

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.

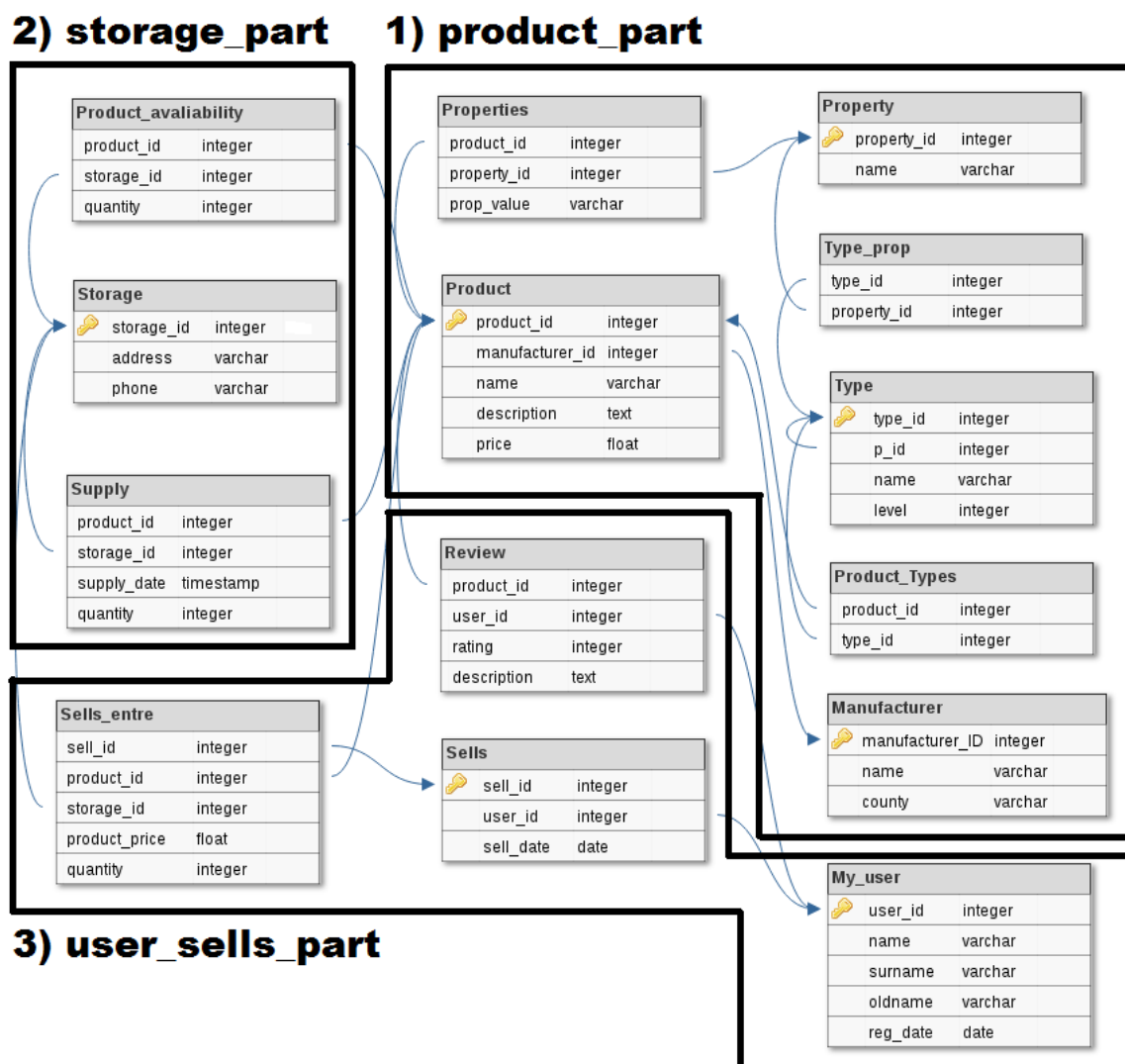


Рис. 1.2: Разделенная на области схема БД

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.
↵ digits) for _ in range(10)))
50
```

```

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

```

```

101
102         #Creating new object and saving it
103         new_sell = Sells(sell_id=new_id, user_id=new_user_id,
↪ sell_date=new_date)
104         new_sell.save()
105
106         i+=1
107         print(str(count)+" row(s) successfully added in table sells.")
108
109     def addManufacturers(self, count, fromFile):
110         #Check if this table is empty
111         if Manufacturer.objects.count()==0:
112             max_id=0
113         else:
114             max_id = Manufacturer.objects.order_by('-manufacturer_id')
↪ [0].manufacturer_id
115
116         #Starting of loop
117         i=1
118         while i<=count:
119             new_id=max_id+i
120             if fromFile:
121                 new_name=self.getRandomLine(MANUFACTURERS)
122                 new_country=self.getRandomLine(COUNTRY)
123             else:
124                 new_name=self.getRandomString()
125                 new_country=self.getRandomString()
126
127             #Creating new object and saving it
128             new_manufacturer = Manufacturer(manufacturer_id=new_id, name
↪ =new_name, country=new_country)
129             new_manufacturer.save()
130
131             i+=1
132             print(str(count)+" row(s) successfully added in table
↪ manufacturer.")
133
134     def addProperty(self, count, fromFile):
135         #Check if this table is empty
136         if Property.objects.count()==0:
137             max_id=0
138         else:
139             max_id = Property.objects.order_by('-property_id')[0].
↪ property_id
140
141         #Starting of loop
142         i=1
143         while i<=count:
144             new_id=max_id+i
145             if fromFile:
146                 new_name=self.getRandomLine(PROPERTY)
147             else:
148                 new_name=self.getRandomString()

```

```

149
150         #Creating new object and saving it
151         new_property = Property(property_id=new_id, name=new_name)
152         new_property.save()
153
154         i+=1
155     print(str(count)+" row(s) successfully added in table property."
↪ )
156
157     def addType_prop(self, count):
158         #Check if there is no data in property or type
159         if Property.objects.count()==0 or Type.objects.count()==0:
160             print('No data in property or type table!')
161             return
162
163         #Check if this table is empty
164         if Type_prop.objects.count()==0:
165             max_id=0
166         else:
167             max_id = Type_prop.objects.order_by('-id')[0].id
168
169         #Variables for generation limits
170         min_property_id=Property.objects.order_by('property_id')[0].
↪ property_id
171         max_property_id=Property.objects.order_by('-property_id')[0].
↪ property_id
172
173         min_type_id=Type.objects.order_by('type_id')[0].type_id
174         max_type_id=Type.objects.order_by('-type_id')[0].type_id
175
176         #Starting of loop
177         i=1
178         while i<=count:
179             new_id=max_id+i
180             new_type_id=random.randint(min_type_id, max_type_id)
181             new_property_id=random.randint(min_property_id,
↪ max_property_id)
182
183             #Creating new object and saving it
184             new_type_prop = Type_prop(id=new_id, property_id=
↪ new_property_id, type_id=new_type_id)
185             new_type_prop.save()
186
187             i+=1
188         print(str(count)+" row(s) successfully added in table type_prop."
↪ ")
189
190     def addSells_entre(self, count):
191         #Check if there is no data in storage or product
192         if Storage.objects.count()==0 or Product.objects.count()==0 or
↪ Sells.objects.count()==0:
193             print('No data in storage or product or sells table!')
194             return

```



```

195
196     #Check if this table is empty
197     if Sells_entre.objects.count()==0:
198         max_id=0
199     else:
200         max_id = Sells_entre.objects.order_by('-id')[0].id
201
202     #Variables for generation limits
203     min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
204     max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
205
206     min_storage_id=Storage.objects.order_by('storage_id')[0].
↪ storage_id
207     max_storage_id=Storage.objects.order_by('-storage_id')[0].
↪ storage_id
208
209     min_sell_id=Sells.objects.order_by('sell_id')[0].sell_id
210     max_sell_id=Sells.objects.order_by('-sell_id')[0].sell_id
211
212     #Starting of loop
213     i=1
214     while i<=count:
215         new_id=max_id+i
216         new_sell_id=random.randint(min_sell_id, max_sell_id)
217         new_product_id=random.randint(min_product_id, max_product_id
↪ )
218         new_storage_id=random.randint(min_storage_id, max_storage_id
↪ )
219         new_product_price=random.uniform(1000, 40000)
220         new_quantity=random.randint(1,100)
221
222         #Creating new object and saving it
223         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)
224         new_sells_entre.save()
225
226         i+=1
227         print(str(count)+" row(s) successfully added in table
↪ sells_entre.")
228
229     def addSupply(self, count):
230         #Check if there is no data in storage or product
231         if Storage.objects.count()==0 or Product.objects.count()==0:
232             print('No data in storage or product table!')
233             return
234
235         #Check if this table is empty
236         if Supply.objects.count()==0:
237             max_id=0
238         else:

```

```

239         max_id = Supply.objects.order_by('-id')[0].id
240
241         #Variables for generation limits
242         min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
243         max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
244
245         min_storage_id=Storage.objects.order_by('storage_id')[0].
↪ storage_id
246         max_storage_id=Storage.objects.order_by('-storage_id')[0].
↪ storage_id
247
248         #Starting of loop
249         i=1
250         while i<=count:
251             new_id=max_id+i
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_supply_date=datetime.date(random.randint(2006,2016),
↪ random.randint(1,12),random.randint(1,28))
255             new_quantity=random.randint(1,500)
256
257             #Creating new object and saving it
258             new_supply = Supply(id=new_id, product_id=new_product_id,
↪ storage_id=new_storage_id, supply_date=new_supply_date, quantity=
↪ new_quantity)
259             new_supply.save()
260
261             i+=1
262             print(str(count)+" row(s) successfully added in table supply.")
263
264     def addProperties(self, count, fromFile):
265         #Check if there is no data in property or product
266         if Property.objects.count()==0 or Product.objects.count()==0:
267             print('No data in property or product table!')
268             return
269
270         #Check if this table is empty
271         if Properties.objects.count()==0:
272             max_id=0
273         else:
274             max_id = Properties.objects.order_by('-id')[0].id
275
276         #Variables for generation limits
277         min_property_id=Property.objects.order_by('property_id')[0].
↪ property_id
278         max_property_id=Property.objects.order_by('-property_id')[0].
↪ property_id
279

```

```

280         min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
281         max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
282
283         #Starting of loop
284         i=1
285         while i<=count:
286             new_id=max_id+i
287             new_product_id=random.randint(min_product_id, max_product_id
↪ )
288             new_property_id=random.randint(min_property_id,
↪ max_property_id)
289             if fromFile:
290                 new_prop_value=self.getRandomLine(PROPERTIES)
291             else:
292                 new_prop_value=self.getRandomString()
293
294             #Creating new object and saving it
295             new_properties = Properties(id=new_id, product_id=
↪ new_product_id, property_id=new_property_id, prop_value=
↪ new_prop_value)
296             new_properties.save()
297
298             i+=1
299             print(str(count)+" row(s) successfully added in table properties
↪ .")
300
301     def addProduct_avaliability(self, count):
302         #Check if there is no data in storage or product
303         if Storage.objects.count()==0 or Product.objects.count()==0:
304             print('No data in storage or product table!')
305             return
306
307         #Check if this table is empty
308         if Product_avaliability.objects.count()==0:
309             max_id=0
310         else:
311             max_id = Product_avaliability.objects.order_by('-id')[0].id
312
313         #Variables for generation limits
314         min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
315         max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
316
317         min_storage_id=Storage.objects.order_by('storage_id')[0].
↪ storage_id
318         max_storage_id=Storage.objects.order_by('-storage_id')[0].
↪ storage_id
319
320         #Starting of loop
321         i=1

```

```

322         while i<=count:
323             new_id=max_id+i
324             new_product_id=random.randint(min_product_id, max_product_id
↪ )
325             new_storage_id=random.randint(min_storage_id, max_storage_id
↪ )
326             new_quantity=random.randint(1,500)
327
328             #Creating new object and saving it
329             new_product_avaliability = Product_avaliability(id=new_id,
↪ product_id=new_product_id, storage_id=new_storage_id, quantity=
↪ new_quantity)
330             new_product_avaliability.save()
331
332             i+=1
333             print(str(count)+" row(s) successfully added in table
↪ product_avaliability.")
334
335     def addProduct_types(self, count):
336         #Check if there is no data in product or type
337         if Product.objects.count()==0 or Type.objects.count()==0:
338             print('No data in product or type table!')
339             return
340
341         #Check if this table is empty
342         if Product_types.objects.count()==0:
343             max_id=0
344         else:
345             max_id = Product_types.objects.order_by('-id')[0].id
346
347         #Variables for generation limits
348         min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
349         max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
350
351         min_type_id=Type.objects.order_by('type_id')[0].type_id
352         max_type_id=Type.objects.order_by('-type_id')[0].type_id
353
354         #Starting of loop
355         i=1
356         while i<=count:
357             new_id=max_id+i
358             new_type_id=random.randint(min_type_id, max_type_id)
359             new_product_id=random.randint(min_product_id, max_product_id
↪ )
360
361             #Creating new object and saving it
362             new_product_type = Product_types(id=new_id, product_id=
↪ new_product_id, type_id=new_type_id)
363             new_product_type.save()
364
365             i+=1

```

```

366         print(str(count)+" row(s) successfully added in table
↪ product_types.")
367
368     def addProduct(self, count, fromFile):
369         #Check if there is no data in manufacturer
370         if Manufacturer.objects.count()==0:
371             print('No data in manufacturer table!')
372             return
373
374         #Check if this table is empty
375         if Product.objects.count()==0:
376             max_id=0
377         else:
378             max_id = Product.objects.order_by('-product_id')[0].
↪ product_id
379
380         #Variables for generation limits
381         min_manufacturer_id=Manufacturer.objects.order_by('
↪ manufacturer_id')[0].manufacturer_id
382         max_manufacturer_id=Manufacturer.objects.order_by('-
↪ manufacturer_id')[0].manufacturer_id
383
384         #Starting of loop
385         i=1
386         while i<=count:
387             new_id=max_id+i
388             new_manufacturer_id=random.randint(min_manufacturer_id,
↪ max_manufacturer_id)
389             if fromFile:
390                 new_name=self.getRandomLine(PRODUCT_NAME)
391                 new_description=self.getRandomLine(PRODUCT_DESCRIPTION)
392             else:
393                 new_name=self.getRandomString()
394                 new_description=self.getRandomString()
395                 new_price=random.uniform(1000, 40000)
396
397             #Creating new object and saving it
398             new_product = Product(product_id=new_id, manufacturer_id=
↪ new_manufacturer_id, name=new_name, description=new_description,
↪ price=new_price)
399             new_product.save()
400
401             i+=1
402             print(str(count)+" row(s) successfully added in table product.")
403
404     def addStorage(self, count, fromFile):
405         #Check if this table is empty
406         if Storage.objects.count()==0:
407             max_id=0
408         else:
409             max_id = Storage.objects.order_by('-storage_id')[0].
↪ storage_id
410

```

```

411         #Starting of loop
412         i=1
413         while i<=count:
414             new_id=max_id+i
415             if fromFile:
416                 new_address=self.getRandomLine(ADDRESS)
417                 new_phone=self.getRandomLine(PHONE)
418             else:
419                 new_address=self.getRandomString()
420                 new_phone=self.getRandomString()
421
422             #Creating new object and saving it
423             new_storage = Storage(storage_id=new_id, address=new_address
↪ , phone=new_phone)
424             new_storage.save()
425
426             i+=1
427             print(str(count)+" row(s) successfully added in table storage.")
428
429     def addReview(self, count, fromFile):
430         #Check if there is no data in product or my_user
431         if Product.objects.count()==0 or My_user.objects.count()==0:
432             print('No data in product or my_user table!')
433             return
434
435         #Check if this table is empty
436         if Review.objects.count()==0:
437             max_id=0
438         else:
439             max_id = Review.objects.order_by('-id')[0].id
440
441         #Variables for generation limits
442         min_product_id=Product.objects.order_by('product_id')[0].
↪ product_id
443         max_product_id=Product.objects.order_by('-product_id')[0].
↪ product_id
444
445         min_my_user_id=My_user.objects.order_by('user_id')[0].user_id
446         max_my_user_id=My_user.objects.order_by('-user_id')[0].user_id
447
448         #Starting of loop
449         i=1
450         while i<=count:
451             new_id=max_id+i
452             new_product_id=random.randint(min_product_id, max_product_id
↪ )
453             new_user_id=random.randint(min_my_user_id, max_my_user_id)
454             new_rating=random.randint(1,5)
455             if fromFile:
456                 new_description=self.getRandomLine(REVIEW_DESCRIPTION)
457             else:
458                 new_description=self.getRandomString()
459

```

```

460         #Creating new object and saving it
461         new_review = Review(id=new_id, product_id=new_product_id,
↪ user_id=new_user_id, rating=new_rating, description=
↪ new_description)
462         new_review.save()
463
464         i+=1
465         print(str(count)+" row(s) successfully added in table review.")
466
467     def addType(self, count, fromFile):
468         #Check if this table is empty
469         if Type.objects.count()==0:
470             max_id=0
471         else:
472             max_id = Type.objects.order_by('-type_id')[0].type_id
473
474         #Starting of loop
475         i=1
476         while i<=count:
477             new_id=max_id+i
478             if fromFile:
479                 new_name=self.getRandomLine(TYPE)
480             else:
481                 new_name=self.getRandomString()
482
483             #50 at 50 if new type will have parent, also checking if
↪ parent is possible
484             if random.randint(0,1)==0 or new_id==1:
485                 new_p_id=None
486                 new_level=1
487             else:
488                 #Random parent
489                 min_p_id=Type.objects.order_by('type_id')[0].type_id
490                 max_p_id=Type.objects.order_by('-type_id')[0].type_id
491                 new_p_id=random.randint(min_p_id, max_p_id)
492                 #Selecting parent level and increasing it
493                 new_level=Type.objects.get(pk=new_p_id).level+1
494                 if new_level>MAXIMUM_LEVEL:
495                     continue
496
497             #Creating new object and saving it
498             new_type = Type(type_id=new_id, p_id=new_p_id, name=new_name
↪ , level=new_level)
499             new_type.save()
500
501             i+=1
502             print(str(count)+" row(s) successfully added in table type.")
503
504     def handle(self, *args, **options):
505         #Reading input options
506         table = options['table']
507         count = int(options['count'])
508

```

```

509     #Checking of options
510     if count<=0:
511         print('Wrong count!')
512         return
513     if table=='my_user':
514         self.addUsers(count, options['fromFile'])
515     elif table=='sells':
516         self.addSells(count)
517     elif table=='manufacturer':
518         self.addManufacturers(count, options['fromFile'])
519     elif table=='property':
520         self.addProperty(count, options['fromFile'])
521     elif table=='type':
522         self.addType(count, options['fromFile'])
523     elif table=='type_prop':
524         self.addType_prop(count)
525     elif table=='supply':
526         self.addSupply(count)
527     elif table=='product':
528         self.addProduct(count, options['fromFile'])
529     elif table=='product_types':
530         self.addProduct_types(count)
531     elif table=='review':
532         self.addReview(count, options['fromFile'])
533     elif table=='properties':
534         self.addProperties(count, options['fromFile'])
535     elif table=='sells_entre':
536         self.addSells_entre(count)
537     elif table=='storage':
538         self.addStorage(count, options['fromFile'])
539     elif table=='product_avaliability':
540         self.addProduct_avaliability(count)
541     elif table=='product_part':
542         self.addManufacturers(count*2, options['fromFile'])
543         self.addProduct(count, options['fromFile'])
544         self.addProperty(count*20, options['fromFile'])
545         self.addProperties(count*10, options['fromFile'])
546         self.addType(count*4, options['fromFile'])
547         self.addType_prop(count*10)
548         self.addProduct_types(count*2)
549     elif table=='storage_part':
550         self.addStorage(count, options['fromFile'])
551         self.addProduct_avaliability(count*10)
552         self.addSupply(count*10)
553     elif table=='user_sells_part':
554         self.addUsers(count, options['fromFile'])
555         self.addSells(count*5)
556         self.addReview(count*10, options['fromFile'])
557         self.addSells_entre(count*10)
558     elif table=='all':
559         self.addUsers(count, options['fromFile'])
560         self.addManufacturers(count, options['fromFile'])
561         self.addSells(count)

```



```
562         self.addProperty(count, options['fromFile'])
563         self.addType(count, options['fromFile'])
564         self.addType_prop(count)
565         self.addProduct(count, options['fromFile'])
566         self.addProduct_types(count)
567         self.addReview(count, options['fromFile'])
568         self.addStorage(count, options['fromFile'])
569         self.addProduct_avaliability(count)
570         self.addProperties(count, options['fromFile'])
571         self.addSupply(count)
572         self.addSells_entre(count)
```

Листинг 1.2: generate.py