Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Рубежный контроль №1

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:
студент группы ИУ5-33Б
Стебунов А.И.

Проверил:
Гапанюк Ю.Е.

2021 г.

Оглавление

Постановка задачи:

Рубежный контроль представляет собой разработку программы на языке Python, которая выполняет следующие действия:

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).

Исходя из выданного задания, требуется:

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

Предметная область:

18	Музыкальное произведение	Оркестр

Текст программы:

```python
# используется для сортировки
from operator import itemgetter

class Composition:
    """Музыкальное произведение"""

    def __init__(self, id, naz, count):
        self.id = id
        self.naz = naz
        self.count = count #год издания.


class Orchestra:
    """Оркестр"""

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

class OrchestraComposition:
    """
    'Исполнители произведения' для реализаци
    связи многие-ко-многим
    """

    def __init__(self, orc_id, comp_id):
        self.orc_id = orc_id
        self.comp_id = comp_id


# Оркестр
orc = [
    Orchestra(1, 'народный'),
    Orchestra(2, 'камерный'),
    Orchestra(3, 'военный'),
    Orchestra(4, 'эстрадный'),
    Orchestra(5, 'симфонический'),
]
```

```python
# Музыкальные композиции
comp = [
    Composition(1, 'Лунная соната', 3),
    Composition(2, 'Танец рыцарей', 4),
    Composition(3, 'Марш Мендельсона', 2),
    Composition(4, 'Пятая симфония', 5),
    Composition(5, 'Щелкунчик', 4),
    Composition(6, 'Новый мерин', 3),
    Composition(7, 'Ария', 2),
    Composition(8, 'Кан-кан', 1)
]

orchestro_composition = [
    OrchestraComposition(1, 1),
    OrchestraComposition(2, 2),
    OrchestraComposition(3, 4),
    OrchestraComposition(4, 1),
    OrchestraComposition(5, 3),
    OrchestraComposition(6, 1),
    OrchestraComposition(7, 2),
    OrchestraComposition(8, 3),
    OrchestraComposition(7, 5),
    OrchestraComposition(6, 2),
]

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

    # Соединение данных один-ко-многим
    one_to_many = [(c.naz, c.count, o.name)
                   for o in orc
                   for c in comp
                   if c.count == o.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(o.name, oc.orc_id, oc.c
                          for o in orc
                          for oc in orchestro_comp
                          if o.id == oc.orc_id]
```

```
    many_to_many = [(c.naz, c.count, orc_name)
                    for orc_name, orc_id, comp_id in many_to_many_temp
                    for c in comp if c.id == comp_id]


    print('Test')#вывод списков со связями 1-м, м-м
    res_0 = (one_to_many)
    print(res_0)
    res_01 = (many_to_many)
    print(res_01)

    print('Задание B1')
    res_11 = []
    for naz, count, orc_name in one_to_many:
        if 'A' in naz[0]:
            res_11.append((naz, orc_name))
    print(res_11)

    print('Задание B2')
    buff = []
    for o in orc:
        #список видов транспорта
        o_names = list(filter(lambda i: i[2] == o.name, one_to_many))
        if len(o_names) > 0:
            o_count = [count for _, count, _ in o_names]
            min_count = min(o_count)
            buff.append((o.name, min_count))
    res_12 = sorted(buff, key=itemgetter(1))
    print(res_12)

    print('Задание B3')
    buff = []
    for naz, count, orc_name in many_to_many:
        buff.append((naz, orc_name))
    res_13 = list(sorted(buff, key=itemgetter(0)))
    print(res_13)

if __name__ == '__main__':
```

Результаты выполнения:

```
Задание B1
[('Ария', 'камерный')]
Задание B2
[('народный', 1), ('камерный', 2), ('военный', 3), ('эстрадный', 4), ('симфонический', 5)]
Задание B3
[('Лунная соната', 'народный'), ('Лунная соната', 'эстрадный'), ('Марш Мендельсона', 'симфонический'), ('Пятая симфония', 'военный'), ('Танец рыцарей', 'камерный')]
```