|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки 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)** |

**Отчет по рубежному контролю № 1 по курсу**

**Разработка интернет-приложений**

**2 вариант**

Студентка группы РТ5-51Б

Бочарова Маргарита

**Задание:**

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).

**Вариант Е.**

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «отдел», и список работающих в них сотрудников.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате. Средняя зарплата должна быть округлена до 2 знака после запятой. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.

|  |  |  |
| --- | --- | --- |
| № | Класс\_1 | Класс\_2 |
| 2 | Школьник | Класс |

# используется для сортировки

from operator import itemgetter

class Schoolchild:

"Школьники"

def \_\_init\_\_(self, id, fio, averege\_mark, class\_id):

self.id = id

self.fio = fio

self.averege\_mark = averege\_mark

self.class\_id = class\_id

class School\_Class:

"Школьный класс"

def \_\_init\_\_(self, id, number, letter):

self.id = id

self.number = number

self.letter=letter

class Schoolchild\_\_School\_Class:

"""Связь многие-ко-многим Школьник-Класс"""

def \_\_init\_\_(self, Schoolchild\_id,School\_Class\_id):

self.Schoolchild\_id = Schoolchild\_id

self.School\_Class\_id = School\_Class\_id

Schoolchild\_list = [

Schoolchild(1, "Давыдов Алексей Иванович", 5, 3),

Schoolchild(2, "Степанова Анастасия Александровна", 3, 4),

Schoolchild(3, "Карелин Владислав Вячеславович", 4.6, 4),

Schoolchild(4, "Зубина Алина Владимировна", 3.8, 1),

Schoolchild(5, "Уменко Михаил Викторович", 2.9, 3),

Schoolchild(6, "Решётникова Марина Витальевна", 4, 2),

Schoolchild(7, "Бережков Кирилл Кириллович", 5, 2),

Schoolchild(8, "Ромашина Юлия Николаевна", 5, 1)

]

listSchool\_Classes = [

School\_Class(1, 10, "A"),

School\_Class(2, 11, "A"),

School\_Class(3, 10, "Б"),

School\_Class(4, 11, "Б")

]

listSchoolchild\_SchoolClass = [

Schoolchild\_\_School\_Class(3, 3),

Schoolchild\_\_School\_Class(8, 2),

Schoolchild\_\_School\_Class(4, 1),

Schoolchild\_\_School\_Class(7, 3),

Schoolchild\_\_School\_Class(2, 3),

Schoolchild\_\_School\_Class(1, 1),

Schoolchild\_\_School\_Class(6, 4),

Schoolchild\_\_School\_Class(5, 4),

]

# Соединение данных один-ко-многим

one\_to\_many = list((School\_Class, Schoolchild)

for Schoolchild in Schoolchild\_list

for School\_Class in listSchool\_Classes

if (Schoolchild.class\_id==School\_Class.id))

many\_to\_many\_temp = list((p.number, p.letter, d.School\_Class\_id, d.Schoolchild\_id)

for p in listSchool\_Classes

for d in listSchoolchild\_SchoolClass

if (p.id==d.School\_Class\_id))

many\_to\_many = list((e.fio, e.class\_id, number, letter, Schoolchild\_id)

for number, letter,School\_Class\_id,Schoolchild\_id in many\_to\_many\_temp

for e in Schoolchild\_list

if (e.id==Schoolchild\_id))

print('Задание E1')

for i in one\_to\_many:

if "A" in i[0].letter:

print(i[1].fio)

print('Задание E2')

Marks\_Avg\_List = list()

for i in listSchool\_Classes:

Schoolchild\_list=list(filter(lambda x: x[0].id==i.id, one\_to\_many))

Marks\_Avg=0

for item in Schoolchild\_list:

Schoolchild=item[1]

Marks\_Avg+=Schoolchild.averege\_mark

Marks\_Avg=round(Marks\_Avg/len(Schoolchild\_list),2)

Marks\_Avg\_List.append((i.number, i.letter, Marks\_Avg))

for item in sorted(Marks\_Avg\_List, key=lambda x: x[0]):

print("Средняя оценка в классе: ", item[0], item[1], "составляет", item[2])

print('Задание E3')

m=''

for i in many\_to\_many:

str\_1=i[0]

for j in range(len(str\_1)):

if j==0 and str\_1[j]=='Р':

m=m+str\_1+', class\_id='+str(i[1])+'\n'

#print(m)

break

else:

break

print(m)

Выполнение программы

