МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по рубежному контролю №1

Выполнил:
Студент группы ИУ5Ц-54Б
Цурин А.П.
Преподаватель:
Гапанюк Ю.Е.

Москва 2025

1. Задания для выполнения

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».
Мой вариант:
Вариант запросов Д.
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.
Варианты предметной области:
Класс 1 - Строка данных, Класс 2 - Таблица данных

2. Листинг программы

```
from operator import itemgetter

class DataRow:
    """Строка данных"""
    def __init__(self, id, name, value, table_id):
        self.id = id
        self.name = name  # название строки
        self.value = value  # числовое значение (аналог зарплаты)
        self.table_id = table_id

class DataTable:
```

```
    """Таблица данных"""
    def __init__(self, id, name):
        self.id = id
        self.name = name  # название таблицы

class RowTable:
    """Строки таблиц для связи многие-ко-многим"""
    def __init__(self, row_id, table_id):
        self.row_id = row_id
        self.table_id = table_id

# Таблицы данных
tables = [
    DataTable(1, "Анализ продаж"),
    DataTable(2, "Бюджет"),
    DataTable(3, "Аудит качества"),
    DataTable(4, "Отчетность"),
    DataTable(5, "Анализ рисков")
]

# Строки данных
rows = [
    DataRow(1, "Иванов", 50000, 1),
    DataRow(2, "Петров", 45000, 1),
    DataRow(3, "Сидоров", 30000, 2),
    DataRow(4, "Кузнецов", 60000, 3),
    DataRow(5, "Александров", 35000, 3),
    DataRow(6, "Сергеев", 55000, 4),
    DataRow(7, "Антонов", 40000, 5),
    DataRow(8, "Николаев", 48000, 5)
]

# Связь многие-ко-многим
row_table = [
    RowTable(1, 1),
    RowTable(2, 1),
    RowTable(3, 2),
    RowTable(4, 3),
    RowTable(5, 3),
    RowTable(6, 4),
    RowTable(7, 5),
    RowTable(8, 5),
    RowTable(1, 3),  # дополнительная связь для многих-ко-многим
    RowTable(4, 5)   # дополнительная связь для многих-ко-многим
]

def main():

    # Соединение данных один-ко-многим
    one_to_many = [(r.name, r.value, t.name)
                   for t in tables
                   for r in rows
                   if r.table_id == t.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(t.name, rt.table_id, rt.row_id)
                         for t in tables
                         for rt in row_table
                         if t.id == rt.table_id]

    many_to_many = [(r.name, r.value, table_name)
                    for table_name, table_id, row_id in many_to_many_temp
                    for r in rows if r.id == row_id]

    # Задание 1: Список всех строк, у которых название заканчивается на «ов», и названия
их таблиц
    print("Задание 1: Строки, оканчивающиеся на 'ов', и их таблицы")
```

```
    res_1 = [(row_name, table_name) for row_name, _, table_name in one_to_many if
row_name.endswith("ов")]
    for row_name, table_name in res_1:
        print(f"Строка: {row_name}, Таблица: {table_name}")
    print()

    # Задание 2: Список таблиц со средним значением строк в каждой таблице,
отсортированный по среднему значению
    print("Задание 2: Таблицы со средним значением строк (отсортировано)")

    # Создаем словарь для хранения сумм и количеств
    table_stats = {}
    for row_name, value, table_name in one_to_many:
        if table_name not in table_stats:
            table_stats[table_name] = {'sum': 0, 'count': 0}
        table_stats[table_name]['sum'] += value
        table_stats[table_name]['count'] += 1

    # Вычисляем средние значения
    res_2_unsorted = []
    for table_name, stats in table_stats.items():
        avg_value = stats['sum'] / stats['count']
        res_2_unsorted.append((table_name, avg_value))

    # Сортируем по среднему значению
    res_2 = sorted(res_2_unsorted, key=itemgetter(1))

    for table_name, avg_value in res_2:
        print(f"Таблица: {table_name}, Среднее значение: {avg_value:.2f}")
    print()

    # Задание 3: Список всех таблиц, у которых название начинается с буквы «А», и список
строк в них
    print("Задание 3: Таблицы, начинающиеся на 'A', и их строки")

    # Фильтруем таблицы, начинающиеся на "A"
    a_table_names = [t.name for t in tables if t.name.startswith("A")]

    # Для каждой такой таблицы находим связанные строки через связь многие-ко-многим
    res_3 = {}
    for table_name in a_table_names:
        # Фильтруем строки для текущей таблицы
        table_rows = list(filter(lambda x: x[2] == table_name, many_to_many))
        # Убираем дубликаты по имени строки
        unique_rows = []
        seen_names = set()
        for row in table_rows:
            row_name, row_value, _ = row
            if row_name not in seen_names:
                unique_rows.append((row_name, row_value))
                seen_names.add(row_name)
        res_3[table_name] = unique_rows

    for table_name, rows_list in res_3.items():
        print(f"\nТаблица: {table_name}")
        for row_name, row_value in rows_list:
            print(f"  - {row_name}: {row_value}")

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

3. Результаты работы программы

Задание 1: Строки, оканчивающиеся на 'ов', и их таблицы
Строка: Иванов, Таблица: Анализ продаж
Строка: Петров, Таблица: Анализ продаж
Строка: Сидоров, Таблица: Бюджет

Строка: Кузнецов, Таблица: Аудит качества
Строка: Александров, Таблица: Аудит качества
Строка: Антонов, Таблица: Анализ рисков

Задание 2: Таблицы со средним значением строк (отсортировано)
Таблица: Бюджет, Среднее значение: 30000.00
Таблица: Анализ рисков, Среднее значение: 44000.00
Таблица: Анализ продаж, Среднее значение: 47500.00
Таблица: Аудит качества, Среднее значение: 47500.00
Таблица: Отчетность, Среднее значение: 55000.00

Задание 3: Таблицы, начинающиеся на 'А', и их строки

Таблица: Анализ продаж
- Иванов: 50000
- Петров: 45000

Таблица: Аудит качества
- Кузнецов: 60000
- Александров: 35000
- Иванов: 50000

Таблица: Анализ рисков
- Антонов: 40000
- Николаев: 48000
- Кузнецов: 60000