

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: Знакомство с ORM на примере Django

Выполнил студент группы 43501/3

_____ Круминьш Д.В.
(подпись)

Преподаватель

_____ Мясов А.В.
(подпись)

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 (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 **ulmart**, а также пользователь **psaer**. Далее был создан проект, следующими командами:

```
1 C:\study\s08\БД\task_1\work>django-admin.py startproject lab_1
2 C:\study\s08\БД\task_1\work>cd lab_1
3 C:\study\s08\БД\task_1\work\lab_1>python manage.py startapp ulmart
```

Листинг 1: Создание проекта

Для изменения стандартных настроек, был открыт файл ...work\lab_1\lab_1\settings.py. Первоначально в нем содержались следующие строки:

```
1 ...
2
3 DATABASES = {
4     'default': {
5         'ENGINE': 'django.db.backends.sqlite3',
```

```

6         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
7     }
8 }
9
10 ...
11
12 INSTALLED_APPS = [
13     'django.contrib.admin',
14     'django.contrib.auth',
15     'django.contrib.contenttypes',
16     'django.contrib.sessions',
17     'django.contrib.messages',
18     'django.contrib.staticfiles',
19 ]
20
21 ...

```

Листинг 2: Часть settings.py до изменений

Которые были заменены на следующие:

```

1 ...
2
3 DATABASES = {
4     'default': {
5         'ENGINE': 'django.db.backends.postgresql_psycopg2',
6         'NAME': 'ulmart',
7         'USER': 'psaer',
8         'PASSWORD': '1234',
9         'HOST': 'localhost',
10        'PORT': '5432',
11    }
12 }
13
14 ...
15
16 INSTALLED_APPS = [
17     'django.contrib.admin',
18     'django.contrib.auth',
19     'django.contrib.contenttypes',
20     'django.contrib.sessions',
21     'django.contrib.messages',
22     'django.contrib.staticfiles',
23     'ulmart',
24 ]
25
26 ...

```

Листинг 3: Часть settings.py после изменений

Далее, для формирования моделей необходимо открыть файл ...work\lab_1\ulmart\models.py. Именно в данном файле, необходимо описать модели, для последующей миграции.

3.2 Миграция

По результатам прошлых лабораторных работ, имеется схема базы-данных, приведенная на рис. 1, согласно которой, будут формироваться соответствующие модели для миграции.

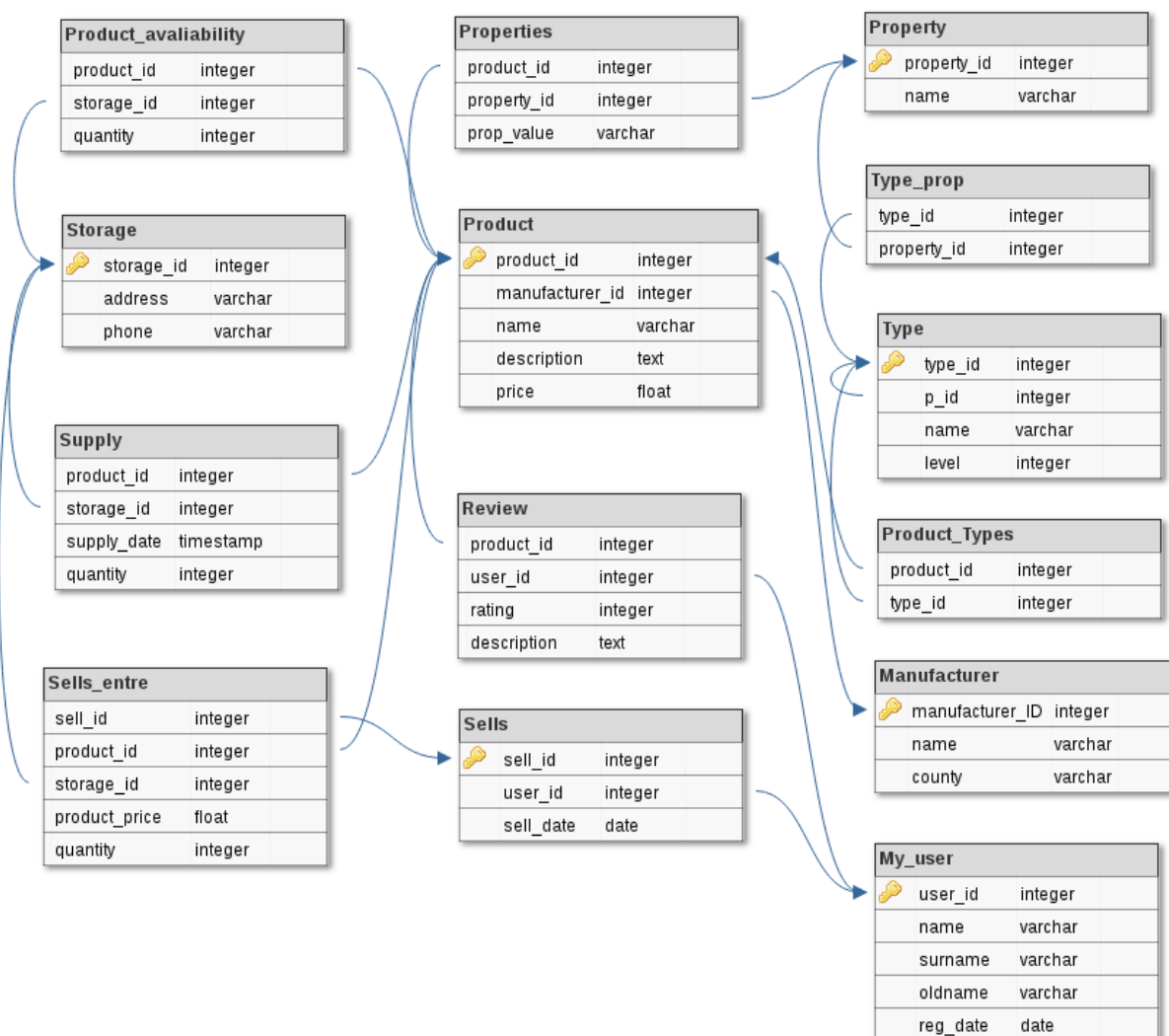


Рис. 1: SQL-схема БД

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

```
1 from django.db import models
2
3 class My_user(models.Model):
4     user_id=models.IntegerField(primary_key=True)
5     name=models.CharField(max_length=200)
6     surname=models.CharField(max_length=200)
7     oldname=models.CharField(max_length=200)
8     reg_date = models.DateField(auto_now=False, auto_now_add=True)
9
10     class Meta:
11         db_table = "My_user"
12
13 class Sells(models.Model):
14     sell_id=models.IntegerField(primary_key=True)
15     user=models.ForeignKey('My_user')
```

```

16     sell_date = models.DateField(auto_now=False, auto_now_add=False)
17
18     class Meta:
19         db_table = "Sells"
20
21 class Manufacturer(models.Model):
22     manufacturer_id=models.IntegerField(primary_key=True)
23     name=models.CharField(max_length=200)
24     country=models.CharField(max_length=200)
25
26     class Meta:
27         db_table = "Manufacturer"
28
29 class Type(models.Model):
30     type_id=models.IntegerField(primary_key=True)
31     p=models.ForeignKey('Type', null=True)
32     name=models.CharField(max_length=200)
33     level=models.IntegerField()
34
35     class Meta:
36         db_table = "Type"
37
38 class Property(models.Model):
39     property_id=models.IntegerField(primary_key=True)
40     name=models.CharField(max_length=200)
41
42     class Meta:
43         db_table = "Property"
44
45 class Type_prop(models.Model):
46     type=models.ForeignKey('Type')
47     property=models.ForeignKey('Property')
48
49     class Meta:
50         db_table = "Type_prop"
51
52 class Product(models.Model):
53     product_id=models.IntegerField(primary_key=True)
54     manufacturer=models.ForeignKey('Manufacturer')
55     name=models.CharField(max_length=200)
56     description=models.CharField(max_length=200)
57     price=models.FloatField()
58
59     class Meta:
60         db_table = "Product"
61
62 class Product_types(models.Model):
63     product=models.ForeignKey('Product')
64     type=models.ForeignKey('Type')
65
66     class Meta:
67         db_table = "Product_types"
68

```

```

69 class Properties(models.Model):
70     product=models.ForeignKey('Product')
71     property=models.ForeignKey('Property')
72     prop_value=models.CharField(max_length=200)
73
74     class Meta:
75         db_table = "Properties"
76
77 class Review(models.Model):
78     product=models.ForeignKey('Product')
79     user=models.ForeignKey('My_user')
80     rating=models.IntegerField()
81     description=models.CharField(max_length=200)
82
83     class Meta:
84         db_table = "Review"
85
86 class Storage(models.Model):
87     storage_id=models.IntegerField(primary_key=True)
88     address=models.CharField(max_length=200)
89     phone=models.CharField(max_length=200)
90
91     class Meta:
92         db_table = "Storage"
93
94 class Product_avaliability(models.Model):
95     product=models.ForeignKey('Product')
96     storage=models.ForeignKey('Storage')
97     quantity=models.IntegerField()
98
99     class Meta:
100         db_table = "Product_avaliability"
101
102 class Supply(models.Model):
103     product=models.ForeignKey('Product')
104     storage=models.ForeignKey('Storage')
105     supply_date = models.DateField(auto_now=False, auto_now_add=True)
106     quantity=models.IntegerField()
107
108     class Meta:
109         db_table = "Supply"
110
111 class Sells_entre(models.Model):
112     sell=models.ForeignKey('Sells')
113     product=models.ForeignKey('Product')
114     storage=models.ForeignKey('Storage')
115     product_price=models.FloatField()
116     quantity=models.IntegerField()
117
118     class Meta:
119         db_table = "Sells_entre"

```

Листинг 4: models.py

Примечание: 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. После написания моделей был запущен процесс миграции.

```
1 C:\study\s08\БД\task_1\work\lab_1>python manage.py makemigrations
   ↳ ulmart
2 Migrations for 'ulmart':
3 ulmart\migrations\0001_initial.py:
4 - Create model Manufacturer
5 - Create model My_user
6 - Create model Product
7 - Create model Product_avaliability
8 - Create model Product_types
9 - Create model Properties
10 - Create model Property
11 - Create model Review
12 - Create model Sells
13 - Create model Sells_entre
14 - Create model Storage
15 - Create model Supply
16 - Create model Type
17 - Create model Type_prop
18 - Add field storage to sells_entre
19 - Add field property to properties
20 - Add field type to product_types
21 - Add field storage to product_avaliability
22
23 C:\study\s08\БД\task_1\work\lab_1>python manage.py migrate ulmart
24 Operations to perform:
25 Apply all migrations: ulmart
26 Running migrations:
27 Applying ulmart.0001_initial... OK
```

Листинг 5: Процесс миграция

По завершению 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 **ulmart** была добавлена директория **management** и сопутствующие файлы. Дерево директории **ulmart**, теперь выглядит следующим образом:

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

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

```
1 from django.core.management.base import BaseCommand
2 from ulmart.models import *
3
4 class Command(BaseCommand):
5     args = '<foo bar ...>'
6     help = 'our help string comes here'
7
8     def _create_property(self):
9         temp1 = Property(property_id=1, name='prop_1')
10        temp1.save()
11        temp2 = Property(property_id=2, name='prop_2')
12        temp2.save()
13        temp3 = Property(property_id=3, name='prop_3')
14        temp3.save()
15
16    def _create_type(self):
17        temp1 = Type(type_id=1, name='type_1', level=1)
18        temp1.save()
19        temp2 = Type(type_id=2, p_id=1, name='type_2', level=2)
20        temp2.save()
21        temp3 = Type(type_id=3, name='type_3', level=1)
22        temp3.save()
23
24    def _create_type_prop(self):
25        temp1 = Type_prop(id=1, property_id=1, type_id=1)
26        temp1.save()
27        temp2 = Type_prop(id=2, property_id=2, type_id=2)
28        temp2.save()
29        temp3 = Type_prop(id=3, property_id=3, type_id=3)
30        temp3.save()
31
32    def _create_manufacturer(self):
33        temp1 = Manufacturer(manufacturer_id=1, name='manufac_1',
↪ country='China')
34        temp1.save()
35        temp2 = Manufacturer(manufacturer_id=2, name='manufac_2',
↪ country='China')
36        temp2.save()
37        temp3 = Manufacturer(manufacturer_id=3, name='manufac_3',
↪ country='China')
38        temp3.save()
39
40    def _create_product(self):
41        temp1 = Product(product_id=1, manufacturer_id=1, name='name_1'
↪ , description='desc_1', price=5000.20)
42        temp1.save()
43        temp2 = Product(product_id=2, manufacturer_id=2, name='name_2'
↪ , description='desc_2', price=8000.20)
44        temp2.save()
45        temp3 = Product(product_id=3, manufacturer_id=3, name='name_3'
↪ , description='desc_3', price=9000.20)
```



```

46         temp3.save()
47
48     def _create_product_types(self):
49         temp1 = Product_types(id=1, product_id=1, type_id=1)
50         temp1.save()
51         temp2 = Product_types(id=2, product_id=2, type_id=2)
52         temp2.save()
53         temp3 = Product_types(id=3, product_id=3, type_id=3)
54         temp3.save()
55
56     def _create_properties(self):
57         temp1 = Properties(id=1, product_id=1, property_id=1,
↪ prop_value='value_1')
58         temp1.save()
59         temp2 = Properties(id=2, product_id=2, property_id=3,
↪ prop_value='value_2')
60         temp2.save()
61         temp3 = Properties(id=3, product_id=3, property_id=3,
↪ prop_value='value_3')
62         temp3.save()
63
64     def _create_my_user(self):
65         temp1 = My_user(user_id=1, name='name_1', surname='surname_1',
↪ oldname='oldname_1', reg_date='2017-10-10')
66         temp1.save()
67         temp2 = My_user(user_id=2, name='name_2', surname='surname_2',
↪ oldname='oldname_2', reg_date='2017-10-10')
68         temp2.save()
69         temp3 = My_user(user_id=3, name='name_3', surname='surname_3',
↪ oldname='oldname_3', reg_date='2017-10-10')
70         temp3.save()
71
72     def _create_review(self):
73         temp1 = Review(id=1, product_id=1, user_id=1, rating=5,
↪ description='desc_1')
74         temp1.save()
75         temp2 = Review(id=2, product_id=2, user_id=2, rating=5,
↪ description='desc_2')
76         temp2.save()
77         temp3 = Review(id=3, product_id=3, user_id=3, rating=5,
↪ description='desc_3')
78         temp3.save()
79
80     def _create_sells(self):
81         temp1 = Sells(sell_id=1, user_id=1, sell_date='2017-10-10')
82         temp1.save()
83         temp2 = Sells(sell_id=2, user_id=2, sell_date='2017-10-10')
84         temp2.save()
85         temp3 = Sells(sell_id=3, user_id=3, sell_date='2017-10-10')
86         temp3.save()
87
88     def _create_storage(self):
89         temp1 = Storage(storage_id=1, address='addr_1', phone='phone_1

```

```

↪ ')
90     temp1.save()
91     temp2 = Storage(storage_id=2, address='addr_2', phone='phone_2
↪ ')
92     temp2.save()
93     temp3 = Storage(storage_id=3, address='addr_3', phone='phone_3
↪ ')
94     temp3.save()
95
96     def _create_product_avaliability(self):
97         temp1 = Product_avaliability(id=1, product_id=1, storage_id=1,
↪ quantity=10)
98         temp1.save()
99         temp2 = Product_avaliability(id=2, product_id=2, storage_id=2,
↪ quantity=10)
100        temp2.save()
101        temp3 = Product_avaliability(id=3, product_id=3, storage_id=3,
↪ quantity=10)
102        temp3.save()
103
104        def _create_supply(self):
105            temp1 = Supply(id=1, product_id=1, storage_id=1, supply_date='
↪ 2017-10-10', quantity=5)
106            temp1.save()
107            temp2 = Supply(id=2, product_id=2, storage_id=2, supply_date='
↪ 2017-10-10', quantity=5)
108            temp2.save()
109            temp3 = Supply(id=3, product_id=3, storage_id=3, supply_date='
↪ 2017-10-10', quantity=5)
110            temp3.save()
111
112            def _create_sells_entre(self):
113                temp1 = Sells_entre(id=1, sell_id=1, product_id=1, storage_id
↪ =1, product_price=413.21, quantity=10)
114                temp1.save()
115                temp2 = Sells_entre(id=2, sell_id=2, product_id=3, storage_id
↪ =2, product_price=442.21, quantity=10)
116                temp2.save()
117                temp3 = Sells_entre(id=3, sell_id=3, product_id=3, storage_id
↪ =3, product_price=5313.21, quantity=10)
118                temp3.save()
119
120
121        def handle(self, *args, **options):
122            self._create_property()
123            self._create_type()
124            self._create_type_prop()
125            self._create_manufacturer()
126            self._create_product()
127            self._create_product_types()
128            self._create_properties()
129            self._create_my_user()
130            self._create_review()

```

```

131         self._create_sells()
132         self._create_storage()
133         self._create_product_avaliability()
134         self._create_supply()
135         self._create_sells_entre()

```

Листинг 6: populate_db.py

Теперь запустим команду и проверим с помощью psql.

```

1 C:\study\s08\БД\task_1\work\lab_1>python manage.py populate_db
2
3 C:\study\s08\БД\task_1\work\lab_1>psql -U postgres
4 Пароль пользователя postgres:
5 psql (9.6.2)
6 Введите "help", чтобы получить справку.
7
8 postgres=# \c ulmart
9 Вы подключены к базе данных "ulmart" как пользователь "postgres".
10 ulmart=# select * from "Product";
11 product_id | name | description | price | manufacturer_id
12 -----+-----+-----+-----+-----
13 1          | name_1 | desc_1      | 5000.2 | 1
14 2          | name_2 | desc_2      | 8000.2 | 2
15 3          | name_3 | desc_3      | 9000.2 | 3

```

Листинг 7: Выполнение manage-команды

Чего и требовалось ожидать, команда успешно добавила данные в каждую из таблиц.

4 Вывод

В результате данной работы было проведено знакомство с миграциями моделей используя Django, а также с manage-командами для наполнения базы данных. К достоинствам миграции Django можно отнести:

- Ускорение процесса изменения схемы базы данных;
- Возможность отслеживания схемы базы данных;
- Поддержка многими бэкендами(PostgreSQL, MySQL, SQLite);
- Возможность отката.

Несмотря на подобные достоинства, я бы все равно предпочел-бы проводить какие-либо операции с базами данных используя прямые sql запросы и команды.

Также был проведен опыт использования manage-команд. Использование данного инструмента позволяет расширить проект, написанием каких-либо собственных команд. Так например, можно написать генератор для заполнения полей таблиц в базе данных.