Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по РК №2
Вариант запросов Б
Вариант предметной области 33

Выполнил:
студент группы ИУ5-33Б
Левкович Леонид

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2025 г.

Листинг кода

rk1 _pcpl_refactored.py

```python
from __future__ import annotations

from dataclasses import dataclass
from typing import Dict, Iterable, List, Tuple


# ------------------------
# Модели данных
# ------------------------

@dataclass(frozen=True)
class DataRow:
    """Строка данных (одна запись)."""
    id: int
    text: str
    length: int          # количественный признак
    table_id: int        # связь 1—М (строка -> таблица)


@dataclass(frozen=True)
class DataTable:
    """Таблица данных."""
    id: int
    name: str


@dataclass(frozen=True)
class RowTableLink:
    """Связь М—М (строка <-> таблица)."""
    row_id: int
    table_id: int


# ------------------------
# Вспомогательные функции
# ------------------------

def build_table_index(tables: Iterable[DataTable]) -> Dict[int, DataTable]:
    return {t.id: t for t in tables}


def build_row_index(rows: Iterable[DataRow]) -> Dict[int, DataRow]:
    return {r.id: r for r in rows}


# ------------------------
# Запросы (чистые функции)
# ------------------------

def query1_rows_with_tables(rows: List[DataRow], tables: List[DataTable]) ->
List[Tuple[str, str]]:
    """
    Запрос 1 (1—М):
    Список (text строки, name таблицы), сортировка по text.
    """
    table_by_id = build_table_index(tables)

    pairs = [(r.text, table_by_id[r.table_id].name) for r in rows]
    return sorted(pairs, key=lambda x: x[0].lower())


def query2_tables_with_row_counts(rows: List[DataRow], tables: List[DataTable])
-> List[Tuple[str, int]]:
```

```
    """
    Запрос 2 (1–М):
    Список (name таблицы, количество строк), сортировка по количеству.
    Важно: считаем по rows.table_id (родная таблица строки).
    """
    table_by_id = build_table_index(tables)

    counts: Dict[int, int] = {}
    for r in rows:
        counts[r.table_id] = counts.get(r.table_id, 0) + 1

    result = [(table_by_id[table_id].name, cnt) for table_id, cnt in
counts.items()]
    return sorted(result, key=lambda x: x[1])


def query3_rows_ending_with_ov_mm(
    rows: List[DataRow],
    tables: List[DataTable],
    links: List[RowTableLink],
) -> List[Tuple[str, str]]:
    """
    Запрос 3 (М–М):
    Список (text строки, name таблицы) для строк, у которых text заканчивается на
"ов".
    Сортировка: сначала по text, затем по name таблицы.
    """
    row_by_id = build_row_index(rows)
    table_by_id = build_table_index(tables)

    pairs_mm: List[Tuple[str, str]] = [
        (row_by_id[link.row_id].text, table_by_id[link.table_id].name)
        for link in links
        if link.row_id in row_by_id and link.table_id in table_by_id
    ]

    filtered = [(text, table_name) for text, table_name in pairs_mm if
text.endswith("ов")]
    return sorted(filtered, key=lambda x: (x[0].lower(), x[1].lower()))


# -------------------------
# Тестовые данные (как в РК1)
# -------------------------

def get_sample_data() -> tuple[list[DataTable], list[DataRow],
list[RowTableLink]]:
    tables = [
        DataTable(1, "Пользователи"),
        DataTable(2, "Заказы"),
        DataTable(3, "Логи"),
    ]

    rows = [
        DataRow(1, "Иванов", 6, 1),
        DataRow(2, "Петров", 6, 1),
        DataRow(3, "Заказ #1024", 10, 2),
        DataRow(4, "Событие: вход", 13, 3),
        DataRow(5, "Сидоров", 7, 1),
    ]

    links = [
        RowTableLink(1, 1),
        RowTableLink(1, 3),
        RowTableLink(2, 1),
        RowTableLink(3, 2),
        RowTableLink(4, 3),
        RowTableLink(5, 1),
```

```
        RowTableLink(5, 2),
    ]

    return tables, rows, links


# ------------------------
# CLI-запуск (не обязателен для тестов)
# ------------------------

def main() -> None:
    tables, rows, links = get_sample_data()

    print("ЗАПРОС 1:")
    for text, table_name in query1_rows_with_tables(rows, tables):
        print(f"- {text} — {table_name}")
    print()

    print("ЗАПРОС 2:")
    for table_name, cnt in query2_tables_with_row_counts(rows, tables):
        print(f"- {table_name}: {cnt}")
    print()

    print("ЗАПРОС 3:")
    for text, table_name in query3_rows_ending_with_ov_mm(rows, tables, links):
        print(f"- {text} — {table_name}")


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

rk1_pcpl_test.py

```
import unittest

from rk1_pcpl_refactored import (
    DataRow,
    DataTable,
    RowTableLink,
    query1_rows_with_tables,
    query2_tables_with_row_counts,
    query3_rows_ending_with_ov_mm,
)


class TestRK1Queries(unittest.TestCase):
    def setUp(self) -> None:
        # Минимально понятные тестовые данные
        self.tables = [
            DataTable(1, "Пользователи"),
            DataTable(2, "Заказы"),
            DataTable(3, "Логи"),
        ]
        self.rows = [
            DataRow(1, "Иванов", 6, 1),
            DataRow(2, "Петров", 6, 1),
            DataRow(3, "Заказ #1024", 10, 2),
            DataRow(4, "Событие: вход", 13, 3),
            DataRow(5, "Сидоров", 7, 1),
        ]
        self.links = [
            RowTableLink(1, 1),
            RowTableLink(1, 3),
            RowTableLink(2, 1),
            RowTableLink(3, 2),
            RowTableLink(4, 3),
            RowTableLink(5, 1),
            RowTableLink(5, 2),
```

```
        ]

    def test_query1_sorted_by_row_text(self) -> None:
        """
        Запрос 1 должен сортировать по тексту строки (по возрастанию).
        Проверяем только порядок text, не привязываясь к конкретной сортировке
таблиц.
        """
        result = query1_rows_with_tables(self.rows, self.tables)
        texts = [t for t, _ in result]
        self.assertEqual(texts, sorted(texts, key=str.lower))

    def test_query2_counts_and_sorting(self) -> None:
        """
        Запрос 2:
        Пользователи: 3 строки (table_id=1)
        Заказы: 1 строка (table_id=2)
        Логи: 1 строка (table_id=3)
        Отсортировано по количеству: 1, 1, 3
        """
        result = query2_tables_with_row_counts(self.rows, self.tables)
        self.assertEqual(result, [("Заказы", 1), ("Логи", 1), ("Пользователи",
3)])

    def test_query3_endswith_ov_mm(self) -> None:
        """
        Запрос 3 (М–М):
        Должны попасть строки на "ов": Иванов, Петров, Сидоров
        И пары по links:
        Иванов: Пользователи, Логи
        Петров: Пользователи
        Сидоров: Пользователи, Заказы
        """
        result = query3_rows_ending_with_ov_mm(self.rows, self.tables,
self.links)
        expected = [
            ("Иванов", "Логи"),
            ("Иванов", "Пользователи"),
            ("Петров", "Пользователи"),
            ("Сидоров", "Заказы"),
            ("Сидоров", "Пользователи"),
        ]
        self.assertEqual(result, expected)


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

Скриншот работы приложения: