МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Отчёт по рубежному контролю №1 по курсу Парадигмы и конструкции языков программирования ГУИМЦ

Исполнитель: **Тошниёзов Жавохир**
студент группы
ИУ5Ц-54Б

Преподаватель: **Гапанюк Ю. Е.**

Москва, МГТУ – 2025

Условия рубежного контроля №1 по курсу ПиК ЯП

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;

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.

Варианты запросов

Мой вариант 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.

Мой вариант предметной области:

№ варианта	Класс 1	Класс 2
32	Колонка данных	Таблица данных

Текст программы:

```
class DataColumn:
    """Класс Колонка данных (аналог Сотрудника)"""
    def __init__(self, column_id, name, data_size, table_id):
        self.column_id = column_id
        self.name = name
        self.data_size = data_size  # количественный признак (размер данных в МБ)
        self.table_id = table_id

    def __repr__(self):
        return f"DataColumn({self.column_id}, '{self.name}', {self.data_size},
{self.table_id})"

class DataTable:
    """Класс Таблица данных (аналог Отдела)"""
    def __init__(self, table_id, table_name):
        self.table_id = table_id
        self.table_name = table_name

    def __repr__(self):
        return f"DataTable({self.table_id}, '{self.table_name}')"
```

```
class ColumnTable:
    """Класс для связи многие-ко-многим"""
    def __init__(self, column_id, table_id):
        self.column_id = column_id
        self.table_id = table_id

    def __repr__(self):
        return f"ColumnTable({self.column_id}, {self.table_id})"
```

```
# Создание тестовых данных
def create_test_data():
    # Таблицы данных
    tables = [
        DataTable(1, "Users"),
        DataTable(2, "Products"),
        DataTable(3, "Аналитика"),
        DataTable(4, "Orders"),
        DataTable(5, "Accounts")
    ]

    # Колонки данных (связь один-ко-многим)
    columns = [
        DataColumn(1, "user_id", 50, 1),
        DataColumn(2, "username", 30, 1),
        DataColumn(3, "user_email", 40, 1),
        DataColumn(4, "product_id", 60, 2),
        DataColumn(5, "product_name", 25, 2),
```

```
        DataColumn(6, "price", 15, 2),
        DataColumn(7, "sale_date", 20, 4),
        DataColumn(8, "amount", 10, 4),
        DataColumn(9, "account_id", 35, 5),
        DataColumn(10, "balance", 12, 5),
        DataColumn(11, "analytics_id", 45, 3),
        DataColumn(12, "report_name", 22, 3)
    ]

    # Связи многие-ко-многим
    column_tables = [
        ColumnTable(1, 1),  # user_id в Users
        ColumnTable(2, 1),  # username в Users
        ColumnTable(3, 1),  # user_email в Users
        ColumnTable(4, 2),  # product_id в Products
        ColumnTable(5, 2),  # product_name в Products
        ColumnTable(6, 2),  # price в Products
        ColumnTable(7, 4),  # sale_date в Orders
        ColumnTable(8, 4),  # amount в Orders
        ColumnTable(9, 5),  # account_id в Accounts
        ColumnTable(10, 5), # balance в Accounts
        ColumnTable(11, 3), # analytics_id в Аналитика
        ColumnTable(12, 3), # report_name в Аналитика
        # Дополнительные связи для многие-ко-многим
        ColumnTable(1, 3),  # user_id также в Аналитика
        ColumnTable(4, 3),  # product_id также в Аналитика
        ColumnTable(7, 3),  # sale_date также в Аналитика
    ]

    return tables, columns, column_tables
```

```
# Запрос 1: Список всех колонок, у которых название заканчивается на «id», и
названия их таблиц
def query_1(tables, columns):
    """Колонки с названием, оканчивающимся на 'id' и их таблицы"""
    result = []
    for column in columns:
        if column.name.lower().endswith('id'):
            table = next((t for t in tables if t.table_id == column.table_id),
None)
            if table:
                result.append((column.name, table.table_name))

    # Альтернативная реализация с list comprehension
    result_lc = [
        (column.name, next(t.table_name for t in tables if t.table_id ==
column.table_id))
        for column in columns
        if column.name.lower().endswith('id')
    ]
```

```
    return result_lc
```

```
# Запрос 2: Список таблиц со средним размером данных колонок в каждой таблице
def query_2(tables, columns):
    """Средний размер данных колонок по таблицам"""
    # Группируем колонки по table_id и вычисляем средний размер
    table_stats = {}

    for column in columns:
        if column.table_id not in table_stats:
            table_stats[column.table_id] = {'total_size': 0, 'count': 0}
        table_stats[column.table_id]['total_size'] += column.data_size
        table_stats[column.table_id]['count'] += 1

    # Вычисляем среднее значение для каждой таблицы
    result = []
    for table_id, stats in table_stats.items():
        table = next(t for t in tables if t.table_id == table_id)
        avg_size = stats['total_size'] / stats['count']
        result.append((table.table_name, avg_size))

    # Сортировка по среднему размеру (по возрастанию)
    result.sort(key=lambda x: x[1])

    return result
```

```
# Запрос 3: Список всех таблиц, у которых название начинается с буквы «А», и
список колонок в них
def query_3(tables, columns, column_tables):
    """Таблицы с названием на 'A' и их колонки (связь многие-ко-многим)"""
    # Находим таблицы, начинающиеся на 'A' (кириллица)
    a_tables = [t for t in tables if t.table_name.startswith('A')]

    result = []
    for table in a_tables:
        # Находим все column_id для этой таблицы из связи многие-ко-многим
        table_column_ids = [ct.column_id for ct in column_tables if ct.table_id
== table.table_id]

        # Находим колонки по их ID
        table_columns = [col for col in columns if col.column_id in
table_column_ids]

        result.append({
            'table': table.table_name,
            'columns': [col.name for col in table_columns]
        })

    return result
```

```python
# Главная функция
def main():
    print("=== РУБЕЖНЫЙ КОНТРОЛЬ №1 ===")
    print("Вариант 32: Колонка данных - Таблица данных")
    print()

    # Создание тестовых данных
    tables, columns, column_tables = create_test_data()

    print("\n" + "="*50)
    print("ЗАПРОС 1: Колонки с названием, оканчивающимся на 'id', и их таблицы")
    result1 = query_1(tables, columns)
    for col_name, table_name in result1:
        print(f"  Колонка: {col_name}, Таблица: {table_name}")

    print("\n" + "="*50)
    print("ЗАПРОС 2: Таблицы со средним размером данных колонок")
    result2 = query_2(tables, columns)
    for table_name, avg_size in result2:
        print(f"  Таблица: {table_name}, Средний размер: {avg_size:.2f} МБ")

    print("\n" + "="*50)
    print("ЗАПРОС 3: Таблицы с названием на 'A' и их колонки (связь многие-ко-
многим)")
    result3 = query_3(tables, columns, column_tables)
    for item in result3:
        print(f"  Таблица: {item['table']}")
        print(f"    Колонки: {', '.join(item['columns'])}")
```

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

Результат выполнения программы:

=== РУБЕЖНЫЙ КОНТРОЛЬ №1 ===

Вариант 32: Колонка данных - Таблица данных

==========

ЗАПРОС 1: Колонки с названием, оканчивающимся на 'id', и их таблицы

Колонка: user_id, Таблица: Users

Колонка: product_id, Таблица: Products

Колонка: account_id, Таблица: Accounts

Колонка: analytics_id, Таблица: Аналитика

==========

ЗАПРОС 2: Таблицы со средним размером данных колонок

Таблица: Orders, Средний размер: 15.00 МБ

Таблица: Accounts, Средний размер: 23.50 МБ

Таблица: Products, Средний размер: 33.33 МБ

Таблица: Аналитика, Средний размер: 33.50 МБ

Таблица: Users, Средний размер: 40.00 МБ

==========

ЗАПРОС 3: Таблицы с названием на 'A' и их колонки (связь многие-ко-многим)

Таблица: Аналитика

Колонки: user_id, product_id, sale_date, analytics_id, report_name