Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю № 1
Вариант запросов: E
Вариант предметной области: 14

Выполнил:
студент группы РТ5-31Б
Романов Ю.И.
Подпись и дата:

Проверил:
к.т.н., доц. каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2025 г.

Описание задания

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».

Результатом рубежного контроля является документ в формате PDF, который содержит текст программы и результаты ее выполнения.

Текст программы

main.py:

```
from operator import itemgetter

# CD-диск
class CD:
    def __init__(self, id, title, capacity, lib_id):
        self.id = id
        self.title = title
        self.capacity = capacity    # объем в МБ
        self.lib_id = lib_id

# Библиотека CD-дисков
class CDLibrary:
    def __init__(self, id, name):
        self.id = id
        self.name = name

# CD-диски в библиотеках для реализации связи многие-ко-многим
class CD_CDLibrary:
    def __init__(self, lib_id, cd_id):
        self.lib_id = lib_id
        self.cd_id = cd_id

# Библиотеки CD-дисков
libraries = [
    CDLibrary(1, 'библиотека мультимедиа'),
    CDLibrary(2, 'архив программного обеспечения'),
    CDLibrary(3, 'библиотека фильмов'),
    CDLibrary(4, 'основной архив'),
    CDLibrary(5, 'коллекция литературы'),
    CDLibrary(6, 'библиотека игр'),
]
```

```
# CD-диски
cds = [
    CD(11, 'Альбом Rock', 700, 1),
    CD(22, 'Мастер и Маргарита', 650, 5),
    CD(33, 'Фильм "Гарри Поттер и Орден Феникса"', 900, 3),
    CD(44, 'Minecraft', 700, 6),
    CD(55, 'Antivirus Pro', 500, 2),
    CD(66, 'Microsoft Office', 650, 4),
]

# Связи многие-ко-многим
cds_libraries = [
    CD_CDLibrary(1, 11),
    CD_CDLibrary(1, 22),
    CD_CDLibrary(1, 33),
    CD_CDLibrary(2, 44),
    CD_CDLibrary(2, 55),
    CD_CDLibrary(2, 66),
    CD_CDLibrary(3, 33),
    CD_CDLibrary(4, 55),
    CD_CDLibrary(4, 66),
    CD_CDLibrary(5, 22),
    CD_CDLibrary(6, 44)
]

def main():
    # Соединение данных один-ко-многим
    one_to_many = [(cd.title, cd.capacity, lib.name)
                   for lib in libraries
                   for cd in cds
                   if cd.lib_id == lib.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(lib.name, cl.lib_id, cl.cd_id)
                         for lib in libraries
                         for cl in cds_libraries
                         if lib.id == cl.lib_id]

    many_to_many = [(cd.title, cd.capacity, lib_name)
                    for lib_name, _, cd_id in many_to_many_temp
                    for cd in cds if cd.id == cd_id]

    print('Задание E1')
    # Список всех библиотек, у которых в названии присутствует слово
"библиотека", и список CD-дисков в них
```

```
    res_e1 = {}
    for lib in libraries:
        if 'библиотека' in lib.name:
            lib_cds = list(filter(lambda i: i[2] == lib.name, one_to_many))
            lib_cds_titles = [x for x, _, _ in lib_cds]
            res_e1[lib.name] = lib_cds_titles

    print(res_e1)

    print('\nЗадание E2')
    # Список библиотек со средней емкостью CD-дисков в каждой библиотеке, отсортированный по средней емкости
    res_e2_unsorted = []
    for lib in libraries:
        lib_cds = list(filter(lambda i: i[2] == lib.name, one_to_many))
        if len(lib_cds) > 0:
            lib_capacities = [capacity for _, capacity, _ in lib_cds]
            lib_avg_capacity = round(sum(lib_capacities) / len(lib_capacities), 2)
            res_e2_unsorted.append((lib.name, lib_avg_capacity))

    # Сортировка по средней емкости
    res_e2 = sorted(res_e2_unsorted, key=itemgetter(1))
    print(res_e2)

    print('\nЗадание E3')
    # Список всех CD-дисков, у которых название начинается с буквы «Ф», и названия их библиотек
    res_e3 = list(filter(lambda i: i[0].startswith('Ф'), many_to_many))
    # Сортировка по названию CD-диска
    res_e3_sorted = sorted(res_e3, key=itemgetter(0))
    print(res_e3_sorted)

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

Экранные формы с примерами выполнения программы