

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)

Лабораторная работа

1.1 Цель работы

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

1.2 Ход работы

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

1. **tableName** - название таблицы или области для которой необходимо сгенерировать данные. В случае ввода **all** будет генерация для всех таблиц.
2. **count** - целочисленное число, обозначающие количество строк, которые необходимо сгенерировать.

Также есть опциональный параметр:


1. **-f** - в случае добавления параметра, данные будут генерироваться случайным образом, а не путем взятия случайных строк из заранее подготовленных текстовых файлов.

Необходимые, для генерации, данные берутся из заранее созданных файлов с соответствующим содержанием. Далее приведен список этих файлов:

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

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

Наиболее интересной таблицей для генерации является таблица **Type**, представленная на рисунке 1.1.




Type		
	type_id	integer
	p_id	integer
	name	varchar
	level	integer

Рис. 1.1: Таблица Type

Если в работах прошлого семестра, поле level не использовалось, то в данной работе, с помощью него контролируется глубина вложенности. Так при генерации новой записи, у родительского элемента(если таковой имеется) проверяется значение поля level. Таким образом можно задать максимальную вложенность. Так-же имеется разделенность, схемы базы данных, по областям в соответствии с их контентом.

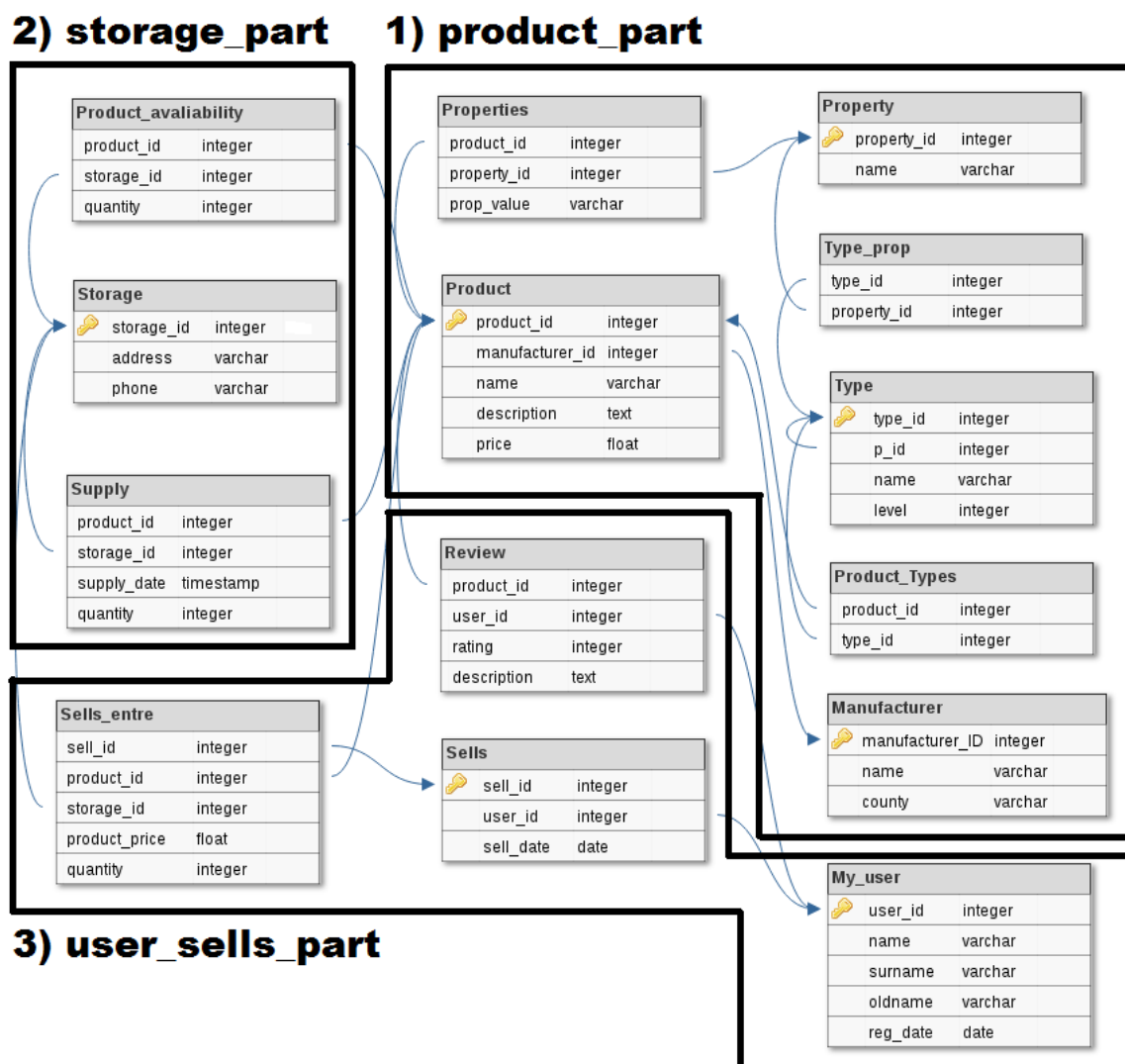


Рис. 1.2: Разделенная на области схема БД

Для каждой из областей также задается параметр count, однако для вложенных таблиц его коэффициент будет несколько изменяться.

1. product_part

- Product -> count*1
- Manufacturer -> count*2
- Property -> count*20
- Properties -> count*10
- Type -> count*4
- Type_prop -> count*10
- Product_types -> count*2

2. storage_part

- Storage -> count*1
- Product_avaliability -> count*10
- Supply -> count*10

3. user_sells_part

- My_user -> count
- Sells -> count*5
- Sells_entre -> count*10
- Review -> count*10

Далее приведен пример использования команды, для генерации 10 новых строчек в каждую из таблиц базы данных.

```
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 1.1: Пример использования команды

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

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.

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 import string
7 import argparse
8
9 NAMES='Data/names.txt'
10 SURNAMES='Data/surnames.txt'
11 OLDNAMES='Data/oldnames.txt'
12 MANUFACTURERS='Data/manufacturers.txt'
13 COUNTRY='Data/country.txt'
14 PROPERTY='Data/property.txt'
15 PROPERTIES='Data/properties.txt'
16 TYPE='Data/type.txt'
17 PRODUCT_NAME='Data/product_name.txt'
18 PRODUCT_DESCRIPTION='Data/product_description.txt'
19 REVIEW_DESCRIPTION='Data/review_description.txt'
20 ADDRESS='Data/address.txt'
21 PHONE='Data/phone.txt'
22
23 MAXIMUM_LEVEL=3
24
25 class Command(BaseCommand):
26     def add_arguments(self, parser):
27         parser.add_argument('table', type=str)
28         parser.add_argument('count', type=int)
29         parser.add_argument("-f", "--fromFile", action="store_true")
30
31     def getLinesCount(self, filename):
32         with open(filename, 'r') as f:
33             return(sum(1 for _ in f))
34
35     def getRandomLine(self, filename):
36         #Random int between 0 and line's count
37         num=random.randint(0,self.getLinesCount(filename)-1)
38
39         #Opening file, and searching for needed line
40         f = open(filename, 'r')
41         i=0
42         for line in f:
43             if i==num:
44                 return(str.strip(line))
45             i+=1
46         return("null")
47
48     def getRandomString(self):
49         return(''.join(random.choice(string.ascii_uppercase + string.
50 ↪ digits) for _ in range(10)))
```

```

51 def addUsers(self, count, fromFile):
52     #Counter
53     added=0
54
55     #Check if this table is empty
56     if My_user.objects.count()==0:
57         max_id=0
58     else:
59         max_id = My_user.objects.order_by('-user_id')[0].user_id
60
61     #Starting of loop
62     i=1
63     while i<=count:
64         new_id=max_id+i
65         if fromFile:
66             new_name=self.getRandomLine(NAMES)
67             new_surname=self.getRandomLine(SURNAMES)
68             new_oldname=self.getRandomLine(OLDNAMES)
69         else:
70             new_name=self.getRandomString()
71             new_surname=self.getRandomString()
72             new_oldname=self.getRandomString()
73         new_date=datetime.date(random.randint(2006,2016), random.
↪ randint(1,12),random.randint(1,28))
74
75         #Creating new object and saving it
76         try:
77             new_user = My_user(user_id=new_id, name=new_name,
↪ surname=new_surname, oldname=new_oldname, reg_date=new_date)
78             new_user.save()
79             added=added+1
80         except:
81             print('Error while trying add new row.')
82             i+=1
83         print(str(added)+" row(s) added in table my_user.")
84
85 def addSells(self, count):
86     #Counter
87     added=0
88
89     #Check if there is no users
90     if My_user.objects.count()==0:
91         print('No users!')
92         return
93
94     #Check if this table is empty
95     if Sells.objects.count()==0:
96         max_id=0
97     else:
98         max_id = Sells.objects.order_by('-sell_id')[0].sell_id
99
100     #Variables for generation limits
101     min_user_id=My_user.objects.order_by('user_id')[0].user_id

```

```

102         max_user_id=My_user.objects.order_by('-user_id')[0].user_id
103
104     #Starting of loop
105     i=1
106     while i<=count:
107         new_id=max_id+i
108         new_user_id=random.randint(min_user_id, max_user_id)
109         new_date=datetime.date(random.randint(2006,2016), random.
↪ randint(1,12),random.randint(1,28))
110
111         #Creating new object and saving it
112         try:
113             new_sell = Sells(sell_id=new_id, user_id=new_user_id,
↪ sell_date=new_date)
114             new_sell.save()
115             added=added+1
116         except:
117             print('Error while trying add new row.')
118             i+=1
119         print(str(added)+" row(s) added in table sells.")
120
121     def addManufacturers(self, count, fromFile):
122         #Counter
123         added=0
124
125         #Check if this table is empty
126         if Manufacturer.objects.count()==0:
127             max_id=0
128         else:
129             max_id = Manufacturer.objects.order_by('-manufacturer_id')
↪ [0].manufacturer_id
130
131     #Starting of loop
132     i=1
133     while i<=count:
134         new_id=max_id+i
135         if fromFile:
136             new_name=self.getRandomLine(MANUFACTURERS)
137             new_country=self.getRandomLine(COUNTRY)
138         else:
139             new_name=self.getRandomString()
140             new_country=self.getRandomString()
141
142         #Creating new object and saving it
143         try:
144             new_manufacturer = Manufacturer(manufacturer_id=new_id,
↪ name=new_name, country=new_country)
145             new_manufacturer.save()
146             added=added+1
147         except:
148             print('Error while trying add new row.')
149             i+=1
150         print(str(added)+" row(s) added in table manufacturer.")

```

```

151
152 def addProperty(self, count, fromFile):
153     #Counter
154     added=0
155
156     #Check if this table is empty
157     if Property.objects.count()==0:
158         max_id=0
159     else:
160         max_id = Property.objects.order_by('-property_id')[0].
↪ property_id
161
162     #Starting of loop
163     i=1
164     while i<=count:
165         new_id=max_id+i
166         if fromFile:
167             new_name=self.getRandomLine(PROPERTY)
168         else:
169             new_name=self.getRandomString()
170
171         #Creating new object and saving it
172         try:
173             new_property = Property(property_id=new_id, name=
↪ new_name)
174             new_property.save()
175             added=added+1
176         except:
177             print('Error while trying add new row.')
178
179         i+=1
180     print(str(added)+" row(s) added in table property.")
181
182 def addType_prop(self, count):
183     #Counter
184     added=0
185
186     #Check if there is no data in property or type
187     if Property.objects.count()==0 or Type.objects.count()==0:
188         print('No data in property or type table!')
189         return
190
191     #Check if this table is empty
192     if Type_prop.objects.count()==0:
193         max_id=0
194     else:
195         max_id = Type_prop.objects.order_by('-id')[0].id
196
197     #Variables for generation limits
198     min_property_id=Property.objects.order_by('property_id')[0].
↪ property_id
199     max_property_id=Property.objects.order_by('-property_id')[0].
↪ property_id

```



```

200
201     min_type_id=Type.objects.order_by('type_id')[0].type_id
202     max_type_id=Type.objects.order_by('-type_id')[0].type_id
203
204     #Starting of loop
205     i=1
206     while i<=count:
207         new_id=max_id+i
208         new_type_id=random.randint(min_type_id, max_type_id)
209         new_property_id=random.randint(min_property_id,
↪ max_property_id)
210
211         #Creating new object and saving it
212         try:
213             new_type_prop = Type_prop(id=new_id, property_id=
↪ new_property_id, type_id=new_type_id)
214             new_type_prop.save()
215             added=added+1
216         except:
217             print('Error while trying add new row.')
218
219         i+=1
220     print(str(added)+" row(s) added in table type_prop.")
221
222     def addSells_entre(self, count):
223         #Counter
224         added=0
225
226         #Check if there is no data in storage or product
227         if Storage.objects.count()==0 or Product.objects.count()==0 or
↪ Sells.objects.count()==0:
228             print('No data in storage or product or sells table!')
229             return
230
231         #Check if this table is empty
232         if Sells_entre.objects.count()==0:
233             max_id=0
234         else:
235             max_id = Sells_entre.objects.order_by('-id')[0].id
236
237         #Variables for generation limits
238         min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
239         max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
240
241         min_storage_id=Storage.objects.order_by('storage_id')[0].
↪ storage_id
242         max_storage_id=Storage.objects.order_by('-storage_id')[0].
↪ storage_id
243
244         min_sell_id=Sells.objects.order_by('sell_id')[0].sell_id
245         max_sell_id=Sells.objects.order_by('-sell_id')[0].sell_id

```

```

246
247     #Starting of loop
248     i=1
249     while i<=count:
250         new_id=max_id+i
251         new_sell_id=random.randint(min_sell_id, max_sell_id)
252         new_product_id=random.randint(min_product_id, max_product_id)
↪ )
253         new_storage_id=random.randint(min_storage_id, max_storage_id)
↪ )
254         new_product_price=random.uniform(1000, 40000)
255         new_quantity=random.randint(1,100)
256
257         #Creating new object and saving it
258         try:
259             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)
260             new_sells_entre.save()
261             added=added+1
262         except:
263             print('Error while trying add new row.')
264
265         i+=1
266         print(str(added)+" row(s) added in table sells_entre.")
267
268     def addSupply(self, count):
269         #Counter
270         added=0
271
272         #Check if there is no data in storage or product
273         if Storage.objects.count()==0 or Product.objects.count()==0:
274             print('No data in storage or product table!')
275             return
276
277         #Check if this table is empty
278         if Supply.objects.count()==0:
279             max_id=0
280         else:
281             max_id = Supply.objects.order_by('-id')[0].id
282
283         #Variables for generation limits
284         min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
285         max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
286
287         min_storage_id=Storage.objects.order_by('storage_id')[0].
↪ storage_id
288         max_storage_id=Storage.objects.order_by('-storage_id')[0].
↪ storage_id
289
290         #Starting of loop

```

```

291         i=1
292         while i<=count:
293             new_id=max_id+i
294             new_product_id=random.randint(min_product_id, max_product_id
↪ )
295             new_storage_id=random.randint(min_storage_id, max_storage_id
↪ )
296             new_supply_date=datetime.date(random.randint(2006,2016),
↪ random.randint(1,12),random.randint(1,28))
297             new_quantity=random.randint(1,500)
298
299             #Creating new object and saving it
300             try:
301                 new_supply = Supply(id=new_id, product_id=new_product_id
↪ , storage_id=new_storage_id, supply_date=new_supply_date, quantity
↪ =new_quantity)
302                 new_supply.save()
303                 added=added+1
304             except:
305                 print('Error while trying add new row.')
306
307             i+=1
308             print(str(added)+" row(s) added in table supply.")
309
310 def addProperties(self, count, fromFile):
311     #Counter
312     added=0
313
314     #Check if there is no data in property or product
315     if Property.objects.count()==0 or Product.objects.count()==0:
316         print('No data in property or product table!')
317         return
318
319     #Check if this table is empty
320     if Properties.objects.count()==0:
321         max_id=0
322     else:
323         max_id = Properties.objects.order_by('-id')[0].id
324
325     #Variables for generation limits
326     min_property_id=Property.objects.order_by('property_id')[0].
↪ property_id
327     max_property_id=Property.objects.order_by('-property_id')[0].
↪ property_id
328
329     min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
330     max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
331
332     #Starting of loop
333     i=1
334     while i<=count:

```

```

335         new_id=max_id+i
336         new_product_id=random.randint(min_product_id, max_product_id
↪ )
337         new_property_id=random.randint(min_property_id,
↪ max_property_id)
338         if fromFile:
339             new_prop_value=self.getRandomLine(PROPERTIES)
340         else:
341             new_prop_value=self.getRandomString()
342
343         #Creating new object and saving it
344         try:
345             new_properties = Properties(id=new_id, product_id=
↪ new_product_id, property_id=new_property_id, prop_value=
↪ new_prop_value)
346             new_properties.save()
347             added=added+1
348         except:
349             print('Error while trying add new row.')
350
351         i+=1
352         print(str(added)+" row(s) added in table properties.")
353
354     def addProduct_avaliability(self, count):
355         #Counter
356         added=0
357
358         #Check if there is no data in storage or product
359         if Storage.objects.count()==0 or Product.objects.count()==0:
360             print('No data in storage or product table!')
361             return
362
363         #Check if this table is empty
364         if Product_avaliability.objects.count()==0:
365             max_id=0
366         else:
367             max_id = Product_avaliability.objects.order_by('-id')[0].id
368
369         #Variables for generation limits
370         min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
371         max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
372
373         min_storage_id=Storage.objects.order_by('storage_id')[0].
↪ storage_id
374         max_storage_id=Storage.objects.order_by('-storage_id')[0].
↪ storage_id
375
376         #Starting of loop
377         i=1
378         while i<=count:
379             new_id=max_id+i

```

```

380         new_product_id=random.randint(min_product_id, max_product_id
↪ )
381         new_storage_id=random.randint(min_storage_id, max_storage_id
↪ )
382         new_quantity=random.randint(1,500)
383
384         #Creating new object and saving it
385         try:
386             new_product_avaliability = Product_avaliability(id=
↪ new_id, product_id=new_product_id, storage_id=new_storage_id,
↪ quantity=new_quantity)
387             new_product_avaliability.save()
388             added=added+1
389         except:
390             print('Error while trying add new row.')
391             i+=1
392         print(str(added)+" row(s) added in table product_avaliability.")
393
394     def addProduct_types(self, count):
395         #Counter
396         added=0
397
398         #Check if there is no data in product or type
399         if Product.objects.count()==0 or Type.objects.count()==0:
400             print('No data in product or type table!')
401             return
402
403         #Check if this table is empty
404         if Product_types.objects.count()==0:
405             max_id=0
406         else:
407             max_id = Product_types.objects.order_by('-id')[0].id
408
409         #Variables for generation limits
410         min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
411         max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
412
413         min_type_id=Type.objects.order_by('type_id')[0].type_id
414         max_type_id=Type.objects.order_by('-type_id')[0].type_id
415
416         #Starting of loop
417         i=1
418         while i<=count:
419             new_id=max_id+i
420             new_type_id=random.randint(min_type_id, max_type_id)
421             new_product_id=random.randint(min_product_id, max_product_id
↪ )
422
423             #Creating new object and saving it
424             try:

```

```

425         new_product_type = Product_types(id=new_id, product_id=
↪ new_product_id, type_id=new_type_id)
426         new_product_type.save()
427         added=added+1
428     except:
429         print('Error while trying add new row.')
430
431         i+=1
432     print(str(added)+" row(s) added in table product_types.")
433
434 def addProduct(self, count, fromFile):
435     #Counter
436     added=0
437
438     #Check if there is no data in manufacturer
439     if Manufacturer.objects.count()==0:
440         print('No data in manufacturer table!')
441         return
442
443     #Check if this table is empty
444     if Product.objects.count()==0:
445         max_id=0
446     else:
447         max_id = Product.objects.order_by('-product_id')[0].
↪ product_id
448
449     #Variables for generation limits
450     min_manufacturer_id=Manufacturer.objects.order_by('
↪ manufacturer_id')[0].manufacturer_id
451     max_manufacturer_id=Manufacturer.objects.order_by('-
↪ manufacturer_id')[0].manufacturer_id
452
453     #Starting of loop
454     i=1
455     while i<=count:
456         new_id=max_id+i
457         new_manufacturer_id=random.randint(min_manufacturer_id,
↪ max_manufacturer_id)
458         if fromFile:
459             new_name=self.getRandomLine(PRODUCT_NAME)
460             new_description=self.getRandomLine(PRODUCT_DESCRIPTION)
461         else:
462             new_name=self.getRandomString()
463             new_description=self.getRandomString()
464         new_price=random.uniform(1000, 40000)
465
466         #Creating new object and saving it
467         try:
468             new_product = Product(product_id=new_id, manufacturer_id
↪ =new_manufacturer_id, name=new_name, description=new_description,
↪ price=new_price)
469             new_product.save()
470             added=added+1

```

```

471         except:
472             print('Error while trying add new row.')
473
474         i+=1
475         print(str(added)+" row(s) added in table product.")
476
477     def addStorage(self, count, fromFile):
478         #Counter
479         added=0
480
481         #Check if this table is empty
482         if Storage.objects.count()==0:
483             max_id=0
484         else:
485             max_id = Storage.objects.order_by('-storage_id')[0].
↪ storage_id
486
487         #Starting of loop
488         i=1
489         while i<=count:
490             new_id=max_id+i
491             if fromFile:
492                 new_address=self.getRandomLine(ADDRESS)
493                 new_phone=self.getRandomLine(PHONE)
494             else:
495                 new_address=self.getRandomString()
496                 new_phone=self.getRandomString()
497
498             #Creating new object and saving it
499             try:
500                 new_storage = Storage(storage_id=new_id, address=
↪ new_address, phone=new_phone)
501                 new_storage.save()
502                 added=added+1
503             except:
504                 print('Error while trying add new row.')
505
506             i+=1
507             print(str(added)+" row(s) added in table storage.")
508
509     def addReview(self, count, fromFile):
510         #Counter
511         added=0
512
513         #Check if there is no data in product or my_user
514         if Product.objects.count()==0 or My_user.objects.count()==0:
515             print('No data in product or my_user table!')
516             return
517
518         #Check if this table is empty
519         if Review.objects.count()==0:
520             max_id=0
521         else:

```

```

522         max_id = Review.objects.order_by('-id')[0].id
523
524         #Variables for generation limits
525         min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
526         max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
527
528         min_my_user_id=My_user.objects.order_by('user_id')[0].user_id
529         max_my_user_id=My_user.objects.order_by('-user_id')[0].user_id
530
531         #Starting of loop
532         i=1
533         while i<=count:
534             new_id=max_id+i
535             new_product_id=random.randint(min_product_id, max_product_id
↪ )
536             new_user_id=random.randint(min_my_user_id, max_my_user_id)
537             new_rating=random.randint(1,5)
538             if fromFile:
539                 new_description=self.getRandomLine(REVIEW_DESCRIPTION)
540             else:
541                 new_description=self.getRandomString()
542
543             #Creating new object and saving it
544             try:
545                 new_review = Review(id=new_id, product_id=new_product_id
↪ , user_id=new_user_id, rating=new_rating, description=
↪ new_description)
546                 new_review.save()
547                 added=added+1
548             except:
549                 print('Error while trying add new row.')
550
551             i+=1
552         print(str(added)+" row(s) added in table review.")
553
554     def addType(self, count, fromFile):
555         #Counter
556         added=0
557
558         #Check if this table is empty
559         if Type.objects.count()==0:
560             max_id=0
561         else:
562             max_id = Type.objects.order_by('-type_id')[0].type_id
563
564         #Starting of loop
565         i=1
566         while i<=count:
567             new_id=max_id+i
568             if fromFile:
569                 new_name=self.getRandomLine(TYPE)

```



```

570         else:
571             new_name=self.getRandomString()
572
573         #50 at 50 if new type will have parent, also checking if
↪ parent is possible
574         if random.randint(0,1)==0 or new_id==1:
575             new_p_id=None
576             new_level=1
577         else:
578             #Random parent
579             min_p_id=Type.objects.order_by('type_id')[0].type_id
580             max_p_id=Type.objects.order_by('-type_id')[0].type_id
581             new_p_id=random.randint(min_p_id, max_p_id)
582             #Selecting parent level and incrasing it
583             new_level=Type.objects.get(pk=new_p_id).level+1
584             if new_level>MAXIMUM_LEVEL:
585                 continue
586
587         #Creating new object and saving it
588         try:
589             ↪ new_name, level=new_level)
590             new_type = Type(type_id=new_id, p_id=new_p_id, name=
591                 new_type.save()
592                 added=added+1
593         except:
594             print('Error while trying add new row.')
595
596         i+=1
597         print(str(added)+" row(s) added in table type.")
598
599     def handle(self, *args, **options):
600         #Reading input options
601         table = options['table']
602         count = int(options['count'])
603
604         #Checking of options
605         if count<=0:
606             print('Wrong count!')
607             return
608         if table=='my_user':
609             self.addUsers(count, options['fromFile'])
610         elif table=='sells':
611             self.addSells(count)
612         elif table=='manufacturer':
613             self.addManufacturers(count, options['fromFile'])
614         elif table=='property':
615             self.addProperty(count, options['fromFile'])
616         elif table=='type':
617             self.addType(count, options['fromFile'])
618         elif table=='type_prop':
619             self.addType_prop(count)
620         elif table=='supply':
621             self.addSupply(count)

```

```

621 elif table=='product':
622     self.addProduct(count, options['fromFile'])
623 elif table=='product_types':
624     self.addProduct_types(count)
625 elif table=='review':
626     self.addReview(count, options['fromFile'])
627 elif table=='properties':
628     self.addProperties(count, options['fromFile'])
629 elif table=='sells_entre':
630     self.addSells_entre(count)
631 elif table=='storage':
632     self.addStorage(count, options['fromFile'])
633 elif table=='product_avaliability':
634     self.addProduct_avaliability(count)
635 elif table=='product_part':
636     self.addManufacturers(count*2, options['fromFile'])
637     self.addProduct(count, options['fromFile'])
638     self.addProperty(count*20, options['fromFile'])
639     self.addProperties(count*10, options['fromFile'])
640     self.addType(count*4, options['fromFile'])
641     self.addType_prop(count*10)
642     self.addProduct_types(count*2)
643 elif table=='storage_part':
644     self.addStorage(count, options['fromFile'])
645     self.addProduct_avaliability(count*10)
646     self.addSupply(count*10)
647 elif table=='user_sells_part':
648     self.addUsers(count, options['fromFile'])
649     self.addSells(count*5)
650     self.addReview(count*10, options['fromFile'])
651     self.addSells_entre(count*10)
652 elif table=='all':
653     self.addUsers(count, options['fromFile'])
654     self.addManufacturers(count, options['fromFile'])
655     self.addSells(count)
656     self.addProperty(count, options['fromFile'])
657     self.addType(count, options['fromFile'])
658     self.addType_prop(count)
659     self.addProduct(count, options['fromFile'])
660     self.addProduct_types(count)
661     self.addReview(count, options['fromFile'])
662     self.addStorage(count, options['fromFile'])
663     self.addProduct_avaliability(count)
664     self.addProperties(count, options['fromFile'])
665     self.addSupply(count)
666     self.addSells_entre(count)

```

Листинг 1.2: generate.py