Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика, искусственный интеллект и системы управления»

Кафедра «Системы обработки информации и управления»

Рубежный контроль 2
по дисциплине «Парадигмы и конструкции языков программирования»

Вариант из РК №1
Вариант: 32
Вариант запросов: Б

Выполнил:
студент ИУ5-33б Косарев А.А.

2025 г.

Код решения на Python:

- database.py

```
from operator import itemgetter

class Column:
    def __init__(self, id: int, id_table: int, name: str, type: str):
        self.id = id
        self.id_table = id_table
        self.name = name
        self.type = type

class Table:
    def __init__(self, id: int, name: str):
        self.id = id
        self.name = name

class TablesToColumns:
    def __init__(self, table_id: int, column_id: int):
        self.table_id = table_id
        self.column_id = column_id

def get_sorted_columns_by_table(tables: list[Table], columns: list[Column]) -
> list[tuple]:
    one_to_many = [
        (tbl.id, tbl.name, col.id, col.name)
        for tbl in tables
        for col in columns
        if tbl.id == col.id_table
    ]
    return sorted(one_to_many, key=itemgetter(2))

def get_table_column_counts(tables: list[Table], columns: list[Column]) ->
list[tuple]:
    column_count_dict = {
        tbl.id: len([col for col in columns if col.id_table == tbl.id])
        for tbl in tables
    }
    sorted counts = sorted(column count dict.items(), key=lambda item:
item[1])
    return sorted_counts

def get_columns_ending_with_ov_by_table(tables: list[Table],
                                        columns: list[Column],
                                        tables_to_columns:
list[TablesToColumns]) -> dict:
    many_to_many_temp = [
        (ttc.table_id, ttc.column_id)
        for ttc in tables_to_columns
    ]

    many_to_many = [
        (tbl.id, tbl.name, col.id, col.name)
        for table_id, column_id in many_to_many_temp
        for tbl in tables if tbl.id == table_id
        for col in columns if col.id == column_id
    ]

    result = {}
    for table_id, table_name, _, col_name in many_to_many:
        if col_name.endswith('ов'):
            result.setdefault(table_name, []).append(col_name)
```

```
    return result

if __name__ == "__main__":
    table_1 = Table(1, "Таблица 1")
    table_2 = Table(2, "Таблица 2")
    table_3 = Table(3, "Таблица 3")
    table_4 = Table(4, "Таблица 4")

    column_1 = Column(1, 3, "Колонка сомов", "Целое")
    column_2 = Column(2, 2, "Колонка кружков", "Целое")
    column_3 = Column(3, 2, "Колонка 3", "Целое")
    column_4 = Column(4, 1, "Колонка крючков", "Целое")
    column_5 = Column(5, 1, "Колонка завозов", "Целое")
    column_6 = Column(6, 1, "Колонка 6", "Целое")

    tab_to_col_1 = TablesToColumns(3, 1)
    tab_to_col_2 = TablesToColumns(2, 2)
    tab_to_col_3 = TablesToColumns(2, 3)
    tab_to_col_4 = TablesToColumns(1, 4)
    tab_to_col_5 = TablesToColumns(1, 5)
    tab_to_col_6 = TablesToColumns(1, 6)

    tables = [table_1, table_2, table_3, table_4]
    columns = [column_3, column_2, column_1, column_6, column_5, column_4]
    tab_to_col = [tab_to_col_3, tab_to_col_2, tab_to_col_1,
                  tab_to_col_6, tab_to_col_5, tab_to_col_4]

    print("Задание 1 (отсортировано по ID колонки):")
    for result in get_sorted_columns_by_table(tables, columns):
        print(f"  Таблица ID:{result[0]} ('{result[1]}') -> "
              f"Колонка ID:{result[2]} ('{result[3]}')")

    print("\nЗадание 2 (отсортировано по количеству колонок):")
    for table_id, count in get_table_column_counts(tables, columns):
        table_name = next(tbl.name for tbl in tables if tbl.id == table_id)
        print(f"  Таблица ID:{table_id} ('{table_name}'): {count} колонок")

    print("\nЗадание 3 (колонки на 'ов' по таблицам):")
    result_dict = get_columns_ending_with_ov_by_table(tables, columns,
tab_to_col)
    for table_name, cols in result_dict.items():
        print(f"  Таблица '{table_name}': {cols}")
```

- test_database.py

```
import unittest
from database import (Column, Table, TablesToColumns,
                      get_sorted_columns_by_table,
                      get_table_column_counts,
                      get_columns_ending_with_ov_by_table)

class TestDatabaseQueries(unittest.TestCase):
    def setUp(self):
        self.tables = [
            Table(1, "Основная"),
            Table(2, "Вспомогательная"),
            Table(3, "Архивная")
        ]
        self.columns = [
            Column(101, 1, "Иванов", "Текст"),
            Column(102, 1, "Петров", "Целое"),
            Column(103, 2, "Сидоров", "Дата"),
            Column(104, 3, "Кузнецов", "Текст"),
            Column(105, 3, "Орлов", "Логический")
```

```
            ]
            self.tables_to_columns = [
                TablesToColumns(1, 101),
                TablesToColumns(1, 102),
                TablesToColumns(2, 103),
                TablesToColumns(3, 104),
                TablesToColumns(3, 105),
                TablesToColumns(1, 104),
            ]

        def test_get_sorted_columns_by_table(self):
            result = get_sorted_columns_by_table(self.tables, self.columns)
            expected_first_column_id = 101
            expected_last_column_id = 105
            self.assertEqual(result[0][2], expected_first_column_id)
            self.assertEqual(result[-1][2], expected_last_column_id)
            for tbl_id, _, col_id, _ in result:
                col = next(c for c in self.columns if c.id == col_id)
                self.assertEqual(col.id_table, tbl_id)

        def test_get_table_column_counts(self):
            result = get_table_column_counts(self.tables, self.columns)
            expected_order = [(2, 1), (1, 2), (3, 2)]
            for (expected_id, expected_count), (actual_id, actual_count) in
    zip(expected_order, result):
                self.assertEqual(expected_id, actual_id)
                self.assertEqual(expected_count, actual_count)

        def test_get_columns_ending_with_ov_by_table(self):
            result = get_columns_ending_with_ov_by_table(self.tables,
    self.columns,

    self.tables_to_columns)
            expected_keys = {"Основная", "Архивная"}
            self.assertSetEqual(set(result.keys()), expected_keys)
            self.assertIn("Кузнецов", result["Основная"])
            self.assertIn("Кузнецов", result["Архивная"])
            self.assertIn("Орлов", result["Архивная"])

    if __name__ == '__main__':
        unittest.main(verbosity=2)
```

Результаты выполнения:

```
PS D:\Users\kosar\Documents\GitHub\Kosarev_PCPL_Labs\RK2> python -m unittest test_database.py -v
test_get_columns_ending_with_ov_by_table (test_database.TestDatabaseQueries.test_get_columns_ending_with_ov_by_table)
test_get_sorted_columns_by_table (test_database.TestDatabaseQueries.test_get_sorted_columns_by_table) ... ok
test_get_table_column_counts (test_database.TestDatabaseQueries.test_get_table_column_counts) ... ok
```