Python Database

1. Catatan

a) Dianjurkan untuk membaca materi Python Dasar terlebih dahulu. Aturan penulisan kode dapat dibaca pada materi Python Dasar.
b) Fokus materi adalah koneksi dan bekerja dengan relational database. Dianjurkan untuk mengetahui dasar-dasar relational database. Dasar-dasar relational database tidak akan dibahas di materi ini.
c) PostgreSQL yang digunakan adalah versi 8.2.x:
   1. Dianjurkan untuk membaca dokumentasi PostgreSQL 8.2.x. Konfigurasi PostgreSQL tidak akan dibahas di materi ini.
   2. User dan database dibuat dengan perintah (di system prompt Linux):
      a) createuser <user>
      b) createdb -U <user> <user>
   3. User dapat login ke prompt PostgreSQL dengan perintah: psql -U <user> -d <user>
d) MySQL yang digunakan adalah versi 5.0.x (storage engine table: MyISAM):
   1. Dianjurkan untuk membaca dokumentasi MySQL 5.0.x. Konfigurasi MySQL tidak akan dibahas di materi ini.
   2. User dan database dibuat dengan perintah (di dalam MySQL prompt, sebagai MySQL root):
      a) create database <user>;
      b) grant all privileges on <user>.* to <user>@localhost;
      c) flush privileges;
   3. User dapat login ke prompt MySQL dengan perintah: mysql -u <user> <user>
e) Pembahasan database akan mencakup PostgreSQL dan MySQL, dengan modul yang mendukung DB-API 2.0.
   1. Fitur non-standard sebisa mungkin dihindari.
   2. Modul untuk PostgreSQL adalah psycopg2, yang dapat didownload dari http://initd.org/pub/software/psycopg/
   3. Modul untuk MySQL adalah MySQLdb, yang dapat didownload dari http://mysql-python.sourceforge.net/
f) Tabel contoh yang digunakan adalah 'books', dengan field:
   1. id: ID buku (integer, auto increment/serial, primary key)
   2. title: judul (varchar 128)
   3. author: pengarang (varchar 128)
   4. Perintah SQL PostgreSQL: create table books (id serial not null, title varchar(128), author varchar(128), primary key(id));
   5. Perintah SQL MySQL: create table books (id integer auto_increment not null,title varchar(128), author varchar(128),primary key(id));

2. DB-API 2.0

- Berbagai relational database datang dengan fungsi umum, disamping fungsi spesifik database system tersebut, namun bisa dengan API yang berbeda-beda antar database (koneksi, query, transaksi, dan lain-lain). API yang berbeda menyebabkan perpindahan ke database lain menjadi sangat merepotkan, karena hampir semua fungsi yang digunakan harus diganti.
- Python datang dengan DB-API (versi 2.0), yang menyediakan API umum untuk beragam modul Python, untuk berbagai database system. Dengan demikian, perpindahan ke database lain menjadi sangat mudah, karena hampir semua fungsi yang digunakan adalah sama. Hanya modulnya saja yang berbeda. Pada penerapan di dunia nyata, terkadang hanya beberapa baris saja yang perlu diubah.
- Developer sebisa mungkin diharapkan tidak mengakses fitur non-standard database system tertentu.
- DB-API 2.0 dapat dibaca pada PEP 249 (http://www.python.org/dev/peps/pep-0249/)
- Tidak semua modul/adapter database kompatibel dengan DB-API 2.0.

3. Koneksi database

Sebelum bekerja dengan PostgreSQL atau MySQL, koneksi ke database server harus dilakukan terlebih dahulu. Walaupun kedua modul (psycopg2 dan mysql-python) kompatibel dengan DB-API 2.0, ada sedikit perbedaan dalam melakukan koneksi.

PostgreSQL

Untuk melakukan koneksi, method connect() psycopg2 digunakan. Parameter untuk connect() dapat berupa string data source name (dsn) ataupun dengan keyword. Training hanya akan membahas penggunaan keyword argument, yang terdiri dari:

- database: nama database
- host: alamat host database (default: UNIX socket)
- port: port database (default: 5432)
- user: nama user yang melakukan koneksi
- password: password user
- sslmode: mode SSL

Setelah koneksi berhasil dilakukan, sebuah objek connection akan dikembalikan.

Setelah selesai bekerja dengan database, koneksi dapat ditutup dengan method close() objek connection. Penting diingat: PostgreSQL adalah database yang mendukung transaksi, apabila koneksi ditutup tanpa melakukan commit (akan dibahas pada query dan transaksi), maka rollback akan dipanggil secara implisit, sehingga perubahan tidak tersimpan.

Contoh koneksi PostgreSQL dapat pula dilihat pada source pgsql_connect.py.

pgsql_connect.py:

```
#!/usr/bin/env python

import psycopg2 as pgsql

conn = pgsql.connect(user='user', database='testing')
print conn
conn.close()
```

MySQL

Untuk melakukan koneksi, method connect() MySQLdb digunakan. Parameter untuk connect() adalah keyword, yang diantaranya terdiri dari:

- db: nama database
- host: alamat host database
- port: port database
- user: nama user yang melakukan koneksi

- passwd: password user
- dan lain sebagainya, yang dapat dibaca pada file connections.py paket MySQLdb.

Setelah koneksi berhasil dilakukan, sebuah objek connection akan dikembalikan.

Setelah selesai bekerja dengan database, koneksi dapat ditutup dengan method close() objek connection.

Penting diingat:
- Apabila menggunakan engine yang mendukung transaksi, maka apabila koneksi ditutup tanpa melakukan commit (akan dibahas pada query dan transaksi), maka rollback akan dipanggil secara implisit, sehingga perubahan tidak tersimpan.
- Apabila menggunakan engine tanpa transaksi, maka sebaiknya commit tetap dilakukan (walau tanpa efek) setiap ada perubahan pada database. Hal ini akan mempermudah apabila database (atau storage engine) yang digunakan, diganti ke yang lainnya.

Contoh koneksi MySQL dapat pula dilihat pada source mysql_connect.py.

mysql_connect.py:
```
#!/usr/bin/env python

import MySQLdb as mysql

conn = mysql.connect(user='user', db='testing')
print conn
conn.close()
```

4. Query

Untuk melakukan query, kita bekerja dengan cursor, yang akan mengatur eksekusi query dan mendapatkan hasilnya.
- Objek Cursor dibuat dengan memanggil method cursor() objek connection.
- Setelah digunakan, cursor bisa ditutup dengan method close() objek Cursor.
- Apabila database mendukung transaksi, maka transaksi akan dimulai otomatis begitu cursor dibuat.

Tidak semua database system mendukung transaksi. Untuk database yang mendukung transaksi, semua perubahan pada database tidak tersimpan sampai commit dilakukan. Ketika bekerja dengan database yang tidak mendukung transaksi, commit sebaiknya juga tetap dilakukan (walau tanpa efek), sehingga dapat mempermudah apabila database yang digunakan akan diganti ke yang lainnya.

Mengirimkan query

Untuk mengirimkan query, gunakan method execute() cursor. Developer bisa mengirimkan query berupa string (keyword query, atau parameter pertama). Hanya, satu hal yang perlu diperhatikan adalah query string yang dikirimkan, apabila melibatkan input dari user, maka developer haruslah berhati-hati. Tidak semua input dari user bisa dipercaya, karena bisa saja merupakan serangan SQL injection.

Agar lebih aman, gunakan parameter substitution. Setiap input bisa diwakili oleh sebuah parameter dan modul database akan otomatis melakukan pemeriksaan dan tindakan lain yang diperlukan agar query dapat dikirimkan dengan aman.

Format parameter yang digunakan bisa berbeda-beda. Developer bisa mengamati format yang digunakan, dengan melihat pada <module>.paramstyle. Berikut ini adalah format yang mungkin:
- Qmark: tanda tanya, contoh: where name=?
- Numeric: parameter berupa nomor (posisi), contoh: where name=:1
- Named: parameter berupa nama, contoh: where name=:name
- Format: format seperti printf C, contoh: where name=%s
- Pyformat: format extended Python, contoh: where name=%(name)s

Untuk materi training, yang akan digunakan adalah format. Sebuah tuple/list berisikan parameter-parameter diberikan sebagai argumen kedua (keyword vars untuk psycopg2 atau keyword args untuk MySQLdb) method execute() cursor.

Contoh-contoh query PostgreSQL dapat dilihat pada pgsql_createtbl.py dan pgsql_query_insert.py. Contoh-contoh query MySQL dapat dilihat pada mysql_createtbl.py dan mysql_query_insert.py.

pgsql_createtbl.py:
```
#!/usr/bin/env python
```

```
import psycopg2 as pgsql

conn = pgsql.connect(user='user', database='testing')

cur = conn.cursor()
query = '''
create table books (id serial not null, title varchar(128), author
varchar(128),
 primary key(id));
'''
cur.execute(query)
conn.commit()

cur.close()
conn.close()
```

pgsql_query_insert.py:

```
#!/usr/bin/env python

import psycopg2 as pgsql

conn = pgsql.connect(user='user', database='testing')

cur = conn.cursor()
cur.execute('''
insert into books(author, title) values(%s, %s)
''', ('author1', 'title1'))
conn.commit()

cur.close()
conn.close()
```

mysql_createtbl.py:

```
#!/usr/bin/env python

import MySQLdb as mysql

conn = mysql.connect(user='user', db='testing')

cur = conn.cursor()
query = '''
create table books (id integer auto_increment not null,title varchar(128),
autho
r varchar(128),primary key(id));
'''
cur.execute(query)

cur.close()
```

```
conn.close()
```

mysql_query_insert.py:

```
#!/usr/bin/env python

import MySQLdb as mysql

conn = mysql.connect(user='user', db='testing')

cur = conn.cursor()
cur.execute('''
insert into books(author, title) values(%s, %s)
''', ('author1', 'title1'))
conn.commit()

cur.close()
conn.close()
```

Catatan:
Dalam contoh-contoh insert, nilai parameter diberikan oleh developer, yang tidak berisikan nilai yang invalid. Dapat pula diberikan langsung tanpa harus disubstitusi. Contoh telah dibuat lebih sederhana tanpa input user.

Mendapatkan hasil query

Untuk mendapatkan hasil query manipulasi berupa insert/update/delete (affected rows), setelah query dikirimkan, akseslah properti rowcount cursor.

Untuk mendapatkan hasil query select, gunakan salah satu method cursor berikut:

- fetchone(): mendapatkan row berikut dalam result set. Akan mengembalikan sequence tunggal, atau None apabila tidak ada data.
- fetchmany([size=cursor.arraysize]): mendapatkan sejumlah row berikut dalam result set. Akan mengembalikan sequence dari sequence, atau sequence kosong apabila tidak ada data.
- fetchall(): mendapatkan semua row tersisa dalam result set. Akan mengembalikan sequence dari sequence, atau sequence kosong apabila tidak ada data.

Contoh-contoh query PostgreSQL dapat dilihat pada pgsql_query_rowcount.py dan pgsql_query_select.py. Contoh-contoh query MySQL dapat dilihat pada mysql_query_rowcount.py dan mysql_query_select.py.

pgsql_query_rowcount.py:

```
#!/usr/bin/env python

import psycopg2 as pgsql

conn = pgsql.connect(user='user', database='testing')
```

```
cur = conn.cursor()
cur.execute('''
insert into books(author, title) values(%s, %s)
''', ('author1', 'title1'))
conn.commit()

print 'Rowcount: %d' %(cur.rowcount)

cur.close()
conn.close()
```

pgsql_query_select.py:

```
#!/usr/bin/env python

import psycopg2 as pgsql

conn = pgsql.connect(user='user', database='testing')

cur = conn.cursor()

print 'Fetch all'
cur.execute('''select * from books''')
books = cur.fetchall()
print books

print 'Fetch one'
cur.execute('''select * from books''')
books = []
while True:
    temp = cur.fetchone()
    if temp:
        books.append(temp)
    else:
        break
print books

print 'Fetch first 3'
cur.execute('''select * from books''')
books = cur.fetchmany(3)
print books

cur.close()
conn.close()
```

mysql_query_rowcount.py:

```
#!/usr/bin/env python

import MySQLdb as mysql
```

```
conn = mysql.connect(user='user', db='testing')

cur = conn.cursor()
cur.execute('''
insert into books(author, title) values(%s, %s)
''', ('author1', 'title1'))
conn.commit()

print 'Rowcount: %d' %(cur.rowcount)

cur.close()
conn.close()
```

mysql_query_select.py:

```
#!/usr/bin/env python

import MySQLdb as mysql

conn = mysql.connect(user='user', db='testing')

cur = conn.cursor()

print 'Fetch all'
cur.execute('''select * from books''')
books = cur.fetchall()
print books

print 'Fetch one'
cur.execute('''select * from books''')
books = []
while True:
    temp = cur.fetchone()
    if temp:
        books.append(temp)
    else:
        break
print books

print 'Fetch first 3'
cur.execute('''select * from books''')
books = cur.fetchmany(3)
print books

cur.close()
conn.close()
```

Representasi tipe data

Salah satu fitur DB-API 2.0 yang sangat menarik adalah representasi tipe data pada database ke tipe data Python, secara otomatis.

Sebagai contoh, apabila query select mengembalikan kolom date/datetime/sejenis, maka sequence hasil kembalian dapat mengandung data dengan tipe datetime. Begitupun halnya dengan bilangan, string dan lainnya (seperti contoh-contoh sebelumnya).

Lihatlah contoh-contoh pgsql_query_date.py dan mysql_query_date.py.

pgsql_query_date.py:
```
#!/usr/bin/env python

import psycopg2 as pgsql

conn = pgsql.connect(user='user', database='testing')

cur = conn.cursor()
cur.execute('''select now()''')
dateinfo = cur.fetchall()
print dateinfo

cur.close()
conn.close()
```

mysql_query_date.py:
```
#!/usr/bin/env python

import MySQLdb as mysql

conn = mysql.connect(user='user', db='testing')

cur = conn.cursor()
cur.execute('''select now()''')
dateinfo = cur.fetchall()
print dateinfo

cur.close()
conn.close()
```

Catatan:
- Tipe NULL SQL diwakili dengan None Python.
- Selengkapnya, bacalah PEP 249.

5. Transaction

Untuk database yang mendukung transaksi, method commit() dan rollback() objek connection dapat digunakan, masing-masing untuk melakukan transaction commit dan rollback.

Lihatlah contoh-contoh pgsql_trans.py dan mysql_trans.py. Terdapat perbedaan yang mencolok, karena PostgreSQL mendukung transaksi dan engine yang digunakan pada MySQL adalah yang tidak mendukung transaksi.

Program contoh akan:
- Mendapatkan semua record dalam tabel books
- Menghapus isi tabel books
- Rollback. Apabila mendukung transaksi, maka penghapusan isi tabel tidak akan disimpan.
- Mendapatkan semua record dalam tabel books.

pgsql_trans.py:
```
#!/usr/bin/env python

import psycopg2 as pgsql

conn = pgsql.connect(user='user', database='testing')

cur = conn.cursor()

print 'Select *'
cur.execute('''select * from books''')
books = cur.fetchall()
print books

print 'Delete from'
cur.execute('''delete from books''')
print 'Rowcount: %d' %(cur.rowcount)
conn.rollback()

print 'Rollback, Select *'
cur.execute('''select * from books''')
books = cur.fetchall()
print books

cur.close()
conn.close()
```

mysql_trans.py:
```
#!/usr/bin/env python

import MySQLdb as mysql

conn = mysql.connect(user='user', db='testing')
```

```
cur = conn.cursor()

print 'Select *'
cur.execute('''select * from books''')
books = cur.fetchall()
print books

print 'Delete from'
cur.execute('''delete from books''')
print 'Rowcount: %d' %(cur.rowcount)
conn.rollback()

print 'Rollback, Select *'
cur.execute('''select * from books''')
books = cur.fetchall()
print books

cur.close()
conn.close()
```