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

Отчет по лабораторной работе №1
Дисциплина: Базы данных
Тема: Знакомство с ORM на примере Django

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 через механизм объектно-реляционного отображения.

2 Программа работы

1. Знакомство с фреймворком Django:
   - установка;
   - создание проекта;
   - создание приложения;
2. Формирование набора моделей, соответствующих схеме БД, полученной по результатам разработки схемы БД и модификации схемы;
3. Знакомство с механизмом миграций: автоматическое формирование схемы БД с помощью миграций;
4. Создание manage-команд для заполнения БД тестовыми данными (по несколько записей в каждой таблице);
5. Написание отчета.

3 Ход работы

3.1 Подготовка

Предварительно был установлен следующий комплекс программ:

- Python 3.6;
- PostgreSQL 9.6.2;
- Psycopg 2.6.2;
- Django 1.10.5.

Для PostgreSQL была создана база данных **mus**, а также пользователь **dan0n**. Далее следующими командами был создан проект:

```
D:\4 course\lastsem\db\lab1>django-admin.py startproject lab1
D:\4 course\lastsem\db\lab1>cd lab1
D:\4 course\lastsem\db\lab1\lab1>python manage.py startapp mus
```

Листинг 3.1. Создание проекта

Для изменения стандартных настроек был открыт файл ...lab1\lab1\lab1\settings.py. Первоначально в нем содержались следующие строчки:

```
...

DATABASES = {
        'default': {
                'ENGINE': 'django.db.backends.sqlite3',
                'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
        }
```

```
}

...

INSTALLED_APPS = [
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
]

...
```

Листинг 3.2. Часть settings.py до изменений

Они были заменены на следующие:

```
...

DATABASES = {
        'default': {
                'ENGINE': 'django.db.backends.postgresql_psycopg2',
                'NAME': 'mus',
                'USER': 'dan0n',
                'PASSWORD': 'myasnov',
                'HOST': 'localhost',
                'PORT': '5432',
        }
}

...

INSTALLED_APPS = [
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
        'mus',
]

...
```

Листинг 3.3. Часть settings.py после изменений

Далее, для формирования моделей необходимо открыть файл ...lab1\lab1\mus\models.py. В данном файле описываются модели для последующей миграции.

3.2 Миграция

По результатам прошлых лабораторных работ имеется схема БД, приведенная на рис. 3.1, согласно которой будут формироваться соответствующие модели для миграции.

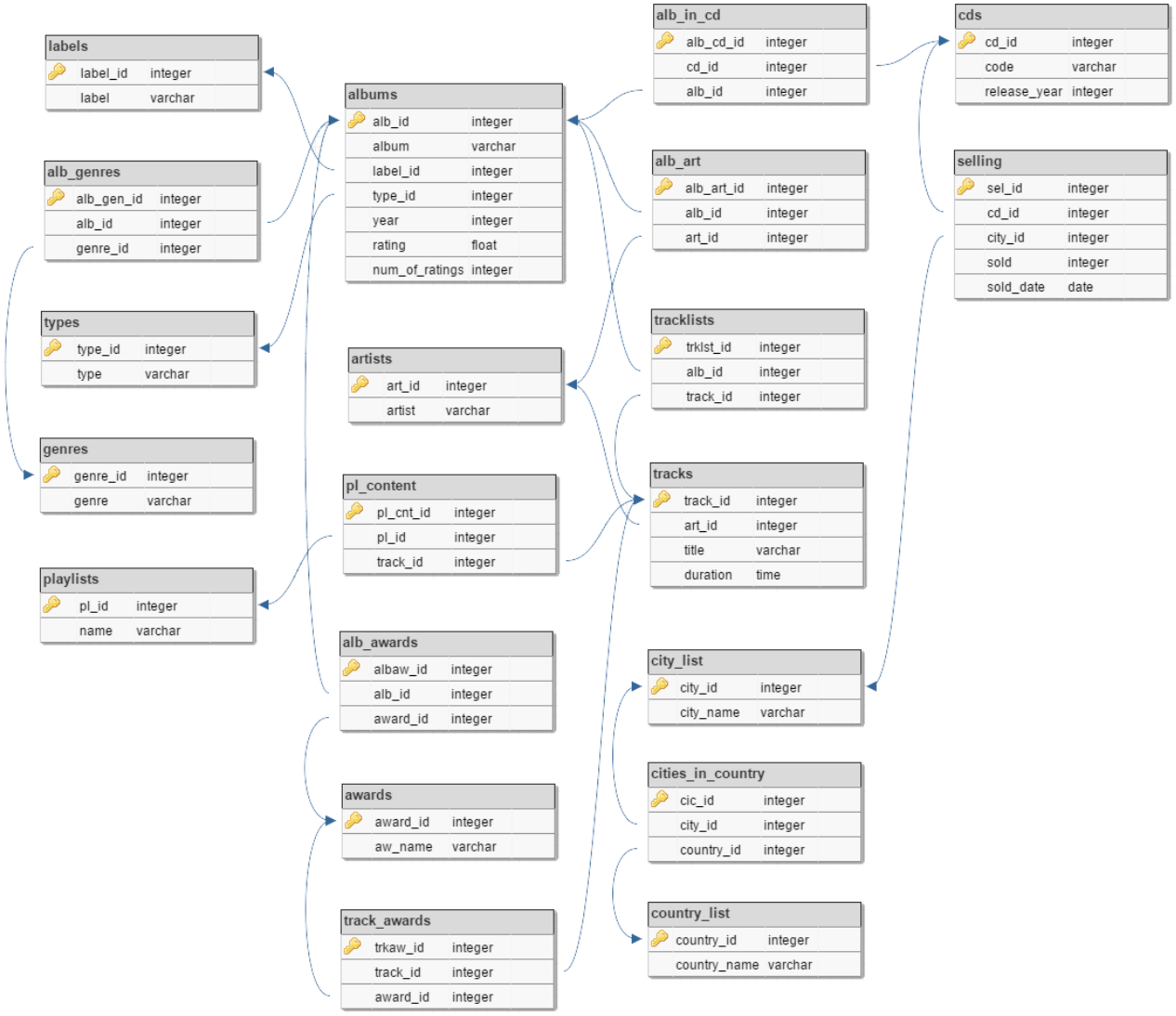

Рис. 3.1. SQL-схема БД

В файл models.py были добавлены модели, соответствующие приведенной базе данных.

```
from django.db import models

# default tables
class Albums(models.Model):
        alb_id = models.IntegerField(primary_key = True)
        album = models.CharField(max_length = 100)
        label = models.ForeignKey('Labels', null = True)
        type = models.ForeignKey('Types', null = True)
        year = models.IntegerField(null = True)
        rating = models.DecimalField(max_digits = 3, decimal_places =
   2, null = True)
```

```
        num_of_ratings = models.IntegerField(null = True)

        class Meta:
                db_table = "albums"

class Artists(models.Model):
        art_id = models.IntegerField(primary_key = True)
        artist = models.CharField(max_length = 100)

        class Meta:
                db_table = "artists"

class Cds(models.Model):
        cd_id = models.IntegerField(primary_key = True)
        code = models.CharField(max_length = 100)
        release_year = models.IntegerField(null = True)

        class Meta:
                db_table = "cds"

class AlbArt(models.Model):
        alb_art_id = models.IntegerField(primary_key = True)
        alb = models.ForeignKey('Albums')
        art = models.ForeignKey('Artists')

        class Meta:
                db_table = "alb_art"

class AlbInCd(models.Model):
        alb_cd_id = models.IntegerField(primary_key = True)
        cd = models.ForeignKey('Cds')
        alb = models.ForeignKey('Albums')

        class Meta:
                db_table = "alb_in_cd"

class Labels(models.Model):
        label_id = models.IntegerField(primary_key = True)
        label = models.CharField(max_length = 100)

        class Meta:
                db_table = "labels"

class Types(models.Model):
        type_id = models.IntegerField(primary_key = True)
        type = models.CharField(max_length = 40)

        class Meta:
                db_table = "types"

class Genres(models.Model):
        genre_id = models.IntegerField(primary_key = True)
        genre = models.CharField(max_length = 40)

```

```
        class Meta:
66              db_table = "genres"

68 class Tracks(models.Model):
        track_id = models.IntegerField(primary_key = True)
70      art = models.ForeignKey('Artists')
        title = models.CharField(max_length = 100)
72      duration = models.DurationField()

74      class Meta:
                db_table = "tracks"
76
  class Tracklists(models.Model):
78      trklst_id = models.IntegerField(primary_key = True)
        alb = models.ForeignKey('Albums')
80      track = models.ForeignKey('Tracks')

82      class Meta:
                db_table = "tracklists"
84
  class Playlists(models.Model):
86      pl_id = models.IntegerField(primary_key = True)
        name = models.CharField(max_length = 200)
88
        class Meta:
90              db_table = "playlists"

92 class PlContent(models.Model):
        pl_cnt_id = models.IntegerField(primary_key = True)
94      pl = models.ForeignKey('Playlists')
        track = models.ForeignKey('Tracks')
96
        class Meta:
98              db_table = "pl_content"

100 class AlbGenres(models.Model):
        alb_gen_id = models.IntegerField(primary_key = True)
102     alb = models.ForeignKey('Albums')
        genre = models.ForeignKey('Genres')
104
        class Meta:
106             db_table = "alb_genres"

108 # modification tables
  class TrackAwards(models.Model):
110     trkaw_id = models.IntegerField(primary_key = True)
        track = models.ForeignKey('Tracks')
112     award = models.ForeignKey('Awards')

114     class Meta:
                db_table = "track_awards"
116
  class Awards(models.Model):
118     award_id = models.IntegerField(primary_key = True)
```

```
        aw_name = models.CharField(max_length = 20)

        class Meta:
                db_table = "awards"

class AlbAwards(models.Model):
        albaw_id = models.IntegerField(primary_key = True)
        alb = models.ForeignKey('Albums')
        award = models.ForeignKey('Awards')

        class Meta:
                db_table = "alb_awards"

class CityList(models.Model):
        city_id = models.IntegerField(primary_key = True)
        city_name = models.CharField(max_length = 10)

        class Meta:
                db_table = "city_list"

class CountryList(models.Model):
        country_id = models.IntegerField(primary_key = True)
        country_name = models.CharField(max_length = 10)

        class Meta:
                db_table = "country_list"

class CitiesInCountry(models.Model):
        cic_id = models.IntegerField(primary_key = True)
        city = models.ForeignKey('CityList')
        country = models.ForeignKey('CountryList')

        class Meta:
                db_table = "cities_in_country"

class Selling(models.Model):
        sel_id = models.IntegerField(primary_key = True)
        cd = models.ForeignKey('Cds')
        city = models.ForeignKey('CityList')
        sold = models.IntegerField()
        sold_date = models.DateField(auto_now = False, auto_now_add =
   False)

        class Meta:
                db_table = "selling"
```

Листинг 3.4. models.py

Если ключ является вторичным, то Django автоматически допишет **_id** к данному полю. Также необходимо указывать класс Meta для каждой таблицы, иначе Django автоматически допишет название проекта к каждой таблице. К тому же необязательно самостоятельно указывать первичные ключи, т.к. Django добавит их автоматически при отсутствии.

После написания моделей был запущен процесс миграции.

```
D:\4 course\lastsem\db\lab1\lab1>python manage.py makemigrations mus
Migrations for 'mus':
  mus\migrations\0001_initial.py:
    - Create model AlbArt
    - Create model AlbAwards
    - Create model AlbGenres
    - Create model AlbInCd
    - Create model Albums
    - Create model Artists
    - Create model Awards
    - Create model Cds
    - Create model CitiesInCountry
    - Create model CityList
    - Create model CountryList
    - Create model Genres
    - Create model Labels
    - Create model Playlists
    - Create model PlContent
    - Create model Selling
    - Create model TrackAwards
    - Create model Tracklists
    - Create model Tracks
    - Create model Types
    - Add field track to tracklists
    - Add field track to trackawards
    - Add field track to plcontent
    - Add field city to citiesincountry
    - Add field country to citiesincountry
    - Add field label to albums
    - Add field type to albums
    - Add field alb to albincd
    - Add field cd to albincd
    - Add field alb to albgenres
    - Add field genre to albgenres
    - Add field alb to albawards
    - Add field award to albawards
    - Add field alb to albart
    - Add field art to albart

D:\4 course\lastsem\db\lab1\lab1>python manage.py migrate mus
Operations to perform:
Apply all migrations: mus
Running migrations:
Applying mus.0001_initial... OK
```

Листинг 3.5. Процесс миграция

По завершении миграции база данных содержит все таблицы из схемы, а также таблицу **django_migrations**, которая необходима для работы системы миграции Django.

3.3 Manage-команды

Для реализации manage-команд по заполнению базы данных, в директорию проекта **mus** была добавлена директория **management** и сопутствующие файлы. Дерево директории mus теперь выглядит следующим образом:

```
mus
├── admin.py
├── __init__.py
├── management
│   ├── commands
│   │   ├── __init__.py
│   │   └── populate_db.py
│   └── __init__.py
├── models.py
└── прочие файлы
```

Файл **populate_db.py** отвечает за исполнение команд. Для заполнения каждой таблицы 3 значениями, в данный файл были внесены соответствующие команды.

```
from django.core.management.base import BaseCommand
from mus.models import *
import datetime

class Command(BaseCommand):
    args = '<foo bar ...>'
    help = 'our help string comes here'

    def _createCds(self):
        temp1 = Cds(cd_id = 1, code = "JNCD-0001", release_year = 2014)
        temp1.save()
        temp2 = Cds(cd_id = 2, code = "JNCD-0002", release_year = 2015)
        temp2.save()
        temp3 = Cds(cd_id = 3, code = "JNCD-0003", release_year = 2015)
        temp3.save()

    def _createLabels(self):
        temp1 = Labels(label_id = 1, label = "J-NERATION")
        temp1.save()
        temp2 = Labels(label_id = 2, label = "Unknown Label")
        temp2.save()
        temp3 = Labels(label_id = 3, label = "LapFox Trax")
        temp3.save()

    def _createTypes(self):
        temp1 = Types(type_id = 1, type = "Single")
        temp1.save()
        temp2 = Types(type_id = 2, type = "EP")
        temp2.save()
        temp3 = Types(type_id = 3, type = "Compilation")
        temp3.save()

    def _createGenres(self):
```

```
        temp1 = Genres(genre_id = 1, genre = "J-Core")
        temp1.save()
        temp2 = Genres(genre_id = 2, genre = "Happy Hardcore")
        temp2.save()
        temp3 = Genres(genre_id = 3, genre = "Dark Abient")
        temp3.save()

    def _createArtists(self):
        temp1 = Artists(art_id = 1, artist = "Nizikawa")
        temp1.save()
        temp2 = Artists(art_id = 2, artist = "Chill")
        temp2.save()
        temp3 = Artists(art_id = 3, artist = "ETIA")
        temp3.save()

    def _createTracks(self):
        temp1 = Tracks(track_id = 1, art_id = 1, title = "Jailbreak",
duration = datetime.timedelta(minutes = 4, seconds = 48))
        temp1.save()
        temp2 = Tracks(track_id = 2, art_id = 2, title = "Do It Right",
 duration = datetime.timedelta(minutes = 5, seconds = 33))
        temp2.save()
        temp3 = Tracks(track_id = 3, art_id = 3, title = "Killer Bee",
duration = datetime.timedelta(minutes = 4, seconds = 20))
        temp3.save()

    def _createPlaylists(self):
        temp1 = Playlists(pl_id = 1, name = "J-NERATION BEST")
        temp1.save()
        temp2 = Playlists(pl_id = 2, name = "Best Hits")
        temp2.save()
        temp3 = Playlists(pl_id = 3, name = "My playlist")
        temp3.save()

    def _createPlContent(self):
        temp1 = PlContent(pl_cnt_id = 1, pl_id = 1, track_id = 2)
        temp1.save()
        temp2 = PlContent(pl_cnt_id = 2, pl_id = 1, track_id = 1)
        temp2.save()
        temp3 = PlContent(pl_cnt_id = 3, pl_id = 1, track_id = 3)
        temp3.save()

    def _createAlbums(self):
        temp1 = Albums(alb_id = 1, album = "J-NERATION", label_id = 1,
type_id = 3, year = 2014, rating = 4, num_of_ratings = 1)
        temp1.save()
        temp2 = Albums(alb_id = 2, album = "J-NERATION 2", label_id =
1, type_id = 3, year = 2015, rating = 5, num_of_ratings = 1)
        temp2.save()
        temp3 = Albums(alb_id = 3, album = "J-NERATION 3", label_id =
1, type_id = 3, year = 2015, rating = 5, num_of_ratings = 1)
        temp3.save()

    def _createAlbGenres(self):
```

```
        temp1 = AlbGenres(alb_gen_id = 1, alb_id = 1, genre_id = 1)
        temp1.save()
        temp2 = AlbGenres(alb_gen_id = 2, alb_id = 2, genre_id = 1)
        temp2.save()
        temp3 = AlbGenres(alb_gen_id = 3, alb_id = 2, genre_id = 1)
        temp3.save()

    def _createAlbInCd(self):
        temp1 = AlbInCd(alb_cd_id = 1, cd_id = 1, alb_id = 1)
        temp1.save()
        temp2 = AlbInCd(alb_cd_id = 2, cd_id = 2, alb_id = 2)
        temp2.save()
        temp3 = AlbInCd(alb_cd_id = 3, cd_id = 3, alb_id = 3)
        temp3.save()

    def _createAlbArt(self):
        temp1 = AlbArt(alb_art_id = 1, alb_id = 1, art_id = 2)
        temp1.save()
        temp2 = AlbArt(alb_art_id = 2, alb_id = 2, art_id = 2)
        temp2.save()
        temp3 = AlbArt(alb_art_id = 3, alb_id = 3, art_id = 2)
        temp3.save()

    def _createTracklists(self):
        temp1 = Tracklists(trklst_id = 1, alb_id = 1, track_id = 1)
        temp1.save()
        temp2 = Tracklists(trklst_id = 2, alb_id = 1, track_id = 2)
        temp2.save()
        temp3 = Tracklists(trklst_id = 3, alb_id = 1, track_id = 3)
        temp3.save()

    def _createAwards(self):
        temp1 = Awards(award_id = 1, aw_name = "Grammy")
        temp1.save()
        temp2 = Awards(award_id = 2, aw_name = "MTV Awards")
        temp2.save()
        temp3 = Awards(award_id = 3, aw_name = "MTV Russia Awards")
        temp3.save()

    def _createTrackAwards(self):
        temp1 = TrackAwards(trkaw_id = 1, track_id = 1, award_id = 1)
        temp1.save()
        temp2 = TrackAwards(trkaw_id = 2, track_id = 2, award_id = 2)
        temp2.save()
        temp3 = TrackAwards(trkaw_id = 3, track_id = 3, award_id = 3)
        temp3.save()

    def _createAlbAwards(self):
        temp1 = AlbAwards(albaw_id = 1, alb_id = 1, award_id = 2)
        temp1.save()
        temp2 = AlbAwards(albaw_id = 2, alb_id = 2, award_id = 2)
        temp2.save()
        temp3 = AlbAwards(albaw_id = 3, alb_id = 3, award_id = 2)
        temp3.save()
```

```
136
    def _createCityList(self):
138         temp1 = CityList(city_id = 1, city_name = "Moscow")
            temp1.save()
140         temp2 = CityList(city_id = 2, city_name = "Bryansk")
            temp2.save()
142         temp3 = CityList(city_id = 3, city_name = "Omsk")
            temp3.save()
144
    def _createCountryList(self):
146         temp1 = CountryList(country_id = 1, country_name = "Russia")
            temp1.save()
148         temp2 = CountryList(country_id = 2, country_name = "USA")
            temp2.save()
150         temp3 = CountryList(country_id = 3, country_name = "Ukraine")
            temp3.save()
152
    def _createCitiesInCountry(self):
154         temp1 = CitiesInCountry(cic_id = 1, city_id = 1, country_id =
    1)
            temp1.save()
156         temp2 = CitiesInCountry(cic_id = 2, city_id = 2, country_id =
    1)
            temp2.save()
158         temp3 = CitiesInCountry(cic_id = 3, city_id = 3, country_id =
    1)
            temp3.save()
160
    def _createSelling(self):
162         temp1 = Selling(sel_id = 1, cd_id = 1, city_id = 1, sold =
    15000, sold_date = datetime.date(day = 1, month = 1, year = 2014))
            temp1.save()
164         temp2 = Selling(sel_id = 2, cd_id = 2, city_id = 1, sold =
    5000, sold_date = datetime.date(day = 10, month = 3, year = 2015))
            temp2.save()
166         temp3 = Selling(sel_id = 3, cd_id = 3, city_id = 1, sold =
    1000, sold_date = datetime.date(day = 8, month = 7, year = 2015))
            temp3.save()
168
    def handle(self, *args, **options):
170         self._createCds()
            self._createLabels()
172         self._createTypes()
            self._createGenres()
174         self._createArtists()
            self._createTracks()
176         self._createPlaylists()
            self._createPlContent()
178         self._createAlbums()
            self._createAlbGenres()
180         self._createAlbInCd()
            self._createAlbArt()
182         self._createTracklists()
            self._createAwards()
```

```
184         self._createTrackAwards()
            self._createAlbAwards()
186         self._createCityList()
            self._createCountryList()
188         self._createCitiesInCountry()
            self._createSelling()
```

Листинг 3.6. populate_db.py

Теперь запустим команду и проверим с помощью psql.

```
1 D:\4 course\lastsem\db\lab1\lab1>python manage.py populate_db
2
  D:\4 course\lastsem\db\lab1\lab1>psql -U postgres
4 Пароль пользователя postgres:
  psql (9.6.2)
6 Введите "help", чтобы получить справку.

8 postgres=# \c mus
  Вы подключены к базе данных "mus" как пользователь "postgres".
10 mus=# SELECT * FROM Tracks;
   track_id |    title    | duration | art_id
12 ----------+-------------+----------+--------
          1 | Jailbreak   | 00:04:48 |      1
14        2 | Do It Right | 00:05:33 |      2
          3 | Killer Bee  | 00:04:20 |      3
```

Листинг 3.7. Выполнение manage-команды

Как и ожидалось, данные были успешно добавлены в БД.

4 Вывод

В результате работы было проведено знакомство с миграциями моделей на примере Django, а также с manage-командами для заполнения БД.

К достоинствам миграции Django можно отнести:

- Отслеживание схемы БД;
- Поддержка различных БД (PostgreSQL, MySQL, SQLite);
- Ускорение процесса изменения схемы базы данных;
- Возможность отката изменений.

К недостаткам можно отнести переход от стандартизированных sql запросов и команд к собственному формату Django, что требует некоторого изучения перед использованием.

Также был проведен опыт использования manage-команд. Использование данного инструмента позволяет расширить проект собственными командами, к примеру, геератором псевдослучайных записей в БД.