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»

Курс «Базовые компоненты интернет-технологий»

ОТЧЁТ ПО

РУБЕЖНОМУ КОНТРОЛЮ №2

Выполнил:
студент группы ИУ5-35Б
Коныгина Дарья

Проверил:
преподаватель
Гапанюк Ю.Е.

Москва

2022

Вариант 8, Д

Жесткий диск	Компьютер

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».

Запросы в соответствии с вариантом:

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 «А», и список работающих в них сотрудников.

Адаптированные запросы под заданную предметную область в соответствии с вариантом:

1. «Компьютер» и «жесткий диск» связаны соотношением один-ко-многим. Выведите список всех жестких дисков, у которых название заканчивается на «ZX» и названия их компьютеров.

2. «Компьютер» и «жесткий диск» связаны соотношением один-ко-многим. Выведите список компьютеров со средним объемом памяти (hdd) жестких дисков в каждом компьютере, отсортированный по среднему объему hdd (отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений).
3. «Компьютер» и «жесткий диск» связаны соотношением многие-ко-многим. Выведите список всех компьютеров, у которых название начинается с буквы «m», и список входящих в них жестких дисков.

Текст программы (содержится в нескольких файлах)

Файл 1. classes.py (описание классов предметной области)

```
# класс "компьютер"
class computer:
    def __init__(self, id, name):
        self.id = id
        self.name = name

# класс "жесткий диск"
class hdd:
    def __init__(self, id_hdd, name_hdd, size_hdd, pc_id):
        self.id = id_hdd
        self.name = name_hdd
        self.size = size_hdd
        self.pc_id = pc_id

# класс "жесткие диски компьютера"
class hddPc:
    def __init__(self, id_hdd, id_pc):
        self.id_hdd = id_hdd
        self.id_pc = id_pc
```

Файл 2. create_connection.py (описание функций для реализации связей один-ко-многим и многие-ко-многим)

```
from rk.classes import hdd, computer, hddPc

# Соединение данных один-ко-многим
def one_to_many(hdds, pcs):
    return [(hdd.name, hdd.size, pc.name)
            for pc in pcs
            for hdd in hdds
            if hdd.pc_id == pc.id
    ]


# Соединение данных многие-ко-многим
def many_to_many(hdds, pcs, hddPc):
    return [(hdd.name, hdd.size, pc.name)
            for hp in hddPc
            for pc in pcs
            for hdd in hdds
```

```
            if pc.id == hp.id_pc and hdd.id == hp.id_hdd
    ]
```

Файл 3. tasks.py (описание функций для выполнения запросов)

```
from rk.classes import hdd, computer, hddPc
from rk.create_connection import one_to_many, many_to_many
def task_1(one_to_many):
    return [x for x in one_to_many if x[0][-2:] == 'ZX']

def task_2(one_to_many):
    comps = []
    dsize = []
    count_disk = []
    for x in one_to_many:
        if x[2] not in comps:
            comps.append(x[2])
            dsize.append(x[1])
            count_disk.append(1)
        else:
            dsize[comps.index(x[2])] += x[1]
            count_disk[comps.index(x[2])] += 1
    #av_s = [round(dsize[i]/count_disk[i], 2) for i in range(len(dsize))]
    #return (sorted(list(zip(comps, av_s)), key=lambda x: x[1], reverse=True))

    return (sorted(list(zip(comps, [round(dsize[i]/count_disk[i], 2) for i in
range(len(dsize))])), key=lambda x: x[1], reverse=True))
def task_3(many_to_many):
    res = {}
    for x in many_to_many:
        if x[2][0] == 'm':
            # список дисков в компе
            disks = list(filter(lambda i: i[2] == x[2], many_to_many))
            # только названия жестких дисков
            disks_name = [x for x,_,_  in disks]
            res[x[2]] = disks_name
    return res
```

Файл 4. test_rk2.py (модульное тестирование)

```
import pytest
import unittest
from rk.classes import hdd, computer, hddPc
from rk.create_connection import one_to_many, many_to_many
from rk.tasks import task_1, task_2, task_3

PCs = [
    computer(1, 'm064'),
    computer(2, 'k05i'),
    computer(3, 'mg6_mt')
    ]

HDDs = [
    hdd(1, 'LM_3', 2, 1),
    hdd(2, 'L0ZX', 2, 1),
    hdd(3, 'LPS_X', 0.5, 2)
]

HddPCs=[
    hddPc(1, 1),
    hddPc(2, 1),
    hddPc(3, 2),
```

```
    ]
one_to_many = one_to_many(HDDs, PCs)
many_to_many = many_to_many(HDDs, PCs, HddPCs)


#TDD
def test_task1():
    assert [('L0ZX', 2, 'm064')] == task_1(one_to_many)

def test_task2():
    assert [('m064', 2)] == task_2(one_to_many)

def test_task3():
    assert {'m064': ['LM_3', 'L0ZX']} == task_3(many_to_many)

if __name__ == "__main__":

    unittest.main()
```

Результаты выполнения:

Скриншот: