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

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).

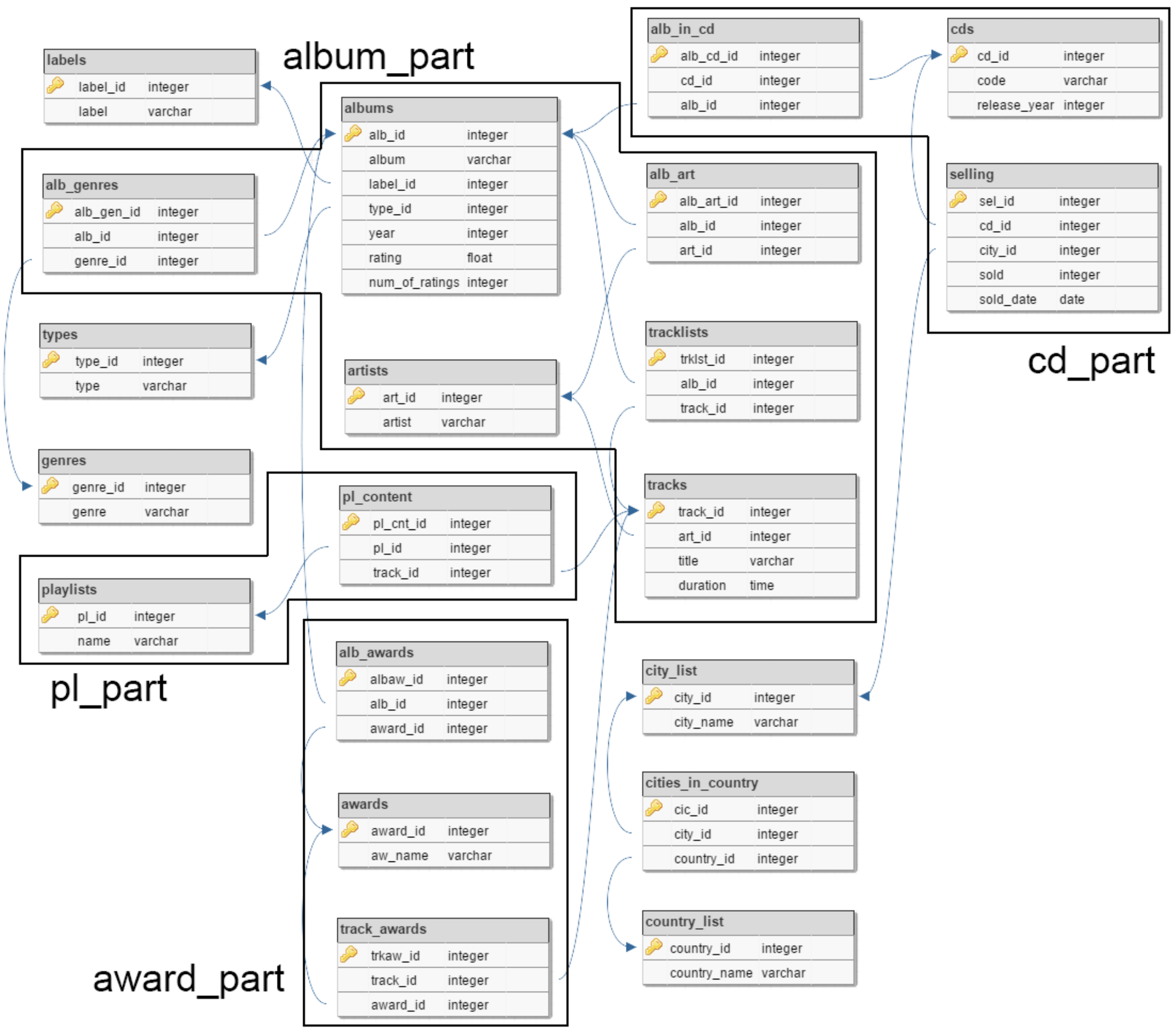

Рис. 2.1. SQL-схема БД

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

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

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

```
D:\4 course\lastsem\db\lab1\lab1>python manage.py generate all 5
5 row(s) addded to cds.
5 row(s) addded to labels.
5 row(s) addded to types.
5 row(s) addded to genres.
5 row(s) addded to artists.
5 row(s) addded to tracks.
5 row(s) addded to playlists.
5 row(s) addded to pl_content.
5 row(s) addded to albums.
5 row(s) addded to alb_genres.
5 row(s) addded to alb_in_cd.
5 row(s) addded to alb_art.
5 row(s) addded to tracklists.
5 row(s) addded to awards.
5 row(s) addded to track_awards.
5 row(s) addded to alb_awards.
5 row(s) addded to city_list.
5 row(s) addded to country_list.
5 row(s) addded to cities_in_country.
5 row(s) addded 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

```python
from django.core.management.base import BaseCommand
from django.db import IntegrityError
from django.db.models import Max, Min
from mus.models import *
import random
import datetime
import string
import argparse

class Command(BaseCommand):
    def add_arguments(self, parser):
        parser.add_argument(’table’, type=str)
        parser.add_argument(’count’, type=int)

    def getRandomString(self):
        return(’’.join(random.choice(string.ascii_uppercase +
  string.digits) for _ in range(random.randint(3, 10))))

    def addCds(self, count):
        # Check if table is empty
        if Cds.objects.count() == 0:
            max_id = 0
        else:
            max_id = Cds.objects.order_by(’-cd_id’)[0].
  cd_id

        # Starting the loop
        i = 1
        error = 0
        while i <= count:
            new_id = max_id + i
            new_code = "{}".format(self.getRandomString())
            new_year = random.randint(1980, 2017)

            # Creating new object and save it
            try:
                entry = Cds(cd_id = new_id, code =
  new_code, release_year = new_year)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

        print("{} row(s) addded to cds.".format(count - error))

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

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

            # Creating new object and save it
            try:
                entry = Labels(label_id = new_id, label = new_label)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

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

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

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

            # Creating new object and save it
            try:
                entry = Types(type_id = new_id, type = new_type)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

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

            i += 1

        print("{} row(s) addded to artists.".format(count - error))

    def addTracks(self, count):
        #Check if there is no entries
        if Artists.objects.count() == 0:
            print(’No artists!’)
            return

        # Check if table is empty
        if Tracks.objects.count() == 0:
            max_id = 0
        else:
            max_id = Tracks.objects.order_by(’-track_id’)[0].track_id

        # Variables for generation limits
        min_art_id = Artists.objects.order_by(’art_id’)[0].art_id
        max_art_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_art_id = random.randint(min_art_id, max_art_id)
            new_title = "{}".format(self.getRandomString())
            new_duration = datetime.timedelta(seconds = random.randint(1, 3600))

            # Creating new object and save it
            try:
                entry = Tracks(track_id = new_id, art_id = new_art_id, title = new_title, duration = new_duration)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

        print("{} row(s) addded to tracks.".format(count - error))

    def addPlaylists(self, count):
        # Check if table is empty
        if Playlists.objects.count() == 0:
            max_id = 0
        else:
```

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

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

            # Creating new object and save it
            try:
                entry = Playlists(pl_id = new_id, name = new_name)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

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

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

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

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

        # Starting the loop
        i = 1
        error = 0
        while i <= count:
            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)

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

            i += 1

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

    def addAlbums(self, count):
        #Check if there is no entries
        if Labels.objects.count() == 0:
            print(’No labels!’)
            return
        if Types.objects.count() == 0:
            print(’No types!’)
            return

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

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

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

```

                # Creating new object and save it
                try:
                    entry = AlbGenres(alb_gen_id = new_id, alb_id = new_alb_id, genre_id = new_genre_id)
                    entry.save()
                except IntegrityError:
                    print("Error while adding new row")
                    error += 1

                i += 1

            print("{} row(s) addded to alb_genres.".format(count - error))

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

            # Check if table is empty
            if AlbInCd.objects.count() == 0:
                max_id = 0
            else:
                max_id = AlbInCd.objects.order_by('-alb_cd_id')[0].alb_cd_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_cd_id = Cds.objects.order_by('cd_id')[0].cd_id
            max_cd_id = Cds.objects.order_by('-cd_id')[0].cd_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_cd_id = random.randint(min_cd_id, max_cd_id)

                # Creating new object and save it
                try:
                    entry = AlbInCd(alb_cd_id = new_id, alb_id = new_alb_id, cd_id = new_cd_id)
                    entry.save()
                except IntegrityError:
                    print("Error while adding new row")
                    error += 1

                i += 1

            print("{} row(s) addded to alb_in_cd.".format(count -
error))

    def addAlbArt(self, count):
            #Check if there is no entries
            if Albums.objects.count() == 0:
                print(’No albums!’)
                return
            if Artists.objects.count() == 0:
                print(’No artists!’)
                return

            # Check if table is empty
            if AlbArt.objects.count() == 0:
                max_id = 0
            else:
                max_id = AlbArt.objects.order_by(’-alb_art_id’)
[0].alb_art_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_art_id = Artists.objects.order_by(’art_id’)[0].
art_id
            max_art_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_alb_id = random.randint(min_alb_id,
max_alb_id)
                new_art_id = random.randint(min_art_id,
max_art_id)

                # Creating new object and save it
                try:
                    entry = AlbArt(alb_art_id = new_id,
alb_id = new_alb_id, art_id = new_art_id)
                    entry.save()
                except IntegrityError:
                    print("Error while adding new row")
                    error += 1

```

```
                     i += 1

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

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

             # Check if table is empty
             if Tracklists.objects.count() == 0:
                     max_id = 0
             else:
                     max_id = Tracklists.objects.order_by('-
trklst_id')[0].trklst_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_track_id = Tracks.objects.order_by('track_id')[0].
track_id
             max_track_id = Tracks.objects.order_by('-track_id')[0].
track_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_track_id = random.randint(min_track_id,
max_track_id)

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

                     i += 1

             print("{} row(s) addded to tracklists.".format(count -
error))
```

```
    def addAwards(self, count):
        # Check if table is empty
        if Awards.objects.count() == 0:
            max_id = 0
        else:
            max_id = Awards.objects.order_by(’-award_id’)
[0].award_id

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

            # Creating new object and save it
            try:
                entry = Awards(award_id = new_id,
aw_name = new_award)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

        print("{} row(s) addded to awards.".format(count -
error))

    def addTrackAwards(self, count):
        #Check if there is no entries
        if Awards.objects.count() == 0:
            print(’No awards!’)
            return
        if Tracks.objects.count() == 0:
            print(’No tracks!’)
            return

        # Check if table is empty
        if TrackAwards.objects.count() == 0:
            max_id = 0
        else:
            max_id = TrackAwards.objects.order_by(’-
trkaw_id’)[0].trkaw_id

        # Variables for generation limits
        min_award_id = Awards.objects.order_by(’award_id’)[0].
award_id
        max_award_id = Awards.objects.order_by(’-award_id’)[0].
award_id
        min_track_id = Tracks.objects.order_by(’track_id’)[0].
track_id
```

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

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

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

                    i += 1

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

    def addAlbAwards(self, count):
            #Check if there is no entries
            if Awards.objects.count() == 0:
                    print(’No awards!’)
                    return
            if Albums.objects.count() == 0:
                    print(’No albums!’)
                    return

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

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

            # Starting the loop
```

```
        i = 1
        error = 0
        while i <= count:
            new_id = max_id + i
            new_award_id = random.randint(min_award_id, max_award_id)
            new_alb_id = random.randint(min_alb_id, max_alb_id)

            # Creating new object and save it
            try:
                entry = AlbAwards(albaw_id = new_id, award_id = new_award_id, alb_id = new_alb_id)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

        print("{} row(s) addded to alb_awards.".format(count - error))

    def addCityList(self, count):
        # Check if table is empty
        if CityList.objects.count() == 0:
            max_id = 0
        else:
            max_id = CityList.objects.order_by(’-city_id’)[0].city_id

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

            # Creating new object and save it
            try:
                entry = CityList(city_id = new_id, city_name = new_name)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

        print("{} row(s) addded to city_list.".format(count - error))

    def addCountryList(self, count):
        # Check if table is empty
```

```
		if CountryList.objects.count() == 0:
			max_id = 0
		else:
			max_id = CountryList.objects.order_by('-
country_id')[0].country_id

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

			# Creating new object and save it
			try:
				entry = CountryList(country_id = new_id
, country_name = new_name)
				entry.save()
			except IntegrityError:
				print("Error while adding new row")
				error += 1

			i += 1

		print("{} row(s) addded to country_list.".format(count
- error))

	def addCitiesInCountry(self, count):
		#Check if there is no entries
		if CityList.objects.count() == 0:
			print('No cities!')
			return
		if CountryList.objects.count() == 0:
			print('No countries!')
			return

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

		# Variables for generation limits
		min_city_id = CityList.objects.order_by('city_id')[0].
city_id
		max_city_id = CityList.objects.order_by('-city_id')[0].
city_id
		min_country_id = CountryList.objects.order_by('
country_id')[0].country_id
		max_country_id = CountryList.objects.order_by('-
country_id')[0].country_id

		# Starting the loop
```

```
        i = 1
        error = 0
        while i <= count:
            new_id = max_id + i
            new_city_id = random.randint(min_city_id,
max_city_id)
            new_country_id = random.randint(min_country_id,
 max_country_id)

            # Creating new object and save it
            try:
                entry = CitiesInCountry(cic_id = new_id
, city_id = new_city_id, country_id = new_country_id)
                entry.save()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

        print("{} row(s) addded to cities_in_country.".format(
count - error))

    def addSelling(self, count):
        #Check if there is no entries
        if CityList.objects.count() == 0:
            print('No cities!')
            return
        if Cds.objects.count() == 0:
            print('No cds!')
            return

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

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

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

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

            # Creating new object and save it
            try:
                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()
            except IntegrityError:
                print("Error while adding new row")
                error += 1

            i += 1

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


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

        # Checking of options
        if count <= 0:
            print('Wrong count!')
            return
        if table == 'cds':
            self.addCds(count)
        elif table == 'labels':
            self.addLabels(count)
        elif table == 'types':
            self.addTypes(count)
        elif table == 'genres':
            self.addGenres(count)
        elif table == 'artists':
            self.addArtists(count)
        elif table == 'tracks':
            self.addTracks(count)
        elif table == 'playlists':
            self.addPlaylists(count)
        elif table == 'pl_content':
            self.addPlContent(count)
        elif table == 'albums':
            self.addAlbums(count)
        elif table == 'alb_genres':
            self.addAlbGenres(count)
        elif table == 'alb_in_cd':
            self.addAlbInCd(count)
```

```python
                elif table == 'alb_art':
                        self.addAlbArt(count)
                elif table == 'tracklists':
                        self.addTracklists(count)
                elif table == 'awards':
                        self.addAwards(count)
                elif table == 'track_awards':
                        self.addTrackAwards(count)
                elif table == 'alb_awards':
                        self.addAlbAwards(count)
                elif table == 'city_list':
                        self.addCityList(count)
                elif table == 'country_list':
                        self.addCountryList(count)
                elif table == 'cities_in_country':
                        self.addCitiesInCountry(count)
                elif table == 'selling':
                        self.addSelling(count)
                elif table == 'cd_part':
                        self.addCds(count)
                        self.addAlbInCd(count * 3)
                        self.addSelling(count * 2)
                elif table == 'album_part':
                        self.addAlbums(count)
                        self.addAlbGenres(random.randint(1, 2))
                        self.addArtists(count)
                        self.addTracks(count * 5)
                        self.addAlbArt(random.randint(1, 2))
                        self.addTracklists(count * 5)
                elif table == 'award_part':
                        self.addAwards(count)
                        self.addAlbAwards(count * 2)
                        self.addTrackAwards(count * 4)
                elif table == 'pl_part':
                        self.addPlaylists(count)
                        self.addPlContent(count * 5)
                elif table == 'all':
                        self.addCds(count)
                        self.addLabels(count)
                        self.addTypes(count)
                        self.addGenres(count)
                        self.addArtists(count)
                        self.addTracks(count)
                        self.addPlaylists(count)
                        self.addPlContent(count)
                        self.addAlbums(count)
                        self.addAlbGenres(count)
                        self.addAlbInCd(count)
                        self.addAlbArt(count)
                        self.addTracklists(count)
                        self.addAwards(count)
                        self.addTrackAwards(count)
                        self.addAlbAwards(count)
                        self.addCityList(count)
```

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

Листинг А.1. generate.py