МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №5

по дисциплине: Базы данных
тема: «Организация взаимодействия с базой данных через консольное приложение»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
ст. пр. Панченко Максим Владимирович

Белгород 2024 г.

Лабораторная работа №5

Организация взаимодействия с базой данных через консольное приложение

Вариант 8

Цель работы: получить навыки подключения к различным системам управления базами данных и взаимодействия с ними. Разработать консольное приложение для взаимодействия с базой данных.

В качестве БД и СУБД был выбран Postgres, подключение к нему можно осуществить через библиотеку psycopg2. Для форматированного вывода использовали tabulate, для получения данных для подключения к СУБД из переменных окружения использовали getenv из os.

```
from tabulate import tabulate
from os import getenv
import psycopg2
```

Класс Repository выполняет взаимодействие с базой данных, для получения отчётов будем использовать классы, наследующие Repository, где мы переопределяем запрос для выборки.

```
class Repository:
    def __init__(self, connection, table, name, cols, cols_names):
        self._cursor = connection.cursor()
        self._table = table
        self._name = name
        self._cols = cols
        self._cols_names = cols_names

    def select(self):
        print(f"Таблица: {self._name}")
        self._cursor.execute(f"SELECT {','.join(self._cols)} FROM {self._table};")
        print(tabulate(self._cursor.fetchall(), headers=self._cols_names, tablefmt="orgtbl"))


class NonPayersRepository(Repository):
    def __init__(self, connection, name, cols_names):
        super().__init__(connection, "", name, [], cols_names)

    def select(self):
        print(f"Таблица: {self._name}")
        self._cursor.execute(f'''
SELECT
  resident.snp,
  SUM(payment.payment) AS debt,
  payment.energy_source
FROM
  resident
  INNER JOIN residents_contracts ON residents_contracts.resident_passport_data =
resident.passport_data
  INNER JOIN contract ON residents_contracts.contract_id = contract.id
  INNER JOIN payment ON payment.contract_id = contract.id
WHERE
  payment.paid_date IS NULL
GROUP BY
  resident.passport_data,
  payment.energy_source
ORDER BY
```

```
  debt DESC;
''')
        print(tabulate(self._cursor.fetchall(), headers=self._cols_names, tablefmt="orgtbl"))


class WorkersRatingRepository(Repository):
    def __init__(self, connection, name, cols_names):
        super().__init__(connection, "", name, [], cols_names)

    def select(self):
        print(f"Таблица: {self._name}")
        self._cursor.execute(f'''
SELECT
  worker.inn AS worker_inn,
  COALESCE (t1.completed, 0) as completed,
  (
    1.0 * COALESCE (t1.completed, 0) / t2.total
  ) as rating
FROM
  worker
  LEFT JOIN (
    SELECT
      workers_tasks.worker_inn as worker_inn,
      COUNT(*) as completed
    FROM
      workers_tasks
      INNER JOIN task ON task.id = workers_tasks.task_id
    WHERE
      workers_tasks.worker_inn = worker_inn
      AND task.completed_date IS NOT NULL
      AND task.until_date > '2004-01-01'
      AND task.until_date < '2040-12-12'
    GROUP BY
      workers_tasks.worker_inn
  ) t1 ON t1.worker_inn = worker.inn
  INNER JOIN (
    SELECT
      workers_tasks.worker_inn as worker_inn,
      COUNT(*) as total
    FROM
      workers_tasks
      INNER JOIN task ON task.id = workers_tasks.task_id
    WHERE
      workers_tasks.worker_inn = worker_inn
      AND task.until_date > '2004-01-01'
      AND task.until_date < '2040-12-12'
    GROUP BY
      workers_tasks.worker_inn
  ) t2 ON t2.worker_inn = worker.inn
ORDER BY
  completed desc;
''')
        print(tabulate(self._cursor.fetchall(), headers=self._cols_names, tablefmt="orgtbl"))


class ProfitHousesRepository(Repository):
    def __init__(self, connection, name, cols_names):
        super().__init__(connection, "", name, [], cols_names)

    def select(self):
        print(f"Таблица: {self._name}")
        self._cursor.execute(f'''
SELECT
  home.address as address,
  (
    COALESCE(t1.plus, 0) - COALESCE(t2.minus, 0)
  ) as profit
```

```
FROM
  home
  LEFT JOIN (
    SELECT
      contract.home AS home,
      SUM(payment.payment) as plus
    FROM
      payment
      LEFT JOIN contract ON contract.id = payment.contract_id
    GROUP BY
      contract.home
  ) as t1 ON t1.home = address
  LEFT JOIN (
    SELECT
      task.home AS home,
      SUM(task.payment) as minus
    FROM
      task
    GROUP BY
      task.home
  ) as t2 ON t2.home = address
ORDER BY
  profit;
''')
        print(tabulate(self._cursor.fetchall(), headers=self._cols_names, tablefmt="orgtbl"))
```

В main создадим маппинг пользовательского ввода и репозиториев, на его основе будем выводить команды и вызывать нужный репозиторий. Если репозиторий недоступен, будем сообщать об ошибке. При вводе 0 выходим из цикла:

```
def main():
    connection = psycopg2.connect(database=getenv("DATABASE"),
                                  user=getenv("USERNAME"),
                                  password=getenv("PASSWORD"),
                                  host=getenv("HOST"),
                                  port=int(getenv("PORT")),
                                  options=f"-c search_path={getenv('SCHEMA')}"
                                  )

    repo_mapping = {
        "1": ("Договоры", Repository(connection, "contract", "Договоры",
                                     ["transaction_date", "until_date", "home", "id"],
                                     ["Дата начала", "Дата окончания", "Дом", "Идентификационный
номер"])),
        "2": ("Дома", Repository(connection, "home", "Дома",
                                 ["address", "commisioning", "floors", "index"],
                                 ["Адрес", "Дата введения в эксплуатацию", "Этажность",
"Индекс"])),
        "3": ("Чеки", Repository(connection, "payment", "Чеки",
                                 ["id", "paid_date", "until_date", "contract_id",
"energy_source", "payment"],
                                 ["УИП", "Дата оплаты", "Срок оплаты", "Ид. ном. договора",
"Энергетический ресурс",
                                  "Сумма"])),
        "4": ("Жильцы", Repository(connection, "resident", "Жильцы",
                                   ["passport_data", "snp", "email", "phone"],
                                   ["Паспортные данные", "ФИО", "Электронная почта", "Номер
телефона"])),
        "5": ("Договоры жильцов", Repository(connection, "residents_contracts", "Договоры
жильцов",
                                             ["resident_passport_data", "contract_id"],
                                             ["Паспортные данные", "Ид. ном. договора"])),
        "6": ("Работы", Repository(connection, "task", "Работы",
                                   ["id", "completed_date", "until_date", "home"],
                                   ["Идентификационный номер", "Дата окончания", "Дедлайн",
"Дом"])),
```

```
        "7": ("Исполнители работ", Repository(connection, "worker", "Исполнители работ",
                                              ["inn", "email", "phone"],
                                              ["ИНН", "Электронная почта", "Номер телефона"])),
        "8": ("Назначения работ", Repository(connection, "workers_tasks", "Назначения работ",
                                             ["worker_inn", "task_id"],
                                             ["ИНН исполнителя", "Ид. ном. работы"])),
        "9": ("Жильцы-неплательщики", NonPayersRepository(connection, "Жильцы-неплательщики",
                                                          ["ФИО", "Долг", "Энергетический
ресурс"])),
        "10": ("Рейтинг рабочих", WorkersRatingRepository(connection, "Рейтинг рабочих",
                                                          ["ИНН Рабочего", "Завершено работ",
"Рейтинг"])),
        "11": ("Прибыль домов", ProfitHousesRepository(connection, "Прибыль домов",
                                                       ["Адрес", "Прибыль"])),

    }

    repo_selects = dict()
    for key, val in repo_mapping.items():
        repo_selects[key] = val[0]

    repo_selects["0"] = "Выход"

    print("=" * 20)
    for key, val in repo_selects.items():
        print(key, val, sep=". ")

    input_value = input("Выберите команду: ")
    print("=" * 20)

    while input_value != "0":
        if input_value in repo_mapping:
            repo_mapping[input_value][1].select()
        else:
            print("Команда не распознана")

        print("=" * 20)
        for key, val in repo_selects.items():
            print(key, val, sep=". ")

        input_value = input("Выберите команду: ")
        print("=" * 20)
```

Результаты выполнения программы:

```
====================
1. Договоры
2. Дома
3. Чеки
4. Жильцы
5. Договоры жильцов
6. Работы
7. Исполнители работ
8. Назначения работ
9. Жильцы-неплательщики
10. Рейтинг рабочих
11. Прибыль домов
0. Выход
Выберите команду: 12321
====================
Команда не распознана
```

```
====================
1. Договоры
2. Дома
3. Чеки
4. Жильцы
5. Договоры жильцов
6. Работы
7. Исполнители работ
8. Назначения работ
9. Жильцы-неплательщики
10. Рейтинг рабочих
11. Прибыль домов
0. Выход
Выберите команду: 10
====================
Таблица: Рейтинг рабочих
|   ИНН Рабочего |   Завершено работ |   Рейтинг |
|----------------+-------------------+-----------|
|   789124114908 |                 1 |       0.5 |
|   182736471829 |                 0 |       0   |
|   918274765621 |                 0 |       0   |
|   928374635461 |                 0 |       0   |
```

```
====================
1. Договоры
2. Дома
3. Чеки
4. Жильцы
5. Договоры жильцов
6. Работы
7. Исполнители работ
8. Назначения работ
9. Жильцы-неплательщики
10. Рейтинг рабочих
11. Прибыль домов
0. Выход
Выберите команду: 3
====================
Таблица: Чеки
|                       УИП | Дата оплаты   | Срок оплаты   |   Ид. ном. договора | Энергетический ресурс                |   Сумма |
|---------------------------+---------------+---------------+---------------------+--------------------------------------+---------|
| 600002182736475819283750 0 |               | 2024-12-20    |                   3 | Were ЖКХ! Were doing what we want to! |     113 |
| 600002182736475819283750 1 |               | 2024-01-19    |                   3 | Were ЖКХ! Were doing what we want to! |    2132 |
| 600002182736475819283749 0 |               | 2024-12-19    |                   1 | Were ЖКХ! Were doing what we want to! |     213 |
| 600002182736475819283747 0 |               | 2024-01-19    |                   1 | Лифт                                 |      31 |
| 600002182736475819283748 0 |               | 2024-12-19    |                   1 | Абыр                                 |      22 |
```

Вывод: в ходе лабораторной работы получили навыки подключения к различным системам управления базами данных и взаимодействия с ними. Разработали консольное приложение для взаимодействия с базой данных.