Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э.Баумана (национальный исследовательский университет)»

Факультет «Информатика и системы управления»

Курс «Базовые компоненты интернет-технологий»

Рубежный контроль №1

Выполнил:

студент группы ИУ5-33Б Николай Горкунов

подпись: , дата:

Проверил:

преподаватель Юрий Гапанюк

подпись: , дата:

2022 г.

Задание.

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.
   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».

Для моей группы вариант запросов В, для меня вариант предметной области 3.

- Вариант В запросов:
  «Отдел» и «Сотрудник» связаны отношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
  «Отдел» и «Сотрудник» связаны отношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
  «Отдел» и «Сотрудник» связаны отношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.
- Вариант 3 предметной области:
  Водитель-Автопарк.

Исходный код.

```
# Используется для сортировки
from operator import itemgetter

class Driver:
    """Водитель"""
    def __init__(self, id, fio, sal, park_id):
        self.id = id
        self.fio = fio
        self.sal = sal
        self.park_id = park_id

class Park:
    """Автопарк"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class DriverPark:
    """
    'Водители автопарка' для реализации
    связи многие-ко-многим
    """
    def __init__(self, park_id, driver_id):
        self.park_id = park_id
        self.driver_id = driver_id

# Автопарки
parks = [
    Park(1, 'Автопарк "Такси"'),
    Park(2, 'Автопарк "Драйв"'),
    Park(3, 'Автопарк "Бумер"'),

    Park(11, 'Автопарк "Такси" (другой)'),
    Park(22, 'Автопарк "Драйв" (другой)'),
    Park(33, 'Автопарк "Бумер" (другой)'),
]

# Водители
drivers = [
    Driver(1, 'Артамонов', 25000, 1),
    Driver(2, 'Петров', 35000, 2),
```

```
    Driver(3, 'Иваненко', 45000, 3),
    Driver(4, 'Иванов', 35000, 3),
    Driver(5, 'Иванин', 25000, 3),
]

drivers_parks = [
    DriverPark(1, 1),
    DriverPark(2, 2),
    DriverPark(3, 3),
    DriverPark(3, 4),
    DriverPark(3, 5),

    DriverPark(11, 1),
    DriverPark(22, 2),
    DriverPark(33, 3),
    DriverPark(33, 4),
    DriverPark(33, 5),
]

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

    # Соединение данных один-ко-многим
    one_to_many = [(d.fio, d.sal, p.name)
        for p in parks
        for d in drivers
        if d.park_id == p.id]

    # Соединение данных многие-ко-многим
    many_to_many_tmp = [(p.name, dp.park_id, dp.driver_id)
        for p in parks
        for dp in drivers_parks
        if p.id == dp.park_id]

    many_to_many = [(d.fio, d.sal, park_name)
        for park_name, park_id, driver_id in many_to_many_tmp
        for d in drivers if d.id == driver_id]

    print('Задание 1')
    res_11 = list(filter(lambda x: x[0].find('A') == 0,
one_to_many))
    print(res_11)
```

```
    print('\nЗадание 2')
    res_12_unsorted = []
    # Перебираем все автопарки
    for p in parks:
        # Список водителей автопарка
        p_drivers = list(filter(lambda i: i[2] == p.name,
one_to_many))
        # Если отдел не пустой
        if len(p_drivers) > 0:
            # Зарплаты водителей автопарка
            p_sals = list(map(lambda x: x[1], p_drivers))
            # Минимальная зарплата водителей автопарка
            p_sals_min = min(p_sals)
            res_12_unsorted.append((p.name, p_sals_min))

    # Сортировка по минимальной зарплате
    res_12 = sorted(res_12_unsorted, key = itemgetter(1))
    print(res_12)

    print('\nЗадание 3')
    # Сортировка по Автопарку
    res_13 = sorted(many_to_many, key = itemgetter(2))
    print(res_13)

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

Пример выполнения.

```
nop@nopc:~/Projects/bmstu_3sem/BKIT_2022$ python3.8 main.py
Задание 1
[('Артамонов', 25000, 'Автопарк "Такси"')]

Задание 2
[('Автопарк "Такси"', 25000), ('Автопарк "Бумер"', 25000), ('Автопарк "Драйв"', 35000)]

Задание 3
[('Иваненко', 45000, 'Автопарк "Бумер"'), ('Иванов', 35000, 'Автопарк "Бумер"'), ('Иванин', 25000, 'Автоп
арк "Бумер"'), ('Иваненко', 45000, 'Автопарк "Бумер" (другой)'), ('Иванов', 35000, 'Автопарк "Бумер" (дру
гой)'), ('Иванин', 25000, 'Автопарк "Бумер" (другой)'), ('Петров', 35000, 'Автопарк "Драйв"'), ('Петров',
 35000, 'Автопарк "Драйв" (другой)'), ('Артамонов', 25000, 'Автопарк "Такси"'), ('Артамонов', 25000, 'Авт
опарк "Такси" (другой)')]
nop@nopc:~/Projects/bmstu_3sem/BKIT_2022$
```