempelajari bahasa yang satu ini.

Dengan Python, kita dapat dengan mudah terhubung ke PostgreSQL. Pilihan modulnya pun cukup banyak. Salah satu yang akan kita gunakan adalah PyGreSQL atau python-pygresql. Setelah melakukan analisis kebutuhan dengan benar dan membuat desain yang benar pula, kita dapat menerapkan aplikasi bisnis kita dengan Python dan menggunakan postgresql untuk mengatur penyimpanan data.

Persiapan PostgreSQL

Pertama-tama, Anda dapat menggunakan paket-paket di distro Anda untuk menginstal PostgreSQL. Jarang sekali ada distro umum yang tidak menyertakan postgresql di dalamnya. Kalaupun tidak, *download*-lah paket untuk distro Anda yang dibuat oleh pihak ketiga. Bagi yang menggunakan Debian, berikan perintah berikut ini untuk instalasi:

```
apt-get install postgresql postgresql-client
postgresql-doc
```

Bagi Anda yang lebih menyukai kompilasi sendiri, *download*-lah *source code* postgresql dari <http://www.postgresql.org>, kemudian ekstrak dan masuklah ke direktori hasil ekstrak tersebut. Setelah itu, lakukanlah langkah-langkah berikut ini.

```
./configure
make
su
make install
adduser postgres
mkdir /usr/local/pgsql/data
chown postgres /usr/local/pgsql/data
su - postgres
/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
/usr/local/pgsql/bin/postmaster -D /usr/local/pgsql/data >logfile 2>&1 &
/usr/local/pgsql/bin/createdb test
/usr/local/pgsql/bin/psql test
```

Instalasi PostgreSQL lebih neko-neko dan kompleks apabila dibandingkan dengan instalasi MySQL.

Salah satu hal yang sedikit berbeda dengan MySQL adalah sistem otentikasi *client*-nya. PostgreSQL memiliki berbagai cara dalam melakukan otentikasi. Cara yang termudah adalah membuat *user* di PostgreSQL yang memiliki nama sama

dengan user di sistem.

Jalankan database server (postmaster) Anda dengan user postgres. Saat ini, kita akan membuat user dengan nama test. Harap diperhatikan, untuk lebih mudahnya, Anda telah membuat dan login dengan user test di Linux Anda.

Jadilah root, dan kemudian jadilah user postgres. Kita akan membuat user dengan nama test, yang dapat membuat database dan tidak bisa membuat user baru lagi.

```
$ su
Password:
# su postgres
$ createuser test
Shall the new user be allowed to create
databases? (y/n) y
Shall the new user be allowed to create
more new users? (y/n) n
CREATE USER
```

Apabila Anda melihat pesan CREATE USER tersebut, maka pembuatan user test telah berhasil. Untuk menghapus seorang user, gunakan program dropuser.

Aktiflah kembali sebagai user test. Kita akan melanjutkan ke persiapan Python untuk melakukan koneksi ke PostgreSQL.

Persiapan Python

Terdapat beberapa modul untuk melakukan koneksi ke PostgreSQL dari Python. Salah satu yang populer adalah PyGreSQL atau umumnya terdapat di distro dengan nama python-pygresql. Anda bisa *download* modul tersebut di <http://www.druid.net/pygresql/>.

Bagi Anda yang menggunakan distro dengan manajemen paket RPM, *download*-lah paketnya di <ftp://ftp.druid.net/pub/distrib/pygresql.i386.rpm>. Bagi Anda yang menggunakan Debian atau turunannya, berikan perintah berikut ini untuk melakukan instalasi:

```
apt-get install python-pygresql
```

Setelah instalasi sukses, kita akan mengujinya dengan menjalankan modul pg di interpreter Python. Harap diingat, jalankanlah Python sebagai *user test*.

```
$ python2.2
Python 2.2.3+ (#1, Aug 10 2003,
10:11:23)
[GCC 3.3.1 (Debian)] on linux2
```

Type "help", "copyright", "credits" or "license" for more information.

```
>>> import pg
>>> from pg import DB
>>> dir(pg)
['DB', 'INV_READ', 'INV_WRITE',
 'RESULT_DDL', 'RESULT_DML',
 'RESULT_DQL', 'RESULT_EMPTY',
 'SEEK_CUR', 'SEEK_END', 'SEEK_SET',
 '__builtins__', '__doc__', '__file__',
 '__name__', '__quote__', 'connect', 'error',
 'get_defbase', 'get_defhost', 'get_defopt',
 'get_defport', 'get_deftty', 'get_defuser',
 're', 'set_defbase', 'set_defhost',
 'set_defopt', 'set_defpasswd',
 'set_defport', 'set_deftty', 'set_defuser',
 'string', 'sys', 'version']
>>> dir(DB)
['__doc__', '__init__', '__module__',
 'clear', 'delete', 'get', 'get_attnames',
 'get_databases', 'get_tables', 'insert',
 'pkey', 'query', 'update']
>>>
```

Apabila Anda mendapatkan tampilan serupa, maka instalasi PyGreSQL telah berhasil. Namun, apabila tampilan yang Anda dapat adalah seperti berikut ini, maka instalasi belum berhasil atau sistem Anda belum memiliki PyGreSQL.

```
>>> import pg
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
ImportError: No module named pg
>>>
```

Koneksi Python dan PostgreSQL

Pertama-tama, sekali lagi, aktiflah sebagai user test di sistem anda. Kita akan membuat sebuah database dengan nama test di postgresQL. Berikanlah perintah createdb sebagai berikut:

```
$ createdb test
CREATE DATABASE
```

Apabila Anda mendapati tampilan serupa, maka pembuatan database test telah berhasil. Kita akan membuat sebuah tabel dengan nama tbl_test dengan dua field: f1 dan f2, masing-masing bertipe integer.

Gunakan client psql untuk masuk ke dalam database server. Berikanlah perintah berikut ini:

```
$ psql test
```

```
File Edit View Terminal Go Help
nopl:~$ psql test
Welcome to psql 7.3.3, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help on internal slash commands
      \g or terminate with semicolon to execute query
      \q to quit

test=> \dt
          List of relations
Schema | Name      | Type  | Owner
-----+-----+-----+-----
public | tbl_test  | table | test
(1 row)

test=> \d tbl_test
          Table "public.tbl_test"
Column | Type      | Modifiers
-----+-----+-----
f1      | integer   |
f2      | integer   |

test=> select * from tbl_test where f1<5;
 f1 | f2
----+---
 1 | 1
 0 | 0
 1 | 1
 2 | 2
 3 | 3
 4 | 4
(6 rows)

test=> select MAX(f1) from tbl_test;
 max
-----
 999
(1 row)

test=>
```

▲ Psql, client untuk PostgreSQL

Welcome to psql 7.3.3, the PostgreSQL interactive terminal.

```
Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help on internal slash
      commands
      \g or terminate with semicolon to
      execute query
      \q to quit
test=>
```

Selanjutnya, berikan perintah berikut ini untuk membuat tabel tbl_test tersebut.

```
test=> CREATE TABLE tbl_test (f1
integer, f2 integer);
CREATE TABLE
test=>
```

Apabila Anda ingin melihat daftar table, ketikkanlah perintah berikut ini:

```
test=> \dt
          List of relations
Schema | Name      | Type  | Owner
-----+-----+-----+-----
public | tbl_test  | table | test
(1 row)

test=>
```

Untuk melihat deskripsi tabel test, berikanlah perintah berikut ini:

```
test=> \d tbl_test
          Table "public.tbl_test"
Column | Type      | Modifiers
-----+-----+-----
f1      | integer   |
```

```
File Edit View Terminal Go Help
nopl:~$ python2.2 psq12.py
usage: psq12.py <DBNAME>
nopl:~$ python2.2 psq12.py test
[test]> select MAX(f1) from tbl_test;
 max
-----
 999
(1 row)

[test]> select * from tbl_test where f1<10;
 f1 | f2
----+---
 1 | 1
 0 | 0
 1 | 1
 2 | 2
 3 | 3
 4 | 4
 5 | 5
 6 | 6
 7 | 7
 8 | 8
 9 | 9
(11 rows)

[test]>
[test]>
[test]>
[test]>
[test]>
[test]>
[test]>
```

▲ Psq12, client PostgreSQL ala Python

```
f2      | integer   |
test=>
```

Kini, jalanlankah interpreter Python Anda. Mulai saat ini sampai seterusnya, kita akan menggunakan modul pg untuk bekerja dengan database postgresQL. Kita akan mulai dengan meng-import class DB dari modul pg. Kemudian, kita membuat instance dari pg.DB dengan nama mydb, yang menunjuk ke database test.

```
>>> from pg import DB
>>> mydb = DB('test')
>>>
```

Selanjutnya, kita akan melihat tabel-tabel apa saja yang terdapat di dalam database test tersebut. Berikanlah kode berikut ini:

```
>>> mydb.get_tables()
['tbl_test']
```

Untuk melihat atribut database test, tuliskan kode berikut ini:

```
>>> mydb.get_attnames('test')
{'oid': 'int'}
```

Apabila Anda ingin melihat atribut tabel tbl_test, tuliskan kode berikut ini:

```
>>> mydb.get_attnames('tbl_test')
{'f1': 'int', 'f2': 'int', 'oid': 'int'}
```

Kita akan melanjutkan ke tahap manipulasi data. Anda dapat memilih untuk menggunakan method dari mydb, ataupun

memberikan sintaks SQL langsung. Untuk mudahnya, kita akan memberikan sintaks SQL langsung.

Contoh pertama kita adalah memasukkan record baru ke dalam `tbl_test`:

```
>>> mydb.query('insert into tbl_test
values(1,1)')
17003
>>>
```

Kemudian, saatnya bagi kita untuk melihat semua record di dalam tabel `tbl_test`:

```
>>> r = mydb.query('select * from
tbl_test')
>>> r
f1 | f2
-- + --
1 | 1
(1 row)
>>> dir(r)
['dictresult', 'fieldname', 'fieldnum',
'getresult', 'listfields', 'ntuples']
>>> r.dictresult()
[{'f1': 1, 'f2': 1}]
>>> r.fieldname(1)
'f2'
>>> r.getresult()
[(1, 1)]
>>> r.listfields()
('f1', 'f2')
>>> r.ntuples()
1
```

Kita amati, `r`, yang merupakan nilai kembalian dari `query` tersebut memiliki sejumlah method. Semuanya kembali kepada Anda untuk memilih method mana yang akan digunakan. Dalam bentuk sederhana, dengan mengakses `r` saja, Anda sudah mendapatkan nilai kembalian `query` tersebut. Umumnya, programmer menggunakan method `dictresult()` ataupun `getresult()`.

Kita juga bisa menggunakan variabel dalam pemberian sintaks SQL seperti berikut ini:

```
>>> for i in range(0,1000L):
>>> mydb.query('insert into tbl_test
values(%d,%d)' %(i,i))
```

Untuk operasi lain seperti `update` dan `delete`, Anda bisa memberikan perintah SQL yang bersesuaian. Contoh program: `psql2`.

Berebekal contoh-contoh di atas, kita akan belajar membuat sebuah client `postgreSQL` sederhana, yang mirip dengan tampilan `psql`.


```
Psq2.py:
#!/usr/bin/python2.2

import sys
from pg import DB

def main(db):
    try:
        mydb = DB(db)
    except:
        sys.exit('error')
    while 1:
        buff = raw_input('[%s]> '
        %db)
        if buff == '': continue
        r = mydb.query(buff)
        print r
    if __name__ == '__main__':
        if len(sys.argv) == 2:
            main(sys.argv[1])
        else:
            sys.exit('usage: %s
<DBNAME>' %sys.argv[0])
```

Pertama-tama, kita perlu memeriksa terlebih dahulu jumlah parameter yang dilewatkan ke program ini. Apabila jumlah paramaternya benar, kita akan mengirimkan parameter terakhir ke fungsi `main()`, yang kemudian akan membuat koneksi. Kita memberikan penanganan kesalahan di sini. Setelah koneksi terjadi, kita akan mengulang terus menerus sambil meminta input. Setiap input yang berisi akan dikirimkan sebagai `query` ke `postgreSQL`.

Program yang satu ini jelas sangat sederhana. Penanganan kesalahannya juga masih sangat minim. Sebenarnya, kita pun dapat menambahkan beberapa perintah spesial seperti `\dt` untuk menampilkan daftar `table`, dengan memanggil method `get_tables()` dari `mydb`.

Kombinasi antara `PostgreSQL` dan `Python` akan menghasilkan aplikasi bisnis yang luar biasa. *Resource free software* yang tersedia sebenarnya sangat banyak. Bagaimana kita menyikapinya adalah hal yang paling. Apabila kita mampu, kita bisa membuat TI di negara kita jauh lebih berkembang. Selamat berkarya! 
Noprianto (noprianto@infolinux.co.id)

IKLAN