

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе №2

Дисциплина: Базы данных

Тема: Создание интерактивного генератора данных

Выполнил
студент гр. 43501/3

_____ Зобков Д. А.
(подпись)

Преподаватель

_____ Мяснов А. В.
(подпись)

“__” _____ 2017 г.

Санкт-Петербург
2017 г.

СОДЕРЖАНИЕ

1	Цель работы	3
2	Ход работы	3
3	Вывод	5
	Приложение А Код генератора данных	6

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. **table** — название таблицы или области для которой необходимо сгенерировать данные. В случае ввода **all** произойдет генерация для всех таблиц.
2. **count** — целочисленное число, обозначающее количество строк, которое необходимо сгенерировать.

Данные генерируются случайным образом в виде случайных чисел, времени, дат и строк, состоящих из случайных символов (заглавные английские буквы и цифры).

Также присутствует возможность генерации данных для таблиц из одной области (рис. 2.1).

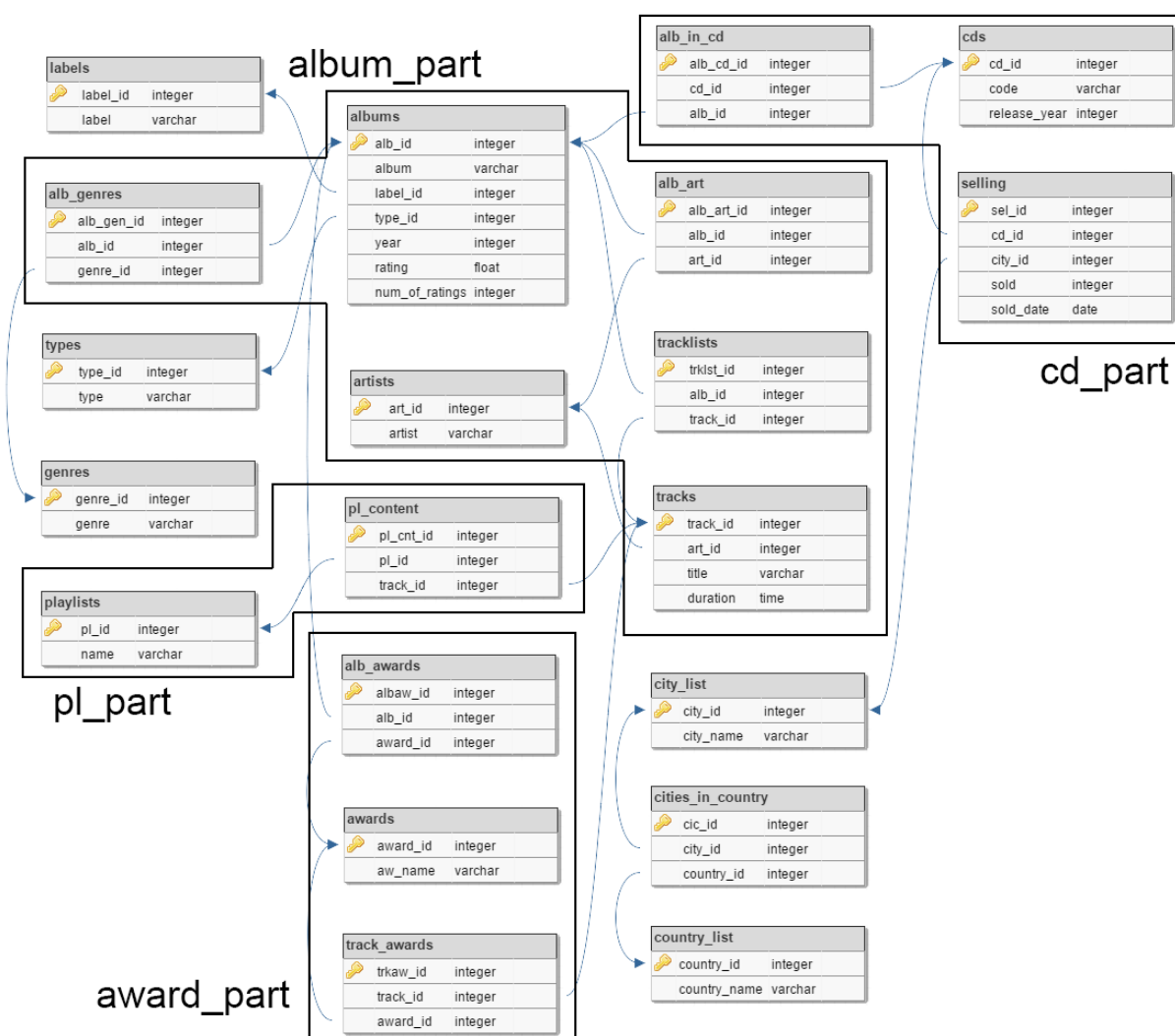


Рис. 2.1. SQL-схема БД

Для каждой из областей также задается параметр count, однако для таблиц, состоящих в области, данный коэффициент модифицируется следующим образом:

1. cd_part
 - cds \rightarrow count
 - alb_in_cd \rightarrow count \times 3
 - selling \rightarrow count \times 2
2. album_part
 - albums \rightarrow count
 - alb_genres \rightarrow count \times (1 || 2)
 - artist \rightarrow count
 - tracks \rightarrow count \times 5
 - alb_art \rightarrow count \times (1 || 2)
 - tracklists \rightarrow count \times 5
3. award_part
 - awards \rightarrow count
 - alb_awards \rightarrow count \times 2
 - track_awards \rightarrow count \times 4
4. pl_part
 - playlists \rightarrow count
 - pl_content \rightarrow count \times 5

Пример использования команды для генерации 5 новых строк в каждую из таблиц базы данных приведен в листинге 2.1.

```
1 D:\4 course\lastsem\db\lab1\lab1>python manage.py generate all 5
2 5 row(s) added to cds.
3 5 row(s) added to labels.
4 5 row(s) added to types.
5 5 row(s) added to genres.
6 5 row(s) added to artists.
7 5 row(s) added to tracks.
8 5 row(s) added to playlists.
9 5 row(s) added to pl_content.
10 5 row(s) added to albums.
11 5 row(s) added to alb_genres.
12 5 row(s) added to alb_in_cd.
13 5 row(s) added to alb_art.
14 5 row(s) added to tracklists.
15 5 row(s) added to awards.
16 5 row(s) added to track_awards.
17 5 row(s) added to alb_awards.
18 5 row(s) added to city_list.
19 5 row(s) added to country_list.
20 5 row(s) added to cities_in_country.
21 5 row(s) added to selling.
```

Листинг 2.1. Пример использования команды

3 Вывод

В ходе данной работы было продолжено создание приложения, работающего с базой данных, путем разработки генератора данных. Собственная реализация в отличие от встроенных в какие-либо СУБД генераторов получается более удобной и гибкой, позволяя дополнять и изменять ее при необходимости.

Генератор случайных строк получается достаточно быстрым по производительности. На время генерации влияют проверки на возможность генерации данных (например, определение диапазона доступных записей по внешнему ключу и проверка на наличие записей в таблице, так как для получения этих данных используются запросы к БД).

Приложение А Код генератора данных

```
1 from django.core.management.base import BaseCommand
2 from django.db import IntegrityError
3 from django.db.models import Max, Min
4 from mus.models import *
5 import random
6 import datetime
7 import string
8 import argparse
9
10 class Command(BaseCommand):
11     def add_arguments(self, parser):
12         parser.add_argument('table', type=str)
13         parser.add_argument('count', type=int)
14
15     def getRandomString(self):
16         return(''.join(random.choice(string.ascii_uppercase +
17 string.digits) for _ in range(random.randint(3, 10))))
18
19     def addCds(self, count):
20         # Check if table is empty
21         if Cds.objects.count() == 0:
22             max_id = 0
23         else:
24             max_id = Cds.objects.order_by('-cd_id')[0].
25 cd_id
26
27         # Starting the loop
28         i = 1
29         error = 0
30         while i <= count:
31             new_id = max_id + i
32             new_code = "{}".format(self.getRandomString())
33             new_year = random.randint(1980, 2017)
34
35             # Creating new object and save it
36             try:
37                 entry = Cds(cd_id = new_id, code =
38 new_code, release_year = new_year)
39                 entry.save()
40             except IntegrityError:
41                 print("Error while adding new row")
42                 error += 1
43
44             i += 1
45
46         print("{} row(s) added to cds.".format(count - error))
47
48     def addLabels(self, count):
49         # Check if table is empty
50         if Labels.objects.count() == 0:
51             max_id = 0
```

```

50         else:
max_id = Labels.objects.order_by('-label_id')
[0].label_id

52         # Starting the loop
i = 1
54         error = 0
while i <= count:
56             new_id = max_id + i
new_label = "{}".format(self.getRandomString())

58             # Creating new object and save it
60             try:
entry = Labels(label_id = new_id, label
= new_label)

62                 entry.save()
except IntegrityError:
64                     print("Error while adding new row")
error += 1

66
68                 i += 1

print("{} row(s) added to labels.".format(count -
error))

70
def addTypes(self, count):
72     # Check if table is empty
if Types.objects.count() == 0:
74         max_id = 0
else:
76         max_id = Types.objects.order_by('-type_id')[0].
type_id

78         # Starting the loop
i = 1
80         error = 0
while i <= count:
82             new_id = max_id + i
new_type = "{}".format(self.getRandomString())

84             # Creating new object and save it
86             try:
entry = Types(type_id = new_id, type =
new_type)

88                 entry.save()
except IntegrityError:
90                     print("Error while adding new row")
error += 1

92
94                 i += 1

print("{} row(s) added to types.".format(count - error
))
96

```

```

    def addGenres(self, count):
        # Check if table is empty
        if Genres.objects.count() == 0:
            max_id = 0
        else:
            max_id = Genres.objects.order_by('-genre_id')
[0].genre_id

        # Starting the loop
        i = 1
        error = 0
        while i <= count:
            new_id = max_id + i
            new_genre = "{}".format(self.getRandomString())

            # Creating new object and save it
            try:
                entry = Genres(genre_id = new_id, genre
= new_genre)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

        print("{} row(s) added to genres.".format(count -
error))

    def addArtists(self, count):
        # Check if table is empty
        if Artists.objects.count() == 0:
            max_id = 0
        else:
            max_id = Artists.objects.order_by('-art_id')
[0].art_id

        # Starting the loop
        i = 1
        error = 0
        while i <= count:
            new_id = max_id + i
            new_artist = "{} {}".format(self.
getRandomString(), self.getRandomString())

            # Creating new object and save it
            try:
                entry = Artists(art_id = new_id, artist
= new_artist)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

```



```

146         i += 1
147
148         print("{} row(s) added to artists.".format(count -
error))
149
150     def addTracks(self, count):
151         #Check if there is no entries
152         if Artists.objects.count() == 0:
153             print('No artists!')
154             return
155
156         # Check if table is empty
157         if Tracks.objects.count() == 0:
158             max_id = 0
159
160         else:
161             max_id = Tracks.objects.order_by('-track_id')
[0].track_id
162
163         # Variables for generation limits
164         min_art_id = Artists.objects.order_by('art_id')[0].
art_id
165         max_art_id = Artists.objects.order_by('-art_id')[0].
art_id
166
167         # Starting the loop
168         i = 1
169         error = 0
170         while i <= count:
171             new_id = max_id + i
172             new_art_id = random.randint(min_art_id,
max_art_id)
173             new_title = "{}".format(self.getRandomString())
174             new_duration = datetime.timedelta(seconds =
random.randint(1, 3600))
175
176             # Creating new object and save it
177             try:
178                 entry = Tracks(track_id = new_id,
art_id = new_art_id, title = new_title, duration = new_duration)
179                 entry.save()
180             except IntegrityError:
181                 print("Error while adding new row")
182                 error += 1
183
184             i += 1
185
186         print("{} row(s) added to tracks.".format(count -
error))
187
188     def addPlaylists(self, count):
189         # Check if table is empty
190         if Playlists.objects.count() == 0:
191             max_id = 0
192
193         else:

```

```

max_id = Playlists.objects.order_by('-pl_id')
[0].pl_id

192
    # Starting the loop
194
    i = 1
    error = 0
196
    while i <= count:
        new_id = max_id + i
198
        new_name = "{}".format(self.getRandomString())

        # Creating new object and save it
        try:
200
            entry = Playlists(pl_id = new_id, name
202
            = new_name)

            entry.save()
204
        except IntegrityError:
            print("Error while adding new row")
206
            error += 1

            i += 1

208

        print("{} row(s) added to playlists.".format(count -
210
        error))

212
    def addPlContent(self, count):
        #Check if there is no entries
214
        if Playlists.objects.count() == 0:
            print('No playlists!')
            return
216
        if Tracks.objects.count() == 0:
            print('No tracks!')
            return
218

220
        # Check if table is empty
222
        if PlContent.objects.count() == 0:
            max_id = 0
224
        else:
            max_id = PlContent.objects.order_by('-pl_cnt_id
226
            ')[0].pl_cnt_id

            # Variables for generation limits
228
            min_pl_id = Playlists.objects.order_by('pl_id')[0].
            pl_id
            max_pl_id = Playlists.objects.order_by('-pl_id')[0].
            pl_id
230
            min_track_id = Tracks.objects.order_by('track_id')[0].
            track_id
            max_track_id = Tracks.objects.order_by('-track_id')[0].
            track_id

232
            # Starting the loop
234
            i = 1
            error = 0
236
            while i <= count:

```

```

238         new_id = max_id + i
        new_pl_id = random.randint(min_pl_id, max_pl_id)
    )
        new_track_id = random.randint(min_track_id,
max_track_id)

240         # Creating new object and save it
242         try:
                entry = PlContent(pl_cnt_id = new_id,
pl_id = new_pl_id, track_id = new_track_id)
244                 entry.save()
        except IntegrityError:
246                 print("Error while adding new row")
                error += 1

248         i += 1

250         print("{} row(s) added to pl_content.".format(count -
error))

252     def addAlbums(self, count):
254         #Check if there is no entries
        if Labels.objects.count() == 0:
256             print('No labels!')
            return
258         if Types.objects.count() == 0:
            print('No types!')
260             return

        # Check if table is empty
262         if Albums.objects.count() == 0:
264             max_id = 0
        else:
266             max_id = Albums.objects.order_by('-alb_id')[0].
alb_id

268         # Variables for generation limits
        min_label_id = Labels.objects.order_by('label_id')[0].
label_id
270         max_label_id = Labels.objects.order_by('-label_id')[0].
label_id
        min_type_id = Types.objects.order_by('type_id')[0].
type_id
272         max_type_id = Types.objects.order_by('-type_id')[0].
type_id

274         # Starting the loop
        i = 1
276         error = 0
        while i <= count:
278             new_id = max_id + i
            new_album = "{}".format(self.getRandomString())
            new_label_id = random.randint(min_label_id,
280             max_label_id)

```

```

new_type_id = random.randint(min_type_id,
max_type_id)
new_year = random.randint(1980, 2017)

# Creating new object and save it
try:
    entry = Albums(alb_id = new_id, album =
new_album, label_id = new_label_id, type_id = new_type_id, year =
new_year, rating = 0, num_of_ratings = 0)
    entry.save()
except IntegrityError:
    print("Error while adding new row")
    error += 1

i += 1

print("{} row(s) added to albums.".format(count -
error))

def addAlbGenres(self, count):
    #Check if there is no entries
    if Albums.objects.count() == 0:
        print('No albums!')
        return
    if Genres.objects.count() == 0:
        print('No genres!')
        return

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

    # Variables for generation limits
    min_alb_id = Albums.objects.order_by('alb_id')[0].
alb_id
    max_alb_id = Albums.objects.order_by('-alb_id')[0].
alb_id
    min_genre_id = Genres.objects.order_by('genre_id')[0].
genre_id
    max_genre_id = Genres.objects.order_by('-genre_id')[0].
genre_id

    # Starting the loop
    i = 1
    error = 0
    while i <= count:
        new_id = max_id + i
        new_alb_id = random.randint(min_alb_id,
max_alb_id)
        new_genre_id = random.randint(min_genre_id,
max_genre_id)

```

```

324         # Creating new object and save it
326         try:
328             entry = AlbGenres(alb_gen_id = new_id,
alb_id = new_alb_id, genre_id = new_genre_id)
330             entry.save()
332         except IntegrityError:
334             print("Error while adding new row")
336             error += 1
338
340             i += 1
342
344         print("{} row(s) added to alb_genres.".format(count -
error))
346
348     def addAlbInCd(self, count):
350         #Check if there is no entries
352         if Albums.objects.count() == 0:
354             print('No albums!')
356             return
358         if Cds.objects.count() == 0:
360             print('No cds!')
362             return
364
366         # Check if table is empty
368         if AlbInCd.objects.count() == 0:
370             max_id = 0
372         else:
374             max_id = AlbInCd.objects.order_by('-alb_cd_id')
[0].alb_cd_id
376
378         # Variables for generation limits
380         min_alb_id = Albums.objects.order_by('alb_id')[0].
alb_id
382         max_alb_id = Albums.objects.order_by('-alb_id')[0].
alb_id
384         min_cd_id = Cds.objects.order_by('cd_id')[0].cd_id
386         max_cd_id = Cds.objects.order_by('-cd_id')[0].cd_id
388
390         # Starting the loop
392         i = 1
394         error = 0
396         while i <= count:
398             new_id = max_id + i
399             new_alb_id = random.randint(min_alb_id,
max_alb_id)
400             new_cd_id = random.randint(min_cd_id, max_cd_id
)
402
404         # Creating new object and save it
406         try:
408             entry = AlbInCd(alb_cd_id = new_id,
alb_id = new_alb_id, cd_id = new_cd_id)
410             entry.save()

```

```

370         except IntegrityError:
371             print("Error while adding new row")
372             error += 1
373
374             i += 1
375
376         print("{} row(s) added to alb_in_cd.".format(count -
error))
377
378     def addAlbArt(self, count):
379         #Check if there is no entries
380         if Albums.objects.count() == 0:
381             print('No albums!')
382             return
383         if Artists.objects.count() == 0:
384             print('No artists!')
385             return
386
387         # Check if table is empty
388         if AlbArt.objects.count() == 0:
389             max_id = 0
390         else:
391             max_id = AlbArt.objects.order_by('-alb_art_id')
[0].alb_art_id
392
393         # Variables for generation limits
394         min_alb_id = Albums.objects.order_by('alb_id')[0].
alb_id
395         max_alb_id = Albums.objects.order_by('-alb_id')[0].
alb_id
396         min_art_id = Artists.objects.order_by('art_id')[0].
art_id
397         max_art_id = Artists.objects.order_by('-art_id')[0].
art_id
398
399         # Starting the loop
400         i = 1
401         error = 0
402         while i <= count:
403             new_id = max_id + i
404             new_alb_id = random.randint(min_alb_id,
max_alb_id)
405             new_art_id = random.randint(min_art_id,
max_art_id)
406
407             # Creating new object and save it
408             try:
409                 entry = AlbArt(alb_art_id = new_id,
alb_id = new_alb_id, art_id = new_art_id)
410                 entry.save()
411             except IntegrityError:
412                 print("Error while adding new row")
413                 error += 1
414

```

```

416         i += 1

        print("{} row(s) added to alb_art.".format(count -
error))

418     def addTracklists(self, count):
420         #Check if there is no entries
        if Albums.objects.count() == 0:
422             print('No albums!')
            return
424         if Tracks.objects.count() == 0:
            print('No tracks!')
426             return

428         # Check if table is empty
        if Tracklists.objects.count() == 0:
430             max_id = 0
        else:
432             max_id = Tracklists.objects.order_by('-
trklst_id')[0].trklst_id

434         # Variables for generation limits
        min_alb_id = Albums.objects.order_by('alb_id')[0].
alb_id
436         max_alb_id = Albums.objects.order_by('-alb_id')[0].
alb_id
        min_track_id = Tracks.objects.order_by('track_id')[0].
track_id
438         max_track_id = Tracks.objects.order_by('-track_id')[0].
track_id

440         # Starting the loop
        i = 1
442         error = 0
        while i <= count:
444             new_id = max_id + i
            new_alb_id = random.randint(min_alb_id,
max_alb_id)
446             new_track_id = random.randint(min_track_id,
max_track_id)

448             # Creating new object and save it
            try:
450                 entry = Tracklists(trklst_id = new_id,
alb_id = new_alb_id, track_id = new_track_id)
                entry.save()
452             except IntegrityError:
                print("Error while adding new row")
                error += 1
454

456             i += 1

458         print("{} row(s) added to tracklists.".format(count -
error))

```

```

460     def addAwards(self, count):
461         # Check if table is empty
462         if Awards.objects.count() == 0:
463             max_id = 0
464         else:
465             max_id = Awards.objects.order_by('-award_id')
466 [0].award_id
467
468         # Starting the loop
469         i = 1
470         error = 0
471         while i <= count:
472             new_id = max_id + i
473             new_award = "{}".format(self.getRandomString())
474
475             # Creating new object and save it
476             try:
477                 entry = Awards(award_id = new_id,
478 aw_name = new_award)
479                 entry.save()
480             except IntegrityError:
481                 print("Error while adding new row")
482                 error += 1
483
484                 i += 1
485
486             print("{} row(s) added to awards.".format(count -
487 error))
488
489     def addTrackAwards(self, count):
490         #Check if there is no entries
491         if Awards.objects.count() == 0:
492             print('No awards!')
493             return
494         if Tracks.objects.count() == 0:
495             print('No tracks!')
496             return
497
498         # Check if table is empty
499         if TrackAwards.objects.count() == 0:
500             max_id = 0
501         else:
502             max_id = TrackAwards.objects.order_by('-
503 trkaw_id')[0].trkaw_id
504
505         # Variables for generation limits
506         min_award_id = Awards.objects.order_by('award_id')[0].
507 award_id
508         max_award_id = Awards.objects.order_by('-award_id')[0].
509 award_id
510         min_track_id = Tracks.objects.order_by('track_id')[0].
511 track_id

```



```

max_track_id = Tracks.objects.order_by('-track_id')[0].
track_id

506
    # Starting the loop
508    i = 1
    error = 0
510    while i <= count:
        new_id = max_id + i
512        new_award_id = random.randint(min_award_id,
max_award_id)
        new_track_id = random.randint(min_track_id,
max_track_id)

514        # Creating new object and save it
516        try:
            entry = TrackAwards(trkaw_id = new_id,
award_id = new_award_id, track_id = new_track_id)
518            entry.save()
        except IntegrityError:
520            print("Error while adding new row")
            error += 1

522            i += 1

524        print("{} row(s) added to track_awards.".format(count
- error))

526    def addAlbAwards(self, count):
528        #Check if there is no entries
        if Awards.objects.count() == 0:
530            print('No awards!')
            return
532        if Albums.objects.count() == 0:
            print('No albums!')
534            return

536        # Check if table is empty
        if AlbAwards.objects.count() == 0:
538            max_id = 0
        else:
540            max_id = AlbAwards.objects.order_by('-albaw_id'
) [0].albaw_id

542        # Variables for generation limits
        min_award_id = Awards.objects.order_by('award_id')[0].
award_id
544        max_award_id = Awards.objects.order_by('-award_id')[0].
award_id
        min_alb_id = Albums.objects.order_by('alb_id')[0].
alb_id
546        max_alb_id = Albums.objects.order_by('-alb_id')[0].
alb_id

548        # Starting the loop

```

```

550         i = 1
551         error = 0
552         while i <= count:
553             new_id = max_id + i
554             new_award_id = random.randint(min_award_id,
555             max_award_id)
556             new_alb_id = random.randint(min_alb_id,
557             max_alb_id)
558
559             # Creating new object and save it
560             try:
561                 entry = AlbAwards(albaw_id = new_id,
562                 award_id = new_award_id, alb_id = new_alb_id)
563                 entry.save()
564             except IntegrityError:
565                 print("Error while adding new row")
566                 error += 1
567
568             i += 1
569
570         print("{} row(s) added to alb_awards.".format(count -
571         error))
572
573     def addCityList(self, count):
574         # Check if table is empty
575         if CityList.objects.count() == 0:
576             max_id = 0
577         else:
578             max_id = CityList.objects.order_by('-city_id')
579             [0].city_id
580
581         # Starting the loop
582         i = 1
583         error = 0
584         while i <= count:
585             new_id = max_id + i
586             new_name = "{}".format(self.getRandomString())
587
588             # Creating new object and save it
589             try:
590                 entry = CityList(city_id = new_id,
591                 city_name = new_name)
592                 entry.save()
593             except IntegrityError:
594                 print("Error while adding new row")
595                 error += 1
596
597             i += 1
598
599         print("{} row(s) added to city_list.".format(count -
600         error))
601
602     def addCountryList(self, count):
603         # Check if table is empty

```

```

596         if CountryList.objects.count() == 0:
597             max_id = 0
598         else:
599             max_id = CountryList.objects.order_by('-
country_id')[0].country_id
600
601         # Starting the loop
602         i = 1
603         error = 0
604         while i <= count:
605             new_id = max_id + i
606             new_name = "{}".format(self.getRandomString())
607
608             # Creating new object and save it
609             try:
610                 entry = CountryList(country_id = new_id
, country_name = new_name)
611                 entry.save()
612             except IntegrityError:
613                 print("Error while adding new row")
614                 error += 1
615
616                 i += 1
617
618         print("{} row(s) added to country_list.".format(count
- error))
619
620     def addCitiesInCountry(self, count):
621         #Check if there is no entries
622         if CityList.objects.count() == 0:
623             print('No cities!')
624             return
625         if CountryList.objects.count() == 0:
626             print('No countries!')
627             return
628
629         # Check if table is empty
630         if CitiesInCountry.objects.count() == 0:
631             max_id = 0
632         else:
633             max_id = CitiesInCountry.objects.order_by('-
cic_id')[0].cic_id
634
635         # Variables for generation limits
636         min_city_id = CityList.objects.order_by('city_id')[0].
city_id
637         max_city_id = CityList.objects.order_by('-city_id')[0].
city_id
638         min_country_id = CountryList.objects.order_by('
country_id')[0].country_id
639         max_country_id = CountryList.objects.order_by('-
country_id')[0].country_id
640
641         # Starting the loop

```

```

642         i = 1
        error = 0
644         while i <= count:
            new_id = max_id + i
646             new_city_id = random.randint(min_city_id,
max_city_id)
            new_country_id = random.randint(min_country_id,
max_country_id)
648             # Creating new object and save it
650             try:
                entry = CitiesInCountry(cic_id = new_id
, city_id = new_city_id, country_id = new_country_id)
652                 entry.save()
            except IntegrityError:
654                 print("Error while adding new row")
                error += 1
656
658                 i += 1

        print("{} row(s) added to cities_in_country.".format(
count - error))
660
        def addSelling(self, count):
662             #Check if there is no entries
            if CityList.objects.count() == 0:
664                 print('No cities!')
                return
666             if Cds.objects.count() == 0:
                print('No cds!')
668                 return

670             # Check if table is empty
            if Selling.objects.count() == 0:
672                 max_id = 0
            else:
674                 max_id = Selling.objects.order_by('-sel_id')
[0].sel_id

676             # Variables for generation limits
            min_city_id = CityList.objects.order_by('city_id')[0].
city_id
678             max_city_id = CityList.objects.order_by('-city_id')[0].
city_id
            min_cd_id = Cds.objects.order_by('cd_id')[0].cd_id
680             max_cd_id = Cds.objects.order_by('-cd_id')[0].cd_id

682             # Starting the loop
            i = 1
684             error = 0
            while i <= count:
686                 new_id = max_id + i
                new_city_id = random.randint(min_city_id,
max_city_id)

```

```

688         new_cd_id = random.randint(min_cd_id, max_cd_id
)
        new_sold = random.randint(1, 100000000)
690         new_date = datetime.date(random.randint
(2000,2016), random.randint(1,12), random.randint(1,28))

        # Creating new object and save it
        try:
694             entry = Selling(sel_id = new_id,
city_id = new_city_id, cd_id = new_cd_id, sold = new_sold, sold_date
= new_date)
            entry.save()
696         except IntegrityError:
            print("Error while adding new row")
698             error += 1

700         i += 1

702         print("{} row(s) added to selling.".format(count -
error))

704

706     def handle(self, *args, **options):
        # Reading input options
708         table = options['table']
        count = int(options['count'])

710         # Checking of options
712         if count <= 0:
            print('Wrong count!')
714             return
        if table == 'cds':
716             self.addCds(count)
        elif table == 'labels':
718             self.addLabels(count)
        elif table == 'types':
720             self.addTypes(count)
        elif table == 'genres':
722             self.addGenres(count)
        elif table == 'artists':
724             self.addArtists(count)
        elif table == 'tracks':
726             self.addTracks(count)
        elif table == 'playlists':
728             self.addPlaylists(count)
        elif table == 'pl_content':
730             self.addPlContent(count)
        elif table == 'albums':
732             self.addAlbums(count)
        elif table == 'alb_genres':
734             self.addAlbGenres(count)
        elif table == 'alb_in_cd':
736             self.addAlbInCd(count)

```

```

738         elif table == 'alb_art':
739             self.addAlbArt(count)
740         elif table == 'tracklists':
741             self.addTracklists(count)
742         elif table == 'awards':
743             self.addAwards(count)
744         elif table == 'track_awards':
745             self.addTrackAwards(count)
746         elif table == 'alb_awards':
747             self.addAlbAwards(count)
748         elif table == 'city_list':
749             self.addCityList(count)
750         elif table == 'country_list':
751             self.addCountryList(count)
752         elif table == 'cities_in_country':
753             self.addCitiesInCountry(count)
754         elif table == 'selling':
755             self.addSelling(count)
756         elif table == 'cd_part':
757             self.addCds(count)
758             self.addAlbInCd(count * 3)
759             self.addSelling(count * 2)
760         elif table == 'album_part':
761             self.addAlbums(count)
762             self.addAlbGenres(random.randint(1, 2))
763             self.addArtists(count)
764             self.addTracks(count * 5)
765             self.addAlbArt(random.randint(1, 2))
766             self.addTracklists(count * 5)
767         elif table == 'award_part':
768             self.addAwards(count)
769             self.addAlbAwards(count * 2)
770             self.addTrackAwards(count * 4)
771         elif table == 'pl_part':
772             self.addPlaylists(count)
773             self.addPlContent(count * 5)
774         elif table == 'all':
775             self.addCds(count)
776             self.addLabels(count)
777             self.addTypes(count)
778             self.addGenres(count)
779             self.addArtists(count)
780             self.addTracks(count)
781             self.addPlaylists(count)
782             self.addPlContent(count)
783             self.addAlbums(count)
784             self.addAlbGenres(count)
785             self.addAlbInCd(count)
786             self.addAlbArt(count)
787             self.addTracklists(count)
788             self.addAwards(count)
789             self.addTrackAwards(count)
790             self.addAlbAwards(count)
791             self.addCityList(count)

```

```
self.addCountryList(count)
self.addCitiesInCountry(count)
self.addSelling(count)
```

Листинг А.1. generate.py