Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана»

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Отчёт по РК №1

Выполнила:
студентка группы ИУ5-35Б
Сухова Мария Андреевна
Подпись:

Дата:

Проверил:
Преподаватель кафедры ИУ5
Гапанюк Юрий Евгеньевич
Подпись:

Дата:

2022 г.

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

Вариант Д 18

18	Музыкальное произведение	Оркестр

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате (отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений).
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «Р», и список работающих в них сотрудников.
Текст программы
Программа разделена на несколько файлов

1) Файл classes.py содержит классы для реализации задания:

```
class music:
    """музыкальное произведение"""

    def __init__(self, id, author, name, music_id, duration):
        self.id = id
        self.author = author
        self.name = name
        self.music_id = music_id
        self.duration = duration

class orchestra:
    """оркестр"""

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


class orchMusic:
    def __init__(self, id_music, id_orch):
        self.id_music = id_music
        self.id_orch = id_orch
```

2) Файл connections.py содержит функции для создания связей один-комногим и многие-ко-многим:

```
def one_to_many(musics, orchectras):
    return [(mus.name, mus.duration, orch.name)
            for mus in musics
            for orch in orchectras
            if mus.orch_id == orch.id]


def many_to_many(musics, orchectras, orchMusic):
    return [(mus.surname, mus.duration, orch.name)
            for oc in orchMusic
            for mus in musics
            for orch in orchectras
            if mus.id == oc.id_mus and orch.id == oc.id_orch]
```

3) Файл processing.py содержит функции для реализации требуемых заданий:

```
def task1(one_to_many):
    return [el for el in one_to_many if el[0].endswith('ов')]


def task2(one_to_many):
    orch = []
    duration = []
    for el in one_to_many:
        if el[2] not in orch:
            orch.append(el[2])
            duration.append((el[1], 1))
        else:
            idx = orch.index(el[2])
            duration[idx] = ((duration[idx][0] + el[1]) /
(duration[idx][1] + 1), duration[idx][1] + 1)
    salary = [i[0] for i in duration]
    return sorted(list(zip(orch, duration)), key=lambda p: p[1],
reverse=True)


def task3(many_to_many):
    orch_R = []
    mus_orch_R = []
    for el in many_to_many:
        if el[2].startswith('P'):
            if el[2] not in orch_R:
                orch_R.append(el[2])
                mus_orch_R.append((el[0],))
            else:
                mus_orch_R[orch_R.index(el[2])] += (el[0],)
    return list(zip(orch_R, mus_orch_R))
```

4) Файл main.py реализует работу программы:

```
5) from processing.classes import music, orchestra, orchMusic
   from processing.connections import one_to_many, many_to_many
   from processing.processing import task1, task2, task3

   # Окрестры
   orchestras = [
       orchestra(1, 'Российский национальный оркестр'),
       orchestra(2, 'Ансамбль песни и пляски'),
       orchestra(3, 'Большой симфонический оркестр'),
       orchestra(4, 'Русская филармония'),
       orchestra(5, 'Виртуозы Москвы'),
       orchestra(6, 'Местные ребята')
   ]

   #Музыкальные произведения
```

```
musics = [
    music(1, 'Глинка', 'Симфония 1', 60, 1),
    music(2, 'Чайковский', 'Этюд 3' , 38, 2),
    music(3, 'Шопен', 'Вальс 7', 12, 4),
    music(4, 'Скрябин', 'Симфония 2', 20, 6),
    music(5, 'Бобер', 'Этюд 4', 28, 3),
    music(6, 'Рахманинов','Вальс 7', 17, 5),
    music(7, 'Мусоргский', 'Симфония 3', 33, 6),
    music(8, 'Прокофьев', 'Этюд 5', 18, 4),
    music(9, 'Серов', 'Вальс 8', 10, 2),
    music(10, 'Бетховен', 'Симфония 4', 57, 1),
    music(11, 'Моцерт', 'Этюд 6', 37, 5),
    music(12, 'Бах', 'Вальс 9', 22, 3),
    music(13, 'Варламов', 'Симфония 5', 7, 3)
]


# Для связи многие-ко-многим
orchMusics = [
    orchMusic(1, 1),
    orchMusic(2, 1),
    orchMusic(3, 2),
    orchMusic(4, 2),
    orchMusic(5, 1),
    orchMusic(6, 3),
    orchMusic(7, 4),
    orchMusic(8, 4),
    orchMusic(9, 5),
    orchMusic(10, 5),
    orchMusic(11, 5),
    orchMusic(12, 6),
    orchMusic(13, 6),
    orchMusic(7, 1),
    orchMusic(8, 2),
    orchMusic(2, 3),
    orchMusic(3, 4),
    orchMusic(4, 5),
    orchMusic(5, 6)
]

# Установка связей
one_to_many = one_to_many(musics, orchestras)
many_to_many = many_to_many(musics, orchestras, orchMusics)


print(*one_to_many, sep='\n', end='\n\n')
print(*many_to_many, sep='\n', end='\n\n')


def main():
    print('Task 1', task1(one_to_many), sep='\n', end='\n\n')
    print('Task 2', task2(one_to_many), sep='\n', end='\n\n')
    print('Task 3', task3(many_to_many), sep='\n', end='\n\n')


if __name__ == '__main__':
    main()
```

Работа программы:

1) После связывания один-ко-многим имеем:
   ('Симфония 1', 60, 'Российский национальный оркестр')
   ('Этюд 3', 38, 'Ансамбль песни и пляски')

('Вальс 7', 12, 'Русская филармония')
('Симфония 2', 20, 'Местные ребята')
('Этюд 4', 28, 'Большой симфонический оркестр')
('Вальс 7', 17, 'Виртуозы Москвы')
('Симфония 3', 33, 'Местные ребята')
('Этюд 5', 18, 'Русская филармония')
('Вальс 8', 10, 'Ансамбль песни и пляски')
('Симфония 4', 57, 'Российский национальный оркестр')
('Этюд 6', 37, 'Виртуозы Москвы')
('Вальс 9', 22, 'Большой симфонический оркестр')
('Симфония 5', 7, 'Большой симфонический оркестр')

2) После связывания многие-ко-многим имеем:
('Симфония 1', 60, 'Российский национальный оркестр')
('Этюд 3', 38, 'Российский национальный оркестр')
('Вальс 7', 12, 'Ансамбль песни и пляски')
('Симфония 2', 20, 'Ансамбль песни и пляски')
('Этюд 4', 28, 'Российский национальный оркестр')
('Вальс 7', 17, 'Большой симфонический оркестр')
('Симфония 3', 33, 'Русская филармония')
('Этюд 5', 18, 'Русская филармония')
('Вальс 8', 10, 'Виртуозы Москвы')
('Симфония 4', 57, 'Виртуозы Москвы')
('Этюд 6', 37, 'Виртуозы Москвы')
('Вальс 9', 22, 'Местные ребята')
('Симфония 5', 7, 'Местные ребята')
('Симфония 3', 33, 'Российский национальный оркестр')
('Этюд 5', 18, 'Ансамбль песни и пляски')
('Этюд 3', 38, 'Большой симфонический оркестр')
('Вальс 7', 12, 'Русская филармония')
('Симфония 2', 20, 'Виртуозы Москвы')
('Этюд 4', 28, 'Местные ребята')

3) Результаты выполнения заданий:
Task 1
[('Рахманинов', 17, 'Виртуозы Москвы'), ('Серов', 10, 'Ансамбль песни и пляски'), ('Варламов', 7, 'Большой симфонический оркестр')]

Task 2
[('Российский национальный оркестр', (58.5, 2)), ('Виртуозы Москвы', (27.0, 2)), ('Местные ребята', (26.5, 2)), ('Ансамбль песни и пляски', (24.0, 2)), ('Русская филармония', (15.0, 2)), ('Большой симфонический оркестр', (10.666666666666666, 3))]

Task 3
[('Российский национальный оркестр', ('Гончаренко', 'Иванов', 'Бобер', 'Галыгин')), ('Русская филармония', ('Галыгин', 'Кенчур', 'Самонян'))]