Отчёт по рубежному контролю №1 по курсу «Базовые компоненты интернет-технологий»

Вариант 8.

Руководитель   
Гапанюк Ю.Е.

Выполнил  
студент группы ИУ5-31Б   
Калинников Даниил  
23.10.2021

МГТУ им. Н.Э. Баумана   
2021 г.

**Задание**

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

* Класс «**Компьютер**», содержащий поля:
* ID записи о студенте;
* Фамилия студента;
* Количество долгов;
* ID записи о группе. (для реализации связи один-ко-многим)
* Класс «**Жесткий диск** », содержащий поля:
* ID записи о группе;
* Наименование группы.
* (Для реализации связи многие-ко-многим) Класс «**Компьютер(2)**», содержащий поля:
* ID записи о студенте;
* ID записи о группе.

2) Необходимо создать списки объектов классов, содержащих тестовые данные, таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

3) Необходимо разработать запросы в соответствии с вариантом. Запросы сформулированы в терминах классов «Студент» и «Группа», которые используются в примере. Нужно перенести эти требования в выданный вариант предметной области.

**Предметная область: Вариант 8 - Жесткий диск - Компьютер**

**Запрос: Вариант А.**

* «**Жесткий диск** » и «**Компьютер**» связаны соотношением один-ко-многим. Выведите список всех связанных студентов и групп, отсортированный по группам, сортировка по студентам произвольная.
* «**Жесткий диск** » и «**Компьютер**» связаны соотношением один-ко-многим. Выведите список групп с суммарным количеством долгов студентов в каждой группе, отсортированный по суммарной количеству долгов.
* «**Жесткий диск** » и «**Компьютер**» связаны соотношением многие-ко-многим. Выведите список всех групп, у которых в названии присутствует индекс «Б», и список учащихся в них студентов.

**Текст программы**

from operator import itemgetter  
  
  
class hd:  
  
 def \_\_init\_\_(self, id, gb, s, pc\_id):  
 self.id = id  
 self.gb = gb  
 self.s = s  
 self.pc\_id = pc\_id  
  
  
class Pc:  
  
 def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class EmpDep:  
  
  
 def \_\_init\_\_(self, pc\_id, hd\_id):  
 self.pc\_id = pc\_id  
 self.hd\_id = hd\_id  
  
  
pcs = [  
 Pc(1, 'PATA'),  
 Pc(2, 'SATA'),  
 Pc(3, 'SSD'),  
  
 Pc(11, 'PATA тип 1'),  
 Pc(22, 'SSD тип 1'),  
 Pc(33, 'SATA тип 3'),  
]  
  
hds = [  
 hd(1, '100 gb', 100, 1),  
 hd(2, '200 gb', 120, 2),  
 hd(3, '4000 gb', 101, 3),  
 hd(4, '1720 gb', 110, 2),  
 hd(5, '1500 gb', 50, 3),  
]  
  
emps\_pcs = [  
 EmpDep(1, 1),  
 EmpDep(2, 2),  
 EmpDep(3, 3),  
 EmpDep(3, 4),  
 EmpDep(3, 5),  
  
 EmpDep(11, 1),  
 EmpDep(22, 2),  
 EmpDep(33, 3),  
 EmpDep(33, 4),  
 EmpDep(33, 5),  
]  
  
  
def main():  
 one\_to\_many = [(e.gb, e.s, d.name)  
 for d in pcs  
 for e in hds  
 if e.pc\_id == d.id]  
   
 many\_to\_many\_temp = [(d.name, ed.pc\_id, ed.hd\_id)  
 for d in pcs  
 for ed in emps\_pcs  
 if d.id == ed.pc\_id]  
  
 many\_to\_many = [(e.gb, e.s, pc\_name)  
 for pc\_name, pc\_id, hd\_id in many\_to\_many\_temp  
 for e in hds if e.id == hd\_id]  
  
 print('Задание А1')  
 res\_11 = sorted(one\_to\_many, key=itemgetter(2))  
 print(res\_11)  
  
 print('\nЗадание А2')  
 res\_12\_unsorted = []  
 for d in pcs:  
 d\_hds = list(filter(lambda i: i[2] == d.name, one\_to\_many))  
 if len(d\_hds) > 0:  
 d\_s = [s for \_, s, \_ in d\_hds]  
 d\_s\_sum = sum(d\_s)  
 res\_12\_unsorted.append((d.name, d\_s\_sum))  
  
 res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)  
 print(res\_12)  
  
 print('\nЗадание А3')  
 res\_13 = {}  
 for d in pcs:  
 if 'SSD' in d.name:  
 d\_hds = list(filter(lambda i: i[2] == d.name, many\_to\_many))  
 d\_hds\_names = [x for x, \_, \_ in d\_hds]  
 res\_13[d.name] = d\_hds\_names  
  
 print(res\_13)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результат выполнения программы**

