Московский государственный технический университет им. Н.Э. Баумана

Факультет “Радиотехнический”
Кафедра ИУ5 “Системы обработки информации и управления”

Отчет по РК1 по курсу
Базовые компоненты интернет технологий

Вариант 11

Выполнил:
Студент группы РТ5-31Б
Корсаков Н.А.
Проверил:
Доцент кафедры ИУ5
Гапанюк Ю.Е.

2021г.

Задание

1) Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

Пример классов данных для предметной области Сотрудник-Отдел:

1. Класс «Сотрудник», содержащий поля:
   - ID записи о сотруднике;
   - Фамилия сотрудника;
   - Зарплата (количественный признак);
   - ID записи об отделе. (для реализации связи один-ко-многим)
2. Класс «Отдел», содержащий поля:
   - ID записи об отделе;
   - Наименование отдела.
3. (Для реализации связи многие-ко-многим) Класс «Сотрудники отдела», содержащий поля:
   - ID записи о сотруднике;
   - ID записи об отделе.

2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов «Сотрудник» и «Отдел», которые используются в примере. Вам нужно перенести эти требования в Ваш вариант предметной области. При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков).

Для реализации запроса №2 введите в класс, находящийся на стороне связи «много», произвольный количественный признак, например, «зарплата сотрудника».

Результатом рубежного контроля является документ в формате PDF, который содержит текст программы и результаты ее выполнения.

Вариант Е.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «отдел», и список работающих в них сотрудников.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате. Средняя зарплата должна быть округлена до 2 знака после запятой (*отдельной функции вычисления среднего значения в Python нет, нужно использовать*

комбинацию функций вычисления суммы и количества значений; для округления необходимо использовать функцию [*https://docs.python.org/3/library/functions.html#round*](https://docs.python.org/3/library/functions.html#round)).

3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.

Текст программы

```
from operator import itemgetter


class  PR:
    def __init__(self, id, name_pr, v, pc_id):
        self.id = id
        self.name_pr = name_pr
        self.v = v
        self.pc_id = pc_id


class PC:
    def __init__(self, id, name):
        self.id = id
        self.name = name


class PR_Pc:
    """
    связь многие ко многим
    """
    def __init__(self, pc_id, pr_id):
        self.pc_id = pc_id
        self.pr_id = pr_id


Pcs = [
    PC(1, 'компьютер Acer'),
    PC(2, 'компьютер Samsung'),
    PC(3, 'компютер Lenovo'),
    PC(4, 'Imac'),
```

```
]

prs = [
    PR(1, 'PyCharm', 224, 3),
    PR(2, 'Photoshop', 2048, 2),
    PR(3, 'Zoom', 289, 1),
    PR(4, 'Telegram', 129, 4),
]

prs_pcs = [
    PR_Pc(1,1),
    PR_Pc(1,4),
    PR_Pc(2,2),
    PR_Pc(2,3),
    PR_Pc(3,4),
    PR_Pc(3,3),
    PR_Pc(4,4),
    PR_Pc(4,1),

]

def main():
    """Основная функция"""

    # Соединение данных один-ко-многим
    one_to_many = [(pr.name_pr, pr.v, p.name)
        for p in Pcs
        for pr in prs
        if pr.pc_id==p.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(p.name, pp.pc_id, pp.pr_id)
        for p in Pcs
```

```
        for pp in prs_pcs
        if p.id==pp.pc_id]

    many_to_many = [(pr.name_pr, pr.v, pc_name)
        for pc_name, pc_id, pr_id in many_to_many_temp
        for pr in prs if pr.id==pr_id]

    print('Задание E1')
    res1 = []
    for name_pr, v, name in one_to_many:
            if 'компьютер' in name:
                res1.append((name, name_pr))
    print(res1)

    print('\nЗадание E2')
    res2_unsorted = []
    for p in Pcs:
        prss = list(filter(lambda i: i[2]==p.name, one_to_many))
        if len(prss) > 0:
            v_ob = [v for _,v,_ in prss]
            sum_a = sum(v_ob)/len(v_ob)
            res2_unsorted.append((p.name, sum_a))
    res2 = sorted(res2_unsorted, key=itemgetter(1))
    print(res2)

    print('\nЗадание E3')
    res3 = []
    for name_pr, v, name in many_to_many:
        if name_pr.find("P") == 0:
            res3.append((name_pr, name))
    print(res3)

if __name__ == '__main__':
```

main()

Результат работы программы

```
Задание E1
[('компьютер Acer', 'Zoom'), ('компьютер Samsung', 'Photoshop')]

Задание E2
[('Imac', 129.0), ('компютер Lenovo', 224.0), ('компьютер Acer', 289.0), ('компьютер Samsung', 2048.0)]

Задание E3
[('PyCharm', 'компьютер Acer'), ('Photoshop', 'компьютер Samsung'), ('PyCharm', 'Imac')]
```