Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе
Дисциплина: Базы данных
Тема: Создание интерактивного генератора данных

Выполнил студент группы 43501/3   _____ Круминьш Д.В.
                                                                    (подпись)

Преподаватель   _____ Мяснов А.В.
                                                                    (подпись)

Санкт-Петербург
2017

1 Цель работы

Получить практические навыки работы с БД путем создания собственного интерактивного генератора данных на языке программирования **python**.

2 Ход работы

Была создана команда **generate**, которая имеет два входных параметра:

1. **tableName** - название таблицы для которой необходимо сгенерировать данные. В случае ввода **all** будет генерация для всех таблиц.

2. **count** - целочисленное число, обозначающие количество строк, которые необходимо сгенерировать.

Для создание большей интерактивности, необходимые данные берутся из заранее созданных файлов. Далее приведен список этих файлов:

- names.txt
- surnames.txt
- oldnames.txt
- manufacturers.txt
- country.txt
- property.txt
- properties.txt
- type.txt
- product_name.txt
- product_description.txt
- review_description.txt
- address.txt
- phone.txt

Например формирование ФИО клиента будет происходить путем взятия случайных строк из файлов **names.txt, surnames.txt, oldnames.txt**.

```
 1  C:\study\s08\BD\task_2\work\lab_1>python manage.py generate all 10
 2  10 row(s) successfully added in table my_user.
 3  10 row(s) successfully added in table manufacturer.
 4  10 row(s) successfully added in table sells.
 5  10 row(s) successfully added in table property.
 6  10 row(s) successfully added in table type.
 7  10 row(s) successfully added in table type_prop.
 8  10 row(s) successfully added in table product.
 9  10 row(s) successfully added in table product_types.
10  10 row(s) successfully added in table review.
```

```
11 │ 10 row(s) successfully added in table storage.
12 │ 10 row(s) successfully added in table product_avaliability.
13 │ 10 row(s) successfully added in table properties.
14 │ 10 row(s) successfully added in table supply.
15 │ 10 row(s) successfully added in table sells_entre.
```
Листинг 1: Пример изпользования команды

Код команды **generate** представлен в приложении 1.

3   Вывод

В ходе данной работы было продолжено создание собственного приложения, которое работает с базой данных. В частности был написан собственный генератор данных. В отличии от встроенных генераторов в какие-либо СУБД, в данном случае генератор в конечном итоге получается более гибким, который можно как-либо изменять.

Хоть и генератор является гибким, у него имеются некоторые проблемы с производительностью, время генерации данных у него заметно выше чем при использовании СУБД. Скорее всего это вызвано тем, что многие данных для записи в таблицы считываются из файлов, что замедляет генерацию.

Так-же на время генерации влияют и некоторые проверки на возможность генерации данных. Например имеются ли в таблице, которая связана с текущей по вторичному ключу данные, и если есть то какой диапазон id имеется в данной таблице.

4   Приложение

Приложение 1

```
1  from django.core.management.base import BaseCommand
2  from django.db.models import Max, Min
3  from ulmart.models import *
4  import random
5  import datetime
6
7  NAMES='Data/names.txt'
8  SURNAMES='Data/surnames.txt'
9  OLDNAMES='Data/oldnames.txt'
10 MANUFACTURERS='Data/manufacturers.txt'
11 COUNTRY='Data/country.txt'
12 PROPERTY='Data/property.txt'
13 PROPERTIES='Data/properties.txt'
14 TYPE='Data/type.txt'
15 PRODUCT_NAME='Data/product_name.txt'
16 PRODUCT_DESCRIPTION='Data/product_description.txt'
17 REVIEW_DESCRIPTION='Data/review_description.txt'
18 ADDRESS='Data/address.txt'
19 PHONE='Data/phone.txt'
20
21 MAXIMUM_LEVEL=3
22
23 class Command(BaseCommand):
24     def add_arguments(self, parser):
```

```
25          parser.add_argument('table')
26          parser.add_argument('count')
27
28      def getLinesCount(self, filename):
29          with open(filename, 'r') as f:
30              return(sum(1 for _ in f))
31
32      def getRandomLine(self, filename):
33          #Random int between 0 and line's count
34          num=random.randint(0,self.getLinesCount(filename)-1)
35
36          #Opening file, and searching for needed line
37          f = open(filename, 'r')
38          i=0
39          for line in f:
40              if i==num:
41                  return(str.strip(line))
42              i+=1
43          return("null")
44
45      def addUsers(self, count):
46          #Check if this table is empty
47          if My_user.objects.count()==0:
48              max_id=0
49          else:
50              max_id = My_user.objects.order_by('-user_id')[0].user_id
51
52          #Starting of loop
53          i=1
54          while i<=count:
55              new_id=max_id+i
56              new_name=self.getRandomLine(NAMES)
57              new_surname=self.getRandomLine(SURNAMES)
58              new_oldname=self.getRandomLine(OLDNAMES)
59              new_date=datetime.date(random.randint(2006,2016), random.
   ↪  randint(1,12),random.randint(1,28))
60
61              #Creating new object and saving it
62              new_user = My_user(user_id=new_id, name=new_name, surname=
   ↪  new_surname, oldname=new_oldname, reg_date=new_date)
63              new_user.save()
64
65              i+=1
66          print(str(count)+" row(s) successfully added in table my_user.
   ↪  ")
67
68      def addSells(self, count):
69          #Check if there is no users
70          if My_user.objects.count()==0:
71              print('No users!')
72              return
73
74          #Check if this table is empty
```

```python
75          if Sells.objects.count()==0:
76              max_id=0
77          else:
78              max_id = Sells.objects.order_by('-sell_id')[0].sell_id
79
80          #Variables for generation limits
81          min_user_id=My_user.objects.order_by('user_id')[0].user_id
82          max_user_id=My_user.objects.order_by('-user_id')[0].user_id
83
84          #Starting of loop
85          i=1
86          while i<=count:
87              new_id=max_id+i
88              new_user_id=random.randint(min_user_id, max_user_id)
89              new_date=datetime.date(random.randint(2006,2016), random.
   ↪ randint(1,12),random.randint(1,28))
90
91              #Creating new object and saving it
92              new_sell = Sells(sell_id=new_id, user_id=new_user_id,
   ↪ sell_date=new_date)
93              new_sell.save()
94
95              i+=1
96          print(str(count)+" row(s) successfully added in table sells.")
97
98      def addManufacturers(self, count):
99          #Check if this table is empty
100         if Manufacturer.objects.count()==0:
101             max_id=0
102         else:
103             max_id = Manufacturer.objects.order_by('-manufacturer_id')
   ↪ [0].manufacturer_id
104
105         #Starting of loop
106         i=1
107         while i<=count:
108             new_id=max_id+i
109             new_name=self.getRandomLine(MANUFACTURERS)
110             new_country=self.getRandomLine(COUNTRY)
111
112             #Creating new object and saving it
113             new_manufacturer = Manufacturer(manufacturer_id=new_id,
   ↪ name=new_name, country=new_country)
114             new_manufacturer.save()
115
116             i+=1
117         print(str(count)+" row(s) successfully added in table
   ↪ manufacturer.")
118
119     def addProperty(self, count):
120         #Check if this table is empty
121         if Property.objects.count()==0:
122             max_id=0
```

```
123        else:
124            max_id = Property.objects.order_by('-property_id')[0].
   ↪  property_id
125
126        #Starting of loop
127        i=1
128        while i<=count:
129            new_id=max_id+i
130            new_name=self.getRandomLine(PROPERTY)
131
132            #Creating new object and saving it
133            new_property = Property(property_id=new_id, name=new_name)
134            new_property.save()
135
136            i+=1
137        print(str(count)+" row(s) successfully added in table property
   ↪  .")
138
139    def addType_prop(self, count):
140        #Check if there is no data in property or type
141        if Property.objects.count()==0 or Type.objects.count()==0:
142            print('No data in property or type table!')
143            return
144
145        #Check if this table is empty
146        if Type_prop.objects.count()==0:
147            max_id=0
148        else:
149            max_id = Type_prop.objects.order_by('-id')[0].id
150
151        #Variables for generation limits
152        min_property_id=Property.objects.order_by('property_id')[0].
   ↪  property_id
153        max_property_id=Property.objects.order_by('-property_id')[0].
   ↪  property_id
154
155        min_type_id=Type.objects.order_by('type_id')[0].type_id
156        max_type_id=Type.objects.order_by('-type_id')[0].type_id
157
158        #Starting of loop
159        i=1
160        while i<=count:
161            new_id=max_id+i
162            new_type_id=random.randint(min_type_id, max_type_id)
163            new_property_id=random.randint(min_property_id,
   ↪  max_property_id)
164
165            #Creating new object and saving it
166            new_type_prop = Type_prop(id=new_id, property_id=
   ↪  new_property_id, type_id=new_type_id)
167            new_type_prop.save()
168
169            i+=1
```

5

```
170         print(str(count)+" row(s) successfully added in table
    ↪ type_prop.")
171
172     def addSells_entre(self, count):
173         #Check if there is no data in storage or product
174         if Storage.objects.count()==0 or Product.objects.count()==0 or
    ↪  Sells.objects.count()==0:
175             print('No data in storage or product or sells table!')
176             return
177
178         #Check if this table is empty
179         if Sells_entre.objects.count()==0:
180             max_id=0
181         else:
182             max_id = Sells_entre.objects.order_by('-id')[0].id
183
184         #Variables for generation limits
185         min_product_id=Product.objects.order_by('product_id')[0].
    ↪ product_id
186         max_product_id=Product.objects.order_by('-product_id')[0].
    ↪ product_id
187
188         min_storage_id=Storage.objects.order_by('storage_id')[0].
    ↪ storage_id
189         max_storage_id=Storage.objects.order_by('-storage_id')[0].
    ↪ storage_id
190
191         min_sell_id=Sells.objects.order_by('sell_id')[0].sell_id
192         max_sell_id=Sells.objects.order_by('-sell_id')[0].sell_id
193
194         #Starting of loop
195         i=1
196         while i<=count:
197             new_id=max_id+i
198             new_sell_id=random.randint(min_sell_id, max_sell_id)
199             new_product_id=random.randint(min_product_id,
    ↪ max_product_id)
200             new_storage_id=random.randint(min_storage_id,
    ↪ max_storage_id)
201             new_product_price=random.uniform(1000, 40000)
202             new_quantity=random.randint(1,100)
203
204             #Creating new object and saving it
205             new_sells_entre = Sells_entre(id=new_id, sell_id=
    ↪ new_sell_id, product_id=new_product_id, storage_id=
    ↪ new_storage_id, product_price=new_product_price, quantity=
    ↪ new_quantity)
206             new_sells_entre.save()
207
208             i+=1
209         print(str(count)+" row(s) successfully added in table
    ↪ sells_entre.")
210
```

```python
    def addSupply(self, count):
        #Check if there is no data in storage or product
        if Storage.objects.count()==0 or Product.objects.count()==0:
            print('No data in storage or product table!')
            return

        #Check if this table is empty
        if Supply.objects.count()==0:
            max_id=0
        else:
            max_id = Supply.objects.order_by('-id')[0].id

        #Variables for generation limits
        min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
        max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id

        min_storage_id=Storage.objects.order_by('storage_id')[0].
↪ storage_id
        max_storage_id=Storage.objects.order_by('-storage_id')[0].
↪ storage_id

        #Starting of loop
        i=1
        while i<=count:
            new_id=max_id+i
            new_product_id=random.randint(min_product_id,
↪ max_product_id)
            new_storage_id=random.randint(min_storage_id,
↪ max_storage_id)
            new_supply_date=datetime.date(random.randint(2006,2016),
↪ random.randint(1,12),random.randint(1,28))
            new_quantity=random.randint(1,500)

            #Creating new object and saving it
            new_supply = Supply(id=new_id, product_id=new_product_id,
↪ storage_id=new_storage_id, supply_date=new_supply_date, quantity
↪ =new_quantity)
            new_supply.save()

            i+=1
        print(str(count)+" row(s) successfully added in table supply."
↪ )

    def addProperties(self, count):
        #Check if there is no data in property or product
        if Property.objects.count()==0 or Product.objects.count()==0:
            print('No data in property or product table!')
            return

        #Check if this table is empty
        if Properties.objects.count()==0:
```

```
254              max_id=0
255          else:
256              max_id = Properties.objects.order_by('-id')[0].id
257
258          #Variables for generation limits
259          min_property_id=Property.objects.order_by('property_id')[0].
    ↪ property_id
260          max_property_id=Property.objects.order_by('-property_id')[0].
    ↪ property_id
261
262          min_product_id=Product.objects.order_by('product_id')[0].
    ↪ product_id
263          max_product_id=Product.objects.order_by('-product_id')[0].
    ↪ product_id
264
265          #Starting of loop
266          i=1
267          while i<=count:
268              new_id=max_id+i
269              new_product_id=random.randint(min_product_id,
    ↪ max_product_id)
270              new_property_id=random.randint(min_property_id,
    ↪ max_property_id)
271              new_prop_value=self.getRandomLine(PROPERTIES)
272
273              #Creating new object and saving it
274              new_properties = Properties(id=new_id, product_id=
    ↪ new_product_id, property_id=new_property_id, prop_value=
    ↪ new_prop_value)
275              new_properties.save()
276
277              i+=1
278          print(str(count)+" row(s) successfully added in table
    ↪ properties.")
279
280     def addProduct_avaliability(self, count):
281          #Check if there is no data in storage or product
282          if Storage.objects.count()==0 or Product.objects.count()==0:
283              print('No data in storage or product table!')
284              return
285
286          #Check if this table is empty
287          if Product_avaliability.objects.count()==0:
288              max_id=0
289          else:
290              max_id = Product_avaliability.objects.order_by('-id')[0].
    ↪ id
291
292          #Variables for generation limits
293          min_product_id=Product.objects.order_by('product_id')[0].
    ↪ product_id
294          max_product_id=Product.objects.order_by('-product_id')[0].
    ↪ product_id
```

```
295
296        min_storage_id=Storage.objects.order_by('storage_id')[0].
    ↪ storage_id
297        max_storage_id=Storage.objects.order_by('-storage_id')[0].
    ↪ storage_id
298
299        #Starting of loop
300        i=1
301        while i<=count:
302            new_id=max_id+i
303            new_product_id=random.randint(min_product_id,
    ↪ max_product_id)
304            new_storage_id=random.randint(min_storage_id,
    ↪ max_storage_id)
305            new_quantity=random.randint(1,500)
306
307            #Creating new object and saving it
308            new_product_avaliability = Product_avaliability(id=new_id,
    ↪  product_id=new_product_id, storage_id=new_storage_id, quantity=
    ↪ new_quantity)
309            new_product_avaliability.save()
310
311            i+=1
312        print(str(count)+" row(s) successfully added in table
    ↪ product_avaliability.")
313
314   def addProduct_types(self, count):
315        #Check if there is no data in product or type
316        if Product.objects.count()==0 or Type.objects.count()==0:
317            print('No data in product or type table!')
318            return
319
320        #Check if this table is empty
321        if Product_types.objects.count()==0:
322            max_id=0
323        else:
324            max_id = Product_types.objects.order_by('-id')[0].id
325
326        #Variables for generation limits
327        min_product_id=Product.objects.order_by('product_id')[0].
    ↪ product_id
328        max_product_id=Product.objects.order_by('-product_id')[0].
    ↪ product_id
329
330        min_type_id=Type.objects.order_by('type_id')[0].type_id
331        max_type_id=Type.objects.order_by('-type_id')[0].type_id
332
333        #Starting of loop
334        i=1
335        while i<=count:
336            new_id=max_id+i
337            new_type_id=random.randint(min_type_id, max_type_id)
338            new_product_id=random.randint(min_product_id,
```

```
                ↪ max_product_id)
339
340                #Creating new object and saving it
341                new_product_type = Product_types(id=new_id, product_id=
                ↪ new_product_id, type_id=new_type_id)
342                new_product_type.save()
343
344                i+=1
345            print(str(count)+" row(s) successfully added in table
                ↪ product_types.")
346
347     def addProduct(self, count):
348            #Check if there is no data in manufacturer
349            if Manufacturer.objects.count()==0:
350                print('No data in manufacturer table!')
351                return
352
353            #Check if this table is empty
354            if Product.objects.count()==0:
355                max_id=0
356            else:
357                max_id = Product.objects.order_by('-product_id')[0].
                ↪ product_id
358
359            #Variables for generation limits
360            min_manufacturer_id=Manufacturer.objects.order_by('
                ↪ manufacturer_id')[0].manufacturer_id
361            max_manufacturer_id=Manufacturer.objects.order_by('-
                ↪ manufacturer_id')[0].manufacturer_id
362
363            #Starting of loop
364            i=1
365            while i<=count:
366                new_id=max_id+i
367                new_manufacturer_id=random.randint(min_manufacturer_id,
                ↪ max_manufacturer_id)
368                new_name=self.getRandomLine(PRODUCT_NAME)
369                new_description=self.getRandomLine(PRODUCT_DESCRIPTION)
370                new_price=random.uniform(1000, 40000)
371
372                #Creating new object and saving it
373                new_product = Product(product_id=new_id, manufacturer_id=
                ↪ new_manufacturer_id, name=new_name, description=new_description,
                ↪  price=new_price)
374                new_product.save()
375
376                i+=1
377            print(str(count)+" row(s) successfully added in table product.
                ↪ ")
378
379     def addStorage(self, count):
380            #Check if this table is empty
381            if Storage.objects.count()==0:
```

```
382          max_id=0
383      else:
384          max_id = Storage.objects.order_by('-storage_id')[0].
     ↪ storage_id
385
386      #Starting of loop
387      i=1
388      while i<=count:
389          new_id=max_id+i
390          new_address=self.getRandomLine(ADDRESS)
391          new_phone=self.getRandomLine(PHONE)
392
393          #Creating new object and saving it
394          new_storage = Storage(storage_id=new_id, address=
     ↪ new_address, phone=new_phone)
395          new_storage.save()
396
397          i+=1
398      print(str(count)+" row(s) successfully added in table storage.
     ↪ ")
399
400  def addReview(self, count):
401      #Check if there is no data in product or my_user
402      if Product.objects.count()==0 or My_user.objects.count()==0:
403          print('No data in product or my_user table!')
404          return
405
406      #Check if this table is empty
407      if Review.objects.count()==0:
408          max_id=0
409      else:
410          max_id = Review.objects.order_by('-id')[0].id
411
412      #Variables for generation limits
413      min_product_id=Product.objects.order_by('product_id')[0].
     ↪ product_id
414      max_product_id=Product.objects.order_by('-product_id')[0].
     ↪ product_id
415
416      min_my_user_id=My_user.objects.order_by('user_id')[0].user_id
417      max_my_user_id=My_user.objects.order_by('-user_id')[0].user_id
418
419      #Starting of loop
420      i=1
421      while i<=count:
422          new_id=max_id+i
423          new_product_id=random.randint(min_product_id,
     ↪ max_product_id)
424          new_user_id=random.randint(min_my_user_id, max_my_user_id)
425          new_rating=random.randint(1,5)
426          new_description=self.getRandomLine(REVIEW_DESCRIPTION)
427
428          #Creating new object and saving it
```

```
429         new_review = Review(id=new_id, product_id=new_product_id,
      ↪ user_id=new_user_id, rating=new_rating, description=
      ↪ new_description)
430         new_review.save()
431
432         i+=1
433     print(str(count)+" row(s) successfully added in table review."
      ↪ )
434
435  def addType(self, count):
436     #Check if this table is empty
437     if Type.objects.count()==0:
438         max_id=0
439     else:
440         max_id = Type.objects.order_by('-type_id')[0].type_id
441
442     #Starting of loop
443     i=1
444     while i<=count:
445         new_id=max_id+i
446         new_name=self.getRandomLine(TYPE)
447
448         #50 at 50 if new type will have parent, also checking if
      ↪ parent is possible
449         if random.randint(0,1)==0 or new_id==1:
450             new_p_id=None
451             new_level=1
452         else:
453             #Random parent
454             min_p_id=Type.objects.order_by('type_id')[0].type_id
455             max_p_id=Type.objects.order_by('-type_id')[0].type_id
456             new_p_id=random.randint(min_p_id, max_p_id)
457             #Selecting parent level and incrasing it
458             new_level=Type.objects.get(pk=new_p_id).level+1
459             if new_level>MAXIMUM_LEVEL:
460                 continue
461
462         #Creating new object and saving it
463         new_type = Type(type_id=new_id, p_id=new_p_id, name=
      ↪ new_name, level=new_level)
464         new_type.save()
465
466         i+=1
467     print(str(count)+" row(s) successfully added in table type.")
468
469  def handle(self, *args, **options):
470     #Reading input options
471     table = options['table']
472     count = int(options['count'])
473
474     #Checking of options
475     if count<=0:
476         print('Wrong count!')
```

12

```
477                return
478            if table=='my_user':
479                self.addUsers(count)
480            elif table=='sells':
481                self.addSells(count)
482            elif table=='manufacturer':
483                self.addManufacturers(count)
484            elif table=='property':
485                self.addProperty(count)
486            elif table=='type':
487                self.addType(count)
488            elif table=='type_prop':
489                self.addType_prop(count)
490            elif table=='supply':
491                self.addSupply(count)
492            elif table=='product':
493                self.addProduct(count)
494            elif table=='product_types':
495                self.addProduct_types(count)
496            elif table=='review':
497                self.addReview(count)
498            elif table=='properties':
499                self.addProperties(count)
500            elif table=='sells_entre':
501                self.addSells_entre(count)
502            elif table=='storage':
503                self.addStorage(count)
504            elif table=='product_avaliability':
505                self.addProduct_avaliability(count)
506            elif table=='all':
507                self.addUsers(count)
508                self.addManufacturers(count)
509                self.addSells(count)
510                self.addProperty(count)
511                self.addType(count)
512                self.addType_prop(count)
513                self.addProduct(count)
514                self.addProduct_types(count)
515                self.addReview(count)
516                self.addStorage(count)
517                self.addProduct_avaliability(count)
518                self.addProperties(count)
519                self.addSupply(count)
520                self.addSells_entre(count)
```

Листинг 2: generate.py