Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет приложений»

Рубежный контроль №1

Вариант №4Б

Выполнил:
студент группы ИУ5-52Б
Ваганов Даниил

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2021 г.

Описание задания:

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).

Вариант Б.

1. «Компьютер» и «Дисплейный класс» связаны соотношением один-ко-многим. Выведите список всех связанных компьютеров и дисплейных классов, отсортированный по компьютерам(установленным процессорам), сортировка по классам произвольная.
2. «Компьютер» и «Дисплейный класс» связаны соотношением один-ко-многим. Выведите список классов с количеством компьютеров в каждом классе, отсортированный по количеству компьютеров.
3. «Компьютер» и «Дисплейный класс» связаны соотношением многие-ко-многим. Выведите список всех процессов, производителя «Intel», и номера классов, в которых они установлены.

Текст программы

```
# используется для сортировки
from operator import itemgetter

#компьютер и дисплейный класс

class Computer:
  """Компьютер"""
  def __init__(self, id, cpu, graphic_card, disp_id):
    self.id = id
    self.cpu = cpu
    self.graphic_card = graphic_card
    self.disp_id = disp_id
```

```
class DispayClass:
  """Дисплейный класс"""
  def __init__(self, id, room_number, square):
    self.id = id
    self.room_number = room_number
    self.square = square

class CompDisp:
  """
    'Компьтеры класса' для реализации
    связи многие-ко-многим
  """
  def __init__(self, disp_id, comp_id):
        self.disp_id = disp_id
        self.comp_id = comp_id

#Компьютеры
comps = [
    Computer(1, 'Intel Core i7', 'GeForce GT710', 1),
    Computer(2, 'Intel Core i5', 'GeForce RTX3080', 1),
    Computer(3, 'Intel Core i4', 'GeForce GTX1660', 1),
    Computer(4, 'AMD Ryzen 5', 'Radeon RX590', 1),
    Computer(5, 'AMD Ryzen 7', 'Radeon RX6900', 2),
    Computer(6, 'AMD Ryzen 9', 'Radeon RX6900', 2),
    Computer(7, 'Intel Core i5', 'GeForce GTX1050', 2),
    Computer(8, 'AMD Ryzen 7', 'Radeon RX6600', 3),
    Computer(9, 'Intel Core i7', 'GeForce GTX1080',3),
    Computer(10, 'Intel Core i5', 'GeForce RTX1080', 3),
]
# Дисплейные классы
disp = [
    DispayClass(1, 'ClRoom-501', 150),
    DispayClass(2, 'ClRoom-306', 140),
    DispayClass(3, 'ClRoom-404', 170),
]

comps_disp = [
    CompDisp(1,1),
    CompDisp(1,2),
    CompDisp(1,3),
    CompDisp(1,4),

    CompDisp(2,5),
    CompDisp(2,6),
    CompDisp(2,7),

    CompDisp(3,8),
    CompDisp(3,9),
    CompDisp(3,10),
]
```

```
def main():
    """Основная функция"""
    # Соединение данных один-ко-многим
    one_to_many = [
      (c.id, c.cpu, c.graphic_card, d.room_number)
        for c in comps
        for d in disp
        if c.disp_id==d.id
    ]
    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.room_number, cd.disp_id, cd.comp_id)
        for d in disp
        for cd in comps_disp
        if d.id==cd.disp_id]

    many_to_many = [(c.cpu, c.graphic_card, d_room)
        for d_room, disp_id, comp_id in many_to_many_temp
        for c in comps if c.id==comp_id]

    print('Задание Б1')
    res1 = sorted(one_to_many, key=itemgetter(1))  #Сортировка по процессорам
    print(res1)

    print('\nЗадание Б2') #Список классов с количеством компьютеров в каждом классе
    res2 = []
    # Перебираем все дисплейные классы
    for d in disp:
        # Список компьютеров класса
        d_comps = list(filter(lambda i: i[3]==d.room_number, one_to_many))
        # Если класс не пустой
        if len(d_comps) > 0:
            res2.append((d.room_number, len(d_comps)))
        res2 = sorted(res2, key=lambda item: item[1], reverse=True)
    print(res2)

    print('\nЗадание Б3') #Список различных процессоры Intel, установленных в
классах
    res3 = {}
    # Перебираем все компьютеры
    for c in comps:
        if 'Intel' in c.cpu:
            # Список классов с Intel'ом
            d_comps = list(filter(lambda i: i[0]==c.cpu, many_to_many))
            # Только номер классов
            d_comps_cpus = [x for _,_,x in d_comps]
            # Добавляем результат в словарь
            # ключ - cpu, значение - номера классов
            res3[c.cpu] = d_comps_cpus
    print(res3)
```

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

Экранная форма с результатом выполнения программы:

```
Задание Б1
[(4, 'AMD Ryzen 5', 'Radeon RX590', 'ClRoom-501'), (5, 'AMD Ryzen 7', 'Radeon RX6900', 'ClRoom-306'), (8, 'AMD Ryzen 7',
'Radeon RX6600', 'ClRoom-404'), (6, 'AMD Ryzen 9', 'Radeon RX6900', 'ClRoom-306'), (3, 'Intel Core i4', 'GeForce GTX1660'
, 'ClRoom-501'), (2, 'Intel Core i5', 'GeForce RTX3080', 'ClRoom-501'), (7, 'Intel Core i5', 'GeForce GTX1050', 'ClRoom-3
06'), (10, 'Intel Core i5', 'GeForce RTX1080', 'ClRoom-404'), (1, 'Intel Core i7', 'GeForce GT710', 'ClRoom-501'), (9, 'I
ntel Core i7', 'GeForce GTX1080', 'ClRoom-404')]
```

```
Задание Б2
[('ClRoom-501', 4), ('ClRoom-306', 3), ('ClRoom-404', 3)]
```

```
Задание Б3
{'Intel Core i7': ['ClRoom-501', 'ClRoom-404'], 'Intel Core i5': ['ClRoom-501', 'ClRoom-306', 'ClRoom-404'], 'Intel Core
i4': ['ClRoom-501']}
```