Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 2

з дисципліни «Бази даних і засоби управління»
«Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»

Виконав:
студент групи КП-81
Бухаленков Дмитро Олександрович
Перевірив: Радченко К.О.

Київ 2020

Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.

2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.

3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.

4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

Головне меню додатку:

```
Menu:


  1 - level
  2 - player
  3 - skin
  4 - player_skin
  5 - Search players by level name with set health and nickname
  6 - Find skins by player health, last online and level
  7 - Count players who own skin by skin name, level title and health
  8 - Fill table "level" by random data
  9 - Exit
```

Меню таблиці:

```
level


  1 - Get
  2 - Delete
  3 - Update
  4 - Insert
  5 - Back
```

Правильний запит:

```
GET player
Enter search criteria:
id , nickname , last_online , health , level_id

health>105
```

id	nickname	last_online	health	level_id
8	PianoCat	2019-11-23	130	5
9	Olexey228	2019-10-14	175	5
10	LeatFingies	2019-12-20	205	6
11	Waterballs	2019-12-23	164	6

```
Query time: 0.002012968063354492
```

Вставка нових значень через інтерфейс додатку:

```
INSERT player
Enter columns, then values
id , nickname , last_online , health , level_id

nickname, last_online, health, level_id
'test_player', '8.8.2018', 110, 5
```

```
 player

 Insert is successful!
```

11	Waterballs	2019-12-23	164	6
12	test_player	2018-08-08	110	5

Обробка помилок при неправильному введенні даних:

```
GET level
Enter search criteria:
id , title , description , blob

some_random_text
```

```
 level

 Invalid search criteria
```

```
DELETE player
 Enter criteria for deletion:
id , nickname , last_online , health , level_id

kajjfhkfgf
```

```
 player

 Invalid deleting criteria


   1 - Get
   2 - Delete
   3 - Update
   4 - Insert
   5 - Back
```

Генерування випадкових значень в таблиці рівнів

```
Enter quantity:
10
```

```
Generated successfully
```

```
6                    |Dump                |Smelly level              |./blob/dump
7                    |Spaceship           |To the space and beyond   |./blob/spaceship
5                    |Swamps              |Woopsie                   |./blob/swamps
8                    |1ba123eec3          |c0dfa668e32500a           |22297d545cd8aaa
9                    |73e21e7955          |24683c39f1bd1e9           |f77906b6cddb149
10                   |5ea66cb5b1          |742cc568365c118           |43fa0c37aacf65d
11                   |5e48606274          |1fe20eb1efc83cc           |7eb88c723af71c9
12                   |bd8eb61339          |c60c2c6a5f56d0b           |18c87c8829efe53
13                   |b70a319681          |12ae0706bb3504c           |94305ad01f10c0d
14                   |a37f82893d          |8b7c8fbc972ec73           |e4f99c4f20c5f63
15                   |e53d5a6aef          |a5b612edbbb3250           |e83bfb8f58948b9
16                   |b9f3bf7c9e          |55070d10f9fa36a           |0b8b42b8d68279f
17                   |0a1b2a0b74          |9d5aecdcd34017b           |149f8763d32dffc
18                   |0aa95f6d0b          |f2defb354a65e68           |3758dda2697abb5
```

Пошук гравців за назвою равня, на якому вони знаходяться, з вказаним діапазоном здоров’я і частиною нікнейму

```
Enter level name
swamps
Enter min health
100
Enter max health
200
Enter nickname
piano
```

```
Enter level name
swamps
Enter min health
100
Enter max health
200
Enter nickname
piano
-----------------------------------------------------------------------------------------------------------
id                      |nickname               |last_online            |health                 |
-----------------------------------------------------------------------------------------------------------
8                       |PianoCat               |2019-11-23             |130                    |
-----------------------------------------------------------------------------------------------------------

Query time: 0.0031464099884033203
```

Пошук скінів, що мають гравці за діапазоном їх здоров’я, діапазоном дати останнього онлайну і назвою равня на якому вони знаходяться

```
Enter minimum health
1
Enter max health
180
Enter min date player was online
8.12.2007
Enter max date player was online
12.12.2020
Enter level name
swamps
-----------------------------------------------------------------------------------
id                      |title                  |blob                   |
-----------------------------------------------------------------------------------
1                       |Dragon Lore            |./blob/dragonlore      |
2                       |Golden Abyss           |./blob/goldenabyss     |
-----------------------------------------------------------------------------------

Query time: 0.006000518798828125
```

Порахувати гравців, що мають скін за його назвою, назвою рівня, на якому знаходиться гравець, та діапазоном здоров’я

```
Enter skin title
golden aby
Enter level name
swamps
Enter min player health
0
Enter max player health
900
------------------------------
count                        |
------------------------------
2                            |
------------------------------

Query time: 0.0029935836791992188
```

model.py:

```
import psycopg2


class Model:
    def __init__(self):
        try:
            self.connection = psycopg2.connect(host="localhost",
                                               database='lab2_test', user='postgres',
password='111223')
            self.cursor = self.connection.cursor()
        except (Exception, psycopg2.Error):
            print("Error connecting to server")

    def get_col_names(self):
        return [d[0] for d in self.cursor.description]

    def get(self, tname, condition):
        try:
            query = f'SELECT * FROM {tname}'
            if condition:
                query += ' WHERE ' + condition
            self.cursor.execute(query)
            return self.get_col_names(), self.cursor.fetchall()
        finally:
            self.connection.commit()

    def insert(self, tname, columns, values):
        try:
            query = f'INSERT INTO {tname} ({columns}) VALUES ({values});'
            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def delete(self, tname, condition):
        try:
```

```
            query = f'DELETE FROM {tname} WHERE {condition};'
            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def update(self, tname, condition, statement):
        try:
            query = f'UPDATE {tname} SET {statement} WHERE {condition}'
            self.cursor.execute(query)
        finally:
            self.connection.commit()

    def search_players_on_level_by_levelname_health_nickname(self,
levelname, a, b, nickname):
        try:
            query = f"
            SELECT player.id, player.nickname, player.last_online, player.health
            FROM level INNER JOIN player ON level.id = player.level_id
            WHERE LOWER(player.nickname) Like '%{nickname.lower()}%'
AND LOWER(level.title) Like '%{levelname.lower()}%' AND player.health
Between {a} And {b}
            "
            self.cursor.execute(query)
            return self.get_col_names(), self.cursor.fetchall()
        finally:
            self.connection.commit()

    def search_skin_by_playerhealth_online_levelname(self, a, b, date1, date2,
levelname):
        try:
            query = f"
            SELECT skin.id, skin.title, skin.blob
            FROM skin INNER JOIN level INNER JOIN player ON level.id =
player.level_id
            INNER JOIN player_skin ON player.id = player_skin.player_id ON
skin.id = player_skin.skin_id
            WHERE player.last_online Between '{date1}' And '{date2}'
```

```
            AND player.health Between {a} And {b} AND LOWER(level.title)
Like '%{levelname.lower()}%' GROUP BY skin.id
            '''
            self.cursor.execute(query)
            return self.get_col_names(), self.cursor.fetchall()
        finally:
            self.connection.commit()

    def count_players_with_skin_by_skin_name_levelname_health(self,
skintitle, levelname, a, b):
        try:
            query = f'''
            SELECT COUNT (*)
            FROM skin INNER JOIN level INNER JOIN player ON level.id =
player.level_id INNER JOIN player_skin ON player.id = player_skin.player_id
ON skin.id = player_skin.skin_id
            WHERE LOWER(skin.title) Like '%{skintitle.lower()}%' AND
LOWER(level.title) Like '%{levelname.lower()}%' AND player.health
Between {a} And {b}
            GROUP BY skin.title
            '''
            self.cursor.execute(query)
            return self.get_col_names(), self.cursor.fetchall()
        finally:
            self.connection.commit()

    def fill_level_by_random_data(self, quantity):
        sql = f'''
        CREATE OR REPLACE FUNCTION randomLevels()
            RETURNS void AS $$
        DECLARE
            step integer  := 0;
        BEGIN
            LOOP EXIT WHEN step > {quantity};
                INSERT INTO level (title, description, blob)
                VALUES (
                    substring(md5(random()::text), 1, 10),
```

```
                substring(md5(random()::text), 1, 15),
                substring(md5(random()::text), 1, 15)
            );
            step := step + 1;
        END LOOP ;
    END;
    $$ LANGUAGE PLPGSQL;
    SELECT randomLevels();
    '''
    try:
        self.cursor.execute(sql)
    finally:
        self.connection.commit()
```

view.py

```
from consolemenu import *
from consolemenu.items import *


class View:
    def print(self, data):
        columns, rows = data
        lineLen = 30 * len(columns)

        self.print_separator(lineLen)
        self.print_row(columns)
        self.print_separator(lineLen)

        for row in rows:
            self.print_row(row)
        self.print_separator(lineLen)

    def print_row(self, row):
        for col in row:
            print(str(col).ljust(26, ' ') + '   |', end='')
        print('')
```

```
    def print_separator(self, length):
        print('-' * length)
```

controller.py

```
from consolemenu import SelectionMenu
import time

from model import Model
from view import View


TABLES_NAMES = ['level', 'player', 'skin', 'player_skin']
TABLES = {
    'level': ['id', 'title', 'description', 'blob'],
    'player': ['id', 'nickname', 'last_online', 'health', 'level_id'],
    'skin': ['id', 'title', 'blob'],
    'player_skin': ['id', 'player_id', 'skin_id']
}


def getInput(msg, tableName=''):
    print(msg)
    if tableName:
        print(' , '.join(TABLES[tableName]), end='\n\n')
    return input()


def getInsertInput(msg, tableName):
    print(msg)
    print(' , '.join(TABLES[tableName]), end='\n\n')
    return input(), input()


class Controller:
    def __init__(self):
        self.model = Model()
        self.view = View()
```

```
    def run_menu(self, message=''):
        selectionMenu = SelectionMenu(
            TABLES_NAMES + ['Search players by level name with set health and
nickname',
                            'Find skins by player health, last online and level',
                            'Count players who own skin by skin name, level title and
health',
                            'Fill table "level" by random data'], title='Menu:',
subtitle=message)
        selectionMenu.show()

        index = selectionMenu.selected_option
        if index < len(TABLES_NAMES):
            tableName = TABLES_NAMES[index]
            self.show_entity_menu(tableName)
        elif index == 4:
            self.search_players_on_level_by_levelname_health_nickname()
        elif index == 5:
            self.search_skin_by_playerhealth_online_levelname()
        elif index == 6:
            self.count_players_with_skin_by_skin_name_levelname_health()
        elif index == 7:
            self.fill_level_by_random_data()
        else:
            print('Closing...')

    def show_entity_menu(self, tableName, msg=''):
        options = ['Get',  'Delete', 'Update', 'Insert']
        functions = [self.get, self.delete, self.update, self.insert]

        selectionMenu = SelectionMenu(options, f'{tableName}',
                                      exit_option_text='Back', subtitle=msg)
        selectionMenu.show()
        try:
            function = functions[selectionMenu.selected_option]
            function(tableName)
```

```
    except IndexError:
        self.run_menu()

def get(self, tableName):
    try:
        condition = getInput(
            f'GET {tableName}\nEnter search criteria:', tableName)
        start_time = time.time()
        data = self.model.get(tableName, condition)
        self.view.print(data)
        print("\nQuery time:", time.time() - start_time)
        input()
        self.show_entity_menu(tableName)
    except Exception:
        self.show_entity_menu(tableName, 'Invalid search criteria')

def insert(self, tableName):
    try:
        columns, values = getInsertInput(
            f"INSERT {tableName}\nEnter columns, then values", tableName)
        self.model.insert(tableName, columns, values)
        self.show_entity_menu(tableName, 'Insert is successful!')
    except Exception as err:
        self.show_entity_menu(tableName, 'Invalid insert arguments')

def delete(self, tableName):
    try:
        condition = getInput(
            f'DELETE {tableName}\n Enter criteria for deletion:', tableName)
        self.model.delete(tableName, condition)
        self.show_entity_menu(tableName, 'Delete is successful')
    except Exception:
        self.show_entity_menu(tableName, 'Invalid deleting criteria')

def update(self, tableName):
    try:
        condition = getInput(
```

```
            f'UPDATE {tableName}\nEnter criteria:', tableName)
        statement = getInput(
            "Enter columns and their new values", tableName)
        self.model.update(tableName, condition, statement)
        self.show_entity_menu(tableName, 'Update is successful')
    except Exception:
        self.show_entity_menu(tableName, 'Invalid update values')

def search_players_on_level_by_levelname_health_nickname(self):
    try:
        levelname = getInput('Enter level name')
        a = getInput('Enter min health')
        b = getInput('Enter max health')
        nickname = getInput('Enter nickname')
        start_time = time.time()
        data =
self.model.search_players_on_level_by_levelname_health_nickname(levelnam
e, a, b, nickname)
        self.view.print(data)
        print("\nQuery time:", time.time() - start_time)
        input()
        self.run_menu()
    except Exception:
        self.run_menu('Invalid search arguments')

def search_skin_by_playerhealth_online_levelname(self):
    try:
        minhealth = getInput('Enter minimum health')
        maxhealth = getInput('Enter max health')
        mindate = getInput('Enter min date player was online')
        maxdate = getInput('Enter max date player was online')
        levelname = getInput('Enter level name')
        start_time = time.time()
        data =
self.model.search_skin_by_playerhealth_online_levelname(minhealth,
maxhealth, mindate, maxdate, levelname)
        self.view.print(data)
```

```
            print("\nQuery time:", time.time() - start_time)
            input()
            self.run_menu()
        except Exception:
            self.run_menu('Invalid search arguments')

    def count_players_with_skin_by_skin_name_levelname_health(self):
        try:
            skintitle = getInput('Enter skin title')
            levelname = getInput('Enter level name')
            minhealth = getInput('Enter min player health')
            maxhealth = getInput('Enter max player health')
            start_time = time.time()
            data =
self.model.count_players_with_skin_by_skin_name_levelname_health(skintitle
, levelname, minhealth, maxhealth)
            self.view.print(data)
            print("\nQuery time:", time.time() - start_time)
            input()
            self.run_menu()
        except Exception:
            self.run_menu('Invalid search arguments')

    def fill_level_by_random_data(self):
        try:
            quantity = getInput('Enter quantity:')
            self.model.fill_level_by_random_data(quantity)
            self.run_menu('Generated successfully')

        except Exception:
            self.run_menu('Invalid quantity')
```